UNIVERSITY OF BAHRAIN

College of Information Technology



Artificial Bee Colony Algorithm for Optimizing Enterprise Network Topology Design with Multiple Objectives

A Thesis Submitted in Partial Fulfillment of the Requirements for the Master Degree in Information Technology

Submitted by

Amani D Saad Raheej

ID number: 20121061

Supervised by

Dr. Amjad Mahmood
(Associate Professor)
University of Bahrain

Dr. Salman A. Khan
(Assistant Professor)
University of Bahrain

Kingdom of Bahrain

December, 2015

DECLARATION

I declare that this work is the result of my own investigation and that it has not already been accepted in substance for any degree, nor is it currently submitted for any degree.

Signed: ---------- Date: 31-12-2015

Amani D. Saad Raheej (Candidate)

The Thesis defense committee considers the Thesis titled:

Artificial Bee Colony Algorithm for Optimizing Enterprise Network Topology Design with Multiple Objectives

Submitted by Amani D. Saad Raheej to the College of Information Technology is satisfactory and acceptable for the Master of Science in Information Technology.

Thesis Defense Committee:

Dr. Amjad Mahmood
(Supervisor)
Department of Computer Science
University of Bahrain

Dr. Salman Khan
(Co-Supervisor)
Department of Computer Engineering
University of Bahrain

Prof. Mostafa I. H. Abd-El-Barr
(Member)
Kuwait University
Kuwait

Prof. Nabil M. Hewahi
(Member)
Department of Computer Science
University of Bahrain

Abstract

The problem of topological design of distributed local area networks (DLAN) is a combinatorial NP-hard problem. The problem complexity is further magnified by the presence of multiple objectives which may be conflicted in nature, that require to be optimized systematically and simultaneously. The improvement in one of the objectives may lead to a certain level of degradation in one or more other objectives. In this thesis, the problem of DLAN tree topology design considers five design objectives: reliability, network availability, average link utilization, average network delay, and monetary cost. There are two aspects of the problem under consideration. First is the multi-objectivity and second is the optimization. The optimum design is governed by both issues.

Historically, many techniques have been proposed over the years to handle the multi-objective aspects of an optimization problem. In this thesis Goal Programming (GP) is adopted to combine the five design objectives into a single overall objective function. The second aspect of the problem is the algorithm that performs the optimization process. Swarm intelligent techniques are proven to be able to solve complex problems within reasonable computational time and thus, have been extensively deployed to solve variants of NP-hard problems. In this thesis, the artificial bee colony optimization algorithm (ABC) is adapted and employed. To ascertain the performance of the adapted ABC algorithm, ant colony optimization algorithm (ACO) is also adapted for the underlying problem as another swarm based algorithm to facilitate the intended comparison. In the results, ACO outperformed ABC for test cases with relatively smaller problem dimensions while for larger dimensions ABC performed better. The results also showed that ABC is a high-performing algorithm with powerful search capabilities and could be tailored to suit large problem dimensions.

This work is dedicated to my beloved mother.

Acknowledgements

First and foremost, praises and thanks to the Almighty God for His limitless blessings throughout my research to complete this work successfully.

I wish to express my enormous gratitude and appreciation to my supervisors, Dr. Amjad Mahmood and Dr. Salman A. Khan, I would especially like to thank them for sharing their knowledge and experience with me. I also acknowledge their thoughtful guidance, critical comments, and correction of the thesis. I have been very fortunate that I could work on my thesis under their supervision.

I am extremely grateful to my beloved parents for their love, caring, and patience. Only due to their prayers I could finish this thesis. I am also extending my heartfelt thanks to all my family and friends for their genuine support and constant encouragement throughout this research work.

Table of Contents

Subject	Page number
Defense Committee's Approval	I
Abstract	II
Dedication	III
Acknowledgment	IV
Table of Contents	V
List of Tables	VIII
List of Figures	IX
List of Symbols	X
List of Appendices	XI
Chapter One Introduction	1
1.1 Background	1
1.2 Preliminary	5
1.3 Problem Statement	7
1.4 Motivation	7
1.5 Research Objectives	9
1.6 Methodology	10
1.7 Contributions of The Thesis	11
1.8 Thesis Outline	13
Chapter Two Literature Review	15
2.1 Network Topological Design Optimization Problem	15
2.2 Artificial Bee Colony Algorithm	17
2.2.1 Fundamental Characteristics of Artificial Bee Colony Algorithm	23
2.3 Ant Colony Optimization Algorithm	25
2.4 Concluding Remarks	29
Chapter Three Problem Formulation	32
3.1 Assumptions	32
3.2 Design Objectives	33

3.2.1 Network Reliability	33
3.2.2 Network Availability	34
3.2.3 Average Link Utilization	36
3.2.4 Monetary Cost	37
3.2.5 Average Network Delay	38
3.3 Constraints	39
3.4 Goal Programming Approach to the DLAN Topology Design Problem	40
3.5 Concluding Remarks	48
Chapter Four **Proposed Algorithms for DLAN Topology Design**	50
4.1 Goal Programming Based ABC Algorithm	50
4.1.1 Initialization (Generation of Bees)	51
4.1.2 Parameter Setting	51
4.1.3 Solution Representation	52
4.1.4 Bees Activity	52
4.1.5 The Framework of ABC Algorithm	56
4.2 Goal Programming Based ACO Algorithm	57
4.2.1 Initialization	58
4.2.2 Ants Activity	58
4.3 Concluding Remarks	59
Chapter Five **Results and Discussion**	61
5.1 Characteristics of Test Cases	61
5.2 Parameter Setting	62
5.3 Results and Discussion Of ABC algorithm	63
5.3.1 Effect of Limit	65
5.3.2 Effect of Colony Size	68
5.4 Results and Discussion for ACO algorithm	74
5.4.1 Effect of Pheromone Deposit and Evaporation	74
5.4.2 Effect of Colony Size	77
5.4.3 Effect of Heuristic Values	81
5.5 Concluding Remarks	79
Chapter Six **Comparison of Techniques**	86

6.1 Results and Discussion.	86
6.2 Concluding Remarks	91
Chapter Seven **Conclusions and Future Work**	94
7.1 Conclusion.	94
7.2 Future Work.	96
References.	99

List of Tables

NO.	Subject	Page
5.1	Characteristics of test cases used in experiments	62
5.2	Network characteristics considered for experimental study	62
5.3	Parameter settings for ABC used in experiments	64
5.4	Limit values used in the experiments	64
5.5	Results for various limit values for all test cases	66
5.6	Percentage differences for different limit sets	67
5.7	Results for *n15* for different population size	69
5.8	Results for *n25* for different population size	69
5.9	Results for *n33* for different population size	70
5.10	Results for *n40* for different population size	70
5.11	Results for *n50* for different population size	70
5.12	Percentage differences for different population sizes for all limit sets	72
5.13	Parameter settings for ACO used in experiments	75
5.14	Results for best and worst average overall deviation and their corresponding DEP setup	76
5.15	Results for *n*15 for different colony sizes	78
5.16	Results for *n*25 for different colony sizes	79
5.17	Results for *n*33 for different colony sizes	79
5.18	Results for *n*40 for different colony sizes	79
5.19	Results for *n*50 for different colony sizes	80
5.20	Percentage differences with regard to increase in colony size	81
5.21	Parameter settings for various heuristic values used in experiments.	82
5.22	Results for $\alpha = 0.2, \beta = 0.8$	82
5.23	Results for $\alpha = 0.8, \beta = 0.2$	82
6.1	Comparison of ACO and ABC	87

List of Figures

NO.	Figure	Page
1.1	Typical Distributed Local Area Network DLAN	6
2.1	Types of agents in ABC algorithm	22
2.2	Pseudo-code of ABC algorithm	25
2.3	Pseudo-code of ACO algorithm	28
3.1	Characteristics of a quality topology	42
3.2	The structure of membership functions used to express the achievement level of design objectives	44
4.1	The proposed Local search operator	54
4.2	Flowchart of the ABC algorithm	56
4.3	Steps of the proposed ABC algorithm	57
5.1	Effect of limit on overall average deviation for (a) n15 (b) n25 (c) n33 (d) n40 (e) n50	68
5.2	Solutions Diversity for n50 for Colony size = 25 and no limit	73
5.3	Effect of different DEP rates on overall average deviation for the case *n*50	78
6.1	Plots of solutions obtained for ABC and ACO for (a) n15 (b) n25 (c) n33 (d) n40 (e) n50	91

List of Symbols

NO.	Symbol
1	α : Influence of pheromone content
2	β: Influence of heuristic value
3	η: The heuristic value of connection
4	τ: The pheromone on connection
5	λ: Traffic on connection
6	γ: Total traffic in the network
7	μ: Value of membership function of an objective
8	μ°: Deviational variable of an objective
9	σ : Coefficients of the objective function
10	∞: No limit

List of Appendices

No.	Appendix
(1)	Appendix A (Statistical Analysis)

Chapter One
Introduction

1.1 Background	1
1.2 Preliminary	5
1.3 Problem Statement	7
1.4 Motivation	7
1.5 Research Objectives	9
1.6 Methodology	10
1.7 Contributions of The Thesis	11
1.8 Thesis Outline	13

Chapter One

Introduction

This chapter provides a brief overview of this work. First, the significance of the problem is stated in the Background, along with preliminary information needed for better understanding of the problem. Later, the chapter covers the statement of the problem, motivation of the thesis, and the research objectives. The contributions of the thesis are also covered thereafter. Finally, the followed methodology and the thesis outline are also presented.

1.1 Background

Today, computer networks are vital in every modern enterprise and organization. They have become the prime channel for information sharing, dissemination, and analysis. In today's businesses world, the costs and risks of losing access to business's information is crucial. Indeed, business critical information is considered as a source of knowledge, which is eventually translated into profit. Such information is not only necessary to achieve competitive advantage, but also to respond faster to the rapidly changing businesses environment and keep abreast with the marketplace. For any organization, the most promising resource of information is "data", which is gathered, shared, transported or retrieved throughout its network. Research revealed that interruptions in network services cost large companies a substantial percentage of annual revenues (Dearborn et al. 2006). However, the disastrous results are not limited to monetary costs but perhaps more importantly, in brand and reputation (Hacki & Lighton 2001).

Modern businesses rely heavily on their computer networks to provide certain level of services that are essential for business survival. The primary and obvious benefits of computer networks

other than storing, accessing and retrieving information, are making the traditional business processes manageable and efficient. Computer networks provide a shared medium that combines both skills and capabilities of diverse components and equipments regardless of the physical location of these components. If the information is to be viewed as food particles, then a network can be viewed as blood cells used to carry these particles to all the enterprise organs. However, without a proper layout of the network, it may suffer from "info-sclerosis" which is defined as the problem of hardening or clogging of its information arteries (McNurlin et al. 1989). Consequently, network services have become essential to the businesses continuity.

There are quite good reasons why an organization should build an efficient and reliable network. Some reasons are cost reduction, high reliability, decreased latency, better availability, and greater flexibility. Furthermore, it is essential to facilitate communication and resources sharing services. In modern networks, efficient network layout configuration is a daunting task that becomes more difficult as the network increases in size and diversity. Currently, the prime challenge for any organization is to build a powerful network architecture that can deliver the required performance quality and reliability in a cost-effective manner. On its most fundamental level, a computer network consists of a number of devices that are connected to each other via a communication channel. These devices are mainly divided into two categories: end-users, which consist of computers, printers, and other peripheral devices, and other network elements such as routers, switches, and hubs. End-users are connected to network elements throughout the communication channel that allows sharing and exchanging of resources and information. In computer networks, networked devices pass data to each other along the communication channel media. The connections are established using either cable media or wireless media. The main

distinguishing factor between wired and wireless technology options in a business is financial costs. Wireless options are often more expensive that makes purchasing and installing wired devices a financial benefit for businesses networks (Fielding et al. 2000).

Computer networks are often organized hierarchically, in which a small network (referred to as a node herein) is connected to a larger one to form a multi-tier, incrementally expanded network. As a result, a huge number of layouts are possible, whereas each such layout is termed as a network topology. Due to the specific nature of computer networking technology, varying network topology significantly affects the entire network performance (Stuck & Arthurs 1984). With many topologies, such as bus networks in which all nodes are connected to a common medium a single failure can cause failure to the entire network. Generally, the more interconnections there are, the more robust the network is, but the more expensive it is to install and configure.

In this thesis, a tree topology is considered for a network layout. In a tree topology, each node is connected to other nodes via a unique path without redundancy, which indicates that a single path exists between two communicating nodes. The objective then is to solve the problem of finding the optimal topology grid layout for a tree topology. Designing an optimum topology is a major and complex task since for n number of connected nodes, there are $2^{\frac{n(n-1)}{2}-1}$ distinct topologies (Khan 2009). Hence, for a network consisting of only ten nodes, there will be 2^{44} distinctive topologies. A typical real network comprises tens of interconnected nodes, which discordantly increases the number of possible topologies. It is, therefore, a computationally complex task to find the best topology. This complexity is further magnified by the presence of various design constraints. Based on these facts, network topology design has been classified as an NP-hard problem (Garey & Johnson 1979).

It is an established fact that network architecture plays a

significant role in achieving appropriate and cost-effective performance (Pandya 2013). The performance of a given topology is usually measured according to a number of predefined and often conflicting objectives such as availability, utilization, reliability, cost, delay, hop count, throughput, and so forth. Although these objectives are closely related to each other, topology optimization often compromises between the various design criteria. For example, management of an organization might be concerned mostly about the financial cost, while for the user, network delay might be a more important issue. Likewise, a network administrator's major concerns might be reliability and availability along with link utilization. Hence, the determination of an optimal topology is a major challenge with respect to these conflicting objectives. In this thesis, five conflicting optimization objectives are considered. These are network reliability, network availability, average link utilization, monetary cost, and network delay.

Due to its sheer complexity, network topology design, with its numerous variants, has been traditionally solved with artificial intelligence techniques. More specifically, nature-inspired and other meta-heuristics have been extensively used for this purpose (Kampstra 2005). Popular algorithms used in the domain include genetic algorithms (Kumar et al. 2002), particle swarm optimization (Khan & Engelbrecht 2012*b*), ant colony optimization algorithms (Khan et al. 2008), simulated annealing (Pierre et al. 1995), tabu search (Dengiz & Alabas 2000), among many others. Artificial bee colony optimization algorithm (ABC) is another relatively new meta-heuristic algorithm that has received much attention throughout the years, and will be the main focus of this thesis (Karaboga & Akay 2009*b*, Karaboga, Gorkemli, Ozturk & Karaboga 2012).

1.2 Preliminary

A computer network interconnects collection of computing devices to provide communication services (sharing of resources and exchange of information). These devices are primarily divided into two types: hosts (or end systems) and other network elements. As computing devices, hosts may include traditional personal computers, workstations, printers, fax machines and other peripherals. To interconnect these hosts, special network elements such as routers, switches, hubs and bridges are used. As the network grows in size and diversity, it may become necessary to combine two or more separate sub-networks. These sub-networks are often called network segments. The prime network that has these interconnected segments is termed as "internetwork". The main reason for network segmentation is to enhance its performance. A network with a chaotic layout may fail to provide the required level of services. In this case, if too many users are accessing and using the shared resources simultaneously, then more time will be required to handle the congested communication channels. As a result, the network will be overloaded. In contrast, dividing the network into other smaller sub-networks simplifies management and controlling aspects and reduces the effect of single device failure in any part of the network.

In modern enterprises, the network is hierarchically structured. That is, segments or sub-networks in the same physical location are usually connected together to form a local area network (LAN). Then, a central backbone is used to connect several LANs over geographically distributed locations to form what is known as distributed local area network DLAN. The arrangement of network components significantly affects its performance (Bisht & Singh 2015). In order to work properly, these components should be arranged in a specific way. Basically, the arrangement usually depend on hardware capabilities and other characteristics of the

required network specifications. Such arrangements design is called network topology design. The layout of a typical DLAN is shown in Figure 1.1.

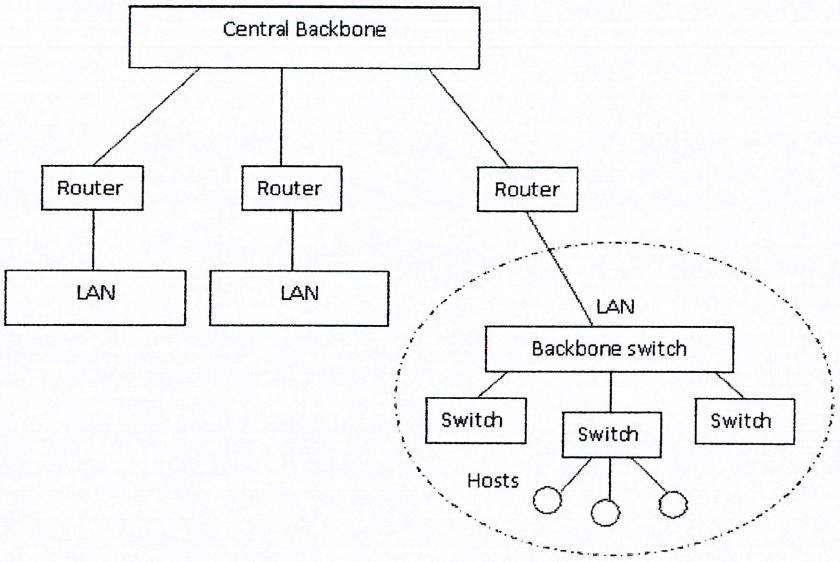


Figure (1.1)
Typical Distributed Local Area Network DLAN

In computer networking domain, the term "network topology" refers to the grid layout of a computer network. The term is often used to provide a description of the actual physical organization of network components. There are many possible topologies for a network (Pandya 2013, Bisht & Singh 2015). One of the most common topologies is the tree topology (which is considered in this thesis). In its simplest form, each node is attached to a central connection node via its own connection cable. The more complex version of a star topology would be comprised of multiple central connection nodes extendedly connected to a central connection node to form a string of sub-trees. The tree topology is highly preferred by most network designers and installers currently. The main advantageous feature of this topology is that the network can be easily expanded by adding devices anywhere on the network. Another benefit is that

it is easy to control such topology as the root node provides centralized management and facilitates monitoring the network (Stuck & Arthurs 1984, Santra & Acharjya 2013).

1.3 Problem Statement

Several formal definitions of the topological design problem can be found in the literature (Ersoy & Panwar 1993, Fencl et al. 2011). Generally, it is the problem of finding a quality feasible topology that satisfies design objectives and adhere to technical constraints. However, in this thesis, topology for a DLAN is considered. That is, if the topology of each LAN is already designed and known, then the aim is to design the backbone topology that connects all LANs without redundancy. Each LAN is referred to as a node. All nodes are connected in a tree topology without any loop. In other words, there is a single path from the root to each node or sub LAN. The overall resulting topology should be a spanning tree topology.

In the above formulation, for each candidate topology, there are two judgemental factors: First, feasibility of the topology that satisfies all the design constraints and second, quality of the topology that optimizes the design objectives. In this thesis, the quality of a feasible topology is measured with respect to five objectives, network reliability, network availability, average link utilization, monetary cost, and average network delay.

1.4 Motivation

The solution to info-sclerosis is to establish an efficient and cost effective dissemination channel. This requires the optimum design of a network, and network topology plays a main role in this process, with consideration of multiple design objectives. Thus, the network topology design problem can be viewed as a multi-objective optimization problem. There are two aspects of the

problem. First is the multi-objectivity and second is the optimization. The optimum design is governed by both issues. In the proposed topology design problem, there are many design objectives which need to be optimized. However, optimizing multiple objectives simultaneously is a complex task due to the *conflicting* nature of these objectives. That is, if one tries to optimize one objective, then another one may be negatively affected. For example, if financial cost is reduced, then it would result in a less reliable network. In this situation, a compromise or trade-off is required between the objectives that would find the optimum balance between the design objectives.

Many techniques have been proposed over the years to handle the multi-objective aspects of an optimization problem. Goal Programming (GP) is one of the famous approaches, proposed by (Charnes et al. 1955). As a multi-objective optimization approach, goal programming was defined to address linear models and has proven to be efficient and effective approach to address various industrial optimization problems. The approach is to set a typical value for each objective parameter to be considered as a targeted goal. Then, the aim is to minimize the absolute deviations of all objective parameters simultaneously. This is achieved by minimizing the sum of the absolute values of the differences between target values and actually achieved values (Coello 1998).

The second aspect of the multi-objective network topology design problem is the algorithm that performs the optimization process. Many optimization algorithms have been proposed over the last fifty years. Some of these are *constructive* and others are *iterative*. Although constructive algorithms are fast in terms of giving a final solution, they do not perform well in situations where multiple objectives are present, coupled with design constraints. In many cases, constructive algorithms fail to even give a feasible solution. In such situations, researchers resort to iterative algorithms which possess the capability to find optimum

or near-optimum solutions in a huge search space within reasonable amount of time. Attempts have been made to efficiently solve the variants of network topology design problem using iterative algorithms mentioned earlier such as genetic algorithms, particle swarm optimization, simulated annealing, tabu search, ant colony optimization, and simulated evolution. However, there are still a list of recent optimization algorithms whose effectiveness levels have yet to be explored, specially in the context of multi-objective network topology design problem. One such algorithm is Artificial Bee Colony (ABC) optimization algorithm which has been applied to a number of optimization problems, but is yet to be applied to network topology design problem considered herein.

Apart from the fact that ABC has not been applied to the network topology design problem as yet, another strong reason to consider ABC for the proposed work is the fact that the algorithm completely fulfils the requirement of being a strong swarm-based optimization algorithm that has been successfully applied to various NP-hard problems (Karaboga & Akay 2009*b*, Karaboga, Gorkemli, Ozturk & Karaboga 2012).

1.5 Research Objectives

The objectives of the proposed research are as follows:

1. To address the problem of topological design of distributed local area network (DLAN) as a multi-objective optimization problem using goal programming approach (GP). Five design objectives are considered, which are: network reliability, network availability, average link utilization, monetary cost, and average network delay.
2. To investigate the applicability and effectiveness of Artificial Bee Colony (ABC) algorithm for multi-objective DLAN topology design as an optimization problem.

3. To adapt a multi-objective Artificial Bee Colony optimization algorithm for the DLAN topology design problem and analyze the performance of the algorithm.
4. To adapt a multi-objective Ant Colony optimization algorithm (ACO) for the DLAN topology design problem and analyze the performance of the proposed algorithm. The aim is to comparatively ascertain the performance of the proposed ABC algorithm.
5. To compare the relative performance of both Artificial Bee Colony algorithm (ABC) and Ant Colony Optimization (ACO) in order to evaluate the performance of the proposed algorithms.

1.6 Methodology

The addressed problem of DLAN topology design is first defined and formulated in terms of five design objectives being considered. These are: network reliability, network availability, average link utilization, monetary cost, and average network delay. Goal programming approach for aggregating the design objectives into single optimization function is defined and explained. Then, ABC algorithm is investigated and tailored to suite the application domain. The performance of the algorithm is experimentally examined and analyzed. Relevant test cases found in the literature are used to assess the achieved performance. Values of dominant control parameters are investigated and optimized to reach the best possible performance level.

To ascertain the performance of the adapted algorithm, the same experiments are to be conducted for ACO which is another population based meta heuristic. For each of the algorithms, different values for control parameters are to be considered when investigating their performance. Given the same computational time interval and same initial seed solutions while setting the control

parameters to its optimal values, the performance of both ACO and ABC algorithms is to be examined. Empirical results collected for both algorithms are then mutually compared against each other. To accomplish this, a code in C++ is developed. Five test cases used in many previous studies are used for experimentation and evaluation. As per the standard practice in the domain, results are reported in terms of averages and standard deviations throughout 30 independent run, each of which comprised of 100 iterations. Furthermore, results are validated statistically using the Wilcoxon ranked-sum test which is a well-known test used for such validations (Hole 2011)."Statistica" package is used for statistical analysis.

1.7 Contributions of The Thesis

Considering the various issues discussed earlier, following are the major contributions of the thesis:

1. **Formulation of multi-objective enterprise network topology design problem.** This involves proposing a new variant of the network topology design problem. Previous studies on the work (Khan & Engelbrecht 2012*c*, Khan 2009) considered four optimization objectives, namely, network financial cost, network latency, maximum hop count between two nodes, and network reliability. The novel aspect of the proposed work is the formulation of a new multi-objective problem considering five optimization objectives. These are network financial cost, average network delay, network reliability, average link utilization, and network availability.
2. **Employing goal programming as a tool for formulating a single multi-objective optimization function for the underlying problem**: The multi-objective aspect of DLAN topology design problem is handled by incorporating goal programming approach. More specifically, this requires

aggregating the aforementioned five optimization objectives into a single optimization function via goal programming approach. This includes applying the fuzzy goal programming variant that is specifically adapted to suit to problem under consideration (Yaghoobi et al. 2008).

3. **Adapting the artificial bee colony optimization algorithm for multi-objective enterprise network topology design**: This includes tailoring the general ABC algorithm to become a multi-objective optimization algorithm considering the five objectives mentioned in point (1) while using the goal programming approach in point (2).
4. **Introducing a goal programming based Ant Colony Optimization algorithm**: A modification of the ant colony optimization algorithm (ACO) proposed in (Khan 2009) employed the fuzzy logic approach for multi-objective optimization. The proposed ACO in (Khan 2009) considered financial cost, network latency, maximum number of hops between a source destination pair, and network reliability as the optimization objectives. In this work, the modifications involves employing goal programming, instead of fuzzy logic, for multi-objective optimization. The modified ACO also considers the five objectives mentioned in point (1) above in the optimization process.
5. **Experimental study and empirical analysis of the adapted algorithms**: An empirical analysis of the proposed goal programming based multi-objective swarm-based algorithms is carried out by investigating various values of control parameters to optimize the performance.
6. **Mutual comparison of the proposed algorithms**: In order to ascertain the effectiveness and efficiency of the proposed ABC algorithm, a comparison is performed with the modified ACO algorithm mentioned in point (5) above.

1.8 Thesis Outline

Chapter 1 provides an introduction to the thesis.

Chapter 2 sheds the light on the existing literature associated with DLAN topology design problem, Artificial Bee Colony algorithm (ABC), and Ant Colony optimization (ACO) algorithm. Important relevant events are presented in a chronological order. Previous studies are also grouped, analyzed, and argued. The literature review related to ACO algorithm is also explored and presented to facilitate the comparison performed in proceeding chapters.

Chapter 3 focuses on DLAN topology design problem considered in this thesis. Firstly, the problem is reviewed along with the main related terminology. Then, the basic associated requirements are adequately addressed in this chapter. This includes formal statement of the problem, notation, assumptions, objective function, and computational method of objectives values.

Chapter 4 provides the detailed description of application design of the adapted algorithms using Goal programming (GP) to the problem under study.

Chapter 5 presents and discusses the results of the conducted experiments. The effect of various control parameters is also investigated and studied. The numerical results are presented, analyzed, and studied.

Chapter 6 presents the mutual comparisons carried out between the adapted ABC and ACO algorithms. The obtained results are analyzed thoroughly and discussed.

Chapter 7 presents the conclusions and potential directions for future work.

Chapter Two
Literature Review

2.1 Network Topological Design Optimization Problem	15
2.2 Artificial Bee Colony Algorithm	17
2.2.1 Fundamental Characteristics of Artificial Bee Colony Algorithm	23
2.3 Ant Colony Optimization Algorithm	25
2.4 Concluding Remarks	29

Chapter Two

Literature Review

In this chapter, the existing literature is reviewed in relevance with the research undertaken in this thesis. Since this work focuses on the DLAN topology design problem using Artificial Bee Colony (ABC) optimization algorithm, the literature review encompasses both aspects. In the first section, historical contributions to the problem of DLAN topology design are presented and discussed. Later, a comprehensive presentation of the classical ABC algorithm is given, followed by a discussion of the literature associated with ABC, in a chronological order. Also, as the Ant Colony Optimization (ACO) algorithm is to be applied to the same problem for mutual comparison, the literature related to this algorithm is briefly presented at the end of this chapter.

2.1 Network Topological Design Optimization Problem

Network topological design problem has received much attention throughout years. Researchers and network engineers extensively tried to define the best possible representation of a network topology (Frank et al. 1970, Gerla & Kleinrock 1977, Elbaum & Sidi 1996, Kumar & Banerjee 2003, Habib 2008, Srivatsa et al. 2009, Pal & Sarma 2010, Fencl et al. 2011). Earlier, the problem was thought of as a single optimization problem (Steiglitz et al. 1969), the main design objective to be optimized was cost and a huge literature was devoted to find the minimal cost topology (Gerla & Kleinrock 1977). Later, other performance-related design objectives arose as important criteria to be considered when solving the problem, specifically, average network delay and

network throughput (Kumar & Banerjee 2003). Clearly, the aim was to minimize packet delay and to maximize the entire network throughput. Later on, several variants of topological design problems appeared that correspond to different adopted design objectives and constrains.

In terms of computational complexity, network topology optimization has been classified as an NP-hard problem, even as a single optimization problem when only cost was considered (Kumar & Banerjee 2003). The problem complexity is further magnified by the presence of multiple optimization objectives which are often conflicting and need to be optimized simultaneously. Apart from this, the technical constraints that should be taken into account further increase the complexity when solving the problem. In this case, resorting to traditional exact methods is not recommended due to the huge number of possible topologies as mentioned earlier in Chapter 1. As a combinatorial optimization problem that grows exponentially with the state space, exact techniques turn out to be non-efficient due to the massive computational time required to examine all possible solutions. Therefore, several iterative approaches have been proposed to solve the network topological design problem. In 2005, a comprehensive survey presented the use of evolutionary algorithms to solve various telecommunication design problems including the topological design aspect (Kampstra 2005). The reason for utilizing the evolutionary algorithms is that these algorithms are capable of finding near-optimal solutions in reasonable amount of time.

Several evolutionary approaches have been proposed to address the topological design problem. For example, the deployment of genetic algorithm has been reported in many studies focusing on local area network topology design (Dengiz et al. 1997, Gen et al. 1998, White et al. 1999, Pierre & Legault 1998, Kumar et al. 2002). Other meta heuristics, such as simulated evolution (Youssef

et al. 2002, Khan & Engelbrecht 2011), particle swarm optimization (Khan & Engelbrecht 2012*b*), ant colony optimization algorithms (Khan et al. 2008), simulated annealing (Pierre et al. 1995), tabu search (Dengiz & Alabas 2000), extremal optimization (Khan 2014), have also been utilized. Simulated annealing has been extensively used in the network design domain (Ersoy & Panwar 1993, Fetterolf 1990, Atiqullah & Rao 1993, Harmatos et al. 2000, Thompson & Bilbro 2000, Dengiz & Alabas 2001, Miyoshi et al. 2002, Khan & Engelbrecht 2009). Swarm intelligent techniques are population based algorithms that simulate the original behavior of swarms found in the nature (Beni & Wang 1993). These algorithms are proven to be able to avoid local optimum trapping and solve complex problems within reasonable computational time (Blum & Li 2008). Thus, they have attracted an increasing attention from researchers and scientists in recent years. Ant colony optimization and particle Swarm optimization are popular swarm-intelligence based algorithms that were applied to the problem of network topology design (Khan & Engelbrecht 2012*a*, Khan et al. 2008). However, no application of ABC has been reported in the literature as far as network topological design is concerned.

2.2 Artificial Bee Colony Algorithm

The artificial bee colony (ABC) algorithm is a swarm based meta heuristic algorithm (Karaboga 2010). ABC has received significant attention in different research domains and has been successfully applied to several real world problems (Karaboga, Gorkemli, Ozturk & Karaboga 2012). Honey bee algorithm as an optimization algorithm was introduced by Tereshko, 2000 who developed a model that simulates the foraging behavior of a honeybee colony based on reaction-diffusion equations and studied how communication in the hive determines this behavior (Tereshko

2000). In 2002, Tereshko and Lee studied how mapping information about the explored environment to the hive determines the foraging behavior. The model proposed by Tereshko and Lee deployed two dominant behaviors of colony's foraging behavior: The recruitment-to and the abandonment-of a located food source (Tereshko & Lee 2002). Later, Tereshko and Loengarov (Tereshko & Loengarov 2005*a*) developed a system by considering the foraging behavior of bees for environmental information gathering and adjusting behaviors accordingly. Since then several variants of the same algorithm have been proposed by simulating various bees' behavioral aspects (Bolaji et al. 2013).

Karaboga (Karaboga 2005), was the first who proposed the Artificial Bee Colony algorithm (ABC) simulating the foraging behavior of honey bee colony. The algorithm was deployed to optimize continuous numerical problems. From amongst all other variants, Karaboga's approach has been most widely studied and applied to solve real world problems (Karaboga et al. 2014). Due to the promising results, the algorithm has highly attracted the interest of researchers all over the world. Studies have shown that more than half of the published relevant literature belongs to this variant (Karaboga & Akay 2009*b*). Right after the development of ABC, many papers were published to demonstrate the framework of this interesting algorithm (Basturk & Karaboga 2006, Karaboga & Basturk 2007*b*, Karaboga & Akay 2009*b*, Karaboga, Gorkemli, Ozturk & Karaboga 2012).

Investigation from analytical perspective showed that the studies performed on ABC algorithm were categorized into three main directions: comparisons and modifications, hybrid models, and applications (Karaboga, Gorkemli, Ozturk & Karaboga 2012). A brief description of each perspective is given below.

1. **Modification and Comparison:** The first direction focused on performance evaluation and comparison of ABC with other

well-known evolutionary algorithms such as Genetic Algorithms (GAs), particle swarm optimization (PSO), and ant colony optimization (ACO) (Karaboga & Akay 2009*a*, Li et al. 2010, Mala et al. 2009). Successive and intensive comparative studies revealed the effectiveness of this algorithm to address almost all targeted problem domains (Karaboga & Basturk 2008). Tailoring the algorithm to meet the requirement of various application domains was the main reason for emergence of the huge variants of ABC (Bolaji et al. 2013).

2. **Hybridization:** The second aspect focused on hybridization and hybrid versions of the algorithm (Ji et al. 2013, Milan n.d.). This ABC type is called hybridized ABC. The performance of the standard algorithm is improved by incorporating other traditional and evolutionary algorithms. Generally, the aim is either to avoid some drawbacks or to borrow some desirable features from other algorithms, as in (Wu & Fan 2011) where chaotic ABC was introduced to avoid local optimum trapping and stagnation.

3. **Deployment and Application:** The third dimension focused on applications and deployment in various fields and disciplines such as in neural networks training (Ozturk & Karaboga 2011), industrial engineering (Pan et al. 2011), electrical engineering (Zhang et al. 2014), mechanical engineering (Şahin et al. 2011), electronic engineering (Ghasemi et al. 2012), civil engineering (Szeto et al. 2011), image processing (Akay & Karaboga 2015), software engineering (Farshidpour & Keynia 2012), control engineering (Gozde et al. 2012), data mining (Ozturk et al. 2015), and also in sensor networks (Karaboga, Okdem & Ozturk 2012). Although the original ABC was originally developed for solving numerical problems, the extended

versions have been introduced for discrete and combinatorial problems.

The classical version of ABC algorithm (Karaboga 2005) consists of two essential components; these are food sources and agents. Food sources represent possible solutions and are selected and prioritized based on several properties such as richness, proximity, and taste. Agents are further divided into two types: employed foragers and unemployed onlookers and scouts. Following is a brief discerption of these components:

Agents: The first component of ABC algorithm is agents (bees recruited by the mother beehive). These agents are categorized into two major groups based on their functionality (Karaboga & Akay 2009*b*). The first group, employed agents, each of which is recruited to utilize a particular food source. The second group, unemployed agents, are further classified into two sub-groups, namely unemployed onlookers and unemployed scouts. The unemployed onlookers wait in the hive. Their duty can be viewed as information sharing channels. Finally, the scouts, are unemployed agents that search globally for new food sources.

Employed Agents: These are also called the foragers that collect nectar iteratively from food sources. An employed forager is recruited at a specific food source that is currently exploiting. Upon reaching the source, the forager generates a new neighbor solution and compare it with the original solution. The forager then keeps the best solution in the current population in a greedy selection procedure. The foragers also carry information about their food sources and share it with other unemployed bees. Such information includes proximity, direction and profitability of discovered food sources. The process of information sharing takes place in the hive and is called the waggle dance. In the waggle dance, the probability of each food source is calculated based on its fitness as shown in the following equation (Karaboga &

Basturk 2007*a*):

$$p_i = \frac{fit_i}{\sum_{j=1}^{N} fit_j} \tag{2.1}$$

Where fit_i is the fitness of i^{th} food source, and N is the total number of food sources.

Each forager is usually associated with a specific food source. The forager exploits its food source by loading the nectar from the food source and unloading at a particular station. Based on the nectar amount, the forager is either instructed to return to the hive or continue foraging. The food source is abandoned if its nectar amount went below a pre-defined threshold value called the "limit". The associated forager of an abandoned food source is turned to an unemployed agent that search globally for new sources.

Unemployed Agents: As mentioned earlier, these agents can be extendedly divided into two groups. The onlookers, wait in the mother hive and participate in waggle dance. The waggle dance is a process of sharing and analyzing information brought by both foragers and scouts. The process varies in its form and frequency to express the quality of the food source (Dyer 2002). Based on information extracted from the waggle dance, an onlooker recruits itself as a forger to utilize a particular food source. The recruitment is proportional to the profitability of a food source (Tereshko & Loengarov 2005*b*).

Onlookers are unemployed agents that wait in the hive and participate in the waggle dance. The waggle dance differs in form and duration to express the food source direction and distance (Dyer 2002). Each onlooker selects to recruit itself to a specific food source in probabilistic selection procedure. In which, the number of onlooker bees recruited at a specific food source is proportional to the fitness of the food source itself. Eventually, onlookers are transformed to foragers and perform the same greedy selection process upon reaching their correspondent food sources.

The second group of unemployed bees are scouts. The scouts search the environment randomly for new food sources (Seeley & Visscher 1988). A food source which could not be improved within a limited number of trials (termed as Limit) will be abandoned. The correspondent forager of an abandoned food source is turned into a scout. Obviously, increasing the number of foragers and onlookers encourages the exploitation process. Increasing the number of scout bees encourages the exploration process. Usually, the number of employed bees is equal to the number of onlooker bees and is the same number of food sources. Figure 2.1 illustrates various types of agents in ABC algorithm.

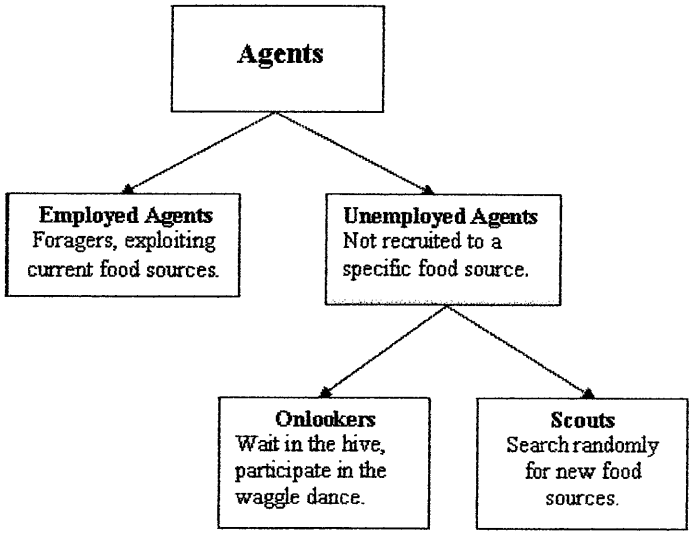


Figure (2.1)
Types of agents in ABC algorithm

Food Sources: A food source represents a possible solution to the problem instant, and the nectar amount of a food source or richness implicitly expresses its profitability, i.e quality of the solution. Several properties are considered when a food source is to be evaluated. In the natural environment of real bees, a food source is characterized by its path availability, richness, proximity, ease of extraction and nectar taste (Karaboga, Gorkemli, Ozturk & Karaboga 2012). In the ABC, a fitness function is defined to evaluate food sources. Usually, the food sources that produce

higher values for the fitness function are preferred. Balancing the process of exploitation of current food sources and exploration of new food sources is governed by the colony decision (Seeley 2009). In the ABC algorithm, the foragers and onlookers carry out the exploitation process while the scouts control the exploration process in the search space.

2.2.1 Fundamental Characteristics of Artificial Bee Colony Algorithm

ABC simulates the natural behavior of a honeybee swarm. The interest in this algorithm has increased significantly in recent years. As a relatively new swarm based algorithm, this fascinating technique maintains a set of desirable features that attract researchers and scientists all over the world. These features are listed below:

1. Millonas stated five principles that should be proven for any swarm to maintain an intelligent behavior (Millonas 1994). These principles are given bellow and are completely satisfied by ABC.

   Proximity Principle: The ability to perform simple space and time computations.

   Quality Principle: The ability to respond to quality factors in the environment.

   Diversity Principle: Diversely respond through broad channels to preserve the fine distribution of resources.

   Stability Principle: The behavioral mode should not change upon every environmental fluctuation.

   Adaptability Principle: The ability to change the behavioral mode when needed.

2. Bonabeau presented two concepts, these are self-organization and division of labor which are fundamental key features for

any algorithm to maintain intelligent swarm behavior (Bonabeau et al. 1997, 1999). ABC employs these concepts to obtain swarm intelligence. These concepts can be defined as:

Self-Organization: The interactions of the swarm's low-level components that leads to a structure at the global level of the swarm. This requires the presence of mutually tolerant agents which are affected by their activities interchangeably to develop a self-decision making behavior.

Division of Labor: The concept of having cooperating specialized agents to perform the various tasks in the swarm simultaneously.

3. Modularity: The system derived from this algorithm can be modularized into number of distinctive components. Each of which might encourage an individual behavioral mode of the algorithm, such as queen bee, drones, workers, mating flight, foraging, dance, nest site selection, navigation, and task selection (Karaboga, Gorkemli, Ozturk & Karaboga 2012).
4. The foundation of algorithmic structure: The role distribution and successively performed exploitation/exploration processes have led to the required algorithmic structure. Figure 2.2 illustrates the pseudo-code for ABC algorithm (Karaboga 2010).
5. Simplicity, flexibility, and robustness: ABC has fewer control parameters when compared to other meta heuristics. In the basic ABC, there are only three control parameters: The number of food sources, which is equal to the number of foragers or onlooker bees, the limit, and a termination criterion. Apart from this, ABC is characterized by its ease of implementation and efficient performance in less computational time (Karaboga & Basturk 2008).

```
Begin;
    Repeat;
            Bees_generation();
            Construct_Solutions();
            Employed Bees Phase();
            Waggle Dance();
            Onlooker Bees Phase();
            Scout Bees Phase();
            Memorize Best Solution();
    Until MCN (Maximum Cyclic Number)
    Return Best Solution;
End;
```

Figure (2.2)
Pseudo-code of ABC algorithm

2.3 Ant Colony Optimization Algorithm

Ant Colony Optimization (ACO) is a well-known swarm based meta heuristic (Dorigo 2007). The algorithm mimics the behavior of real ants found in nature. Since its emergence, ACO has been applied to a wide range of optimization problems such as travelling salesman problem, scheduling problems, structural and concrete engineering problems, electrical engineering problems, clustering problems and many others (Dorigo & Stützle 2003, Dorigo et al. 2006, Mohan & Baskaran 2012, Behravan 2012). Various research studies proven the efficiency and robustness of ACO algorithm to be one of the most successful and high-performing meta heuristic techniques (Dorigo et al. 2006, Selvi & Umarani 2010). Goss was the first who experimentally examined the behavior of real ants in 1989 (Goss et al. 1989). In experiments, the ant colony showed a representation of the notion of self-organization and division of labor, which could be simulated as an intelligent system called Ant System (AS). The real ants were able to select the shortest path from amongst multiple available alternatives. Also, the ants acted as intelligent

agents and memorized the shortest paths by laying a chemical substance called "pheromone". The amount of pheromone on a path depends on the quality and quantity of the food source and the pheromone concentration laid on a specific path is proportional to the richness of this food source (Dorigo & Stützle 2004). Therefore, the ants tend to prefer paths with higher pheromone concentration. Dorigo and colleagues introduced the first algorithm based on the foraging behavior of ants in the early 1990s (Colorni et al. 1991, Dorigo 1992). Later in 1997, Dorigo and Gambardella proposed a major improvement to the legacy algorithm by introducing the Ant Colony System (ACS) and used the so called pseudo-random-proportional rule that is used to balance the local and global search abilities of the AS. In 1998, first conference dedicated to the ACO algorithms was held.

Simulating the behavior of real ants, the algorithm recruits a set of computational agents called artificial ants. The artificial ants incrementally construct solutions for the instance under consideration. The solutions' construction process is stochastic and is biased by a predefined decision policy. According to this policy, the selection of a specific path depends on the interpretation of a set of artificial information. This information includes pheromone trail, and heuristic value (Dorigo & Stützle 2004). The pheromone trail symbolizes the notion of memorizing and utilizing past experience. The aim of pheromone adaptation scheme is to increase the pheromone concentration associated with quality solutions and decrease those that are associated with relatively poorer ones. Therefore, the pheromone deposited on a link represents the preferability of this particular link and consequently it is more likely to be favored and selected repeatedly. Collectively, for a number of links, the accumulated pheromone concentration implicitly favors exploitation of previously traversed links. Thus, implements a greedy selection procedure, which encourages the exploitation process in the search

space. The greedy selection procedure is counter-balanced by pheromone evaporation. In fact, the procedure of pheromone evaporation implements a useful form of forgetting the previously explored paths, which encourages the exploration process in the search space.

Another important factor to be considered when constructing the solutions is the random heuristic value given to the link. At each step, the ants make a probabilistic decision. The probabilistic decision is biased by pheromone information and other possibly available heuristic information. That is, the higher the pheromone and the heuristic values associated with a specific link, the higher the probability that this particular link will be selected. The values of pheromone concentration and heuristic values are continuously adjusted by the ants during the run time. In the literature, several variations of ACO algorithm were proposed that adopt different ways for pheromone adaptation process (Dorigo & Gambardella 1997, Dorigo et al. 2006).

The meta heuristic consists of three components as well as the initialization. These components are: solutions construction, pheromone update, and daemon action (optional). These components are executed repeatedly until the termination condition is satisfied, as illustrated in Figure 2.3.

For the sake of brevity and since the daemon action is not applied herein only the other three steps are discussed below:

- **Initialization**: Each agent is responsible of constructing an individual solution. Herein, the solution is a matrix that depicts a complete feasible network topology. The ant starts solution construction from a random node referred to as a root node. Then, at each step, the current partial solution is incrementally extended by adding another node from the set of feasible neighbor nodes. The constraints are also checked at each step to guarantee solution feasibility. When all nodes are connected, the solution is fully constructed. At the end

```
Begin;
    Repeat;
            Ants_Generation_and_Activity();
                    Construct_solutions();
                    pheromone_update();
             daemon_actions(); (optional)
             Memorize the best solution found so far();
    Until MCN (Maximum Cyclic Number)
    Return best solution;
End;
```

Figure (2.3)
Pseudo-code of ACO algorithm

of this phase the resulting number of solutions is equal to the number of employed agents (i.e there is one solution per ant).

- **Ants Activity**: Once the initialization phase is finished and the set of initial solutions is constructed. The next step is to calculate the probability associated with each link and update pheromone values. Each ant constructs a solution using the link selection probability function given in the following equation (Dorigo et al. 2006):

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha_{ant}}[\eta_{ij}]^{\beta_{ant}}}{\sum_{l \in N_i}[\tau_{ij}(t)]^{\alpha_{ant}}[\eta_{ij}]^{\beta_{ant}}} \qquad (2.2)$$

The deposit rules are as expressed in the following equations (Dorigo & Stützle 2004), respectively:

$$\tau_{ij} \leftarrow \tau_{ij} + \triangle\tau^{k} \qquad (2.3)$$

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} \qquad (2.4)$$

Where N_i is the set of neighbors of node i, $p_{ij}^{k}(t)$, is the probability of selecting a connection t between nodes i and j for the k^{th} ant, τ_{ij} is the pheromone on connection t, and

η_{ij} is the heuristic value of connection t. The exponents α_{ant} and β_{ant} represent the influence of pheromone content and heuristic value respectively. $\triangle\tau^k$ is the amount of pheromone to be added, p is a parameter between 0 and 1.

The effect of the parameters α and β can be clarified as follows: α represents the effect of pheromone concentration level on connection t. If α is neglected (if α is taken to be 0), then the effect of accumulated pheromone τ_{ij} is ignored. In this case, only the influence of heuristic value η_{ij} is considered, which means that the selected candidate links are heuristically determined regardless of the pheromone concentration. This scheme encourages the exploration process of new links in the search space. In contrast, If β is neglected then, only pheromone intensification is considered, that is only pheromone information is used to direct the search, without any heuristic bias. Then the previously experienced links with a relatively extra amount of pheromone concentration are most likely to be selected again and again. This scheme encourages the exploitation process and corresponds to the classic stochastic greedy selection procedure that may generally lead to rather poor results (Dorigo & Stützle 2004).

For the work done in this thesis, ACO has been specifically chosen for the sake of comparison with ABC, since ACO is also a population-based swarm intelligence algorithm similar to ABC. Another motivation for choosing ACO is that, a multi-objective ACO has been applied to a variant of the network topology design problem (Khan 2009) and exhibited the best performance when compared with many other iterative optimization algorithms.

2.4 Concluding Remarks

This chapter reviewed the literature relevant to both the topological design problem of Distributed Local Area Networks

(DLANs) and the Artificial Bee Colony algorithm (ABC) along with that related to Ant Colony Optimization algorithm (ACO). First, the chapter discussed the literature related to the DLAN topological design problem. Later, the chapter provided an overview of the existing literature related to ABC algorithm. Finally, as the Ant Colony Optimization (ACO) was chosen for the intended comparison with artificial bee colony, the chapter also provided a brief overview of the literature related to ACO algorithm.

Chapter Three
Problem Formulation

3.1 Assumptions	32
3.2 Design Objectives	33
3.2.1 Network Reliability	33
3.2.2 Network Availability	34
3.2.3 Average Link Utilization	36
3.2.4 Monetary Cost	37
3.2.5 Average Network Delay	38
3.3 Constraints	39
3.4 Goal Programming Approach to the DLAN Topology Design Problem	40
3.5 Concluding Remarks	48

Chapter Three

Problem Formulation

This chapter addresses the problem considered in this thesis. In Chapter 1, an overview of basic concepts and terminology was provided in the background. Then, a formal description of the problem was also provided. This chapter provides a detailed discussion on the design objectives. There are five design objectives to be optimized simultaneously, namely, network reliability, network availability, average link utilization, monetary cost, and average network delay. The assumptions and the technical constraints are also clarified and discussed. Later on, goal programming approach for aggregating the individual design objectives is explained.

3.1 Assumptions

The *following are assumed for the* DLAN *topology design* problem:

1. A tree topology is considered for network design.
2. Each node represents a LAN (i.e. a local site) connected via a router to other LANs and finally to the central backbone.
3. LANs are assumed to be already defined and exist. The topology of each LAN is assumed to be known.
4. Distances between LANs are known. The location of each node can be represented by its cartesian coordinates with respect to the root node.
5. The root node acts as a central backbone and consists of switches and fiber optic links.
6. In the network, each link is bidirectional.

7. The reliability is known for each link.
8. Nodes are fault-free. Only links are subject to failure.
9. Fiber optic cables are used to connect LANs.
10. The maximum utilization of a link should stay below a specified threshold.
11. The capacity of a networking device is limited by the number of ports available on it. The ports are used to connect other hosts to the networking device.

3.2 Design Objectives

This section describes the design objectives considered in this thesis. As mentioned earlier, five design objectives to be optimized simultaneously. Following is a description for each of them.

3.2.1 Network Reliability

Vast number of definitions and measures can be found in the literature when network reliability is concerned. In general, reliability can be defined as the probability of proper delivery of information with respect to the expected loss in case of any occasional occurrences of network failure. In short, it is the probability of reliability fulfillment (Jereb 1998). Theoretically, a reliable network is totally free of technical errors. In practice, however, any component will eventually fail. As time elapses, network physical components suffer from physical degradation caused by the intensive use or by sudden crises. Severe problems can be reported when network reliability is concerned. Therefore, for network engineers and administrators, it is essential to predict the reliability of the network and to analyze how these levels of reliability can be optimized. The aim is to maximize the network reliability. In fact, reliability is a key metric when a network performance is to be judged. It reflects the quality of the network design. However, the exact determination of a network reliability

level is a complex mission. Due to this fact, huge literature was devoted to address this issue (Jereb 1998).

Reliability of a distributed communications network depends on the sustainability of both nodes and individual links. In terms of components, it is relatively easier to predict the reliability and availability levels. Particularly, for components that have been used previously, it is likely that there is abundant information available for the statistical distribution of failure times, and the distribution of repair times.

For a tree topology, network reliability can be calculated as the product of reliabilities of all individual links present in the topology (assuming that nodes are perfect), as shown in the following equation (Khan 2009):

$$R_S = \prod_{i=1}^{L} R_i \tag{3.1}$$

Where, R_S is the reliability of the entire network. R_i is the reliability of i^{th} link in the network. And L is the total number of links in the network. R_i ranges from 0 to 1.

3.2.2 Network Availability

The term "availability" refers to the probability that a device will continue to fulfil its required functionality without failures under defined conditions for a defined period of time. In fact, availability is a key characteristic of any system design. It equally becomes a crucial and complex issue for a network topology design problem. Note that, a typical tree network topology consists of links and nodes, in which each node has an access to every other node via a unique path. Even a small fault in a cable or a network device can threat the availability of the entire network. Commercially, a network service provider must ensure provision of maximum availability, since it may serve as a critical factor in attracting customers. Therefore, the usual accepted value

of availability in the industry is very high, usually expressed as number of nines such as 99.999% (also known as 5 nines), as adopted for networks in military sectors (Vargas & BluePrints 2000). For example, if the availability of a network per day unit was 99.993%, it indicates the expected unavailability to be 6.04 seconds per day. Thus, the accumulated annual unavailability would be approximately 36.8 minutes per year. For a network involving sensitive applications and critical data handling (such as in stock exchanges or space mission program), this may lead to disastrous results (Dearborn et al. 2006).

Network engineers and administrators are facing growing challenges to drive network availability to highest levels. However, to achieve the "five nines", network availability requires configuration of an infrastructure that supports continuous availability. Redundancy has been used to elevate availability. Yet, as modules are added to a network, both complexity and cost increases. Network topology plays a major role in achieving appropriate and cost effective availability. As an optimization problem, the objective is to maximize availability. However, for ease of calculation, the unavailability is considered as an objective to be minimized. The availability A of a topology can be defined by the following equations (Tornatore et al. 2003):

$$A = 1 - U_l \times U_d \times 100\% \tag{3.2}$$

where, U_l is the unavailability of all links in the network, U_d is the unavailability of all networking devices. The unavailability of links U_l is the summation of unavailability of all links that appear in the topology and can be calculated as follows.

$$U_l \simeq \sum_{i=1}^{L} U_{l_i} \tag{3.3}$$

Where L is the total number of links in the network. U_{l_i}, is the

unavailability of i^{th} link. Similarly, the unavailability of devices U_d, is defined as follows.

$$U_d \simeq \sum_{j=1}^{D} U_{d_j} \tag{3.4}$$

Where D is the total number of devices in the network. U_{d_j} is the unavailability of j^{th} device.

For a massively distributed network, the prediction of its reliability and availability levels is a major challenge due to its characteristics of heterogeneity and dynamism. The problem is too complex to be optimized directly with exact methods. Therefore, approximate optimization methods are suitable alternatives to optimize network reliability (Quan & Li 2014). In designing any distributed system, the reliability and availability of the communication paths between all components is a primary consideration (Juan et al. 2010). Reliability and availability in a distributed system is a function of both: (a) the reliability and availability of each individual component, and (b) how these components are connected (Medhi 1999). Thus, this work takes both reliability and availability into consideration when a network topology design is to be optimized.

3.2.3 Average Link Utilization

Link utilization can be defined as the ratio of currently used bandwidth to maximum capacity that the link can handle. While high link utilization indicates that a link is highly utilized, low utilization indicates that the link is under-utilized. A link may even be idle with no traffic on it. The aim is to keep individual link utilization within the permissible level. Utilization is another key decision parameter used to indicate if link capacity is sufficient to ensure that the communication quality meets the user's requirements (Igai & Oki 2011).

As an optimization problem, the aim is to maximize link utilization to maintain a reasonably acceptable performance level.

However, a very high link utilization often compromises latency (Stanojević & Shorten 2007, Jasem et al. 2010*a*,*b*). That is, under normal conditions, exceeding a threshold value increases collisions. This can cause severe congestion levels (Shakkottai et al. 2008), resulting in delay. Several approaches have been proposed in literature to estimate the upper bound of link utilization (Stanojević & Shorten 2007, Jasem et al. 2010*b*, Igai & Oki 2011). One such approach is by Igai and Oki (Igai & Oki 2011), whose work aimed to balance both utility and congestion. The following equation is used to calculate link utilization, $U_{(l)}$.

$$U_{(l)} = \frac{T(l)}{C_{(l)}} * 100\% \tag{3.5}$$

where $T(l)$ is traffic on link l, $C_{(l)}$ is the maximum capacity of the same link.

3.2.4 Monetary Cost

The ability to guarantee a quality level of services is fundamental to attract new customers and satisfy current users. However, it comes at extra monetary costs. The operator can choose to spend more to equip his network with high quality hardware. Though, improving the quality of these elements above the levels of present commercial systems is hard and usually beyond the control of the operator. The goal is then to find the topology with minimum cost. In order to calculate the cost of the topology design, the cost of all cable links should be considered along with the cost per unit of the used cable. For this purpose the following equation is used:

$$Cost = Length \times C_{cable}. \tag{3.6}$$

where Cost is the link cost per unit and *length* represents the total length of cable, and C_{cable} represents the cost per unit of the cable used. The cost of a network C_{total} is then calculated as:

$$C_{total} = \sum_{i=1}^{L} Cost_i \tag{3.7}$$

Where L is the total number of links in the network. It should be noted that, in this case, networking devices costs do not play any role in topology optimization since, first, network devices are assumed to be fault-free, and second, any tree topology will consist of the same networking devices. The only difference between various topologies of same dimension will be in term of cables.

3.2.5 Average Network Delay

Network delay is one of the most important network performance metrics. It can be defined as the amount of time it takes a bit to be transmitted from the source to its intended destination. The aim is to minimize the average network delay. There are several reasons behind network delay (Jasem et al. 2010*a*). The transmission speed of the network is affected by congestion levels and routing changes. Given a specific source and destination, the performance of the network may vary drastically over time, producing various delay rates. The most important factors that contributes to this fluctuation in network performance are, network topology changes and traffic changes (Pucha et al. 2007). In all cases, delay must be avoided. The objective is to find the network topology with lowest delay. To calculate the average network delay for a given topology, both link delay and device delay should be taken into account. If a link is required to connect two local sites i and j, the following equations were used to calculate the average network delay (Khan 2009):

$$D_{total} = D_{link} + D_{device} \tag{3.8}$$

$$D_{total} = \frac{1}{\gamma}\sum_{i=1}^{L}\frac{\lambda_i}{\lambda_{max,i} - \lambda_i} + \frac{1}{\gamma}\sum_{i=1}^{D}\sum \gamma_{ij}{}^{B_{ij}} \qquad (3.9)$$

Where, L is the total number of links in the topology. λ_i is traffic on link i in bits per second (bps), and $\lambda_{max,i}$ is the capacity of link i in (bps). γ is the total traffic in the network. D is the total number of networking devices in the network. B_{ij} is the delay per bit resulted from devices and is equal to $B_{i,j} = b_{i,j}/w$, where $b_{i,j}$ is the delay per packet and w is the average packet size.

3.3 Constraints

The constraints considered in this thesis, are as follows.

1. The maximum number of nodes attached to any networking device must not exceed its capacity stated by the available number of ports for this device.
2. The traffic on any link must not exceed the maximum link bandwidth.
3. Maximum number of hops from source to destination should not exceed the maximum allowed number of hops. Practically, the maximum allowed number of hops is 15. If the hop count exceeds this number, then the destination is considered unreachable (Sportack & Fairweather 1999).
4. Other user specified constrains that affect the hierarchical structure such as, certain nodes must be leaf/terminal nodes and cannot be attached directly to the central backbone.

In Summary, the aim is to maximize the network reliability, network availability, and to minimize monetary cost and average network delay, while keeping the average link utilization within the range of its threshold values.

3.4 Goal Programming Approach to the DLAN Topological Design Problem

As mentioned earlier in Section 2.1, that DLAN topology design problem is an NP-hard optimization problem. Due to the multi-objective nature of this problem, a suitable approach is required that can efficiently handle the multiple objectives. Historically, several approaches have been proposed in the literature that aim to find the best possible solution for optimization problems with multiple objectives (MOO). Well-known approaches include, weighted sum approach (Zadeh 1963), lexicographic ordering (Coello 1998), fuzzy logic (Zadeh 1965), goal attainment (Gembicki 1974), and goal programming (Johnsen 2012).

In this work, goal programming is proposed as the underlying MOO approach for DLAN topology design problem. Of all the above mentioned approaches, goal programming is characterized by its computational efficiency and ease of implementation as well the ability to handle multiple objectives simultaneously. Goal programming (GP) approach was developed by Charnes in 1955 (Charnes et al. 1955). As a multi objective optimization approach (MOO), GP was proposed as a technique to address linear models and has proven to be efficient and effective to address various complex industrial optimization problems. Since then, GP has been extensively employed in an impressive number of applications in various sectors. Because of its flexibility and relatively lower computational complexity GP is considered as one of the most widely used multi objectives optimization approaches (Aouni & Kettani 2001).

From methodological development perspective, several extensions and variants of goal programming have been proposed (Jones & Tamiz 2010). Some of these variants are, integer and

binary goal programming, lexicographic goal programming, weighted goal programming, chebyshev goal programming, fuzzy goal programming, fractional goal programming and many others. In this thesis, the Fuzzy Goal Programming (FGP) variant is considered. This variant utilizes the fuzzy set theory (Zadeh 1965). Its major benefit over other variants is its ability to deal levels of imprecision in the goal programming model (Jones & Tamiz 2010).

The approach starts by initializing a set of targeted goals. Each goal has a maximum achievement level that is determined *a priori*. The level of fuzziness around each goal is measured using a membership function. There are various possibilities for measuring the fuzziness around the targeted goals, each of which leads to a different fuzzy membership function. The membership function is used to express the achievement of a particular objective around the goal. Later, the deviational variable of each objective is calculated. The deviational variable measures the distance of the achieved level form its desired goal. The aim is then to minimize the resulted overall deviation.

Narasimhan was the first who incorporated the fuzzy set theory in the goal programming model in 1980 (Narasimhan 1980). In 1981, Hannan simplified Narasimhan's approach to a simpler liner method (Hannan 1981). In the literature, many fuzzy goal programming variants were reported that differs in calculating the deviational variables (Chanas & Kuchta 2002). One variant is the weighted fuzzy goal programming variant which assigns weights to deviational variables to differently prioritize the underlying objective of each deviational variable (Yaghoobi et al. 2008). Variants of this approach have been extensively employed in the literature to solve problems in different application domains (Güneş & Umarosman 2005, Mekidiche & Belmokaddem 2012, Mekidiche et al. 2013).

In this thesis a simpler version of this variant was applied, all

deviations are aggregated into a single final evaluation function by linearly adding all produced individual deviations. All deviational variables were given the same priority. The reason was to give equal importance to all design objectives and formulate a general version of the problem that considers all point of views of users, administrators and managers. The application of this approach to the problem under consideration is explained in the rest of this chapter. For the five design objectives under consideration, the goal is to find a high quality topology that maintains high reliability, high availability, efficient link utilization, low average network delay and low monetary cost. Figure 3.1 illustrates the characteristics of a quality topology:

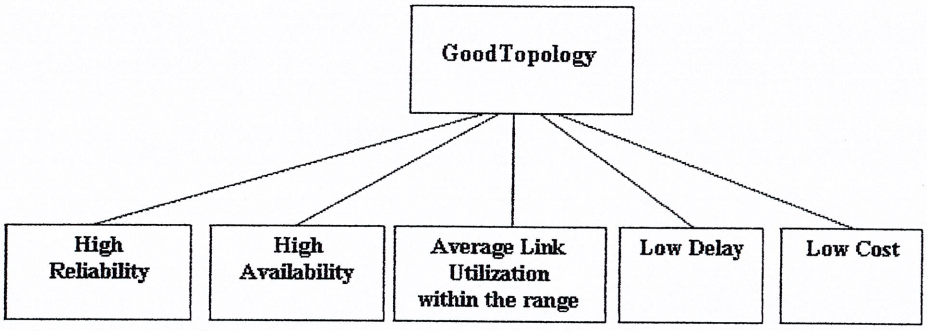


Figure (3.1)
Characteristics of a quality topology

The quality of the overall topology is expressed and evaluated using fuzzy goal programming approach. The evaluation process consists of four steps as explained below.

1. **Define the Goals**

   Ideal values for each objective should be stated in advance. Following this, the quality of the achieved level for each objective function is evaluated by calculating the deviation from its corresponding ideal value (i.e the target). The less the deviation, the better the solution. Therefore, the best deviation that can be achieved is 0 which signifies achieving

the target. While positive deviation indicates over-achievement of a pre-set goal, negative deviation indicates under-achievement of this goal. Ideally, both deviations are unfavorable. The deviation is penalized according to the membership function which should be calculated at the second step. In this work, the ideal values of each design objective (i.e goals) are determined by considering data found in relevant literature (Khan 2009). The details of the adopted ideal values are to be discussed in Section 5.1.

2. **Calculate the Membership Function**

The achievement level of each objective is evaluated through a membership function. Thus, a membership function should be defined for all objectives individually. The membership function takes a value that ranges from 0 to 1, to express the achieved satisfaction level. While the value of 0 indicates total dissatisfaction, the value of 1 indicates complete satisfaction. There are various possibilities for measuring the achievement of a targeted goal, each of which leads to a different membership function. Several membership functions can be defined that differs in algebraic structure (Narasimhan 1980, Hannan 1981). The structure of a membership function determines the penalization pattern of undesired deviation.

To define the membership function for reliability, the minimum and maximum values are determined first. The left-sided function is used as shown in Figure 3.2(b).

In this figure, the two values denoted as 'A' and 'B' refer to the minimum reliability *'MinR'* and the maximum reliability *'MaxR'*, respectively. The membership value for reliability of solution x, $\mu_r(x)$, can be formulated as follows.

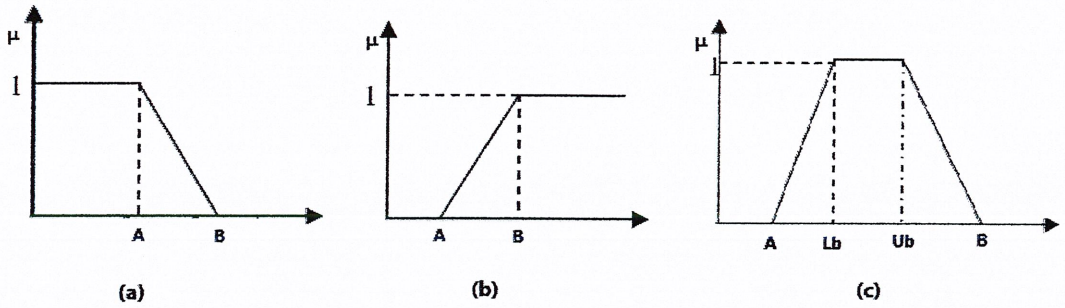


Figure (3.2)

The structure of membership functions used to express the achievement level of design objectives

$$\mu_r(x) = \begin{cases} 1 & \text{if } Rel(x) \geq MaxR \\ \frac{Rel(x)-MinR}{MaxR-MinR} & \text{if } MinR \leq Rel(x) < MaxR \\ 0 & \text{if } Rel(x) < MinR \end{cases} \qquad (3.10)$$

where the term *'Rel(x)'* represents the reliability of the current solution. The above equation can be represented as fuzzy rule as follows. If the reliability of a current solution is larger than maximum reliability, then the membership function of reliability will be 1. Else, if reliability is less than the minimum value, then the membership function will be 0. Otherwise, the membership function will be given a value between 0 and 1 as defined by the expression $\frac{Rel(x)-MinR}{MaxR-MinR}$.

In a similar way, the membership function for availability, $\mu_a(x)$, can be defined. The two extreme values of availability *'MaxA'* and *'MinA'* define the maximum and minimum values, respectively. In Figure 3.2(b), *'MinA'* corresponds to 'A' and *'MaxA'* corresponds to 'B'. The membership function for availability is determined as follows.

$$\mu_a(x) = \begin{cases} 1 & \text{if } Av(x) \geq MaxA \\ \frac{Av(x)-MinA}{MaxA-MinA} & \text{if } MinA \leq Av(x) < MaxA \\ 0 & \text{if } Av(x) < MinA \end{cases} \qquad (3.11)$$

where the term *'Av(x)'* represents the availability of the current solution. The above equation can be represented as fuzzy rule as follows. If the availability of a current solution is larger than maximum availability, then the membership function will be 1. Else, if availability is less than the minimum value, then the membership function will be 0. Otherwise, the membership function will be given a value between 0 and 1 as defined by the expression $\frac{Av(x)-MinA}{MaxA-MinA}$.

The membership function for link utilization denoted as $\mu_l(x)$ is a special case, since the ideal value of link utilization is a range of values rather than a single value (i.e 50% - 70%). In such case, the trapezoidal shaped function is used as illustrated in Figure 3.2(c), where the minimum *'MinL'* and maximum *'MaxL'* values correspond to 'A' and 'B', respectively. *'Lb'* and *'Ub'* correspond to lower bound and upper bound of ideal values range, respectively. The membership function for individual link utilization is formulated as follows.

$$\mu_l(x) = \begin{cases} 1 & \text{if } Lb \leq Lu(x) \leq Ub \\ \frac{Lu(x)-MinL}{Lb-MinL} & \text{if } MinL \leq Lu(x) < Lb \\ \frac{MaxL-Lu(x)}{MaxL-Ub} & \text{if } Ub < Lu(x) \leq MaxL \\ 0 & \text{if } MaxL < Lu(x) < MinL \end{cases} \tag{3.12}$$

where *Lu(x)* represents the utilization of link *'x'* in the current solution. The above equation can be represented as fuzzy rule as follows. If the link utilization is falls in the ideal range of link utilization, then the membership function will be 1. Else, if link utilization is less than the lower bound in the range it will be given a value between 0 and 1 as in the expression $\frac{Lu(x)-MinL}{Lb-MinL}$. The membership function of link utilization will be 0 if the it was lower or greater

than the peak values.

The average link utilization for a solution consists of n links, can be computed as follows.

$$\mu_{avl}^{o} = \frac{\sum_{i=1}^{n} \mu_l(i)}{n} \tag{3.13}$$

where μ_{avl}^{o} is the average link utilization for all links within the current solution.

The membership function for delay, $\mu_d(x)$, can be defined in a way similar to that of reliability and availability. The values *'MaxD'* and *'MinD'* define the maximum and minimum delay, respectively. In Figure 3.2(a), it 'A' corresponds to *'MinD'* and *'B'* corresponds to *'MaxD'*. Then, the membership value for delay is determined as follows.

$$\mu_d(x) = \begin{cases} 1 & \text{if } Delay(x) \leq MinD \\ \frac{MaxD - Delay(x)}{MaxD - MinD} & \text{if } MinD < Delay(x) \leq MaxD \\ 0 & \text{if } Delay(x) > MaxD \end{cases} \tag{3.14}$$

where the term $Delay(x)$ represents the average delay of the current solution. The above equation can be represented as fuzzy rule as follows. If the delay of a current solution is larger than maximum delay, then the membership function will be 0. Else, if delay is less than the minimum value, then the membership function will be 1. Otherwise, the membership function will be given a value between 0 and 1 as defined by the expression $\frac{MaxD - Delay(x)}{MaxD - MinD}$.

Finally, the membership function for cost is defined by finding the minimum and maximum values. Again, the function shown in figure 3.2(a) is used. In this figure, the two values shown as 'A' and 'B' refer to the minimum cost

'MinC' and the maximum cost *'MaxC'*, respectively. The membership value for cost of solution x, $\mu_c(x)$, is then computed as follows.

$$\mu_c(x) = \begin{cases} 1 & \text{if } Cost(x) \leq MinC \\ \frac{MaxC - Cost(x)}{MaxC - MinC} & \text{if } MinC < Cost(x) \leq MaxC \\ 0 & \text{if } Cost(x) > MaxC \end{cases} \quad (3.15)$$

where the term *'Cost(x)'* represents the cost of the current solution. The above equation can be represented as fuzzy rule as follows. If the cost of a current solution is larger than maximum cost, then the membership function will be 0. Else, if cost is less than the minimum value, then the membership function will be 1. Otherwise, the membership function will be given a value between 0 and 1 as defined by the expression $\frac{MaxC - Cost(x)}{MaxC - MinC}$.

3. **Calculate the Deviational Variables**

After calculating the membership function of all the design objectives, $\mu(x)$, individually, the deviational variable $\mu^o(x)$ of each design objective should be calculated. The deviational variable of an objective represents unwanted deviation from the target and is computed as follows.

$$\mu^o(x) = 1 - \mu(x) \quad (3.16)$$

The lower the deviation $\mu^o(x)$, the better the solution. The problem then is to minimize the deviation variable of all design objectives.

4. **Calculate the Overall Achievement Function**

The fuzzy goal programming approach places all deviational variables $\mu^o(x)$, into a single, normalized deviation function. Thus, the deviational variables of design objectives are

aggregated in a single overall deviation function, which measures the degree of minimization of overall deviational variables (Romero 2004). In short, the overall deviation function measures the "lack" or deviation of achieved goals. The mathematical representation of overall deviation function is given in the following equation.

$$\mu(x)_{Total} = \sigma_1 \mu_r^o(x) + \sigma_2 \mu_a^o(x) + \sigma_3 \mu_l^o(x) + \sigma_4 \mu_d^o(x) + \sigma_5 \mu_c^o(x) \qquad (3.17)$$

In Equation (3.17) above, the $\mu(x)_{Total}$ represents the overall deviation for all objectives collectively. Also in this equation, $\mu_r^o(x)$, $\mu_a^o(x)$, $\mu_l^o(x)$, $\mu_d^o(x)$, $\mu_c^o(x)$, represent the deviational variables of reliability, availability, average link utilization, delay and cost, of solution x, respectively. The solution which results in the minimum value for the Equation (3.17) is recorded as the best solution found.

In the above equation, the coefficients σ_1, σ_2, σ_3, σ_4, and σ_5, are weights used to indicate the priority of a specific design objective. In this work, a general problem is considered that cares equally for all design objectives. Therefore, all coefficients were given a value equal to 1.

3.5 Concluding Remarks

In this chapter, the problem of topological design for distributed local area network was defined and discussed. The necessary background was also provided. The problem was formulated as a multi-objective optimization problem. All design objectives, constraints, and assumptions were presented and discussed. Furthermore, this chapter introduced the fuzzy goal programming approach to calculate the normalized deviational variables and combine them into a single overall deviation function.

Chapter Four
Adapted Algorithms

4.1 Goal Programming Based ABC Algorithm	50
4.1.1 Initialization (Generation of Bees)	51
4.1.2 Parameter Setting	51
4.1.3 Solution Representation	52
4.1.4 Bees Activity	52
4.1.5 The Framework of ABC Algorithm	56
4.2 Goal Programming Based ACO Algorithm	57
4.2.1 Initialization	58
4.2.2 Ants Activity	58
4.3 Concluding Remarks	59

Chapter Four

Adapted Algorithms

This chapter presents the algorithms applied for the DLAN topology design problem, namely, the Artificial Bee Colony (ABC) algorithm and the Ant Colony Optimization (ACO) algorithm. The original algorithms were adapted to address the multi-objective nature of DLAN topology design using goal programming approach. A comprehensive introduction to the basic characteristics of both algorithms was provided in Chapter 2. This chapter discusses the algorithmic implementation of goal programming based ABC and ACO algorithms for DLAN topology design problem. This is followed by an in-depth explanation of the proposed framework along with the necessary modifications applied to the original algorithms to solve the problem under consideration.

4.1 Goal Programming Based ABC Algorithm

Since its emergence at 2005, the interest in ABC has increased considerably. Researchers all over the world studied the algorithm and its application to real world problems (Dorigo & Stützle 2003, Bolaji et al. 2013). Originally, ABC optimization algorithm was proposed for optimization of continuous problems (Karaboga 2005). Later on, the algorithm was adapted for discrete problems (Krause et al. 2013). Solving discrete optimization problems is, however, a big challenge for ABC (Xu et al. 2013). Certain yet non-simple modifications and/or adaptations are necessary in the original ABC algorithm in order to be applicable to discrete problems (Li et al. 2011, Pan et al. 2011, Wang et al. 2012).

Another major challenge was to extend the algorithm for solving constrained optimization problems (Karaboga & Basturk 2007*a*, Karaboga & Akay 2011). In this work, the original ABC was adapted to meet with requirements of DLAN topology design problem. The discrete multi-objective nature of the problem and the presence of technical constraints should be taken into account when ABC is to be tailored to address the DLAN topology design.

To make the algorithm applicable to the problem under consideration, a discrete version of ABC is proposed. Also, goal programming approach is employed to address the multi-objectivity of DLAN topology design. To achieve reasonable performance level, certain modifications were necessary to be made. For example, in the original ABC, only one scout is permitted in a generation (Karaboga 2005). This may, however, burden the exploration capabilities of the algorithm and lead to a stagnation problem of having all bees locked and hence, producing the same solution again and again. Furthermore, in the original ABC, a constant value for the limit was used. Whereas, several experimental results showed that the results obtained by using more than one scout bee and different limit values, are better than the results of original ABC (Karaboga & Basturk 2008). The stages of proposed ABC are explained in this section.

4.1.1 Initialization (Generation of Bees)

In this phase, the initial population of food sources is randomly generated by the scout bees after setting the control parameters.

4.1.2 Parameter Setting

As any other swarm-based algorithm, the control parameters play an important role in determining the algorithm's performance. Consequently, fine tuning for the control parameters of ABC is necessary to achieve reasonable performance. Compared to other

heuristics, ABC has relatively less control parameters, which is one of the main advantages of ABC algorithm. Initially, only three control parameters should be set a priori. These are the colony size, which defines the number of employed and unemployed bees that corresponds to the number of food sources, the limit, which defines the number of trials after which the food source cannot be improved further, and a termination criterion.

4.1.3 Solution Representation

In the proposed scheme, each solution corresponds to a food source. Hence, the terms “food source" and “solution" are often used interchangeably. The solution is represented by a matrix that depicts a complete feasible topology. New solutions are generated by scout bees. Each scout starts the solution construction process by selecting a random node or a LAN as a root. The solution is successively extended by adding one node at time. At each step, the set of technical constraints are checked to ensure the feasibility of the newly produced solution. When all nodes are connected, the solution is completely constructed. At the end of this phase, the current population of initial solutions is generated. The number of solutions constructed is equal to the number of employed bees and is the same number of onlooker bees.

4.1.4 Bees Activity

In this section, the bees activity is described and discussed as in the following phases.

Employed Bees Phase

The employed bees search locally for new neighboring solutions. In this work, a local search operator was specifically designed for the problem under consideration. The employed bees utilize the local search operator to produce new neighboring food sources around the original ones. The proposed operator borrows the

mutation concept from genetic algorithms (GA). A mutant solution is generated from original one by selecting a random node and replacing the connection link with another randomly selected link. The feasibility of the newly produced solution is checked. A non-feasible neighbor solution is neglected and the original one is restored. If the produced neighbor solution is feasible, then it is evaluated and compared with the original one. The evaluation is done by calculating the solution fitness using Equation 3.17. The fittest solution is memorized in the current population in a greedy selection procedure. Therefore, the employed bee is responsible of the exploitation process using the proposed local search operator as illustrated in Figure 4.1.

Waggle Dance

A fitness function is defined to calculate the quality of each food source. Herein, the fitness function is as expressed in Equation 3.17. A better solution often depicts better topology design. This information is shared with other onlooker bees in the hive by means of waggle dance process. To symbolize this process, first, the probability for all solutions in the current generation is calculated using Equation 2.1. As the value of Equation 3.17 of food source decreases, the probability with which the food source is preferred by onlookers increases. Thus, solutions with lower overall deviation have higher chances to be selected and utilized. The process is similar to roulette wheel selection procedure.

Onlooker Bees Phase

Through the waggle dance, onlooker bees are directed to exploit food sources in a probability based selection procedure. The number of onlooker bees allocated to a particular food source depends on the fitness of this food source. Herein, more than one onlooker can be recruited to exploit a specific food source with probability proportional to its fitness. After selecting the food

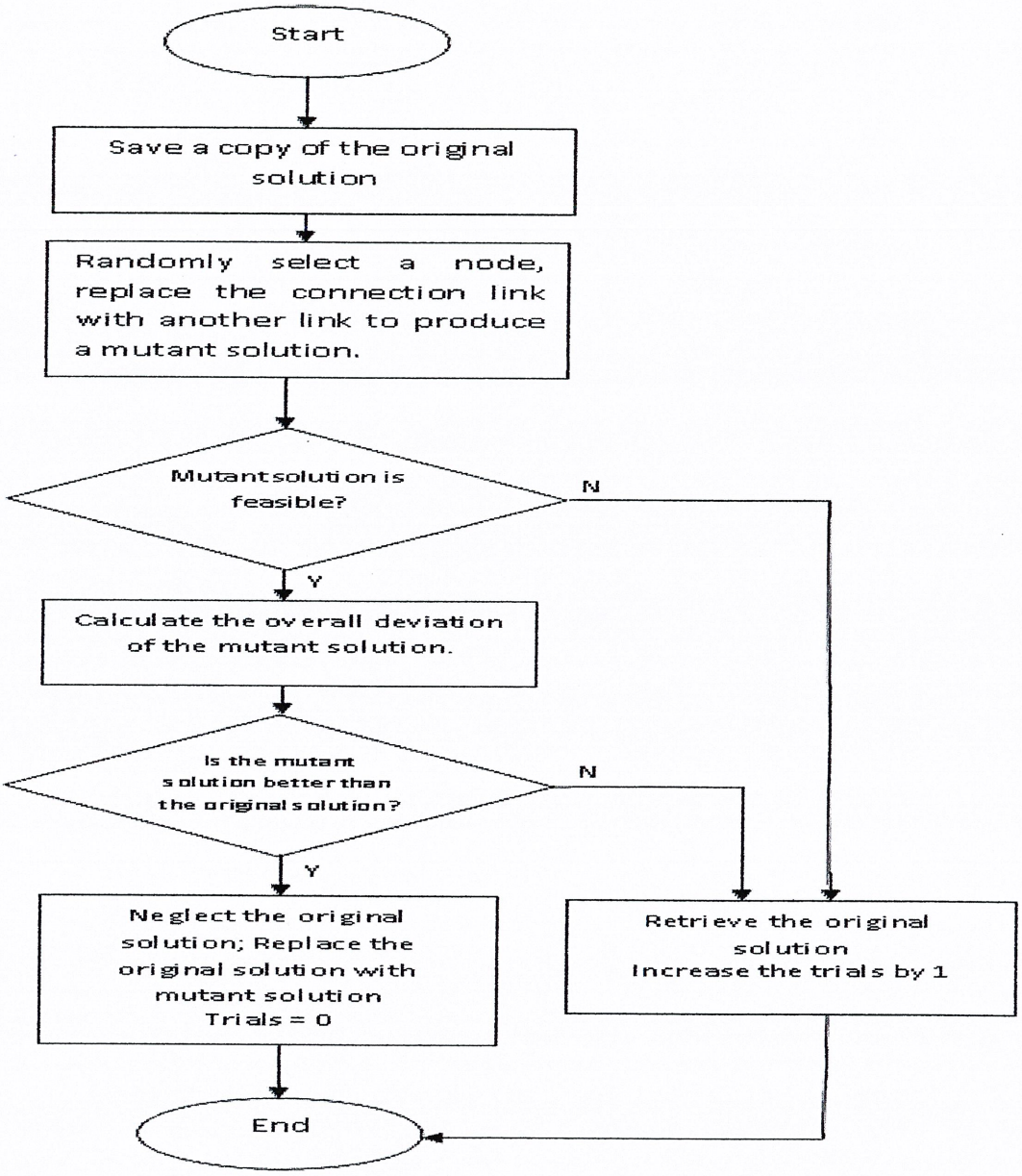


Figure (4.1)
The proposed Local search operator

source, each onlooker bee performs the same local search scheme of the employed bees and searches for a new neighboring food source. Again, the fittest solution between the original solution and the new neighboring solution is memorized in the population.

Scout Bees Phase

The scout bees search globally for new solutions. In the population, each food source is identified by a unique index *i*. A

trial number that indicates the number of attempts to improve a food source is also recorded (denoted as t_i). At each time when an employed bee or an onlooker bee tries to improve the food source, the trials number is incremented. The increment is done whether the employed bee (forager or onlooker) succeeds to generate a better solution or not. The trials number serves as a counter. At each step, the trials number is compared to the value of "Limit". When the trials number associated with a food source exceeds the limit, this food source is abandoned and replaced by another randomly generated food source. The employed bee associated with an abandoned food source is transformed into a scout. The scout bees are responsible of generating random new solutions. The purpose of scout bees is to diversify solutions and avoid local optima trapping. Thus, the number of trials for discarding the food source is equal to the limit.

In ABC, there are two mechanisms to diversify solutions in the population. First, a mutant solution is created by modifying the randomly selected part of the original solution. This mutation rate is relatively small and is used to perform fine tuning. This mechanism provides the ABC algorithm with a powerful local search ability.

Secondly, rather than modifying a part of the solution, an entirely new solution is generated from scratch and inserted in the population by a scout. This mechanism strengthens the ABC algorithm's global search ability and prevents premature convergence. Most importantly, this feature lessens the dependency of the algorithms' performance on the population size (Karaboga & Akay 2009*a*). Therefore, there is a good balance between the local search process conducted by employed bees and the global search process carried out by scouts. This feature is considered as one of the prominent advantages of ABC over other algorithms. The flowchart in Figure 4.2 shows the stages of the ABC algorithm applied to the problem under consideration.

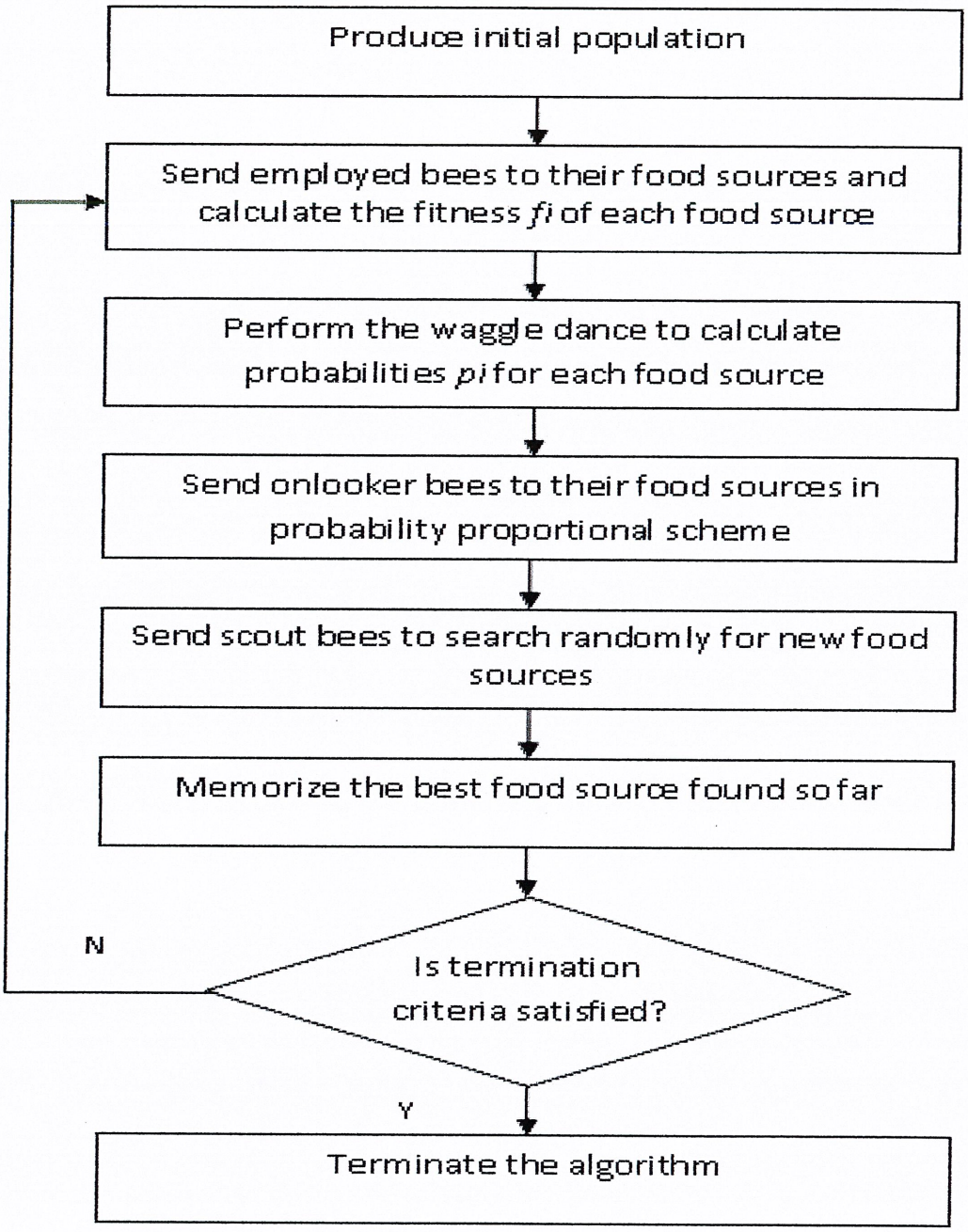


Figure (4.2)
Flowchart of the ABC algorithm

4.2 The Framework of ABC Algorithm

The detailed steps of the proposed ABC algorithm as applied to the problem of DLAN topology design problem with multiple objectives, are shown in Figure 4.3 as follows:

Begin;
Step 1: Initialization phase;
Step 1.1: Set the system parameters;
Step 1.2: Produce initial population;
Step 2: Recursion;
Step 2.1: If the stop criteria is met:
Step 2.2: While stop criteria is not satisfied:
perform step 3 to step 6;
Step 3: Employed bees phase;
Step 3.1: Send bees onto their food sources;
Step 3.2: For each employed bee, produce a new neighboring solution.
Step 3.3: If the neighbor solution is feasible:
goto Step 3.4.
Else, restore the original solution, go to Step 3.5.
Step 3.4: Calculate deviation of neighboring solution;
compare to the original solution.
Record the best solution in the current population.
Step 3.5: neglect any food source if its trials exceeds the limit.
The corresponding employed bee should become a scout and perform step 6.
Step 4: The Waggle dance phase.
Step 4.1: Calculate probabilities for final solutions produced by employed bees.
Step 5: Onlooker bee phase.
Step 5.1: For each onlooker bee, select a food source based on its probability.
Step 5.2: Go to Step 3.1
Step 6: Scout bee phase.
Step 6.1: Randomly generate new solutions.
Step 6.2: *Go to Step 2.*
End;

Figure(4.3)
Steps of the proposed ABC algorithm

4.3 Goal Programming Based ACO Algorithm

This section presents the proposed multi-objectives ant colony optimization algorithm (ACO) for the DLAN topology design problem. The section first provides preliminary information needed to understand the specific nature of design application. Then, the

adopted fuzzy goal programming approach (FGP) as applied to ACO is explained. The phases of the proposed ACO algorithm are explained as follows.

4.3.1 Initialization

In this phase, the control parameters are set and the initial population is randomly generated. Each agent is responsible of constructing an individual solution. Herein, the solution is a matrix that depicts a complete feasible network topology. The ant starts solution construction from a random node referred to as a root node. Then, at each step, the current partial solution is incrementally extended by adding another node from the set of feasible neighbor nodes. The constraints are also checked at each step to guarantee solution feasibility. When all nodes are connected, the solution is fully constructed. At the end of this phase, the resulting number of solutions is equal to the number of agents (i.e there is one solution per ant).

4.3.2 Ants Activity

As mentioned earlier in Section 2.3, ACO utilizes a set of information to guide the search. This information includes, pheromone concentration and heuristic information. Several variants of ACO are found that adopt different pheromone adaptation schemes (Dorigo & Stützle 2004). In this thesis, the elitist ant per iteration approach is assumed (Dorigo et al. 1996). That is, for each iteration, the ant that constructs the best solution is taken as the elitist ant (the ant that produce the lowest value for equation 3.17). According to the elitist ant approach, only the elitist ant is allowed to update the pheromone values. Each candidate link that appears in the elitist ant solution is rewarded by increasing the current pheromone concentration of that link. On the other hand, pheromone evaporation decreases the pheromone concentration on all links (if these links appeared in the elitist ant solution or not).

The updated values of pheromone are then passed to following ants of the next generation.

After initial solutions construction, the pheromone levels are updated for all links. Later, the probability of each link is calculated using Equation 2.2. This process is repeatedly applied until a predefined termination criterion is satisfied. Pheromone deposit favors exploitation of previously traversed links. The ants are guided to focus the search more likely near the best solutions found so far. However, evaporation process is useful to avoid local minima trapping by decreasing the pheromone levels on all links allowing the ant to explore new regions in the search space.

4.4 Concluding Remarks

In this chapter, the proposed algorithms for DLAN topology design using fuzzy goal programming approach were presented. The necessary adaptations made for both algorithms to suit the design application were also covered in sufficient details. The proposed framework required for suitable design implementation was clarified. For, ABC, a local search operator was exclusively designed to suit the problem under concern. For ACO, the elitist ant approach was adopted as found in relevant literature. Both algorithms were tailored to meet the design specification to deal with the five design objectives considered in this thesis.

Chapter Five
Results and Discussion

5.1 Characteristics of Test Cases	61
5.2 Parameter Setting	62
5.3 Results and Discussion Of ABC algorithm	63
5.3.1 Effect of Limit	65
5.3.2 Effect of Colony Size	68
5.4 Results and Discussion for ACO algorithm	74
5.4.1 Effect of Pheromone Deposit and Evaporation	74
5.4.2 Effect of Colony Size	77
5.4.3 Effect of Heuristic Values	81
5.5 Concluding Remarks	79

Chapter Five

Results and Discussion

In this chapter the empirical results are presented and discussed to evaluate the performance of the proposed algorithms. The performance is evaluated based on the quality of the produced solutions. First, the experimental study related to ABC algorithm is presented. The effect of different control parameters of ABC algorithm is investigated, which include the *colony size* and the *limit value*. Later, the experiments and empirical results related to ACO are presented and discussed. The effect of ACO control parameters is also investigated. These parameters include, *pheromone deposit and evaporation rate*, *colony size*, and *heuristic values*. The main purpose of the experiments was to test the algorithms' performance. Another motive was to evaluate the overall performance by revealing any complimentary capabilities of various parameters configurations.

5.1 Characteristics of Test Cases

The test cases adopted in this thesis followed the same scheme found in the literature (Khan 2009). Five test cases were used for the experimental study. Each of which represents a DLAN with local networks or nodes. In the test cases, the number of nodes ranged from 15 (referred to as n15) to 50 (referred to as n50). Other parameters of test cases such as traffic flow, ports capacity, and cost of cables were also taken from relevant literature. In (Khan 2009), the values of these parameters where experimentally collected from real sites. In this work, this data is used to determine ideal values of objectives. The characteristics of test cases are summarized in Tables 5.1 and 5.2. In 5.1, MinC is in minimum cost in US$, MinD is minimum delay in milliseconds,

and traffic is in mbps. The test cases described are used for experimental study performed in this chapter. Also, to enable the intended mutual comparison between ABC algorithm and ACO algorithm, it should be mentioned that, both algorithms are given the same seed solutions to start with.

Table(5.1)

Characteristics of test cases used in experiments

Test Case	# of Local Sites	MinC	MinD	MaxR	Traffic
n15	15	4640	2.14296	0.868746	24.63
n25	25	5120	2.15059	0.785678	74.12
n33	33	8158	2.15444	0.72498	117.81
n40	40	9646	2.08757	0.675729	144.76
n50	50	11616	2.08965	0.611117	164.12

Table(5.2)*

Network characteristics considered for experimental study

Parameter	Characteristic
Cost of fiber optic cable	$ 5 per meter
Delay per bit due to networking device	250μ sec.
Maximum link utilization	70
Maximum link Bandwidth	100 mbps
Average packet size	500 bytes
Type of networking device	Router, switch, or hub
Number of ports on a networking device	4, 8, or 12
Maximum network availability	99.999%

*(Khan 2009)

5.2 Parameter Setting

It is proven that for any swarm based algorithm, the setting parameters are essential when the performance is to be evaluated (Stützle et al. 2012). The behavior of these algorithms depends heavily on the values given to control parameters. Moreover, it is proven that a parameter set that performs perfectly in some cases may fail in other cases (Akay & Karaboga 2009). Therefore, the conducted experiments also focused on studying the effect of

various setting parameters schemes. The parameters considered in this thesis are the most prevalent ones for both ABC and ACO algorithms. The best solution is the one that produces the lowest value for Equation 3.17. The lowest possible value for this equation is 0 and it represents that all objectives achieved their aspiration level. While, the maximum value for this equation is 5 and represents that all objectives failed to achieve an accepted level. To study the effect of a specific control parameter set up, default values were assumed for all other parameters. Only the parameter under concern was varied in each experiment. Other parameters were kept fixed at their default values.

5.3 Results and Discussion Of ABC Algorithm

This section presents the conducted experiments and the empirical results related to ABC algorithm. In the experiments, the goal programming based ABC was applied to the five test cases described previously. The performance of the ABC algorithm was examined and evaluated with respect to a number of control parameters, namely, the colony size and the limit value. In all experiments, the number of employed bees was taken to be equal to the number of processed solutions. For best selection of limit values that suit the underlying problem, expressions were defined to determine the value of the limit (as shown in Table 5.4). For each parameter setup, thirty independent runs were made. Each run comprised of 100 iterations (as per the standards in the domain). The average of lowest deviations found in each run was reported, along with the standard deviation. The results were statistically validated using Wilcoxon ranked-sum test (see Appendix A). The default value for the colony size was assumed to be 20 agent. The parameter values used in the experiments are given in Table 5.3.

In the classical ABC algorithm, only one scout is permitted

Table(5.3)
Parameter settings for ABC used in experiments

Control Parameter	Parameter set
Colony size	10, 15, 20, 25, 30
Employed foragers	50% of the colony
Employed onlookers	50% of the colony
Scout bees	Adaptive (varied dynamically)
Limit	Proportional to colony size and problem dimension
Mutation rate	1%

per a generation. However, this may burden the global search capabilities of the algorithm and could lead to premature convergence. To avoid this, a modified scouts production scheme was proposed. Instead of having a single scout bee per iteration as in the original ABC, the number of scouts was set through an expression. The aim is to increase the global search capabilities of the algorithm and avoid local optimum trapping by maintaining better diversity in the population. The expressions used to determine the limit values are shown in Table 5.4. It should be noted that the term problem dimension in this case refers to number of nodes to be connected or number of LANs in the DLAN.

Table(5.4)
Limit values used in the experiments

Limit	Adopted value
Lim1	0.01 × Food sources × Problem dimension
Lim2	0.05 × Food sources × Problem dimension
Lim3	2 × Problem dimension
Lim4	0.5 × Food sources × Problem Dimension
Lim5	No limit or (Limit = ∞)

Usually, the values of the limit are heuristically determined (Karaboga 2005, Karaboga & Basturk 2008, Akay & Karaboga 2009, Kiran & Gündüz 2014). Earlier, in (Karaboga 2005), a fixed value for the limit was used (i.e. 100). Other studies revealed that better results were obtained with fine tuning of limit values (Akay & Karaboga 2009, Kiran & Gündüz 2014). In this work, the limit values were set to be proportional to colony size and problem

dimension, similar scheme was followed in previous studies found in relevant literature (Karaboga & Basturk 2008, Kiran & Gündüz 2014). In Table 5.4, five limit sets were given, denoted as lim1 to lim5. The value of the limit increases gradually with each set. For example, for case instance of *n15*, and number of agents = 10 bees, the corresponding limit values will be, lim1 = 1, lim2 = 7, lim3 = 30, lim4 = 75, and lim5 = ∞ (the fractions are neglected).

5.3.1 Effect of Limit

The limit is an important control parameter of ABC (Karaboga 2005). The number of trials after which a food source is abandoned is equal to the limit. The employed bee of an abandoned food source is transformed to a scout bee. Thus, the limit controls the scouts' production frequency in the population. In fact, the number of permitted scouts in the population is inversely proportional to the value of the limit. Lower limit values are interpreted as less permitted trials to exploit the food source, and hence producing more scouts within the same time interval. This scheme encourages the exploration abilities of ABC. In contrast, higher limit values allow more trials and cause scouts not to occur so often. This, encourages the exploitation abilities of ABC. However, a balanced exploration/exploitation process is required to enhance the algorithm's performance. Table 5.5 is showing the experimental results for all test cases with all limit sets. In this table, the first column lists the test cases. The rest of the columns show the results obtained for the five limit sets (denoted as lim1 to lim5). The colony size was kept fixed at its default value (employed bees = onlooker bees = 20). DEV is the average overall deviation with standard deviation. Time is average run time in seconds.

From Table 5.5, it is evident that the best results were obtained in all cases with moderate value of limit (i.e. Lim3). Extremely

Table(5.5)
Results for various limit values for all test cases

Case	Lim1		Lim2		Lim3		Lim4		lim5	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time	DEV	Time
n15	2.106 ±0.012	6.8	2.091 ±0.014	5.4	2.089 ±0.015	4.8	2.106 ±0.022	3.0	2.127 ±0.020	2.5
n25	1.886 ±0.013	26.2	1.869 ±0.015	21.6	1.865 ±0.015	19.3	1.888 ±0.028	10.2	1.890 ±0.023	10.4
n33	1.877 ±0.013	235.1	1.863 ±0.017	59.9	1.867 ±0.011	51.9	1.878 ±0.035	42.9	1.887 ± 0.026	36.1
n40	1.896 ±0.014	155.6	1.887 ±0.018	124.7	1.878 ±0.017	108.5	1.886 ±0.024	81.1	1.890 ±0.029	78.6
n50	1.969 ±0.012	364.2	1.948 ±0.019	251.1	1.937 ±0.020	193.2	1.941 ±0.030	159.8	1.944 ± 0.033	182.4

lower and higher values for the limit have worsened the algorithm's performance with lim1, lim2, lim4 and lim5 sets. For example, for the case *n15*, the lowest achieved deviation was 2.089 with limit = lim3. For the same instance of *n15* the highest deviation of 2.127 resulted with highest limit value limit = lim5. The same trend was observed for cases of *n25* and *n33*. For cases *n40* and *n50* the trend was, however, different. In case *n40*, the lowest deviation of 1.878 was achieved with limit = lim3, while the highest deviation of 1.896 was reported with limit value of limit = lim1.

To summarize, in all cases, the best results were produced with limit = lim3. This could be attributed to the following justification: For a tree topology of a DLAN with number of nodes N, the number of links required to connect all nodes is N-1. A neighbor solution is created close to the original solution by replacing one link at a time. When a link to be changed is selected randomly, there will be only a finite number of possible changes. The presence of constraints that are checked every time before changing any link, has further reduced the number of possible changes. This can lead the algorithm to enter an infinite loop of applying the same changes repeatedly. Therefore, it can be confidently claimed that a moderate value for the limit seemed to be more appropriate in context of the problem. The proposed value around 2*N was therefore heuristically determined and experimentally tested. The results in Table 5.5 support the above

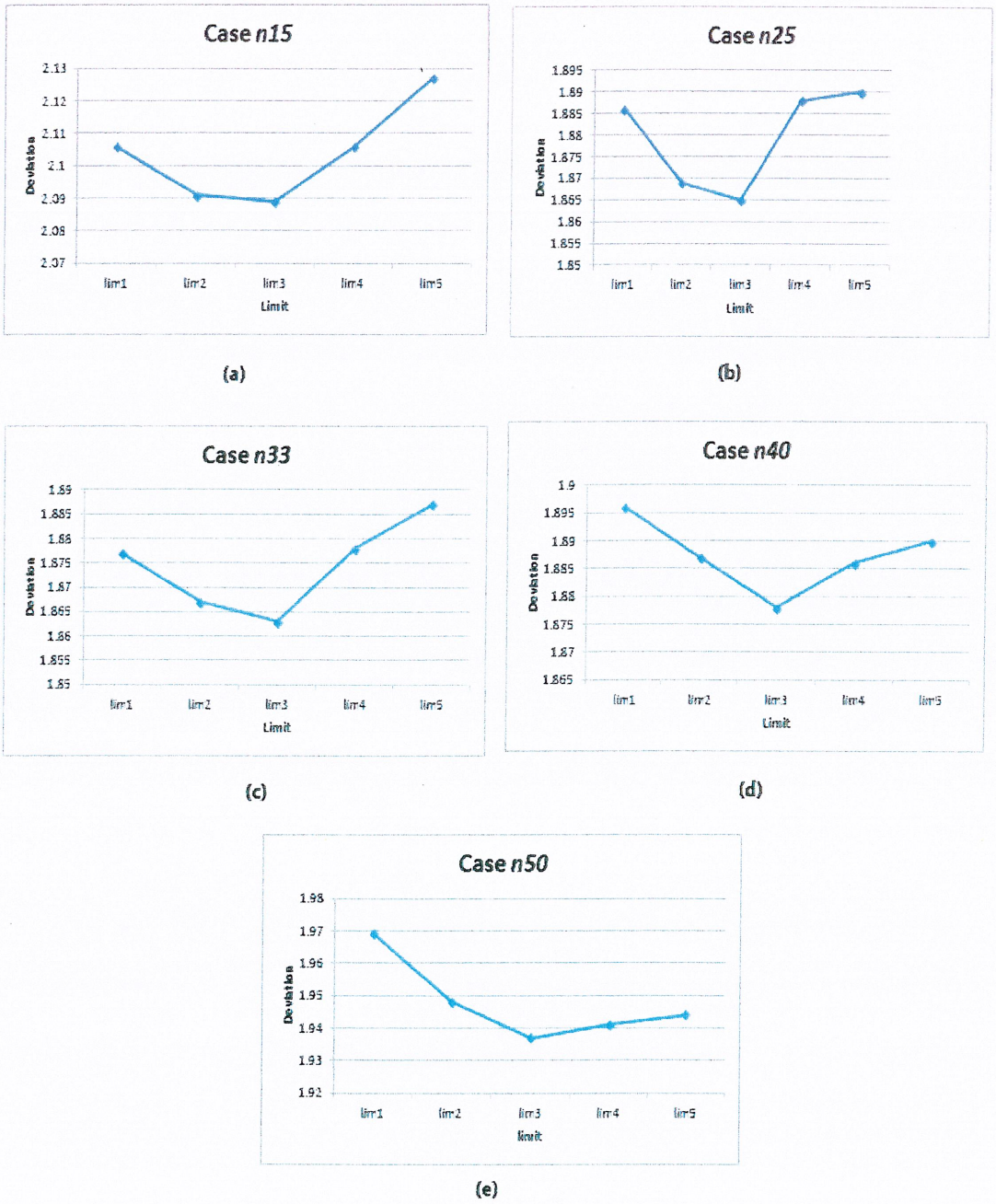
claim. The percentage differences for all test cases is given in Table 5.6. Statistically significant differences are shown in italics.

Table(5.6)
Percentage differences for different limit sets

Case	*% Difference*
n15	*1.79*
n25	*1.32*
n33	*1.27*
n40	*0.63*
n50	*1.62*

In Table 5.6 the percentage difference for each test is calculated by considering the data shown in Table 5.5. The absolute percentage difference between the best achieved result and the worst recorded result is calculated and presented in table 5.6. For the case *n15* the best and worst values were 2.089 and 2.127 and the percentage difference was 1.79%.

For all test cases, the worst results were obtained with lowest and highest limit values, which were, lim1 and lim5, respectively. This trend appears to be logical. As higher limit values cause scouts not to occur often and this downgrades the exploration capability of the ABC algorithm. Moreover, when the value of limit is set to infinity or (no limit), no scouts are permitted at all. The entire exploration process is omitted and all bees are directed to search locally for neighbor solutions. Applying only the exploitation process leads to the problem of local optimum trapping. On the other hand, lower limit values cause food sources to be abandoned too early. Newly constructed solutions might not be utilized sufficiently. That causes more scouts to be produced than needed. When control parameter limit is very low, the population would be comprised of so many random solutions. Figure 5.1 shows the effect of limit on average overall deviation for all test cases.



Figure(5.1)

Effect of limit on overall average deviation for (a) n15 (b) n25 (c) n33 (d) n40 (e) n50

5.3.2 Effect of Colony Size

Tables 5.7 to 5.11 show the effect of colony size on the quality of solutions. In these tables, the first column lists the colony size, which ranges from 10 to 30. The rest of the columns display the overall average deviations and corresponding average run time

obtained using parameter setups for all limit values shown in Table 5.4. For each parameter setup, the the average overall deviation of 30 runs and the standard deviation are reported. The corresponding average run time in seconds is also given. Also in Tables 5.7 to 5.11, DEV is the average overall deviation with standard deviation. Time is the average run time in seconds. The best (minimum overall deviation) is shown in bold.

Table(5.7)
Results for *n15* for different population size

Number of Bees	Limit 1		Limit 2		Limit 3		Limit 4		No limit	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	2.109 ±0.011	4.4	2.104 ±0.014	3.3	2.089 ±0.016	2.6	2.091 ±0.019	2.2	2.116 ±0.027	1.6
15	2.107 ±0.011	4.6	2.096 ±0.015	3.4	2.091 ±0.015	2.9	2.096 ±0.018	1.8	2.133 ±0.021	1.4
20	2.106 ±0.012	6.8	2.091 ±0.014	5.4	2.089 ±0.015	4.8	2.106 ±0.022	3.0	2.127 ±0.020	2.5
25	2.098 ±0.012	7.9	2.093 ±0.013	6.1	2.092 ±0.015	5.7	2.109 ±0.019	3.1	2.138 ±0.016	2.3
30	2.093 ±0.010	9.0	2.093 ±0.013	7.4	**2.085** ±0.017	7.0	2.136 ±0.017	3.2	2.139 ±0.021	2.5

Table(5.8)
Results for *n25* for different population size

Number of Bees	Limit 1		Limit 2		Limit 3		Limit 4		No limit	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.893 ±0.020	13.7	1.879 ±0.015	11.1	1.863 ±0.016	8.9	1.860 ±0.019	6.4	1.868 ±0.028	5.7
15	1.888 ±0.013	21.6	1.876 ±0.018	16.9	1.865 ±0.020	14.0	1.871 ±0.024	9.2	1.884 ±0.021	8.5
20	1.886 ±0.013	26.2	1.869 ±0.015	21.6	1.865 ±0.015	19.3	1.888 ±0.028	10.2	1.890 ±0.023	10.4
25	1.879 ±0.013	32.8	1.876 ±0.015	26.9	1.863 ±0.017	24.4	1.881 ±0.028	14.6	1.891 ±0.028	14.6
30	1.874 ±0.014	36.6	1.860 ±0.017	29.3	**1.858** ±0.015	28.1	1.902 ±0.028	14.5	1.887 ±0.023	14.8

It has been shown in previous studies, that ABC algorithm does not need fine tuning for the colony size in order to preserve satisfactory good results (Karaboga & Akay *2009a*). The reason behind is that ABC maintains two powerful mechanisms to diversify solutions. That is, either part of the solution is replaced to produce new neighboring solutions by foragers and onlookers. Or, entirely new solutions are created from scratch by scouts.

Table(5.9)
Results for *n33* for different population size

Number of Bees	Limit 1		Limit 2		Limit 3		Limit 4		No limit	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.889 ±0.013	40.8	1.868 ±0.015	32.2	1.867 ±0.020	25.0	1.874 ±0.022	19.6	1.867 ± 0.034	18.1
15	1.883 ±0.010	59.2	1.866 ±0.012	39.3	1.872 ±0.016	38.6	1.877 ±0.018	22.7	1.882 ±0.027	27.4
20	1.877 ±0.013	235.1	1.867 ±0.017	59.9	1.863 ±0.011	51.9	1.878 ±0.035	42.9	1.887 ± 0.026	36.1
25	01.875 ±0.011	284.7	1.866 ±0.013	260.4	1.865 ±0.015	252.5	1.886 ±0.025	207.7	1.880 ±0.027	211.2
30	1.869 ±0.013	297.6	1.868 ±0.016	268.9	**1.861** ±0.014	264.3	1.899 ±0.028	95.9	1.905 ± 0.029	228.6

Table(5.10)
Results for *n40* for different population size

Number of Bees	Limit 1		Limit 2		Limit 3		Limit 4		No limit	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.915 ±0.009	70.9	1.893 ±0.014	59.4	1.882 ±0.017	43.5	1.881 ±0.013	37.9	**1.864** ± 0.024	32.2
15	1.898 ±0.012	144.6	1.887 ±0.018	103.0	1.879 ±0.024	85	1.885 ±0.023	68.9	1.869 ± 0.028	51.3
20	1.896 ±0.014	155.6	1.887 ±0.018	124.7	1.878 ±0.017	108.5	1.886 ±0.024	81.1	1.890 ±0.029	78.6
25	1.899 ±0.011	170.8	1.882 ±0.013	134.8	1.873 ±0.015	101.9	1.896 ±0.025	92.1	1.883 ±0.027	96.5
30	1.893 ±0.015	227.5	1.876 ±0.016	188.5	1.874 ±0.029	146.8	1.887 ±0.027	118.5	1.883 ±0.022	115.3

Table(5.11)
Results for *n50* for different population size

Number of Bees	Limit 1		Limit 2		Limit 3		Limit 4		No limit	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.978 ±0.014	219.5	1.957 ±0.020	148.8	1.945 ±0.025	95.4	1.950 ±0.027	93.8	1.950 ± 0.025	93.3
15	1.975 ±0.012	316.3	1.954 ±0.024	221.6	1.948 ±0.029	155.3	1.935 ±0.033	121.8	1.949 ±0.025	122.4
20	1.969 ±0.012	364.2	1.948 ±0.019	251.1	1.937 ±0.020	193.2	1.941 ±0.030	159.8	1.944 ± 0.033	182.4
25	1.964 ±0.015	439.5	1.949 ±0.020	314.6	1.954 ±0.023	258.1	1.944 ±0.033	227.3	**1.927** ±0.027	226.3
30	1.963 ±0.014	516.3	1.954 ±0.019	326.4	1.946 ±0.020	332.0	1.945 ±0.023	253.9	1.936 ±0.028	287.0

Thus, ABC does not require huge colony sizes to solve problems with big dimensions (Karaboga & Basturk 2008) "*Since ABC algorithm uses exploitative process efficiently to converge minima and explorative process to provide sufficient diversity in the population for a given colony size, ABC algorithm does not need big number of colony size to solve optimization problems with high dimensions.*"

In Table 5.7 considering lim1, lim2 and lim3, a general trend is that increasing the colony size has a positive impact on the quality of the produced solutions. For example, in Table 5.7 with lim1, the deviation of 2.109 was obtained with colony size equal to 10. Then, the deviation decreased gradually to 2.107, 2.106, and 2.098 with colony of sizes 15, 20, and 25 respectively. In this instance, the lowest deviation achieved was 2.093 with colony of size 30. Also in Table 5.7, the same trend was somewhat observed with lim3 and lim4. The deviation slightly decreased upon increasing the colony size and reached its lowest with colony size equal to 30. However, some exceptions were found, in lim2, the deviation of 2.091 with colony of size 20 was obtained, while with bigger colony size of 25, the deviation was 2.093. Yet, the small difference can be neglected as it is shown that ABC is not very sensitive to colony size as a control parameter. The same trend was also observed in Tables 5.8, 5.9 related to cases *n25*, *n33*, respectively. The percentage differences for all test cases, with different colony sizes and different limit values are given in Table 5.12. Statistically significant differences are shown in italics. The percentage difference is calculated by considering the difference between the best and the worst deviations for each case instance with different colony sizes. For example in the case of *n15* for limit value equal to lim1 the worst deviation was 2.109 and the best deviation was 2.093 therefore the percentage difference in this case was 0.76%. Similarly, the percentage difference for the rest of the cases were calculated.

It is also observed that in Tables 5.7 to 5.11 with limits equal to lim4 and lim5, increasing the colony size did not achieve any improvement in the results. In this case, having very high values for limit reduced the effect of colony size, as a control parameter. Very high values for limit or even no limit would encourage the exploitation process. Consequently, very few or no scouts are allowed to occur. All agents are directed to search the space

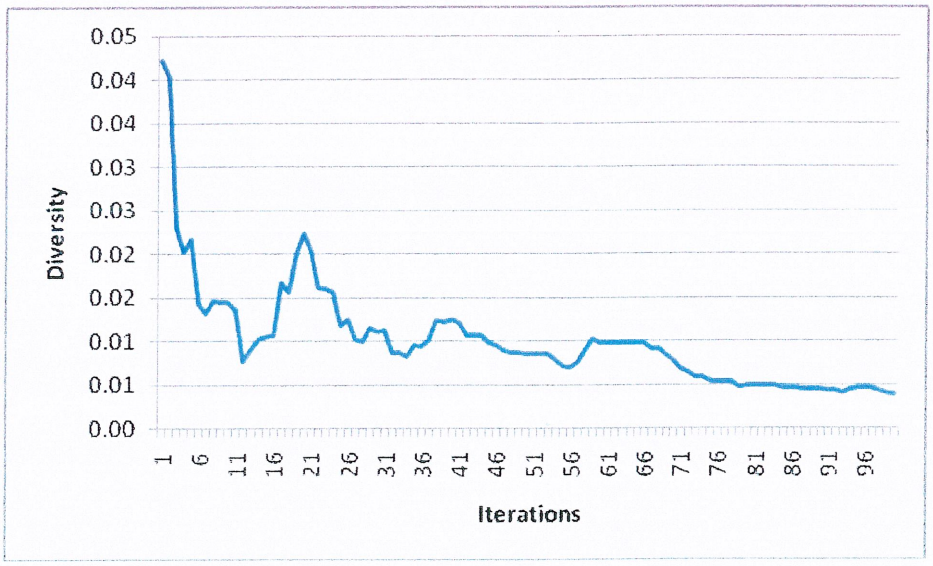
Table(5.12)

Percentage differences for different population sizes for all limit sets

Case	% Difference				
	Lim1	Lim2	Lim3	Lim4	Lim5
n15	*0.76*	*0.62*	*0.33*	*2.10*	*0.96*
n25	*1.00*	*1.01*	*0.37*	*2.2*	*1.21*
n33	*1.05*	*0.10*	*0.58*	*0.69*	*1.99*
n40	*1.15*	*0.90*	*0.48*	*0.79*	*1.38*
n50	*0.57*	*0.46*	*0.87*	*0.76*	*1.18*

locally for better solutions. In this case, the quality of the produced results will depend more on solutions diversity and the quality of initial solutions, rather than the population size. In other words, if the scouts are not allowed to occur, then, no matter how big the colony is, the quality of the produced solutions depends on the richness of current food sources.

Another interesting observation in Tables 5.7 to 5.11 is that for all test cases, the achieved overall deviation improved when the value of the limit gradually increased through lim2, to reach its ideal value with lim3. Then in most of instances, this achieved deviation again worsened with high values of limits lim4, and lim5. Nevertheless, in some cases even better results were obtained with high values of limit, such as the cases of *n40* and *n50*, the results in these cases were different than the dominant trend. For example, in case of *n40*, the best average overall deviation of 1.864, was obtained with limit = lim5, and number of agents = 10. Also, in case of *n50*, specifically, for 25 agents, the overall deviation decreased from 1.964 to 1.949, when the limit was gradually increased. A possible reason of this observation could be that, the diversity of the individuals in the population was sufficient and the solutions proceed with evolution even when no scouts are allowed. In this case, the algorithm did not need scouts to explore new regions in the search space to produce satisfactory good results. The population diversity of case *n50* is shown in Figure 5.2 below.



Figure(5.2)
Solutions diversity for n50 for colony size = 25 and no limit

To summarize, control parameter limit is an essential parameter of ABC algorithm dictating the occurrence of scout bees that are responsible of providing diversity in the population. For relatively smaller populations, control parameter limit should not be so low in order to avoid the population to be comprised of so many random solutions. However, if large populations have sufficient diversity and improved in their progress, limit parameter does not stand out in this situations.

Also, it can be noticed that, for all test cases, the computational time significantly decreases as the limit increases. This is could be related to the fact that for some instances, specially for relatively larger test cases, when the limit increases, scouts are less often employed. Generating scout bees consumes large computational time. Compared to other agents, scouts are generating solutions from scratch which is much more complex and time consuming than generating a neighbor solution by changing one link in a solution.

As seen from numerical results and figures belonged to the experiments, when the diversity in the initial population is sufficient, effects of scout bee and limit values decrease. The

population distribution provides enough diversity for the algorithm. In conclusion, the exploration and exploitation abilities in the ABC algorithm should be balanced using limit and population size in order to obtain better results in a reasonable computational time.

5.4 Results and Discussion for ACO Algorithm

In this section the empirical results related to ACO are presented and analyzed to evaluate the performance of the proposed algorithm. The fuzzy goal programming based ACO was applied to the same five test cases described earlier in this chapter. In the experiments, the colony sizes were taken as 10, 15, 20, 25, and 30 ants. The adopted combinations of pheromone deposit and evaporation rates, and the heuristic values are illustrated in Table 5.13, and Table 5.21 respectively. For each parameter set up, thirty independent runs were conducted. Each test instance was run for 100 iterations. The average of best solutions found in each run was reported, along with the standard deviation. The results were statistically validated using Wilcoxon ranked-sum test (see Appendix A). To study the effect of pheromone deposit and evaporation rate, default values were assumed for colony size and heuristic values. The default colony size was taken as 20 ants. While, to give equal importance to pheromone intensification on a link α_{ant} and heuristic value of the link β_{ant}, both heuristic values α_{ant}, and β_{ant} were given equal values and were taken as 0.5.

5.4.1 Effect of Pheromone Deposit and Evaporation

Table 5.13 shows the pheromone deposit and evaporation rates considered in the experiments. In this table, the deposit and evaporation rates are given in columns 2 and 3, respectively. Column 4 shows the difference between the pheromone deposit

and evaporation rates, denoted as "DEP". For example, in set 1, the deposit is 0.2 and no evaporation takes place and hence, the DEP is 0.2 and so on. The difference between pheromone deposit and evaporation rates (i.e. DEP) increases gradually with each set. For example, in set 1, the difference between deposit and evaporation was 0.2. Then, gradually increased in each set to reach a maximum of 0.5 in set 4.

Table(5.13)

Parameter settings for ACO used in experiments

Parameter set	Pheromone rates		DEP
	Deposit	Evaporation	
Set 1	0.2	0.0	0.2
Set 2	0.4	0.1	0.3
Set 3	0.6	0.2	0.4
Set 4	0.8	0.3	0.5

Table 5.14 shows the best and worst average overall deviations labelled as Min.Deviation and Max.Deviation, respectively. For each test case instance, the corresponding pheromone deposit and evaporation rates and the average run time in seconds are given. The last column gives the percentage difference obtained when the two situations were compared. As an example, in the case of *n*33, where the best overall average deviation of 1.887 was achieved with DEP = 0.2 (i.e. deposit = 0.2 and evaporation = 0), while for DEP = 0.5 (i.e. deposit = 0.8, and evaporation = 0.3), the overall deviation was 1.938. The difference was 2.7 % and average run time was 38.1 seconds for both instances. These results were obtained when pheromone deposit was small and no evaporation occurred. In Table 5.14, Time is the average run time (in seconds), % Dif is the percentage difference. Statistically significant differences are presented in italics.

In Table 5.14 it can be noticed that, the best (minimum) overall deviations were obtained with DEP = 0.2 for all test cases. Whereas, the worst (maximum) deviations for almost all test cases occurred with DEP = 0.5. An exception was case *n*40, where the

Table(5.14)

Results for best and worst average overall deviation and their corresponding DEP setup

Case	DEP	Min. Deviation	Time	DEP	Max. Deviation	Time	% Dif
n15	0.2	2.031 ± 0.010	5.6	0.5	2.068 ± 0.016	5.8	*1.8*
n25	0.2	1.921 ± 0.015	15.6	0.5	1.978 ± 0.014	15.9	*3.0*
n33	0.2	1.887 ± 0.015	38.1	0.5	1.938 ± 0.017	38.1	*2.7*
n40	0.2	1.894 ± 0.019	437.1	0.4	1.938 ± 0.011	374.3	*2.3*
n50	0.2	1.913 ± 0.016	7537.3	0.5	1.991 ± 0.012	3035.5	*4.1*

maximum deviation was reported with DEP = 0.4. Furthermore, all percentage differences in Table 5.14 were found to be statistically significant, as validated by Wilcoxon ranked-sum test.

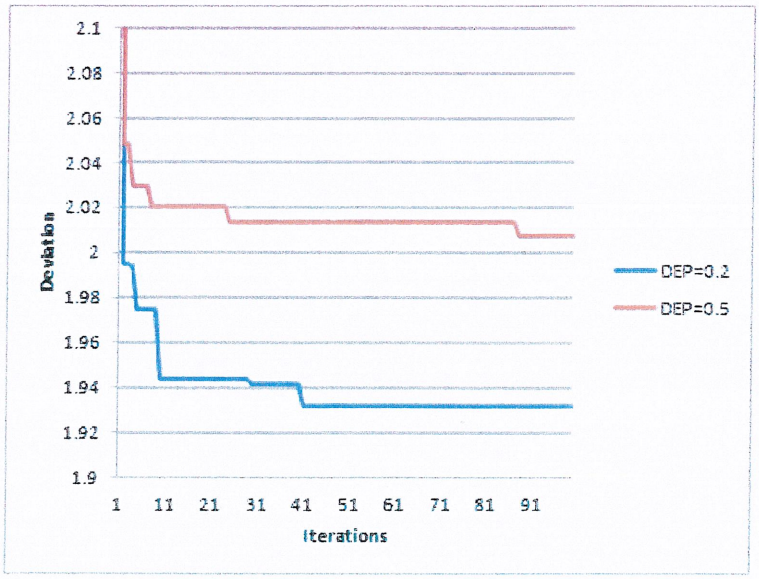
The results presented in Table 5.14 can be justified as follows: With DEP = 0.2, the pheromone deposit rate is small (deposit rate = 0.2), and no evaporation takes place (evaporation rate = 0). As mentioned earlier in Section 2.3 and as found in the literature (Dorigo & Stützle 2004), pheromone evaporation symbolizes a forgetting scheme for past experience, and thus, favoring the exploration of new regions in the search space. When no evaporation takes place, then the pheromone contents on links that appear in the elitist ant solution would be higher than the pheromone contents on other links. As time elapses, this behavior continues. Therefore, the pheromone contents on candidates links that appear repeatedly in the elitist ants solutions would increase excessively. Other links that have not been chosen frequently by the elitist ant would preserve none to very low pheromone contents. Eventually, there would be few links which have accumulated a remarkable amount of pheromone, and therefore would be selected again and again. Due to the big differences among probabilities favoring links with higher pheromone contents, the exploitation process is favored, forcing the ants to search more rigorously near previously discovered good solutions. Generally, this behavior may lead to the problem of local optimum trapping. Specially, if the search process is directed too early (when

adopting high DEP rates). Yet, the small deposit rate adopted in this case (i.e. deposit = 0.2) contributes in preventing the problem of premature convergence to suboptimal solutions. The pheromone levels on selected links were adjusted unhurriedly. Hence, in the beginning of the search, there will be a little difference between the probabilities, $pk_ij(t)$, which makes selection of links more random. Later, only quality links that have accumulated a noticeable amount of pheromone would be selected, and thus the search will be directed towards best profound regions, which in turn produced better solutions.

In contrast, in the case of DEP = 0.5, the pheromone deposit rate is high (deposit rate = 0.8), and the evaporation rate is low (evaporation rate = 0.3), this resulted in a problem of premature convergence. Good links were rewarded by adding huge amount of pheromone too early, with less amount of evaporation. The differences between probabilities will be very high and the ants are forced to focus the search in narrow areas very soon. Thus, worse results were obtained in this case. Figure 5.3 supports the claims mentioned above. This figure shows the behavior of the algorithm with DEP = 0.2 and DEP = 0.5 for test case *n50*. It can be easily observed that with DEP = 0.2, the overall achieved deviation is less than that obtained with DEP = 0.5 for the same number of iterations.

5.4.2 Effect of Colony Size

The effect of colony size (i.e number of ants) is shown in Tables 5.15 to 5.19. Each table consists of nine columns. The first column displays the number of ants ranging from 10 to 30. The rest of the columns show the achieved average deviation and the average run time for each DEP set values of 0.2, 0.3, 0.4, and 0.5. The heuristic values were kept at their default (i.e., 0.5). The results are shown in a separate table for each test case. In these tables, DEV is the Average Overall Deviation with standard



Figure(5.3)

Effect of different DEP rates on overall average deviation for the case *n*50

deviation. Time is the average run time in seconds. The best (minimum overall deviation) is shown in bold.

Table(5.15)

Results for *n*15 for different colony sizes

Number of Ants	DEP = 0.2		DEP = 0.3		DEP = 0.4		DEP = 0.5	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	2.037 ±0.010	3.5	2.60 ±0.016	3.3	2.069 ±0.012	2.7	2.082 ±0.020	2.8
15	2.033 ±0.011	4.1	2.070 ±0.021	4.1	2.066 ±0.017	4.1	2.074 ±0.017	4.1
20	2.031 ±0.010	5.6	2.045 ±0.017	5.8	2.057 ±0.016	5.8	2.068 ±0.016	5.8
25	2.029 ±0.009	7.6	2.043 ±0.013	7.7	2.057 ±0.011	7.6	2.062 ±0.014	7.6
30	**2.025** ±0.008	9.4	2.042 ±0.015	9.5	2.050 ±0.016	9.4	2.057 ±0.014	9.4

A general trend in Tables 5.15 to 5.19 suggests that, increasing the number of ants enhanced the quality of obtained solutions. Better results (lower deviation) were obtained in all test cases when the number of ants was increased. For example, in the first column of Table 5.17, it is observed for case *n33* with DEP = 0.2 that the overall average deviation was 1.902 for 10 ants, which decreased gradually as the number of ants increased.

Table(5.16)
Results for $n25$ for different colony sizes

Number of Ants	DEP = 0.2		DEP = 0.3		DEP = 0.4		DEP = 0.5	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.931 ±0.027	8.3	1.982 ±0.011	8.6	1.983 ±0.018	8.7	1.991 ±0.011	8.3
15	1.930 ±0.015	12.7	1.972 ±0.015	12.7	1.972 ±0.017	12.7	1.982 ±0.031	12.5
20	1.921 ±0.015	15.6	1.969 ±0.013	16.0	1.978 ±0.011	16.0	1.978 ±0.014	15.9
25	1.916 ±0.012	27.0	1.966 ±0.013	27.2	1.972 ±0.011	27.1	1.973 ±0.014	26.6
30	**1.915** ±0.010	28.6	1.966 ±0.012	28.8	1.915 ±0.014	27.7	1.971 ±0.014	27.9

Table(5.17)
Results for $n33$ for different colony sizes

Number of Ants	DEP = 0.2		DEP = 0.3		DEP = 0.4		DEP = 0.5	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.902 ±0.015	37.0	1.940 ±0.011	36.8	1.947 ±0.013	37.2	1.948 ±0.015	37.4
15	1.895 ±0.015	46.6	1.935 ±0.012	46.5	1.941 ±0.014	46.7	1.939 ±0.013	46.1
20	1.887 ±0.033	38.1	1.935 ±0.023	38.2	1.935 ±0.039	38.1	1.938 ±0.024	38.1
25	1.880 ±0.018	988.7	1.931 ±0.009	990.5	1.933 ±0.013	989.8	1.940 ±0.011	989.9
30	**1.878** ±0.015	999.2	1.925 ±0.011	999.0	1.927 ±0.018	999.8	1.936 ±0.008	998.4

Table(5.18)
Results for $n40$ for different colony sizes

Number of Ants	DEP = 0.2		DEP = 0.3		DEP = 0.4		DEP = 0.5	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.910 ±0.014	224.9	1.943 ±0.014	207.9	1.945 ±0.009	246.7	1.944 ±0.012	226.5
15	1.905 ±0.015	376.7	1.934 ±0.012	359.4	1.942 ±0.014	368.9	1.933 ±0.013	352.8
20	1.894 ±0.019	437.1	1.930 ±0.018	419.7	1.983 ±0.011	374.3	1.931 ±0.017	441.4
25	1.889 ±0.014	2459.0	1.932 ±0.012	2484.7	1.931 ±0.013	2468.9	1.931 ±0.013	2471.1
30	**1.888** ±0.014	2456.5	1.927 ±0.011	2457.1	1.934 ±0.013	2514.5	1.930 ±0.013	2510.4

Finally, an overall average deviation of 1.878 was resulted with 30 ants. Similarly, the rest of the tables showed the same pattern. An

Table(5.19)
Results for $n50$ for different colony sizes

Number of Ants	DEP = 0.2		DEP = 0.3		DEP = 0.4		DEP = 0.5	
	DEV	Time	DEV	Time	DEV	Time	DEV	Time
10	1.929 ±0.016	5873.0	1.995 ±0.012	3525.0	1.990 ±0.012	3525.5	1.993 ±0.012	3527.2
15	1.917 ±0.011	3773.6	1.990 ±0.013	2367.6	1.994 ±0.013	2369.1	1.993 ±0.014	2521.1
20	1.913 ±0.016	7537.3	1.984 ±0.013	3063.6	1.988 ±0.014	3053.7	1.991 ±0.012	3035.5
25	1.911 ±0.014	17862.6	1.982 ±0.011	8262.5	1.985 ±0.012	8262.7	1.990 ±0.010	9218.4
30	**1.911** ±0.008	41913.9	1.981 ±0.010	17566.6	1.982 ±0.013	17570.9	1.986 ±0.010	18759.9

exception was the case $n50$ with DEP = 0.4, when lower deviation of 1.990 achieved with number of ants that is equal to 10 ants compared to deviation of 1.994 with number of ants equal to 15 ants. Yet, the difference in the overall deviations in this case was small (0.004) as well as the difference in the number of ants. Such cases were also noticed in the case $n33$ with DEP = 0.5 and also $n40$ with DEP =0.3. However, in all cases the lowest achieved deviation was reported with number of ants = 30. The percentage differences are as illustrated in Table 5.20. Significant percentage differences are italicized.

The above trends can be logically explained as follows. As increasing the number of ants ensures better diversity in initial population, more ants are recruited to search simultaneously for better solutions. Consequently, more solutions are created and evaluated at each iteration. The probability of finding better elitist ant also increases. Thus, finally leads to better searching capabilities.

Looking again at Tables 5.15 to 5.19, considering the same number of ants for various DEP setup, it can be observed that, for all test cases, and for all colony sizes, better solutions were obtained with DEP = 0.2. For DEP = 0.3 to DEP =0.5, the solutions' quality decreased gradually as DEP rate increased. In short, the solutions quality were inversely proportional to DEP

rate. In most cases, the worst results were obtained with DEP = 0.5 and colony size = 10. This behavior appears to be logical due to the problem of premature convergence to suboptimal solutions with higher DEP rates as discussed earlier in Section 5.4.1. However, increasing the colony size with higher DEP rates has enhanced the solutions quality but only up to a certain level of achievement. The percentage differences for all test cases and all DEP sets are shown in Figure 5.20. Statistically significant differences are shown in italics.

Table(5.20)

Percentage differences with regard to increase in colony size

Case	% Difference			
	DEP = 0.2	DEP = 0.3	DEP = 0.4	DEP = 0.5
n15	*0.59*	*0.80*	*3.42*	*1.00*
n25	*0.82*	*0.80*	*3.40*	*1.00*
n33	*1.26*	*0.77*	*1.02*	*0.61*
n40	*1.15*	*0.82*	*0.71*	*0.72*
n50	*0.933*	*0.70*	*0.60*	*0.35*

5.4.3 Effect of Heuristic Values

This section studies the effect of exponents α and β. As mentioned earlier in Section 2.3 α depicts the influence of pheromone trail and β depicts the influence of heuristic information. The experiments were aimed at understanding the impact of parameter settings on the performance of the algorithm. Three parameter settings were considered for the experiments as shown in Table 5.21. In set 1, both α and $\beta$are given the same importance, and hence, both were given an equal value of 0.5. In set 2, emphasizes the influence of heuristic value given to the link β, and thus β was taken as 0.8 while α was set lower value of 0.2. The opposite in set 3, which emphasizes the effect of pheromone trail. The colony size was set at its default value (i.e. 20 ants).

Table(5.21)

Parameter settings for various heuristic values used in experiments

Parameter set	α	β
Set 1	0.5	0.5
Set 2	0.2	0.8
Set 3	0.8	0.2

The performance of ACO was compared for the considered parameter settings. The experimental results for parameter set 1 were shown previously in Table 5.14. Tables 5.22 and 5.23 are showing the results for the second and third parameter sets, respectively. Results for $\alpha = 0.2$, $\beta = 0.8$ best and worst average overall deviation and their corresponding pheromone deposit and evaporation rate setup. Time = Average run time (in seconds), % Dif = percentage difference. Statistically significant differences are presented in italics.

Table(5.22)

Results for $\alpha = 0.2$, $\beta = 0.8$

Case	DEP	Min. Deviation	Time	DEP	Max. Deviation	Time	% Dif
n15	0.2	2.044 ± 0.007	6.6	0.5	2.072 ± 0.012	6.3	*1.4*
n25	0.2	1.928 ± 0.014	20.2	0.5	1.977 ± 0.012	24.4	*2.5*
n33	0.2	1.891 ± 0.015	967.4	0.5	1.939 ± 0.011	969.4	*2.5*
n40	0.2	1.895 ± 0.013	1600.6	0.5	1.939 ± 0.011	1585.8	*2.3*
n50	0.2	1.915 ± 0.012	28284.4	0.5	1.993 ± 0.008	11513.9	*3.9*

Table(5.23)

Results for $\alpha = 0.8$, $\beta = 0.2$

Case	DEP	Min. Deviation	Time	DEP	Max. Deviation	Time	% Dif
n15	0.2	2.013 ± 0.006	6.3	0.5	2.069 ± 0.013	6.6	*2.7*
n25	0.2	1.920 ± 0.016	24.3	0.5	1.977 ± 0.008	24	*2.8*
n33	0.2	1.886 ± 0.014	968.1	0.5	1.940 ± 0.011	968.9	*2.8*
n40	0.2	1.902 ± 0.017	400.9	0.4	1.940 ± 0.011	400.6	*2.0*
n50	0.2	1.916 ± 0.015	26385.2	0.5	1.994 ± 0.009	17446.3	*3.9*

In Tables 5.14, 5.22, and 5.23, it can be observed that for all test cases, the best results were obtained with DEP = 0.2 and the worst results were produced with DEP = 0.5. As observed in these tables, a general trend is that for the first three test cases

(*n15*, *n25*, and *n33*) the best solutions were obtained with the third parameter set (i.e α = 0.8, β = 0.2). For example, considering the case *n15*, the best solution of 2.013 was obtained with the third parameter set, while, the second best solution of 2.031 was obtained with the first parameter set. The third best solution, which was 2.044 was obtained with the second parameter set. The test cases *25n* and *33n* followed the same pattern. While for the last two test cases, *n40* and *n50*, the first parameter set produced better results. Generally, the second and third parameter settings complemented each other, performing better on each other's instances. In summary, the conducted comparisons suggest that the third parameter set resulted in better solutions for relatively smaller dimension test cases (*n15*, *n25*, and *n33*) whereas for larger problem dimensions of *n40* and *n50*, the first parameter set produced better results. The second parameter setting performed poorly compared to the other two parameter sets.

The results presented in Tables 5.14, 5.22, and 5.23, can be justified as follows. Emphasizing pheromone trail α for smaller test cases produced better results. The reason behind, could be related to the limited number of possible links available at each stage of solution construction process. This could be further attributed to the fact that emphasizing the influence of pheromone information encourages the selection of previously experienced links. Specifically, better links accumulate extra amount of pheromone and thus get selected repeatedly. Thus, good links become strong candidates throughout the search process. This scheme would produce better results for relatively smaller test cases, due to the limited number of possible links at each step when the solution is to be created. On the other hand, for larger test cases, emphasizing the influence of pheromone trails α may implicitly deprive unexplored good links from participating in solution construction process. For the underlying problem, a moderate value for heuristic information was sufficient to balance

the search and produce better results when larger problem dimensions were considered.

5.5 Concluding Remarks

In this chapter, an experimental study was performed on both of the proposed algorithms, ABC and ACO. Experimental results were collected for the five test cases mentioned previously. The performance of the algorithms was tested and analyzed. The effect of different control parameters was also examined. For ABC, a discrete version of ABC algorithm for solving DLAN topology design problem was proposed. A modified ABC was proposed considering the multi-objective nature of the problem. Mutation operator was specifically designed for the proposed algorithm. Adaptive scouts production scheme was also proposed to enhance the global search ability of ABC. In the present work, the performance of the original ABC algorithm was enhanced using ranges of control parameters that were re-scaled to suit the problem considered. Rather than having a fixed value, expressions were defined to determine a range of "Limit" values. For ACO, Important setting parameters such as, pheromone deposit and evaporation rate, number of ants, and heuristic values were examined. Empirical analysis was done to study the effect of these parameters. Results were statistically validated using Wilcoxon ranked-sum test.

Chapter Six
Comparison of Techniques

6.1 Results and Discussion	86
6.2 Concluding Remarks	91

Chapter Six

Comparison of Techniques

This chapter mutually compares the performance of the proposed artificial bee colony algorithm to that of the ant colony optimization algorithm. First, An initial comparison was conducted between the two algorithms considering the results obtained previously in Chapter 5. However, to make a fair comparison, the experiments were conducted for both algorithms being given the same time interval while setting the control parameters to its optimum values. The optimum values of control parameters were taken as the ones that produced the best results in Chapter 5. Finally, the results of the comparison are presented, discussed and analyzed in this chapter.

6.1 Results and Discussion

This section describes the experiments conducted to compare the performance of both ACO and ABC. Experiments carried out previously in Chapter 5, showed that varying the control parameters affects the performance of both algorithms significantly. Initially, the performance of both algorithms was evaluated by considering the results obtained and presented previously in Chapter 5. In these results, it is observed that ACO performed better in the case *n15*, while in cases *n25*, *n33*, and *n40* ABC produced better results. Again, in case *n50*, ACO performed better. In these cases, all percentage differences were statistically significant. It is also observed that, the computational time required to finish thirty runs in ACO was much larger than that in ABC. As was found in relevant literature and in order to make the comparison fair, experiments have been conducted, both algorithms were run for the same time interval while setting

control parameters to its optimum values (Mohiuddin et al. 2014, Khan et al. 2007, Khan & Engelbrecht 2007). Each experiment, consisted of thirty independent runs and terminated after predetermined time interval. Considering results obtained previously in Chapter 5, time was set to the best time achieved for each test case instance. In all experiments, the same test cases described in Chapter 5 were used. The average deviations produced have been recorded along with the standard deviation. The results were statistically validated using Wilcoxon ranked-sum test. Table 6.1 shows the results obtained. In this table, DEV is the average overall deviation with standard deviation. Time is the average run time in seconds. Dif% denotes percentage differences. Also, CS is the Colony Size, DEP is pheromone deposit and evaporation rate, HS is Heuristic values set. Statistically significant differences are shown in italics.

Table(6.1)
Comparison of ACO and ABC

Case	ACO					ABC				Dif
	DEV	Time	DEP	CS	HS	DEV	Time	Limit	CS	%
N15	2.062 ±0.010	6.3	0.2	20	Set 3	2.085 ±0.013	6.3	Lim3	30	*1.10*
N25	1.919 ±0.011	28.1	0.2	30	Set 3	1.863 ±0.017	28.1	Lim3	30	*2.9*
N33	1.858 ±0.011	246.3	0.2	30	Set 3	1.836 ±0.012	246.3	Lim3	30	*1.18*
N40	1.904 ±0.022	65	0.2	32	Set 1	1.880 ±0.009	32	Lim5	10	*1.26*
N50	1.948 ±0.013	226.3	0.2	30	Set 1	1.935 ±0.026	226.3	Lim5	25	*0.66*

From Table 6.1 it observed that, in the case *n15* ACO performed better while in the remaining test cases ABC out-performed ACO. Generally, the performance of ABC became clearer for relatively larger test cases. This trend could be related to the following justification: considering the fundamental characteristics distinguishing ACO and ABC algorithms, each algorithm has different distinctive features to generate and process

solutions. For example, the process used to diversify solutions in the population is different in both algorithms. ACO deploys a unified process to diversify solutions which implies the generation of new solution from scratch every time a solution is to be created. While, ABC adopts a dual diversification process that can be clarified as follows.

- In ABC, the solution is either modified by selecting and changing a link randomly to produce a neighbor solution. This scheme provides ABC algorithm with a powerful local search ability and is also useful for fine tuning.

- Or, rather than changing part of a solution, the whole solution is removed and replaced by a new random solution generated from scratch. This scheme provides ABC with global search ability and prevents the problem of premature convergence.

Consequently, comparing the two algorithms, ACO maintains a powerful global search operator to diversify solutions. Since, in ACO new candidate solutions are always built from scratch. Thus, encouraging exploration process in search space. The exploitation process in ABC is controlled by the pheromone information and heuristic values, which are determined heuristically.

In ABC algorithm the scheme is different, ABC employs an advanced local search operator that produces a mutant solution from its original version by selecting and modifying a random link. Since the mutation rate adopted in this thesis is small (changing one link at a time). The process of mutant solution creation is simple and fast, and creates a large number of solutions in less computational time. The process of mutant solution creation is computationally less expensive than creating a whole new solution. Hence, it is logical to have ABC consuming less computational time to complete an independent run compared to ACO. The fact that ABC does not need to build a solution

from scratch every time it creates a solution, has also led to increasing the convergence speed of the search. Though, the exploration process in ABC is controlled by the limit, which is determined heuristically.

Furthermore, in ABC the best solution found so far is kept in the population and it can be used repeatedly for producing new mutant solutions until it reaches the limit and replaced with a randomly generated solution by a scout. In ACO, the best solution discovered so far is not held as an active candidate solution in the population since it is replaced with a newly produced solution by the corresponding ant in the next iteration. The pheromone information is used to pass attributes of good links to the next iteration. Therefore, the solution does not contribute directly to the creation of new candidate solutions.

Moreover, ABC employs a greedy selection process between the current mutant solution and the original solution. In ACO, pheromone concentration and heuristic information are used to keep track of good solutions. In ABC, in the employed bee phase, a mutant solution is produced for each solution in the population regardless of the quality of solutions. Later, in the onlooker bees phase the solutions with the higher fitness value are utilized more often than those with less fitness values to produce other mutant solutions. This leads to having the promising regions around the best produced solutions searched thoroughly in shorter time.

From the above discussion it can also observed that, ACO has a powerful exploration operator since it frequently creates new solutions in the search space. However, the local search operator depends upon both pheromone concentration and heuristic information to keep track of the best obtained solutions. For ABC, the scheme is different, ABC has a powerful exploitation operator. Yet, the global search operator in ABC is controlled by the limit control parameter which is often determined heuristically. ABC applies small changes trying to improve solution. In contrast,

ACO adopt large step wise scheme and builds an entire solution from scratch every time a solution is to be created.

Apart from the termination condition and population size, ABC algorithm has only one control parameter (limit) which controls the scouts production frequency. While, ACO, has two control parameter apart from the termination condition and population size these are, pheromone deposit and evaporation rate, and the heuristic values. Therefore, it can be concluded that, ABC is simple and flexible and also employs less control parameters. A key factor for successful application of ABC is to establish a good balance between the local search process carried out by employed bees and the global search process performed by scout bees. Figure 6.1 shows the performance of ACO versus that of ABC. In this figure, it can be seen that for case *n15* ACO produced better overall achieved deviation. While for the rest of the test cases, ABC produced better results. The above discussion and observations suggest that, a possible reason for having ACO performed better for relatively smaller test cases, is that for smaller test cases creating a solution from scratch is easier and more affordable than changing a solution link to create a mutant solution. In contrast, for larger problem dimensions, creating a mutant solution seemed to be more reasonable, since constructing a whole new solution is both computationally expensive and time consuming process, this provides a comprehensive reason for having ABC performing better for larger test cases.

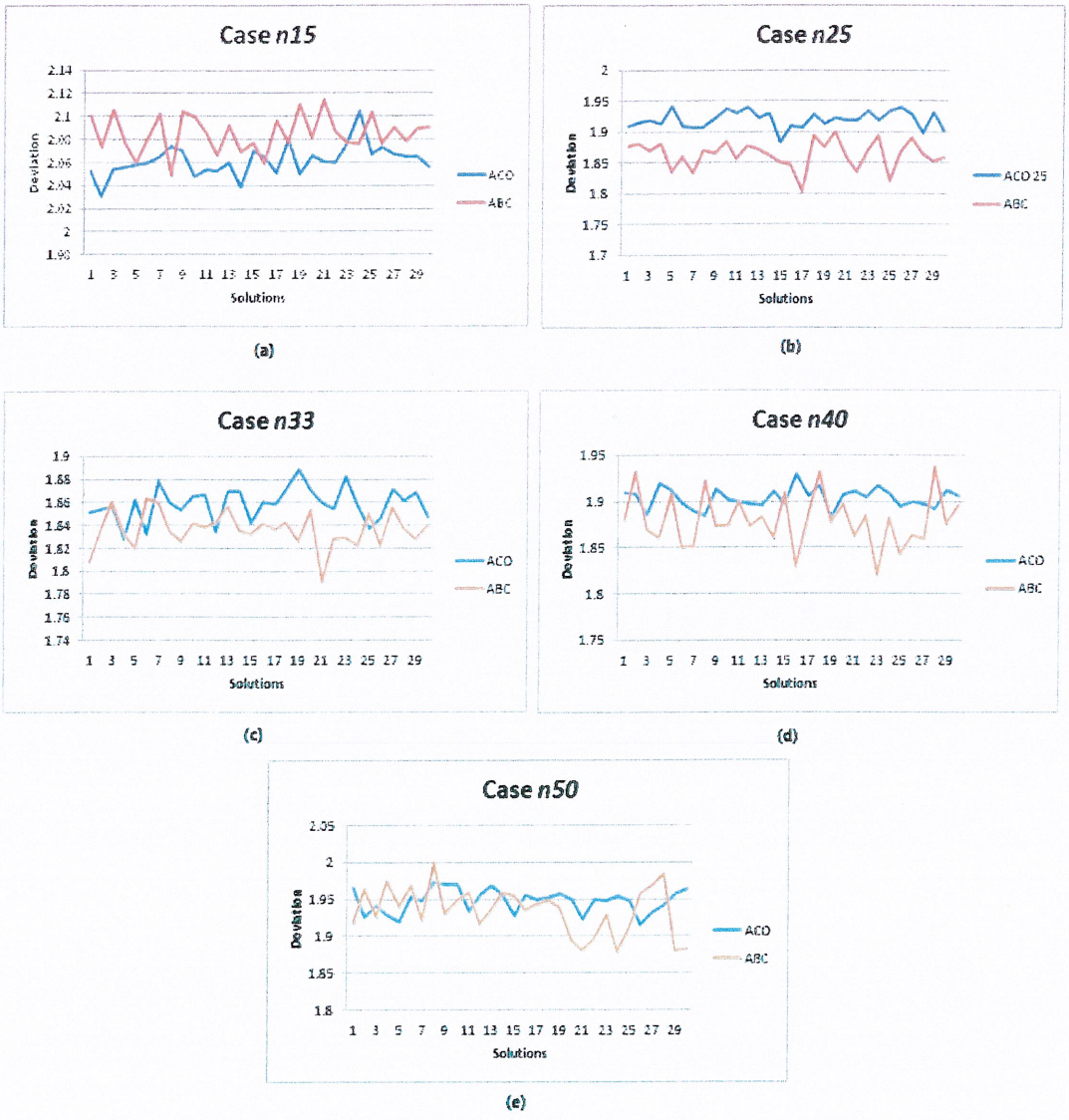


Figure (6.1)
Plots of solutions obtained for ABC and ACO for (a) n15 (b) n25 (c) n33 (d) n40 (e) n50

6.2 Concluding Remarks

In this chapter, a mutual comparison of ant colony optimization ACO and artificial bee colony ABC was performed and presented. The performance of both algorithms was investigated. First, the results obtained previously in Chapter 5 were used for initial comparison. Then, to make a fair comparison both algorithms were run for the same time interval while setting the control parameters to its optimal values. The comparison revealed that, the

performance of ABC algorithm was better than that of ACO algorithm for larger test cases (i.e. *n25*, *n33*, *n40* and *n50*). While ACO performed better for smaller test cases (i.e. *n15*). A logical reason might be that, in smaller problem dimensions as *n15*, the construction of a whole new solution is more affordable and less expensive than generating a mutant solution. Moreover, ABC uses less control parameters and it can be efficiently tailored for solving larger test cases in less computational time.

Chapter Seven
Conclusions and Future Work

7.1 Conclusion.	94
7.2 Future Work.	96

Chapter Seven

Conclusions and Future Work

This thesis addressed the topological design problem of distributed local area networks. The problem was modelled as a multi-objective optimization problem considering five design objectives which were, network reliability, network availability, average link utilization, average network delay and cost. The main objective of this thesis was to design and analyze the artificial bee colony algorithm (ABC) to address the DLAN topology design problem. This objective was accomplished by adapting the ABC algorithm to suit the problem under consideration. Another objective was to address the multi-objective nature of the problem and this was accomplished by using goal programming as a multi-objective optimization approach to combine the five individual objectives into a single objective function. The performance of the proposed algorithm was also compared to another algorithm, namely, Ant colony optimization, which is also a population based optimization algorithm (ACO). ACO was chosen for this comparison and again was engineered to address the problem considered using goal programming as multi-objective optimization approach.

7.1 Conclusion

This thesis discussed the problem of DLAN topology design with sufficient details in Chapter 3. All the necessary information was provided including the assumptions, constraints, and the design objectives. The fuzzy goal programming approach was also proposed and discussed for the purpose of aggregation of multiple design objectives into a single overall objective function.

A goal programming based multi-objective artificial bee colony

optimization algorithm for the DLAN topology design problem was proposed. The algorithm performance was examined and evaluated. Since the limit value and the colony size are important parameters that play a significant role in determining the algorithm performance, empirical analysis was conducted to study the effect of these parameters. As far as limit was concerned, better results were obtained with moderate limit values. A very low and very high values of the limit produced the worst results. However, an exception to be considered was for larger problem dimensions. For relatively larger problem dimensions and with sufficient diversity in initial population, the higher values of the limit produced the best results. As for the colony size, increasing the colony size had a positive impact on the results obtained. Yet, the improvement in solutions quality was not very significant with small increments in colony size.

Furthermore, a goal programming based multi-objective ant colony optimization algorithm (ACO) was proposed and discussed. The performance of the proposed algorithm was investigated and analyzed. Empirical analysis were done to study the effect of control parameters, namely, pheromone deposit and evaporation rate, heuristic values, and colony size. With respect to pheromone deposit and evaporation rate, better results were obtained when the difference in pheromone deposit and evaporation rates was low. With respect to heuristic values, generally better results were obtained when the search was balanced between exploration and exploitation process. With respect to colony size, it was found that better results were obtained with larger colony sizes. Finally, an overall comparison of the ant colony optimization algorithm and the artificial bee algorithm was conducted. The comparison revealed that, generally, ACO performed better for relatively smaller problem dimensions, while ABC had a better performance for medium and larger problem dimensions in less computational time.

7.2 Future Work

Some potential directions for future research are presented below.

- **Efficient Approach for Dynamic Parameters Setting**

  Chapter 5 investigated the effect of varying control parameters for ACO and ABC. The parameters in ACO and ABC were statically assigned in advance. More research is also needed to develop mechanisms to dynamically adjust the parameter values during the run time.

- **Enhanced ABC Algorithm**

  Several advances have been proposed for the original ABC to enhance the algorithm performance. For example, (Mezura-Montes & Velez-Koeppel 2010) proposed the Elitist Artificial Bee Colony (EABC) and proved its efficiency over the original ABC. Other enhancement of the algorithm may include enhancing the local search operator. The proposed local search operator borrows the mutation phase from genetic algorithm. A research is needed on varying mutation rates and its effect on exploitation process of ABC algorithm. Also, in the proposed ABC, the number of employed bees was considered to be equal to that of onlooker bees and equal to the number of food sources. Further research is required to study possibilities of having various number of employed and onlookers and to specify the number each of them dynamically during the run time.

- **Exploring Hybridization Schemes**

  Several researches suggested that hybridizing heuristics with each other resulted in better solutions. Results in this thesis showed that ACO has a powerful global search abilities while ABC has efficient local search abilities. Incorporating features of one algorithm into another to form an enhanced

algorithm is a recommended future work. Moreover, other heuristics to be partially or completely incorporated with ACO and/or ABC is yet to be explored.

- **Other Aggregation Techniques**

  In this thesis, the multi-objective nature of the DLAN topology design problem was addressed using goal programming. However, other multi-objective optimization approaches presented in Chapter 3 can also be explored and employed.

- **Application of the ABC to Other Multi-Objective Problems**

  The experimental study of ACO and ABC showed that ABC outperformed ACO for medium and large problem dimensions while ACO performed better with relatively smaller problem dimension. Consequently, the application of ACO and ABC to other multi-objective optimization problems should also be studied to verify this behavior.

- **Application of Other Techniques to the Problem of DLAN Topology Design Problem**

  Other algorithms such as artificial immune system (AIS), Glowworm swarm optimization (GSO), teaching-learning-based optimization (TLBO) algorithm, gravitational search algorithm (GSA), Bat algorithm (BA), shuffled frog leaping (SFL), scatter search (SS), firefly algorithm (FA), bird flocking, Fish School Search (FSS), Cell mass, Gen strings, memetic algorithms (MAs) should also be explored. These algorithms have been applied to a number of optimization problems. A suggestion is to design these algorithms to the DLAN topology design problem and a comparison study can be conducted with algorithms applied in this thesis.

- **Other Extensions for the Problem of Topology Design Problem** The multi-objective DLAN topology design introduced earlier in Chapter 3 considered five design objectives. The problem can extended to involve other design objectives such as network survivability, and network sustainability. Also, the maximum problem dimension considered was 50 nodes. Another plan is to consider larger problem dimensions and study the effect of problem dimension as a determinant factor.

References

Akay, B. & Karaboga, D. (2009), Parameter tuning for the artificial bee colony algorithm, *in* 'Computational collective intelligence. Semantic web, social networks and multiagent systems', Springer Berlin Heidelberg, pp. 608–619.

Akay, B. & Karaboga, D. (2015), 'A survey on the applications of artificial bee colony in signal, image, and video processing', *Signal, Image and Video Processing,* **9**(4), 967–990.

Aouni, B. & Kettani, O. (2001), 'Goal programming model: A glorious history and a promising future', *European Journal of Operational Research,* **133**(2), 225–231.

Atiqullah, M. & Rao, S. (1993), 'Reliability Optimization of Communication Networks Using Simulated Annealing', *Microelectron Reliability,* **33**(9), 1303–1319.

Basturk, B. & Karaboga, D. (2006), An artificial bee colony (abc) algorithm for numeric function optimization, *in* 'IEEE swarm intelligence symposium', Vol. 8, pp. 687–697.

Behravan, H. (2012), 'Swarm intelligence/ant colonies through applications', *Computational Intelligence II, University of Estern Finland* .

Beni, G. & Wang, J. (1993), Swarm intelligence in cellular robotic systems, *in* 'Robots and Biological Systems: Towards a New Bionics?', Springer, pp. 703–712.

Bisht, N. & Singh, S. (2015), 'Analytical study of different network topologies', *International Research Journal of Engineering and Technology (IRJET),* **2**(1).

Blum, C. & Li, X. (2008), *Swarm intelligence in optimization*, Springer, Berlin Heidelberg.

Bolaji, A. L., Khader, A. T., Al-Betar, M. A. & Awadallah, M. A. (2013), 'Artificial bee colony algorithm, its variants and applications: A survey.', *Journal of Theoretical & Applied Information Technology* **47**(2).

Bonabeau, E., Dorigo, M. & Theraulaz, G. (1999), *Swarm intelligence: from natural to artificial systems*, Oxford university press.

Bonabeau, E., Sobkowski, A., Theraulaz, G. & Deneubourg, J.-L. (1997), Adaptive task allocation inspired by a model of division of labor in social insects., *in* 'BCEC', pp. 36–45.

Chanas, S. & Kuchta, D. (2002), 'Fuzzy goal programming-one notion, many meanings', *Control and Cybernetics* **31**(4), 871–890.

Charnes, A., Cooper, W. W. & Ferguson, R. O. (1955), 'Optimal estimation of executive compensation by linear programming', *Management science* **1**(2), 138–151.

Coello, C. A. C. (1998), 'A comprehensive survey of evolutionary-based multiobjective optimization techniques', *Knowledge and Information Systems* **1**(3), 269 – 308.

Colorni, A., Dorigo, M., Maniezzo, V. et al. (1991), Distributed optimization by ant colonies, *in* 'Proceedings of the first European conference on artificial life', Vol. 142, Paris, France, pp. 134–142.

Dearborn, R., Napolitan, R., Whitcomb, L. & Wilson, J. (2006), 'The costs of downtime: North american medium businesses 2006', *Infonetics Research Press Release* .

Dengiz, B. & Alabas, C. (2000), 'A tabu search algorithm for computer networks design', *Problems in Modern Applied Mathematics* **4**(2), 363–366.

Dengiz, B. & Alabas, C. (2001), A simulated annealing algorithm for design of computer communication networks, *in* 'World Multiconference on Systemics, Cybernetics, and Informatics', Vol. 5, pp. 188–193.

Dengiz, B., Altiparmak, F. & Smith, A. (1997), 'Local Search Genetic Algorithm for Optimal Design of Reliable Network', *IEEE Transactions on Evolutionary Computation* pp. 179–188.

Dorigo, M. (1992), 'Optimization, learning and natural algorithms', *Ph. D. Thesis, Politecnico di Milano, Italy* .

Dorigo, M. (2007), 'Ant colony optimization', *Scholarpedia* **2**(3), 1461.

Dorigo, M., Birattari, M. & Stützle, T. (2006), 'Ant colony optimization', *Computational Intelligence Magazine, IEEE* **1**(4), 28–39.

Dorigo, M. & Gambardella, L. M. (1997), 'Ant colony system: a cooperative learning approach to the traveling salesman problem', *Evolutionary Computation, IEEE Transactions on* **1**(1), 53–66.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996), 'Ant system: optimization by a colony of cooperating agents', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **26**(1), 29–41.

Dorigo, M. & Stützle, T. (2004), *Ant Colony Optimization*, Massachusetts Institute of Technology,Cambridge, MA.

Dorigo, M. & Stützle, T. (2003), The ant colony optimization metaheuristic: Algorithms, applications, and advances, *in* 'Handbook of metaheuristics', Springer, pp. 250–285.

Dyer, F. C. (2002), 'The biology of the dance language', *Annual review of entomology* **47**(1), 917–949.

Elbaum, R. & Sidi, M. (1996), 'Topological design of local-area networks using genetic algorithms', *IEEE/ACM Transactions on Networking (TON)* **4**(5), 766–778.

Ersoy, C. & Panwar, S. (1993), 'Topological Design of Interconnected LAN/MAN Networks', *IEEE Journal on Selected Area in Communications* pp. 1172–1182.

Farshidpour, S. & Keynia, F. (2012), 'Using artificial bee colony algorithm for mlp training on software defect prediction', *Oriental Journal of Computer science & Technology* **5**(2).

Fencl, T., Burget, P. & Bilek, J. (2011), 'Network topology design', *Control Engineering Practice* **19**(11), 1287–1296.

Fetterolf, P. (1990), Design of Data Networks with Spanning Tree Bridges, *in* 'IEEE International Conference on Systems, Man, and Cybernetics', pp. 298–300.

Fielding, R., Meeks, G. & Nair, P. (2000), 'Wired versus wireless', *OECD publishing* (2000/08), 4.

Frank, H., Frisch, I. T. & Chou, W. (1970), Topological considerations in the design of the arpa computer network, *in* 'Proceedings of the May 5-7, 1970, spring joint computer conference', ACM, pp. 581–587.

Garey, M. & Johnson, D. (1979), *Computers and intractability: a guide to the theory of NP-completeness*, San Francisco: Freeman.

Gembicki, F. (1974), Vector optimization for control with performance and parameter sensitivity indices, PhD thesis, Ph. D. Thesis, Case Western Reserve Univ., Cleveland, Ohio.

Gen, M., Ida, K. & Kim, J. (1998), A Spanning Tree-Based Genetic Algorithm for Bicriteria Topological Network Design, *in* 'IEEE International Conference on Evolutionary Computation', pp. 164–173.

Gerla, M. & Kleinrock, L. (1977), 'On the topological design of distributed computer networks', *Communications, IEEE Transactions on* **25**(1), 48–60.

Ghasemi, A., Masnadi-shirazi, M., Biguesh, M., Jahromi, A. F. & Ghasemi, F. (2012), Spectrum allocation based on artificial bee colony in cognitive radio networks, *in* 'Telecommunications (IST), 2012 Sixth International Symposium on', IEEE, pp. 182–187.

Goss, S., Aron, S., Deneubourg, J.-L. & Pasteels, J. M. (1989), 'Self-organized shortcuts in the argentine ant', *Naturwissenschaften* **76**(12), 579–581.

Gozde, H., Taplamacioglu, M. C. & Kocaarslan, İ. (2012), 'Comparative performance analysis of artificial bee colony algorithm in automatic generation control for interconnected reheat thermal power system', *International Journal of Electrical Power & Energy Systems* **42**(1), 167–178.

Güneş, M. & Umarosman, N. (2005), 'Fuzzy goal programming approach on computation of the fuzzy arithmetic mean', *Mathematical and Computational applications* **10**(2), 211–220.

Habib, S. J. (2008), 'Redesigning network topology with technology considerations', *International Journal of Network Management* **18**(1), 1–13.

Hacki, R. & Lighton, J. (2001), 'The future of the networked company', *The McKinsey Quarterly* p. 26.

Hannan, E. L. (1981), 'On fuzzy goal programming*', *Decision Sciences* **12**(3), 522–531.

Harmatos, J., Szentes, A. & Godor, I. (2000), Planning of Tree-topology UMTS Terrestrial Access Networks, *in* 'Proceedings of

the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Vol. 1 ', pp. 353 – 357.

Hole, G. (2011), 'Graham hole research skills', **1**. Last accessed: 2015-02-12.
URL: *http://users.sussex.ac.uk/ grahamh/RM1web/WilcoxonHandoout2011*

Igai, K. & Oki, E. (2011), 'A simple estimation scheme for upper bound of link utilization based on rtt measurement', *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)* pp. 10–16.

Jasem, H. N., Zukarnain, Z. A., Othman, M. & Subramaniam, S. (2010*a*), 'Evaluation study for delay and link utilization with the new-additive increase multiplicative decrease congestion avoidance and control algorithm', *arXiv preprint arXiv:1001.2848* .

Jasem, H. N., Zukarnain, Z. A., Othman, M. & Subramaniam, S. (2010*b*), 'On the delay and link utilization with the new-additive increase multiplicative decrease congestion avoidance and control algorithm', *Scientific Research and Essays* **5**(23), 3719–3729.

Jereb, L. (1998), Network reliability: models, measures and analysis, *in* 'Proceedings of the 6th IFIP Workshop on Performance Modelling and Evaluation of ATM Networks, Tutorial Papers, Ilkley, UK', p. T02.

Ji, J., Wei, H., Liu, C. & Yin, B. (2013), 'Artificial bee colony algorithm merged with pheromone communication mechanism for the 0-1 multidimensional knapsack problem', *Mathematical Problems in Engineering* **8**.

Johnsen, E. (2012), 'A. charnes and ww cooper: Management models and industrial applications of linear programming, vol. i. and vol. ii, ialt 859 sider. john wiley and sons, inc. london 1961. lait 183 sh/-.', *Ledelse og Erhvervsøkonomi* **26**.

Jones, D. & Tamiz, M. (2010), Goal programming variants, *in* 'Practical Goal Programming', Springer, pp. 11–22.

Juan, A. A., Marquès, J. M., Ionescu, D. & Faulin, J. (2010), Reliability and availability issues in large-scale distributed systems, *in* 'Proceedings of the Winter Simulation Conference', Winter Simulation Conference, pp. 2915–2926.

Kampstra, P. (2005), 'Evolutionary computing in telecommunications, a likely ec success story', *BMI paper, August* .

Karaboga, D. (2005), An idea based on honey bee swarm for numerical optimization, Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Karaboga, D. (2010), 'Artificial bee colony algorithm', **5**(3), 6915.

Karaboga, D. & Akay, B. (2009*a*), 'A comparative study of artificial bee colony algorithm', *Applied Mathematics and Computation* **214**(1), 108–132.

Karaboga, D. & Akay, B. (2009*b*), 'A survey: algorithms simulating bee swarm intelligence', *Artificial Intelligence Review* **31**(1-4), 61–85.

Karaboga, D. & Akay, B. (2011), 'A modified artificial bee colony (abc) algorithm for constrained optimization problems', *Applied Soft Computing* **11**(3), 3021–3031.

Karaboga, D. & Basturk, B. (2007*a*), Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems, *in* 'Foundations of Fuzzy Logic and Soft Computing', Springer, pp. 789–798.

Karaboga, D. & Basturk, B. (2007*b*), 'A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm', *Journal of global optimization* **39**(3), 459–471.

Karaboga, D. & Basturk, B. (2008), 'On the performance of artificial bee colony (abc) algorithm', *Applied soft computing* **8**(1), 687–697.

Karaboga, D., Gorkemli, B., Ozturk, C. & Karaboga, N. (2012), 'A comprehensive survey: artificial bee colony (abc) algorithm and applications', *Artificial Intelligence Review* pp. 1–37.

Karaboga, D., Gorkemli, B., Ozturk, C. & Karaboga, N. (2014), 'A comprehensive survey: artificial bee colony (abc) algorithm and applications', *Artificial Intelligence Review* **42**(1), 21–57.

Karaboga, D., Okdem, S. & Ozturk, C. (2012), 'Cluster based wireless sensor network routing using artificial bee colony algorithm', *Wireless Networks* **18**(7), 847–860.

Khan, S. A. (2009), Design and analysis of evolutionary and

swarm intelligence techniques for topology design of distributed local area networks, PhD thesis, University of Pretoria.

Khan, S. A. (2014), 'A goal programming based extremal optimization algorithm for topology design of enterprise networks', *International Journal of Computing and Digital Systems* (2), 123–132.

Khan, S. A. & Engelbrecht, A. P. (2007), 'A new fuzzy operator and its application to topology design of distributed local area networks', *Information Sciences* **177**(13), 2692–2711.

Khan, S. A. & Engelbrecht, A. P. (2009), ' Fuzzy Hybrid Simulated Annealing Algorithms for Topology Design of Switched Local Area Networks', *Soft Computing* **3**(1), 45–61.

Khan, S. A. & Engelbrecht, A. P. (2011), Assessment of the "Evaluation" function in the simulated evolution algorithm, *in* 'IEEE 7th International Conference on Natural Computation', pp. 1062–1066.

Khan, S. A. & Engelbrecht, A. P. (2012*a*), 'A fuzzy particle swarm optimization algorithm for computer communication network topology design', *Applied Intelligence* **36**, 161–177.

Khan, S. A. & Engelbrecht, A. P. (2012*b*), 'A fuzzy particle swarm optimization algorithm for computer communication network topology design', *Applied Intelligence* **36**(1), 161–177.

Khan, S. A. & Engelbrecht, A. P. (2012*c*), 'A fuzzy particle swarm optimization algorithm for computer communication network topology design', *Applied Intelligence* **36**(1), 161–177.

Khan, S., Baig, Z. et al. (2007), A simulated evolution-tabu search hybrid metaheuristic for routing in computer networks, *in* 'Evolutionary Computation, 2007. CEC 2007. IEEE Congress on', IEEE, pp. 3818–3823.

Khan, S., Engelbrecht, A. P. et al. (2008), A fuzzy ant colony optimization algorithm for topology design of distributed local area networks, *in* 'Swarm Intelligence Symposium, 2008. SIS 2008. IEEE', IEEE, pp. 1–7.

Kiran, M. S. & Gündüz, M. (2014), 'The analysis of peculiar control parameters of artificial bee colony algorithm on the numerical optimization problems', *Journal of Computer and Communications,* **2**(04), 127.

Krause, J., Cordeiro, J., Parpinelli, R. S. & Lopes, H. S. (2013), 'A survey of swarm algorithms applied to discrete optimization problems', *Swarm Intelligence and Bio-inspired Computation: Theory and Applications. Elsevier Science & Technology Books* pp. 169–191.

Kumar, R. & Banerjee, N. (2003), Multicriteria network design using evolutionary algorithm, *in* 'Genetic and Evolutionary ComputationŮGECCO 2003', Springer, pp. 2179–2190.

Kumar, R., Parida, P. P. & Gupta, M. (2002), Topological design of communication networks using multiobjective genetic optimization, *in* 'wcci', IEEE, pp. 425–430.

Li, H., Liu, K. & Li, X. (2010), A comparative study of artificial bee colony, bees algorithms and differential evolution

on numerical benchmark problems, *in* 'Computational Intelligence and Intelligent Systems', Springer, pp. 198–207.

Li, J.-Q., Pan, Q.-K. & Gao, K.-Z. (2011), 'Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems', *The International Journal of Advanced Manufacturing Technology* **55**(9-12), 1159–1169.

Mala, D. J., Kamalapriya, M., Shobana, R. & Mohan, V. (2009), A non-pheromone based intelligent swarm optimization technique in software test suite optimization, *in* 'Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009. International Conference on', IEEE, pp. 1–5.

McNurlin, B. C., Sprague, R. H. & Bui, T. X. (1989), *Information systems management in practice*, Prentice-Hall International.

Medhi, D. (1999), 'Network reliability and fault-tolerance', *Wiley Encyclopedia of Electrical and Electronics Engineering* .

Mekidiche, M. & Belmokaddem, M. (2012), 'Application of weighted additive fuzzy goal programming approach to quality control system design', *International Journal of Intelligent Systems and Applications (IJISA)* **4**(11), 14.

Mekidiche, M., Belmokaddem, M. & Djemmaa, Z. (2013), 'Weighted additive fuzzy goal programming approach to aggregate production planning', *International Journal of Intelligent Systems and Applications (IJISA)* **5**(4), 20.

Mezura-Montes, E. & Velez-Koeppel, R. E. (2010), Elitist artificial bee colony for constrained real-parameter optimization,

in 'Evolutionary Computation (CEC), 2010 IEEE Congress on', IEEE, pp. 1–8.

Milan, T. (n.d.), 'Artificial bee colony (abc) algorithm with crossover and mutation', *Advances In Computer Science* .

Millonas, M. M. (1994), Swarms, phase transitions, and collective intelligence, *in* 'SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS', Vol. 17, ADDISON-WESLEY PUBLISHING CO, pp. 417–417.

Miyoshi, T., Shimizu, S. & Tanaka, Y. (2002), Fast Topological Design with Simulated Annealing for Multicast Networks, *in* '7th International Symposium on Computers and Communications', pp. 959–966.

Mohan, B. C. & Baskaran, R. (2012), 'A survey: Ant colony optimization based recent research and implementation on several engineering domain', *Expert Systems with Applications* **39**(4), 4618–4627.

Mohiuddin, M. A., Khan, S. A. & Engelbrecht, A. P. (2014), 'Simulated evolution and simulated annealing algorithms for solving multi-objective open shortest path first weight setting problem', *Applied Intelligence* **41**(2), 348–365.

Narasimhan, R. (1980), 'Goal programming in a fuzzy environment', *Decision sciences* **11**(2), 325–336.

Ozturk, C., Hancer, E. & Karaboga, D. (2015), 'Dynamic clustering with improved binary artificial bee colony algorithm', *Applied Soft Computing* **28**, 69–80.

Ozturk, C. & Karaboga, D. (2011), Hybrid artificial bee colony algorithm for neural network training, *in* 'Evolutionary Computation (CEC), 2011 IEEE Congress on', IEEE, pp. 84–88.

Pal, S. K. & Sarma, S. S. (2010), 'Computer network topology design in limelight of pascal graph property', *arXiv preprint arXiv:1003.5432* .

Pan, Q.-K., Tasgetiren, M. F., Suganthan, P. N. & Chua, T. J. (2011), 'A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem', *Information sciences* **181**(12), 2455–2468.

Pandya, K. (2013), 'Network structure or topology', *International Journal* **1**(2).

Pierre, S., Hyppolite, M.-A., Bourjolly, J.-M. & Dioume, O. (1995), 'Topological design of computer communication networks using simulated annealing', *Engineering Applications of Artificial Intelligence* **8**(1), 61–69.

Pierre, S. & Legault, G. (1998), 'A Genetic Algorithm for Designing Distributed Computer Network Topologies', *IEEE Transactions on Systems, Man, Cybernetics* **28**(2), 249–258.

Pucha, H., Zhang, Y., Mao, Z. M. & Hu, Y. C. (2007), Understanding network delay changes caused by routing events, *in* 'ACM SIGMETRICS Performance Evaluation Review', Vol. 35, ACM, pp. 73–84.

Quan, X. & Li, H. (2014), 'Computer simulation-based network

topology genetic algorithm reliability study', *Journal of Chemical & Pharmaceutical Research* **6**(7).

Romero, C. (2004), 'A general structure of achievement function for a goal programming model', *European Journal of Operational Research* **153**(3), 675–686.

Şahin, A. Ş., Kılıç, B. & Kılıç, U. (2011), 'Design and economic optimization of shell and tube heat exchangers using artificial bee colony (abc) algorithm', *Energy Conversion and Management* **52**(11), 3356–3362.

Santra, S. & Acharjya, P. P. (2013), 'A study and analysis on computer network topology for data communication', *International Journal of Emerging Technology and Advanced Engineering* **3**(1).

Seeley, T. D. (2009), *The wisdom of the hive: the social physiology of honey bee colonies*, Harvard University Press.

Seeley, T. D. & Visscher, P. K. (1988), 'Assessing the benefits of cooperation in honeybee foraging: search costs, forage quality, and competitive ability', *Behavioral Ecology and Sociobiology* **22**(4), 229–237.

Selvi, V. & Umarani, D. R. (2010), 'Comparative analysis of ant colony and particle swarm optimization techniques', *International Journal of Computer Applications (0975–8887)* **5**(4).

Shakkottai, S., Shakkottai, S. G. & Srikant, R. (2008), *Network optimization and control*, Now Publishers Inc.

Sportack, M. A. & Fairweather, J. (1999), *IP routing fundamentals*, Cisco Press.

Srivatsa, S. et al. (2009), 'Topological design of minimum cost survivable computer communication networks, bipartite graph method', *arXiv preprint arXiv:0908.1033* .

Stanojević, R. & Shorten, R. (2007), How expensive is link utilization?, *in* 'Network Control and Optimization', Springer, pp. 54–64.

Steiglitz, K., Weiner, P. & Kleitman, D. (1969), 'The design of minimum-cost survivable networks', *Circuit Theory, IEEE Transactions on* **16**(4), 455–460.

Stuck, B. W. & Arthurs, E. (1984), *A computer and communications network performance analysis primer*, Prentice-Hall, Inc.

Stützle, T., López-Ibánez, M., Pellegrini, P., Maur, M., de Oca, M. M., Birattari, M. & Dorigo, M. (2012), Parameter adaptation in ant colony optimization, *in* 'Autonomous search', Springer, pp. 191–215.

Szeto, W., Wu, Y. & Ho, S. C. (2011), 'An artificial bee colony algorithm for the capacitated vehicle routing problem', *European Journal of Operational Research* **215**(1), 126–135.

Tereshko, V. (2000), Reaction-diffusion model of a honeybee colonyŠs foraging behaviour, *in* 'Parallel Problem Solving from Nature PPSN VI', Springer, pp. 807–816.

Tereshko, V. & Lee, T. (2002), 'How information-mapping patterns determine foraging behaviour of a honey bee colony', *Open Systems & Information Dynamics* **9**(02), 181–193.

Tereshko, V. & Loengarov, A. (2005*a*), 'Collective decision making in honey-bee foraging dynamics', *Computing and Information Systems* **9**(3), 1.

Tereshko, V. & Loengarov, A. (2005*b*), 'Collective decision making in honey-bee foraging dynamics', *Computing and Information Systems* **9**(3), 1.

Thompson, D. & Bilbro, G. (2000), 'Comparison of a Genetic Algorithm with a Simulated Annealing Algorithm for the Design of an ATM Network', *IEEE Communications Letters* **4**(8), 267–269.

Tornatore, M., Maier, G. A., Pattavina, A., Villa, M., Righetti, A., Clemente, R. & Martinelli, M. (2003), Availability optimization of static path-protected wdm networks, *in* 'Optical Fiber Communication Conference', Optical Society of America, p. FA5.

Vargas, E. & BluePrints, S. (2000), 'High availability fundamentals', *Sun Blueprints series* .

Wang, L., Zhou, G., Xu, Y., Wang, S. & Liu, M. (2012), 'An effective artificial bee colony algorithm for the flexible job-shop scheduling problem', *The International Journal of Advanced Manufacturing Technology* **60**(1-4), 303–315.

White, A., Mann, J. & Smith, G. (1999), 'Genetic Algorithms and

Network Ring Design', *Annals of Operations Research* **6**(1), 347–371.

Wu, B. & Fan, S.-h. (2011), Improved artificial bee colony algorithm with chaos, *in* 'Computer Science for Environmental Engineering and EcoInformatics', Springer, pp. 51–56.

Xu, Y., Fan, P. & Yuan, L. (2013), 'A simple and efficient artificial bee colony algorithm', *Mathematical Problems in Engineering* **9**.

Yaghoobi, M., Jones, D. F. & Tamiz, M. (2008), 'Weighted additive models for solving fuzzy goal programming problems', *Asia-Pacific Journal of Operational Research* **25**(05), 715–733.

Youssef, H., Sait, S. & Khan, S. (2002), 'Topology Design of Switched Enterprize Networks using a Fuzzy Simulated Evolution Algorithm', *Engineering Applications of Artificial Intelligence* **15**, 327–340.

Zadeh, L. (1963), 'Optimality and non-scalar-valued performance criteria', *Automatic Control, IEEE Transactions on* **8**(1), 59–60.

Zadeh, L. A. (1965), 'Fuzzy sets', *Information and control* **8**(3), 338–353.

Zhang, X., Zhang, X., Ho, S. & Fu, W. (2014), 'A modification of artificial bee colony algorithm applied to loudspeaker design problem', *Magnetics, IEEE Transactions on* **50**(2), 737–740.

Appendix A

Statistical Analysis-Wilcoxon ranked-sum test

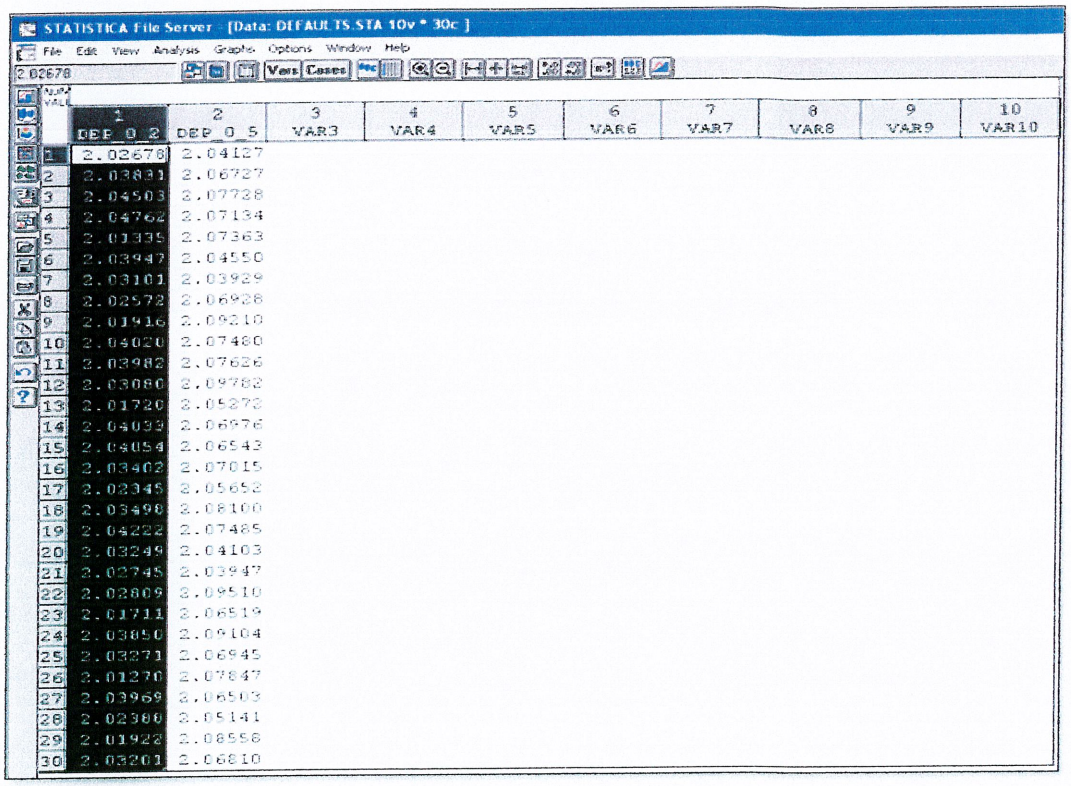
Best results of ACO for 30 runs for the case n15, DEP= 0.2, 20 Ants

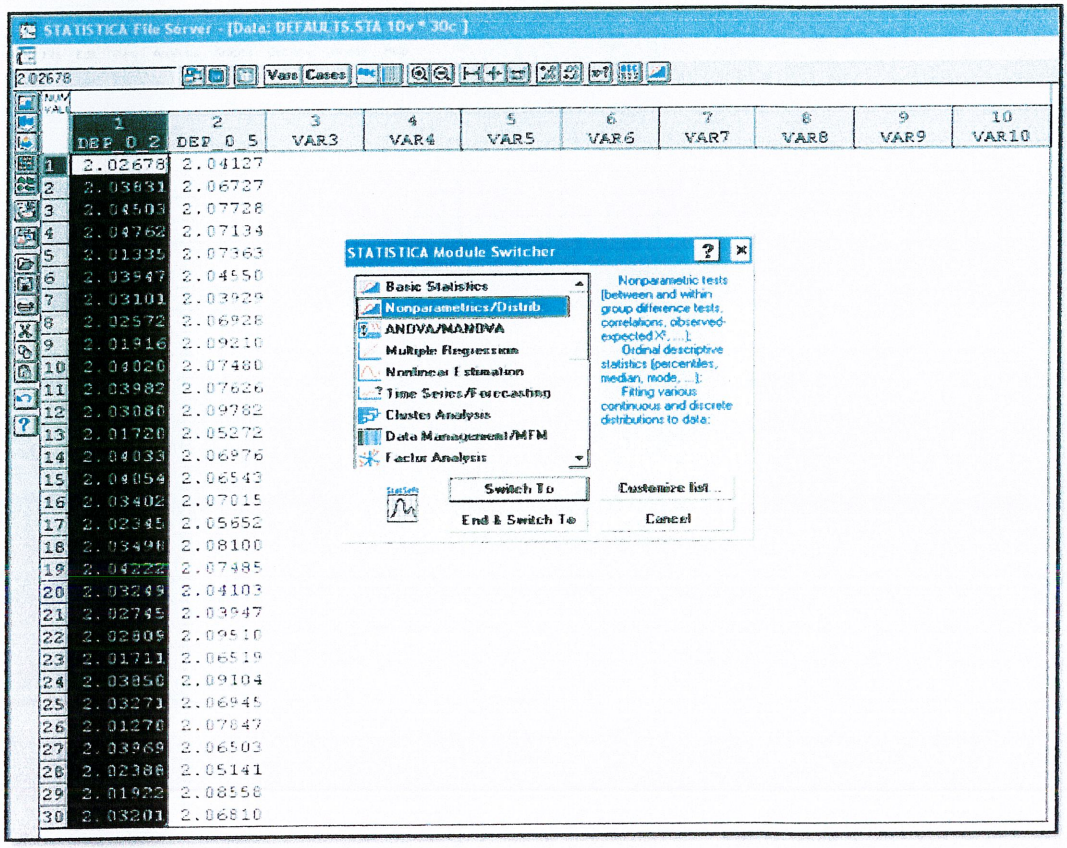
BoBSoLn	BoBCost	BOBReliability	BOBDelay	BOBAvailability	BOBUtilization	Run Time
2.02678	7080	0.531076	0.00364498	92.5	6.02429	7
2.03831	7580	0.531076	0.00395172	94.9	6.52929	6
2.04503	7650	0.501572	0.00408128	95.6	6.73571	5
2.04762	7040	0.501572	0.00382043	94.3	6.30857	5
2.01335	6850	0.56058	0.00391574	95.1	6.46357	5
2.03947	7310	0.509618	0.00350215	94.6	5.78786	5
2.03101	7070	0.591724	0.00446433	95	7.36786	6
2.02572	7280	0.616638	0.00448887	95.1	7.40429	5
2.01916	6920	0.591724	0.00426375	93.9	7.03786	5
2.0402	7630	0.591724	0.0037842	95.3	6.25143	5
2.03982	7210	0.551729	0.00374783	93.4	6.19	6
2.0308	6740	0.489028	0.00391942	92.9	6.46929	5
2.0172	6770	0.531076	0.00429081	95.2	7.07857	6
2.04033	7270	0.529437	0.00398404	94.9	6.57929	5
2.04054	7160	0.481306	0.0040356	94.9	6.66071	5
2.03402	7260	0.591724	0.00410815	96.1	6.78071	6
2.02345	6980	0.591724	0.0036776	94.3	6.07429	5
2.03498	7250	0.56058	0.00400408	93.9	6.60786	6
2.04222	7180	0.501572	0.00402862	94.1	6.65214	5
2.03249	6720	0.482796	0.00409267	94.3	6.75143	6
2.02745	6900	0.531076	0.00352324	96	5.825	6
2.02809	7460	0.591724	0.00373192	93.9	6.16214	5
2.01711	6490	0.56058	0.00417959	93.9	6.89571	5
2.0385	7590	0.56058	0.00412694	93.5	6.80714	6
2.03271	7280	0.529437	0.00398241	94.6	6.57643	7
2.0127	6560	0.537931	0.0040357	96	6.66214	6
2.03969	6970	0.472068	0.00370944	93.7	6.12786	7
2.02388	6590	0.509618	0.00413835	95.8	6.82786	6
2.01922	6620	0.624597	0.00375764	94.1	6.20786	6
2.03201	7020	0.487518	0.00413719	92.5	6.82714	6

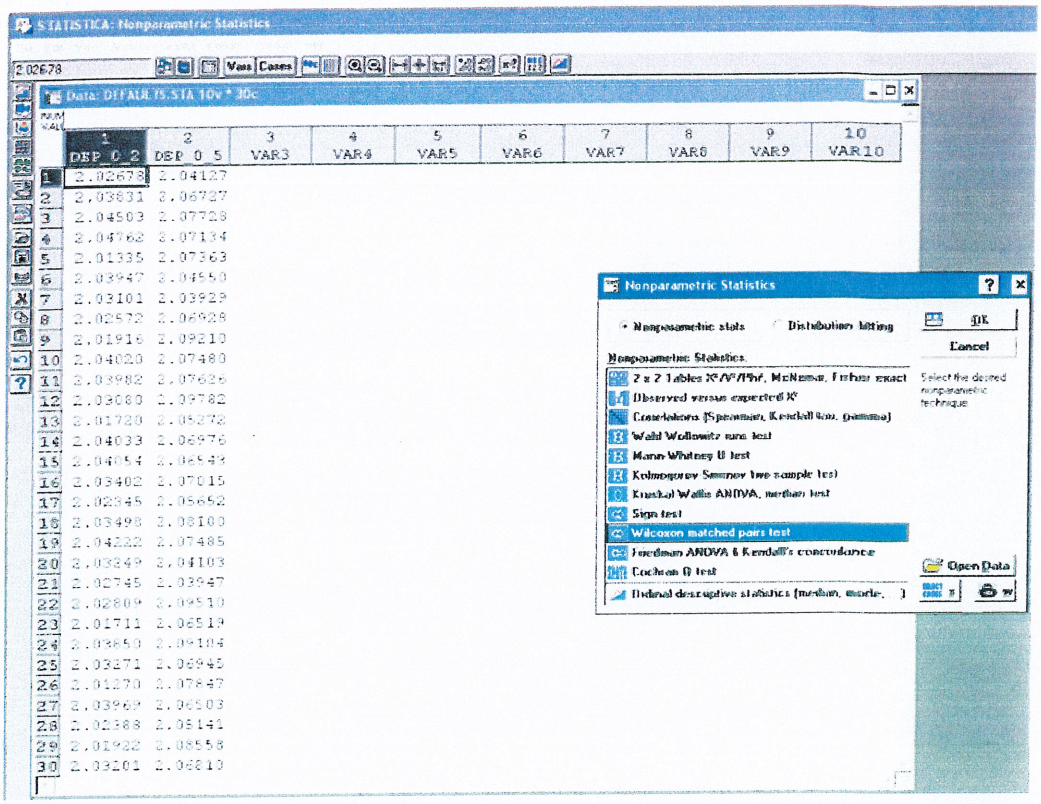
Best results of ACO for 30 runs for the case n15, DEP=0.5, 20Ants

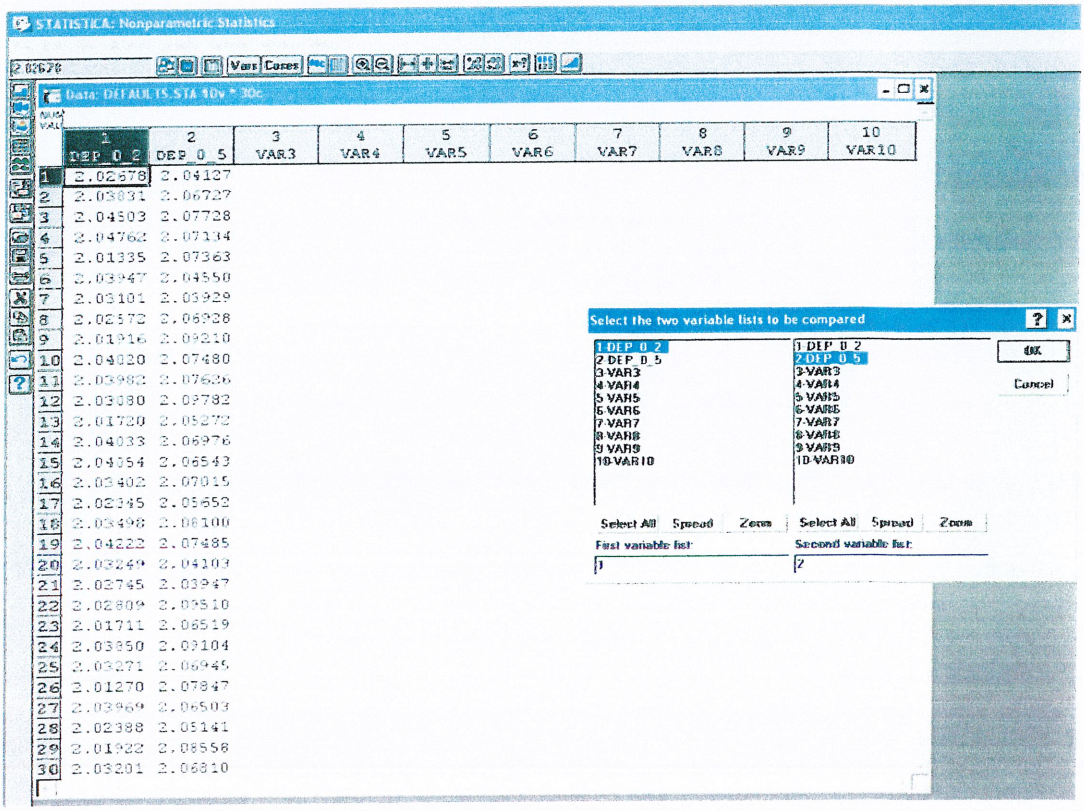
BoBSoLn	BoBCost	BOBReliability	BOBDelay	BOBAvailability	BOBUtilization	Run Time
2.04127	7440	0.56058	0.00389966	96.1	6.44	7
2.06727	7800	0.531076	0.00370512	94.1	6.12071	6
2.07728	8610	0.503125	0.00373004	92.6	6.16357	5
2.07134	7690	0.509618	0.00383665	95.2	6.335	5
2.07363	7780	0.46186	0.00377228	94.2	6.23357	5
2.0455	7140	0.509618	0.00344694	94	5.69786	5
2.03929	7500	0.584184	0.00414524	93.9	6.845	6
2.06928	8190	0.531076	0.00406875	92	6.71929	5
2.0921	8330	0.385312	0.00332408	93.2	5.49357	5
2.0748	7380	0.522691	0.00400293	93.8	6.61	5
2.07626	8420	0.529437	0.00351158	95.3	5.80714	6
2.09782	8550	0.551729	0.00459307	93.8	7.57786	5
2.05272	7410	0.584184	0.00433481	93	7.15286	6
2.06976	7740	0.501572	0.00396151	94.2	6.54429	5
2.06543	7330	0.482796	0.0036284	94.5	5.99571	5
2.07015	7950	0.475173	0.0038408	95.8	6.34357	6
2.05652	7940	0.582381	0.00379925	95.2	6.275	5
2.081	8240	0.455974	0.00399788	94.3	6.60071	6
2.07485	7960	0.584184	0.00344353	94.8	5.69571	5
2.04103	7270	0.553437	0.0042512	94.9	7.01357	6
2.03947	6940	0.489028	0.00382479	93.9	6.31286	6
2.0951	8740	0.56058	0.0039998	93	6.6	5
2.06519	7680	0.413243	0.00359016	94.7	5.935	5
2.09104	8010	0.473707	0.00395601	93.6	6.53214	6
2.06945	8090	0.642602	0.00385812	95.8	6.37143	7
2.07847	7530	0.406565	0.00380115	94	6.28357	6
2.06503	8140	0.473707	0.00420075	93.8	6.92857	7
2.05141	7640	0.413243	0.00402344	94.1	6.64143	6
2.08558	8470	0.495181	0.00403246	92.2	6.66071	6
2.0681	7510	0.553437	0.00366669	94.3	6.06	6

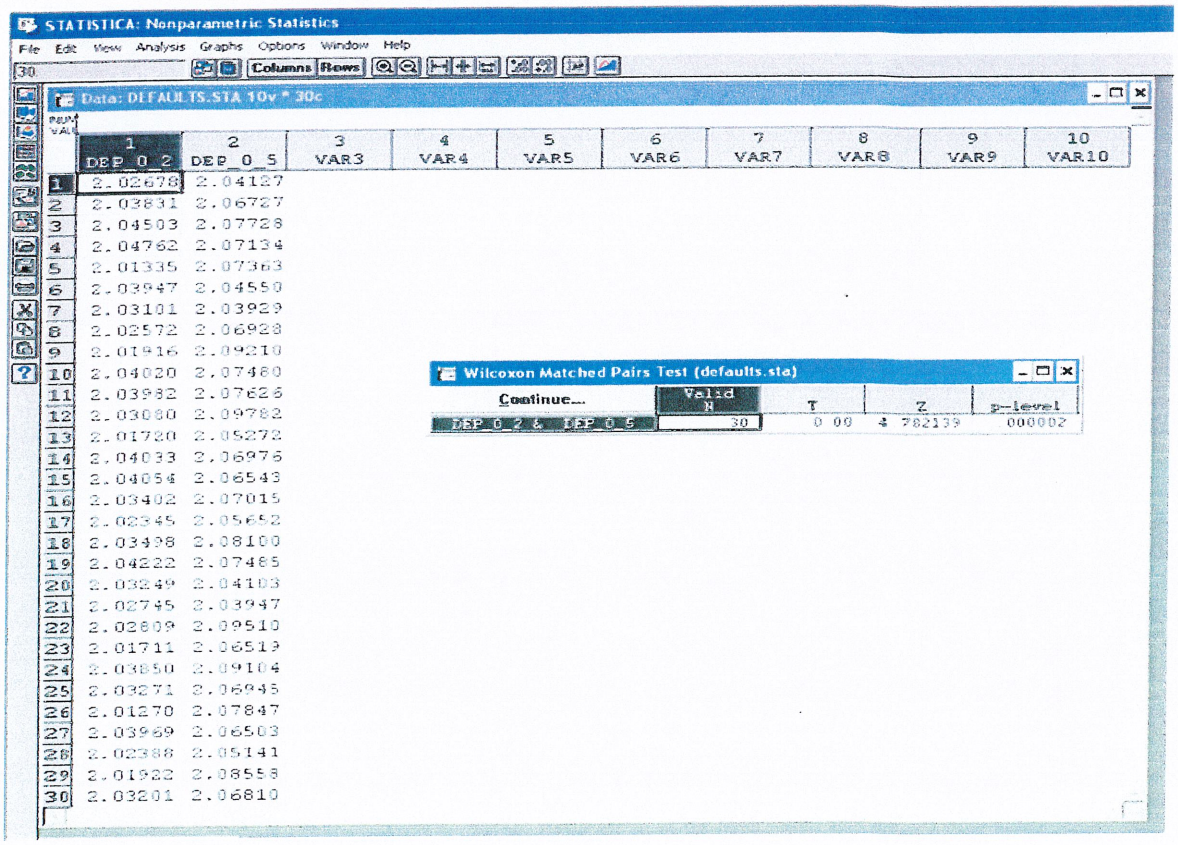
Statistical Analysis performed on previously presented ACO cases

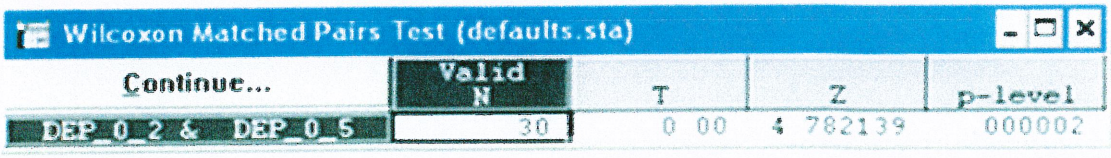





STATISTICA: Nonparametric Statistics
Data: DEFAULTS.STA 10v * 30c
Nonparametric Statistics
Nonparametric stats
Distribution fitting
OK
Cancel
Nonparametric Statistics
2 x 2 Tables (Fisher exact, McNemar)
Observed versus expected
Wald-Wolfowitz runs test
Mann-Whitney U test
Kolmogorov-Smirnov two-sample test
Kruskal-Wallis ANOVA, median test
Sign test
Wilcoxon matched pairs test
Friedman ANOVA & Kendall's concordance
Cochran Q test
Open Data


STATISTICA: Nonparametric Statistics
Data: DEFAULTS.STA 10v * 30c
Select the two variable lists to be compared
1 DEP_0_2
2 DEP_0_5
3 VAR3
4 VAR4
5 VAR5
6 VAR6
7 VAR7
8 VAR8
9 VAR9
10 VAR10
OK
Cancel
Select All
Spread
Zero
First variable list:
Second variable list:


STATISTICA: Nonparametric Statistics
File Edit View Analysis Graphs Options Window Help
Data: DEFAULTS.STA 10v * 30c
1 DEP_0_2
2 DEP_0_5
3 VAR3
4 VAR4
5 VAR5
6 VAR6
7 VAR7
8 VAR8
9 VAR9
10 VAR10
1 2.02678 2.04127
2 2.03831 2.06727
3 2.04503 2.07728
4 2.04762 2.07134
5 2.01335 2.07363
6 2.03947 2.04550
7 2.03101 2.03929
8 2.02572 2.06928
9 2.01916 2.09210
10 2.04020 2.07480
11 2.03982 2.07626
12 2.03080 2.09782
13 2.01720 2.05272
14 2.04033 2.06976
15 2.04054 2.06543
16 2.03402 2.07015
17 2.02345 2.05652
18 2.03498 2.08100
19 2.04222 2.07485
20 2.03249 2.04103
21 2.02745 2.03947
22 2.02809 2.09510
23 2.01711 2.06519
24 2.03850 2.09104
25 2.03271 2.06945
26 2.01270 2.07847
27 2.03969 2.06503
28 2.02388 2.05141
29 2.01922 2.08558
30 2.03201 2.06810
Wilcoxon Matched Pairs Test (defaults.sta)
Continue...
Valid N
T
Z
p-level
DEP_0_2 & DEP_0_5
30
0 00
4 782139
000002


Wilcoxon Matched Pairs Test (defaults.sta)
Continue...
Valid N
T
Z
p-level
DEP_0_2 & DEP_0_5
30
0 00
4 782139
000002

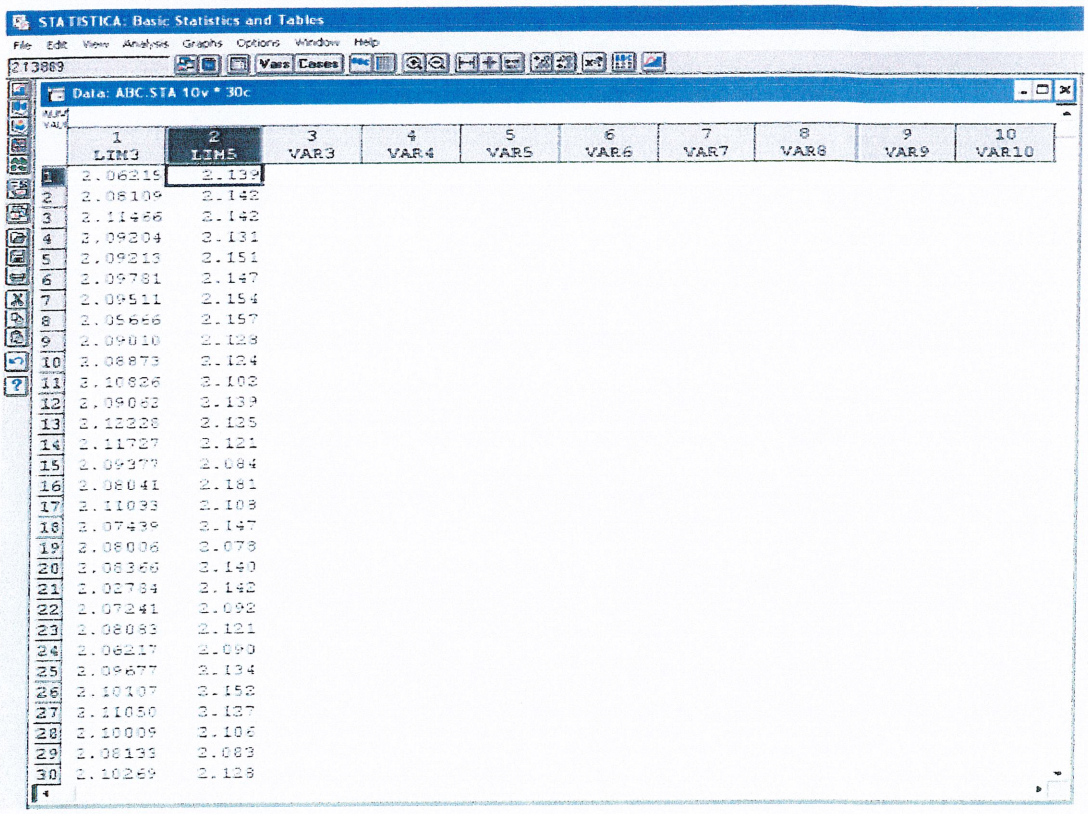
Best results of ABC for 30 runs for the case n15, Limit 3, Bees 20

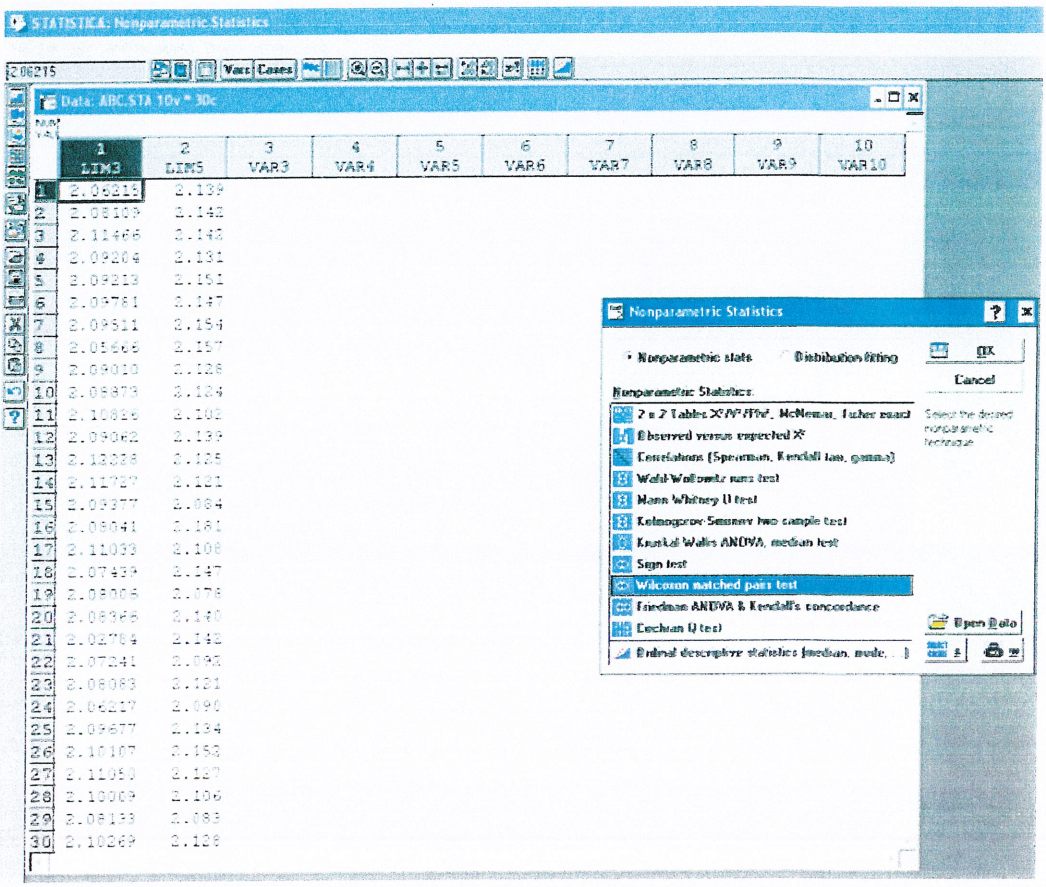
BoBSoLn	BoBCost	BOBReliability	BOBDelay	BOBAvailability	BOBUtilization	Run Time
2.06215	8610	0.267889	0.005806	94.1	9.56429	4
2.08109	8570	0.253086	0.005543	93	9.13143	5
2.11466	9890	0.40911	0.005623	93.3	9.25143	5
2.09204	9650	0.396546	0.005665	94.6	9.32429	6
2.09213	9320	0.286711	0.005354	95.1	8.82357	6
2.09781	9390	0.192023	0.006134	93	10.1029	6
2.09511	10090	0.177367	0.005247	94.3	8.65714	5
2.05666	8100	0.332903	0.005314	93.8	8.76071	6
2.0901	8780	0.246166	0.005433	93.3	8.94929	6
2.08873	8780	0.280339	0.004957	94.7	8.17071	5
2.10826	9780	0.354804	0.005408	94.6	8.89786	6
2.09062	8130	0.252385	0.004427	93.9	7.31571	6
2.12228	9290	0.302639	0.004207	95.5	6.94429	5
2.11727	11190	0.175593	0.007688	93.1	12.6364	5
2.09377	9170	0.258143	0.005189	95.1	8.55143	6
2.08041	9130	0.347002	0.00555	94.5	9.14071	5
2.11033	9550	0.288597	0.003944	92.7	6.51	5
2.07439	8470	0.245314	0.005368	93.9	8.84571	5
2.08006	10710	0.452994	0.004871	93.7	8.04286	4
2.08366	8220	0.286711	0.004464	94.6	7.36286	4
2.02784	7720	0.35814	0.005753	93.7	9.46071	4
2.07241	8030	0.25083	0.004987	93.6	8.22	4
2.08083	7830	0.228027	0.004565	93.5	7.53	4
2.06217	9090	0.35579	0.006254	93.9	10.28	4
2.09677	10440	0.231592	0.006966	94.2	11.4486	4
2.10107	10950	0.238345	0.003557	93.2	5.88571	4
2.1105	8950	0.374516	0.004163	95	6.87571	4
2.10009	8360	0.270782	0.004026	95	6.64286	4
2.08133	9310	0.345931	0.005652	94.8	9.29571	4
2.10269	8220	0.211354	0.003974	94.8	6.55857	4

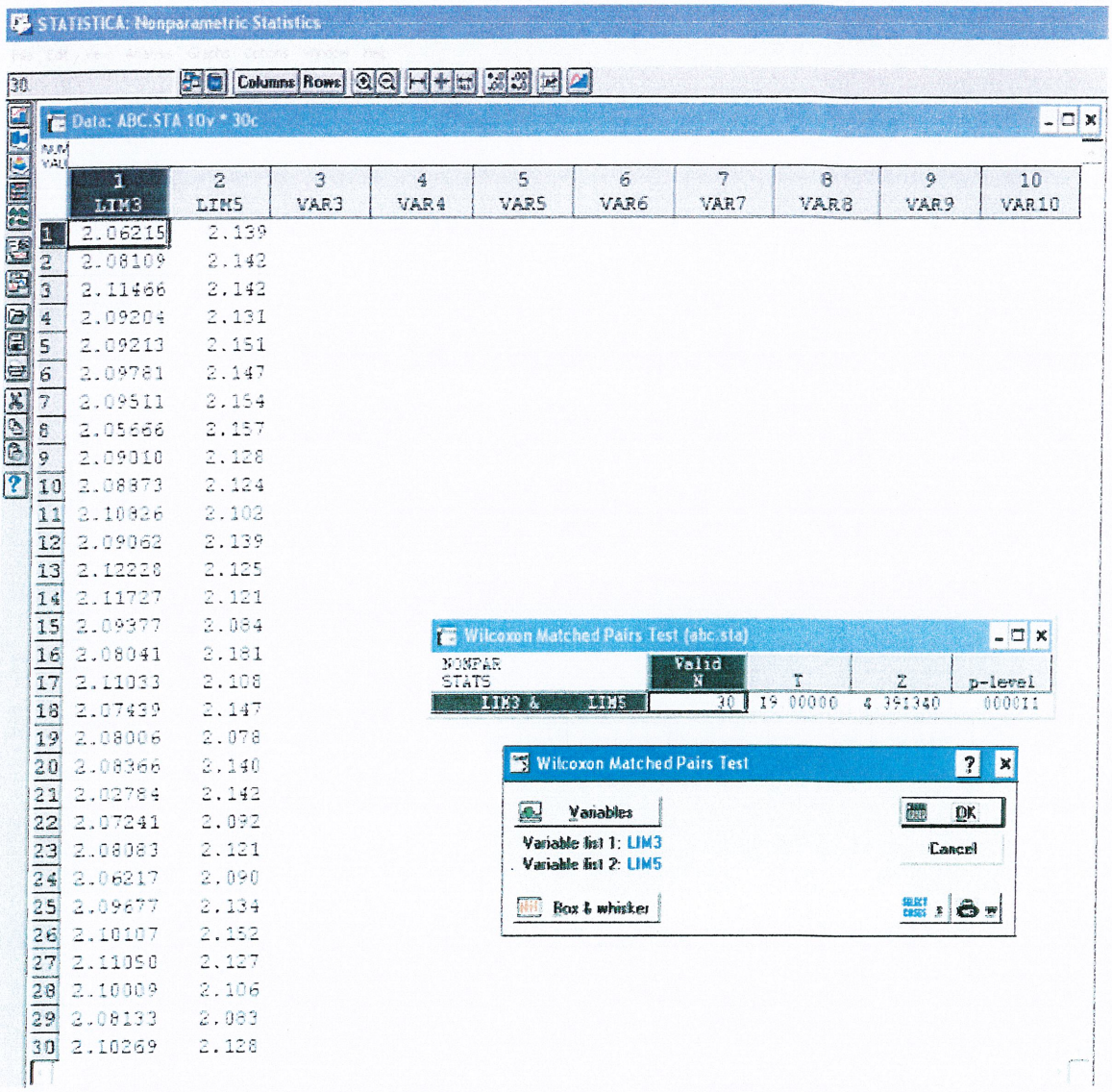
Best results of ABC for 30 runs for the case n15, Limit 5, bees 20

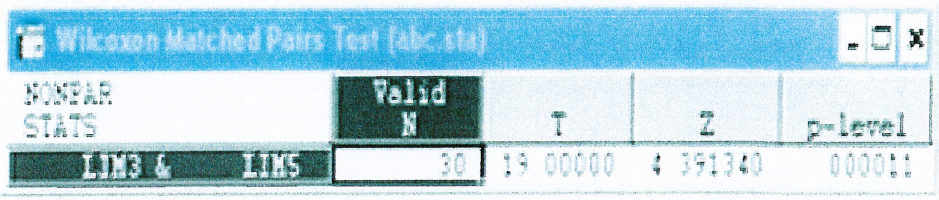
BoBSoLn	BoBCost	BOBReliability	BOBDelay	BOBAvailability	BOBUtilization	Run Time
2.13889	9800	0.350182	0.004847	93.4	7.99286	3
2.14169	11110	0.341041	0.005725	92.3	9.42571	3
2.14189	10070	0.203172	0.005378	93.5	8.86286	2
2.13096	10430	0.166232	0.006492	92.7	10.6821	3
2.15091	11110	0.141791	0.006986	91.9	11.4893	3
2.14686	10270	0.245138	0.005649	92.6	9.30643	3
2.15374	12320	0.333694	0.007578	93	12.4336	4
2.15688	9210	0.147003	0.004224	92.8	6.97429	2
2.12837	10940	0.158583	0.006038	93.1	9.93429	3
2.12354	10480	0.198091	0.007027	91.7	11.56	3
2.10246	10770	0.220013	0.00796	91.8	13.0821	2
2.13949	12610	0.101563	0.005542	91.2	9.14429	1
2.12476	9960	0.295582	0.005794	92.8	9.54786	3
2.12092	10030	0.333694	0.005823	93.1	9.58643	2
2.08412	9630	0.217894	0.006679	93.4	10.9771	3
2.18141	10730	0.272375	0.004829	93.3	7.96357	3
2.10826	9780	0.354804	0.005408	94.6	8.89786	2
2.14672	11920	0.206944	0.00559	90.9	9.21286	3
2.07773	9230	0.26137	0.006293	93.4	10.3607	2
2.14026	10150	0.307273	0.005063	94.2	8.355	2
2.14228	10420	0.244459	0.006233	91.8	10.2521	1
2.09155	10390	0.243529	0.006919	94.6	11.3693	3
2.12089	10500	0.203878	0.006682	93.1	10.9921	3
2.09044	10490	0.303358	0.007249	93.6	11.915	2
2.1336	10880	0.219855	0.007007	92.1	11.5371	1
2.1519	11870	0.396546	0.003585	93.7	5.92214	3
2.1265	9750	0.227936	0.005492	93.3	9.04071	2
2.10607	9900	0.182422	0.006221	93.9	10.2479	3
2.08266	9780	0.307371	0.006948	92.6	11.4257	2
2.12791	9890	0.314382	0.005637	92.6	9.28143	3

Statistical Analysis performed on previously presented ABC case






STATISTICA: Nonparametric Statistics
Columns
Rows
Data: ABC.STA 10v * 30c
1 LIM3
2 LIM5
3 VAR3
4 VAR4
5 VAR5
6 VAR6
7 VAR7
8 VAR8
9 VAR9
10 VAR10
1 2.06215 2.139
2 2.08109 2.142
3 2.11466 2.142
4 2.09204 2.131
5 2.09213 2.151
6 2.09781 2.147
7 2.09511 2.154
8 2.05666 2.157
9 2.09010 2.128
10 2.08873 2.124
11 2.10826 2.102
12 2.09062 2.139
13 2.12220 2.125
14 2.11727 2.121
15 2.09377 2.084
16 2.08041 2.181
17 2.11033 2.108
18 2.07439 2.147
19 2.08006 2.078
20 2.08366 2.140
21 2.02784 2.142
22 2.07241 2.092
23 2.08083 2.121
24 2.06217 2.090
25 2.09677 2.134
26 2.10107 2.152
27 2.11050 2.127
28 2.10009 2.106
29 2.08133 2.083
30 2.10269 2.128
Wilcoxon Matched Pairs Test (abc.sta)
NONPAR STATS
Valid N
T
Z
p-level
LIM3 & LIM5
30
19 00000
4 391340
000011
Wilcoxon Matched Pairs Test
Variables
Variable list 1: LIM3
Variable list 2: LIM5
Box & whisker
OK
Cancel


Wilcoxon Matched Pairs Test (abc.sta)
NONPAR STATS
Valid N
T
Z
p-level
LIM3 & LIM5
30
19 00000
4 391340
000011

ملخص الإطروحة

مشكلة التصميم التوبولوجي لشبكات المناطق المحلية الموزعة جغرافياً هي مشكله معقدة. تاريخياً، حتى عندما يكون الهدف المرجو من وراء التصميم واحداً، تم تشخيص المشكلة على انها من المشاكل المعقدة جداً. ويضاعف من تعقيد المشكلة وجود أهداف متعددة اثناء محاولة التصميم والتي قد تتعارض في طبيعتها وتتأثر ببعضها البعض سلباً أو إيجاباً. يتطلب تحسين مثل هذه الشبكات نظاماً متزامناً ومتناسقاً. إذ قد يؤدي التحسن في احد هذه الأهداف الى إلى مستوى معين من تدهور في واحد أو أكثر من الأهداف الأخرى.

هذه الأطروحة تتناول مشكلة تصميم التوبولوجي الشبكات المناطق المحلية الموزعة بأعتبار خمسة أهداف للتصميم المقترح: وهي: الموثوقية، وتوافر الشبكة ومتوسط استخدام الرابط، و متوسط التأخيرفي الشبكة، والتكلفة النقدية. هناك نوعان من أبعاد المشكلة قيد النظر. البعد الأول هو تعدد الاهداف المراد تحسينها والثاني هو الأمثلية او التجويد. يتأثر التصميم الأمثل بكل من القضايا السابق ذكرها. وقد تم اقتراح العديد من التقنيات على مر السنين لمعالجة جوانب متعددة وبأعتبار اهداف مختلفة لهذه المشكلة. هذه الرسالة تقترح طريقه الاهداف المبرمجه لمعالجة البعد الأول في جمع وتوحيد أهداف التصميم الخمسة إلى وظيفة موضوعية شاملة واحدة. طريقة الاهداف المبرمجة معروفة بكونها فعالة حسابياً وكثيراً ما تستخدم لمعالجة مختلف المشاكل االتحسين الصناعي. البعد الثاني للمشكلة هو الخوارزمية التي تنفذ عملية التحسين. وأثبتت الأبحاث الحديثة ان التقنيات المستنبطة من أسراب الكائنات الحية قادرة على حل المشاكل المعقدة في غضون فترة زمنية معقولة حسابياً وبالتالي، قد نشرت على نطاق واسع إلى حل عدد كبير من المشاكل المعقدة.

هذه الأطروحة تقترح خوارزمية مستعمرة النحل الاصطناعي وهي خوارزمية حديثة نسبيا تلقت اهتماما متزايدا من الباحثين في جميع أنحاء العالم بسبب خواصها ومميزاتها المتفردة. رغم ذلك، لم ترد تقارير عن التطبيقات بالمقدار الكافي لهذه الخوارزمية وخاصةً في معالجة مشاكل التصميم التوبولوجي للشبكات الموزعة جغرافياً. للتحقق من أداء الخوارزمية اقترحت هذه الاطروحة ايضاً خوارزمية مستعمرة النمل. يستنتج من البحث المجرى في هذه الأطروحة أن خوارزمية مستعمرة النحل هي خوارزمية عالية الأداء مع قدرات بحث قوية، ويمكن أن تكون مصممة لتناسب مشكلة كبيرة الأبعاد.

جامعة البحرين

كلية تقنية المعلومات

خوارزمية مستعمرة النحل الاصطناعية لتجويد التصميم التوبولوجي لشبكات الحاسوب متعددة الاهداف

مخطط بحث لاستكمال متطلبات الحصول على درجة الماجستير في تقنية المعلومات

إعداد

أماني ضمد سعد رهيج

الرقم الجامعي: 20121061

إشراف

الدكتور أمجد محمود

(أستاذ مشارك)

جامعة البحرين

الدكتور سلمان أحمد خان

(أستاذ مساعد)

جامعة البحرين

مملكة البحرين

ديسمبر 2015