BIO-INSPIRED SOLUTIONS FOR BANDWIDTH PACKING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TALHA KORUK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JANUARY 2018

Approval of the thesis:

BIO-INSPIRED SOLUTIONS FOR BANDWIDTH PACKING

submitted by **TALHA KORUK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Ertan Onur _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Ertan Onur
Computer Engineering Department, METU _____

Assist. Prof. Dr. Hüseyin Polat
Computer Engineering Department, Gazi University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: TALHA KORUK

Signature :

ABSTRACT

BIO-INSPIRED SOLUTIONS FOR BANDWIDTH PACKING

Koruk, Talha
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Ertan Onur

JANUARY 2018, 95 pages

Sharing channel capacity among multi-rate sensors or synonymously packing bandwidth while satisfying quality of service requirements stays as an important challenge. We present bio-inspired solutions to this problem by reducing it to the NP-hard multiple-choice knapsack problem. We employ various bio-inspired population-based meta-heuristics to allocate capacity to the requesting nodes in a single-hop sensor network. In this thesis, we present the controlled lab experiments for determining the capacity of a wireless channel and then discuss the feasibility of meta-heuristic solutions. The runtime and closeness to the optimal solutions results are presented and discussed. Artificial bee colony optimisation provides the fastest solution although the convergence rate per generation is slower.

Keywords: Bin Packing, Multiple-Choice Knapsack Problem, Bio-Inspired Meta-Heuristics

ÖZ

DOĞA ESİNİMLİ BANT GENİŞLİĞİ PAYLAŞTIRMA ÇÖZÜMLERİ

Koruk, Talha
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ertan Onur

Ocak 2018 , 95 sayfa

Kablosuz alıcı ağlarında, bir yandan servis kalitesi gerekliliklerini sağlarken diğer yandan sınırlı kanal kapasitesinin paylaşımı ya da başka bir deyişle bant genişliğinin optimize bir biçimde doldurulması sorunu önemini korumaktadır. Bu çalışmada polinom zamanlı çözümü olmayan (NP) kombinatoriyal problemi, çoktan seçmeli sırt çantası problemine indirgeyerek problemin çözülmesinde doğa-esinimli üst-sezgisel yöntemlerden yararlanılabileceğini gösterdik. Muhtelif popülasyon tabanlı doğa-esinimli algoritmaları bu tek sıçramalı sensör ağlarında kapasite dağıtımında görevlendirdik. Biz bu gördüğünüz tez çalışmasında; oluşturduğumuz kablosuz ağın kanal kapasitesini bulduktan sonra üst-sezgisel yöntemlerin yardımıyla belirlediğimiz kontrollü laboratuvar çalışmalarını yürüttük ve bu çözümlerin uygulanabilirliğini sorguladık. Çözümleri, sonuca ulaşma süreleri ve doğru çözüme yakınlıkları bakımından inceledik ve Yapay Arı Kolonisi optimizasyonunun, ıraksama oranı düşük olsa da en hızlı ve güvenilir sonuçları sağladığını gözlemledik.

Anahtar Kelimeler: Kutu Paketleme, Çoktan Seçmeli Çanta Problemi, Doğa-Esinli Üst-Sezgisel Yöntemler

I dedicate this thesis to My Family. Endless thanks to my family.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Assoc. Prof. Dr. Ertan Onur, for understanding, patience and providing experience in course of thesis study. His technical support, understanding and patience were remarkable.

I would also like to thank Alitan Bayramoğlu, Mehmet Baskın, Bora Baydar, Ali Ata, Alparslan Lorasdağı and Zeki Bozkurt for their tremendous friendship and understanding at all times.

Lastly, sincerest thanks to each of my family members for supporting and believing in me all the way through my academic life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
API	Application Programming Interface
APS	Application Support Sub-Layer
BBA	Binary Bat Algorithm
COA	Criss-Cross Optimization
FTDI	Future Technology Device Interface
GA	Genetic Algorithm
GS	Generation Size
GT	Guard Time
HP	Horizontal Probability
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
KP	Knapsack Problem
LF	Learning Factor
MAC	Medium Access Control
MCKP	Multi-Choice Knapsack Problem
MP	Mutation Probability
PAN	Personal Area Network
PSO	Particle Swarm Optimization
RF	Radio Frequency
PPS	Parent Pool Size
PS	Population Size
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WFQ	Weighted Fair Queue
WMSN	Wireless Multimedia Sensor Network
WPAN	Wireless Personal Area Network

WSN	Wireless Sensor Network
QOS	Quality Of Service
VP	Vertical Probability
ZDO	ZigBee Device Object

CHAPTER 1

INTRODUCTION

Wireless sensor networks (WSN) have been developed to monitor the environmental conditions and to operate controllable devices. These networks are generally comprised of "nodes", a term that is used to define smart devices used to gather and relay information. These nodes may contain different types of sensors that are put to work to evaluate some form of scalar measurement or to gather multimedia information. While mainly a sensor node behaves as some kind of an end-device which only collects data and sends information, some of them are tasked as coordinators to manage the network and gather information from other sensor nodes. Assembled information then gets evaluated in order to form an efficient network formation and to trigger the desired sensors. This evaluation process tend to bear some significance, as performance and efficiency of a WSN is affected by many factors. If we were to name a few; network topology, hardware constraints, energy consumption, the platform's operating system or the communication protocol of the sensor node comes to mind. We should add that with the relevant technology in cameras and microphones advancing in recent years, these devices have become more and more applicable for low-bandwidth demanding WSNs. As a result, these types of multimedia sensors have been implemented to wireless sensor networks and have enabled the development of Wireless Multimedia Sensor Networks (WMSN). A WMSN, in addition to its monitoring capabilities, makes it possible to retrieve video and audio streams, picture and scalar data. WMSNs not only facilitate communication between nodes but they can also store information and evaluate them in real-time and actuate desired sensors. In this way, WMSNs become applicable for various areas such as smart homes and hospitals [58], transportation [62], environmental monitoring and industrial process

control [4, 7].

Mostly WMSNs are designed in terms of reliability and energy efficiency [28, 73] due to scalar transmission. Nevertheless, in recent years, quality of services (QoS) requirements regarding concerns of multimedia transmission through prioritizing data packets, providing different services, scheduling resource efficiently [25, 26, 27, 74].

However, involvement of multimedia sensors brings about new technical challenges, usually regarding bandwidth allocation. Due to the limited bandwidth, all of the sensors of a network may not be active at any given time. WMSNs are obliged to deal with the selection of sensors with higher service quality requirements and efficient use of limited bandwidth capacity. In other words, WMSNs have to cope with dynamic changes in sensor activities and bandwidth variations [68].

1.1 Motivation

Smart homes always have been the dream of mankind for generations [16]. Today smart homes became a reality thanks to recent developments in home automation technology which consists of electronically controllable units such as heaters, ventilators, lights and streaming video as shown in Figure 1.1. These units are designed to provide better life quality in terms of security, energy efficiency and comfort. Meanwhile each unit has its own duty, home automation appliances tend to consolidate the control of these units into a single device. In order to achieve this, each unit must have communication with the other members of the network. Residents in a smart home are given the chance to monitor the events in real-time and control components remotely via their own mobile device, such as switching the lights on or off. This type of instant control also tends to bring about a decrease in electricity, water and gas consumption. On top of simple wireless device control, users tend to expand their smart home experience with the help of video streaming, voice calls and more. The industry is evolving into what we can call home automation with multimedia capabilities. Therefore there is an increase in bandwidth in wireless networks installed in smart houses partly because of these resource consuming applications [47]. Some sensors may have a fixed position, however, some of them need to be mobile. There-

fore, power demand and connection of all units by cable does not seem possible. More sensors of different understanding of service requirement of each sensor type often leads to complex communication. All this communication can be carried out with wireless signals, which eliminates the need for wires. This means WSNs play a important role in our lives through applications such as home automation [63].

Figure 1.1: Smart Home Example

1.2 Problem Definition

Problem definition is examined into two topics. Firstly, we provide a realistic smart home example to better understand the challenge. Secondly, we define the problem formally for a specific scenario.

1.2.1 Smart Home as a Realistic Scenario

In the smart home scenario [21] the end-user requires an overview of the sensors' present situation using a mobile device. The user may change the security settings, alter the states of lights, curtains, heater or humidifier. The environment in this case is a standard flat.

Let us review this scenario where a WMSN is installed in order to facilitate as a smart home application as shown in Figure 1.1. Sections of the flat for the purposes of this example are kitchen, living room, bedroom, baby room, bathroom, entrance and entertainment room. Each room has a different number of nodes which employ sensors. Each node is connected to the central device acting as the sink. The main entrance of the building puts use to a kit that consists of a camera, a microphone, a door lock trigger and a motion detector sensor. This kit is powered from electricity of the main building itself. The motion detector sensor stays active at all times to turn on the light and inform the user. The user may then activate the camera and the microphone in order to get video streamed from downstairs to the sink device. The resolution of the camera is adjustable. The user is also given the control over the lock on the main entrance. The user can monitor whether the door is unlocked or not. The entrance of the apartment is equipped with a control panel with a security number pad.

The kitchen employs a fire sensor (battery-powered), a smoke sensor (battery-powered), a gas sensor (battery-powered), a motion detector sensor (battery-powered), a temperature sensor (battery-powered), a humidity sensor (battery-powered), a photo detector (battery-powered), a camera (main-powered) and a microphone (main-powered). Among these units; the fire sensor, the smoke sensor and the gas sensor are always active and of higher importance. The camera and the microphone may be activated by the user with once again adjustable quality. Information coming from the temperature sensors is used to adjust the room temperature. The TV can be controlled by the motion detection sensors that resides in several positions around the flat. The system monitors the input coming from these sensors and can decide whether the user is in the kitchen or in the bathroom etc. and can turn the TV off or leave it on. Also the lights around the house can be triggered from the motion detector sensor or mi-

crophone sensor. System may turn the TV off and the lights in 10 minute after the user leaves. The curtains may be opened depending on the photo detector sensor, i.e. whether there is sunlight coming from outside the house.

Living room has a motion detector sensor (main-powered), a temperature sensor (battery-powered), a photo detector (battery-powered), a camera (main-powered) and a microphone (main-powered). Curtains are automatically controlled depending on photon sensors. The motion detector activates or deactivates the lights, the television and the music box. Each wall has a temperature sensor. These temperature sensors activate radiator or air conditioning to adjust room temperature to the desired value. Lights, television and music box are closed in about 10 minutes after the user leaves the room. The user may activate camera and microphone.

Baby room is equipped with a motion detection sensor (main-powered), a temperature sensor (battery-powered), a humidity sensor (battery-powered), a photon detector (battery-powered), a camera (main-powered) and a microphone (main-powered). The microphone is always active in case the baby cries. The user may activate the camera to get video streaming or picture. The room temperature is adjusted to the desired value automatically. Humidifier works also automatically depending on humidity readings. The user may activate the camera to get video streaming or picture.

The requirements of each sensor is ideally the same but contains various parameters which are listed in Table 1.1. The system is a single-hop star-topology dynamic wireless sensor network. Each sensor behaves rather independent regarding to other sensors that they only communicate with the coordinator (sink).

1.2.2 Problem Definition

In a stationary sensor network, let us assume that a sink node (e.g., $Coordinator$ in Figure 1.2) has N sensors connected to it forming a star topology in a single collision domain as shown in Figure 1.2. Each sensor may be active or inactive at a time. When the i^{th} sensor s_i is active, it generates a traffic with rate λ_{ij} bps where i = 1, 2, . . . , N , j = 1, 2, . . . , M_i and M_i represent the distinct number of rates that may be potentially generated by s_i. For example, if the sensor is a video camera,

Table 1.1: Sensor Information

Sensor Information					
Name	**Type**	**Priority 1**	**Priority 2**	**Priority 3**	**Power Status**
		Rate 1	**Rate 2**	**Rate 3**	
Camera	Multimedia	100	150	200	Main-Powered
		16 $KBit/s$	32 $KBit/s$	48 $KBit/s$	
Microphone	Multimedia	57	73	91	Main-Powered
		8 $KBit/s$	16 $KBit/s$	32 $KBit/s$	
Motion Detector	Scalar	23	37	57	Low-Powered
		4 $KBit/s$	8 $KBit/s$	12 $KBit/s$	
Temperature	Scalar	23	37	57	Low-Powered
		1 $KBit/s$	2 $KBit/s$	4 $KBit/s$	
Gas Detector	Scalar	200	200	200	Low-Powered
		1 $KBit/s$	1 $KBit/s$	1 $KBit/s$	
Fire Detector	Scalar	200	200	200	Low-Powered
		1 $KBit/s$	1 $KBit/s$	1 $KBit/s$	
Door Lock	Scalar	41	83	131	Low-Powered
		2 $KBit/s$	4 $KBit/s$	6 $KBit/s$	

the traffic requirement depends on the compression algorithm of the video source and the camera may support discrete number of compression rates where the number of different compression rates is M_i. The sensors connected to the coordinator node shares a channel with capacity C bits per second. We assume static channelization and time-division multiple access (TDMA) scheme [46].

Depending on the scenario, some of the sensors will be active and others will be inactive. The traffic requirements of the active sensors will be sustained on the shared channel. The devices hosting sensors will communicate with the coordinator to convey the sensor measurements. The traffic generated by the sensors will have discrete integer priorities p_{ij} between 10 and 200, respectively. The coordinator C will collect all the traffic requirements from the newly registered sensors and change the traffic rate of each already connected sensors accordingly. The traffic rates of the sensors will determine the duty-cycle as well. We assume that the coordinator is aware of the types of sensors including possible λ_{ij} and p_{ij} values. Coordinator will determine which sensor reading or stream will occupy the channel bandwidth based on the priorities and traffic requirements in the run-time using an online algorithm.

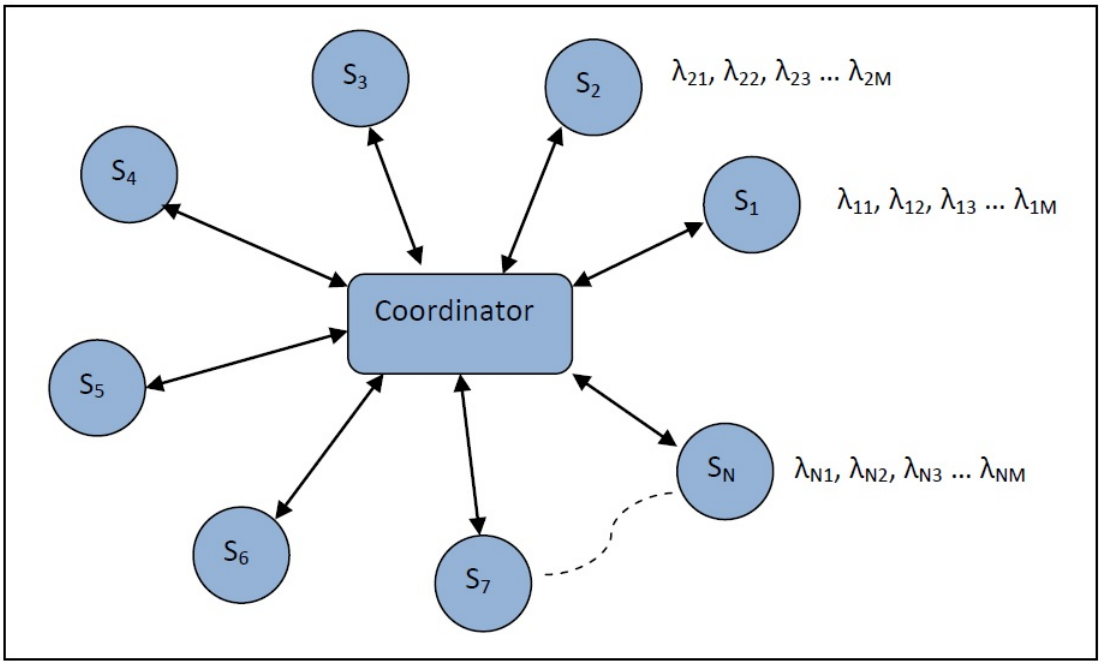

Figure 1.2: A Scenario Where N Sensors Which Supports Multi-Rates are Connected to a Coordinator

Let's sketch a simple scenario. Assume $N = 4$; only four sensors S_1, S_2, S_3 and S_4 exist in a stationary sensor network connected to a coordinator following star topology. As shown in Figure 1.3, first sensor S_1 is equipped with a motion detector (scalar sensor) and S_2 a camera sensor that supports multiple codecs. The third sensor S_3 is equipped with a temperature sensor and S_4 is a microphone sensor. Let λ_1 , λ_2 , λ_3 , λ_4 represent the traffic requirement of S_1's motion detector sensor, S_2's camera sensor, S_3's temperature sensor and S_4's microphone sensor, respectively.

The timeline of a simple scenario is shown in Figure 1.4. Assume that this scenario runs over $C = 22$ Kbps. At the beginning, only the motion detector sensor and the temperature sensor are active generating λ_1 (6 Kbps), λ_3 (4 Kbps) units of traffic respectively. At the 20^{th} second the motion detector goes off and video sensor S_2 is activated. Starting from the 20^{th} second on the total traffic requirement is increased by λ_2 (16 Kbps) units. Then, at the 60^{th} second microphone sensor is activated depending on video stream. λ_4 (8 Kbps) rate is added to total demand.

When the traffic rate of these four sensors are equally allocated it may not be possible to utilize the channel for streaming the video. We need to adapt the rate of the sensors

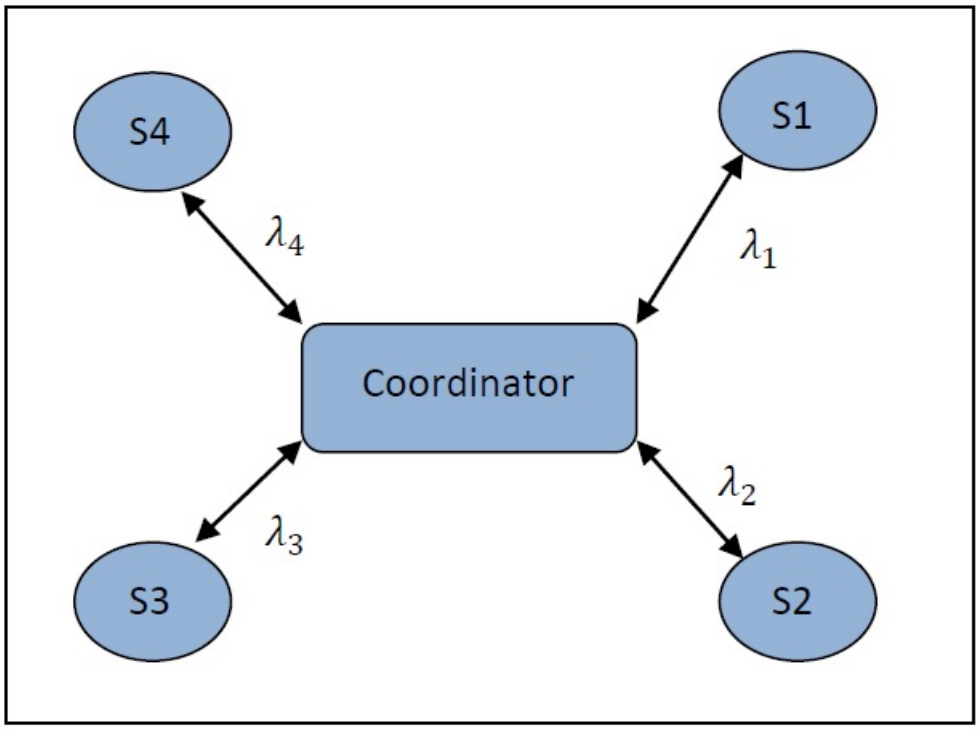

Figure 1.3: An Example Configuration Where Four Sensors S_1, S_2, S_3 and S_4 with Their Pre-determined Rates are Connected to the Coordinator C

to dynamically changing traffic requirements with different priority values. How to determine shared channel allocation in real-time based on the traffic requirements and priorities thereof is called as the bandwidth packing problem. In this problem, we assume all the traffic requirements and their discrete priorities are known upfront; that is why we can refer to this as an offline problem. However, the solution has to be computed in real-time with minimal delay.

Bandwidth packing problem is a specific example of variable-sized one-dimensional priority-based bin packing problem. The objective is to pack the traffic requirements on a fixed capacity channel by considering the discrete priorities. However, the objective is accompanied by another important concern. If we can carefully schedule the traffic, it may be possible to let sensors sleep more and conserve energy. In this thesis, we concentrate on showing meta-heuristic methods to satisfy these objectives in short time.

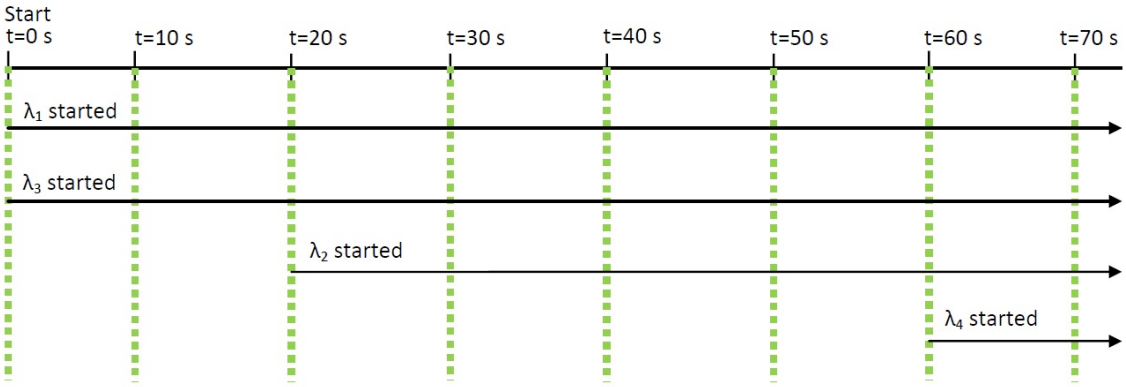

Figure 1.4: An Example Scenario of Sensor Activity in Time

1.3 Problem Analysis

In this section, we will show that our proposed problem can be reduced to the multiple choice knapsack problem. Initially, we should elaborate on what the multiple choice knapsack problem [5, 48, 54, 75] really is. Then, we use meta-heuristic methods to solve for multi-rate bandwidth packing problem in short time to present the applicability for real life systems.

1.3.1 The Problem Complexity

The main goal of the proposed problem is to find the best sensor configuration which provides the maximum priority and optimum bandwidth allocation within the capacity of wireless sensor networks. Having large numbers of sensors in the network, with each sensor having different discrete rates and superorities against each other, obtaining the optimal sensor configuration may prove somewhat difficult given the capacity constraints and priority motives. Thus, ensuring bandwidth efficiency and optimally prioritized sensor configurations in a constrained wireless sensor network through allocating rates to sensors is the main requirement of the proposed problem.

Now that we have somewhat defined what inconveniences configuring wireless sensor networks may present, we must draw attention to a very similar problem in many ways [5, 48, 54, 75]; the Multiple-Choice Knapsack Problem (MCKP). The Knapsack Problem is a renowned NP-Hard problem [22, 45, 52] in computer science due to its

various applications, and has an inception that dates back into 1890's [31], and the MCKP is a mere subproblem that introduces a class system upon the foundation that is knapsack problem. If we were to define what the knapsack problem is used for; we can use an allegory where we are given a knapsack that can only carry a limited amount of weight. We are to fill this knapsack with items that each have their own weight and their own value; with the goal of having the maximum value inside the bag while not exceeding the weight limit. This problem has been reinterpreted over a number of cases that involves basically two important factors; resource allocation within fixed constraints.

The Multi Choice Knapsack Problem however, has further limitations in what items can be put in this metaphorical bag; by implementing a class attribute into the set of items in hand, and limiting the user into selecting only one item of each given class to be put into this bag. This further complicates the NP-Hard problem [22, 45, 52] and provides insight on how to solve many resource allocation problems by using meta-heuristics such as the case we have in our hands, distributing and allocating bandwidth and prioritizing data among the nodes in wireless multimedia sensor networks.

1.3.2 Formal Definition of Multi-Rate Bin Packing Problem

We can derive a solution to the NP-Hard QoS-aware bandwidth packing problem by examining the solutions for multiple-choice knapsack problem [30, 33, 41]. Even though a sensor may utilise more than one data rate, since we are adapting to the class system that is introduced in multi-choice knapsack problem; we can ease the workforce of the coordinator node by appointing a class to each data rate and necessitating that each sensor may have only one class of rate in use.

The formulation of the QoS-aware multi-rate bandwidth packing problem becomes finding x_{ij} to maximize

$$Z = \sum_{i=1}^{N} \sum_{j=1}^{M_i} x_{ij} p_{ij}$$

subject to

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} x_{ij} \lambda_{ij} \leq C, \qquad i = 1, 2, \ldots, N \tag{1.1}$$

$$\sum_{j=1}^{S_i} x_{ij} = 1, \qquad j = 1, 2, \ldots, M_i \tag{1.2}$$

where M_i is the number of distinct rates of the i^{th} sensor, $x_{ij} = 1$ if $a_i = 1$ and s_i is allocated the j^{th} rate, otherwise it is zero, v_{ij} is the value of allocating rate λ_{ij} to s_i, C is the capacity of the shared channel when $a_i = 1$ if the i^{th} sensor is activated, otherwise $a_i = 0$ that sensor is off. The first constraint ensures that the channel capacity is not exceeded with the rate allocations and the second constraint guarantees that only one rate (compression rate, coding rate, etc.) is allocated to sensor s_i if it is active (i.e., $a_i = 1$). We assume that the network is stationary, the topology is known and the environmental conditions do not change through the period of study. The end nodes convey measurement results over a single hop to the coordinator in a star topology and all the nodes are in the same collision domain. The types and possible rates of sensors are known upfront. These guide us to acquire problem solution through meta-heuristic optimization techniques [11, 44, 79] such as Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Binary Bat Algorithm (BBA), Criss-Cross Optimisation (COA), Genetic Algorithm (GA) and Particle Swarm optimization (PSO).

1.4 Literature Review

Bandwidth packing problem still proves to be a real issue when dealing with telecommunications and the Internet. Recent applications are presented in a survey [69]. Most of the researchers consider bandwidth packing solution is possible through Poisson traffic forming an M/M/1 queuing model. Therefore, they reduce their problem to the knapsack problem and present their own heuristic solutions to meet the QoS requirements [24, 51].

Delay Guaranteed Routing and MAC (DGRAM) [65] is a TDMA-based delay guaranteed protocol which is based on re-using the allocated time slots. This way, this is also considered as energy efficient. CSMA and TDMA based Intelligent Hybrid MAC (IH-MAC) [3] is proposed for reduction of delay and better energy-efficiency through utilizing CSMA and TDMA advantages. This is provided by the usage of broadcasting and link scheduling intelligently. Bitmap-assisted efficient and scalable TDMA-based MAC (BEST-MAC) [1] is proposed for supporting diverse traffic without any loss and delay. These studies are the source of inspiration for us in terms of usage of mini-slot, ordinary knapsack algorithms, management of the slots and member registration.

In the literature, there are a few studies on WSN over Xbee-Zigbee series 2. Most of these works debates performance evaluation on different conditions [14]. For example, [6, 8] consider performance of ZigBee in terms of end-to-end delay, throughput, energy consumption and packet delivery ratio. Especially, [59] may be considered as a special study due to it conducting real-life performance tests over Xbee-ZigBee series 2. However, most of the performance evaluation studies did not address the delay factor caused by TDMA slot size.

Few researchers that we have come across show that Xbee-ZigBee presents its features such as low power consumption or self-healing can be applied in real world applications without much of an optimization when using it only on controlling environmental factors [10, 23, 40]

Some studies are about high quality video, picture, audio streaming in real-time to add multimedia features. In order to find a better solution for real-time multimedia

streaming, various researchers studied the problem from different aspects such as optimal queue scheduling [61], the network layer, the routing scheme [49, 67, 72] and the effect of packet size on throughput performance [70]. Also there are studies which try to demonstrate that high resolution video, picture and voice transmission using WSN without any specific design is not possible [32]. However, in [2] the author shows that low quality real-time video transmission over Xbee-ZigBee by making optimization on routing path is possible.

On the other hand, some researchers study frame packing to increase throughput and provide the desired Quality of Service. For example, [60] presents the impact of optimal frame packing regarding performance of the WSN. Another example, in [66] the author approaches more specifically into designing next fit decreasing algorithm according to the frame's deadline and transmission time.

There are also works on MAC layer optimization [12, 19, 56, 77]. For example, in [76] the author considered bandwidth packing at MAC layer level. Although there is not enough studies for data-link layer optimization for black box systems such as Xbee-ZigBee, researchers may be inspired by MAC layer optimization studies. Over the MAC layer, packet classification and node scheduling is presented as a solution [50]; nevertheless, quality of services is disregarded.

In [57] authors mention the impact of queue delay on WSN performance. Inter-frame delay may be considered as an another type of controllable parameter that impacts the queueing system.

As a result, the majority of works focus on mathematical modelling on existing WSN equipment and computer simulations of QoS instead of real life measurements on current global hardware such as Xbee-ZigBee. Therefore, we have centered our research on the combination of TDMA-based optimization for maximum throughput with optimal energy consumption and the selection of the appropriate rates while scheduling mini-slots efficiently using meta-heuristic methods depending on pre-determined rates and discrete priority values of sensors.

1.5 Contribution

In this thesis, we have determined nearly the best traffic scheduling configuration for multi-sensor having various discrete rates wireless network systems through meta-heuristic solutions in short time depending on priorities of sensors' rates. We have found six applicable algorithms, analyzed parameters effects of each algorithm and compared them. Artificial Bee Colony method is applicable for real life wireless sensor networks. Results show that ABC could generate solutions for network configurations in reasonable time. Moreover, providing nominal deviated solution is another advantage. Through lower solution time and better deviations from the optimal solution, we ensure the quality of service, in the meantime causing less delay and maintaining better energy efficiency for star topology networks.

1.6 Thesis Outline

We have stated the problem in detail at the Introduction Chapter. In the Chapter 2, background information about XBee-ZigBee and FIT IoT-Lab are given and their uses have been explained roughly. Next, we have made a research for obtaining a realistic network capacity to provide as input; in order to do that, XBee-ZigBee and IoT-Lab network environments are selected then the bandwidth capacities are obtained in Chapter 3. Brief information of meta-heuristic methods are given meanwhile evaluating their applicabilities for the network configuration case in Chapter 4. Simulation results are graphed and discussed; comparing a number of algorithms' deviation from the optimal solution and how they behave under multi sensor scenarios while at the same time taking their run times into account was demonstrated in Chapter 5. Lastly, we have concluded this thesis by declaring contributions and future work details.

CHAPTER 2

BACKGROUND

2.1 General Information About ZigBee

In this section, ZigBee protocol, XBee module and Fit IoT-Lab background information are given in details.

2.1.1 What is ZigBee?

As a need of controlling and monitoring environment, communication type between remotely controlled sensors gain importance in terms of energy consumption, data rate and security. The possibility of the existence of lots of remotely controlled sensors interaction shows that the need of standardized communication interface. Using this interface, all modules may form a Personal Area Network (PAN) and interconnect through it. XBee-ZigBee [17] is one of the standardized protocols which is based on IEEE 802.15.4.

IEEE 802.15.4 is a global standard which is defined by the Institute of Electrical and Electronics Engineers. This is developed for Wireless Personal Area Network (WPAN) which provides low data rates. This comprises definition of "Physical Layer" and "Mac Layer". It uses CSMA/CA in the MAC layer. There are three operating frequencies namely 2.4 GHz, 915 MHz and 868 MHz in the physical layer. Mainly; it operates at 2.4 GHz with 250 Kbps theoretic limit.

ZigBee is a protocol which operates above IEEE 802.15.4 physical layer. ZigBee protocol is developed by ZigBee Alliance memberships and is aimed to target low

bandwidth applications such as home automation. The ZigBee protocol provides efficient energy consumption, cost utilization, low data rate, security, self-healing, self-management network configuration and reliability.

2.1.2 ZigBee Stack Architecture

ZigBee stack is formed of various layers above MAC layer as shown in Figure 2.1. These are network layer (NWK), application layer which is separated into parts which are ZigBee Device Object (ZDO), Application Support Sub-layer (APS) and application objects. Network layer of ZigBee is responsible for managing network, security of network and routing. APS is responsible for maintaining of routing tables and forwarding messages. ZDO is responsible for determining roles of devices, initiating a device and discovering other devices on a network.

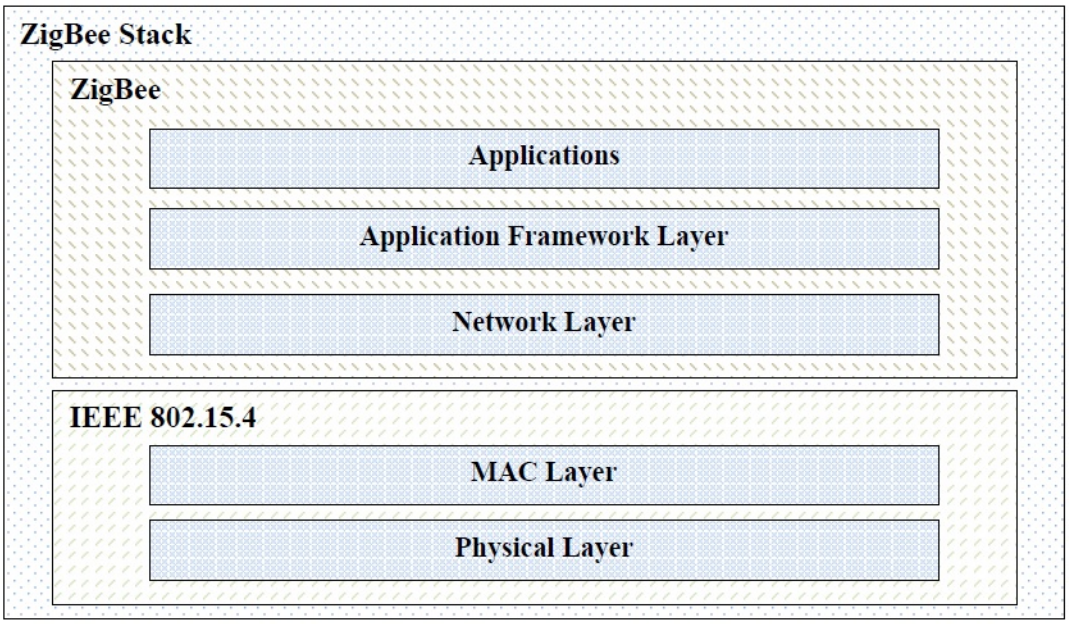

Figure 2.1: The ZigBee Stack [55]

2.1.3 ZigBee Node Types

ZigBee may have three types of roles in a WSN. These are coordinator, router and end-device roles as shown in Figure 2.2. WSNs have to contain only one coordinator

which is the manager. Also, a WSN may have one or more routers and end-devices limited according to addressing method.

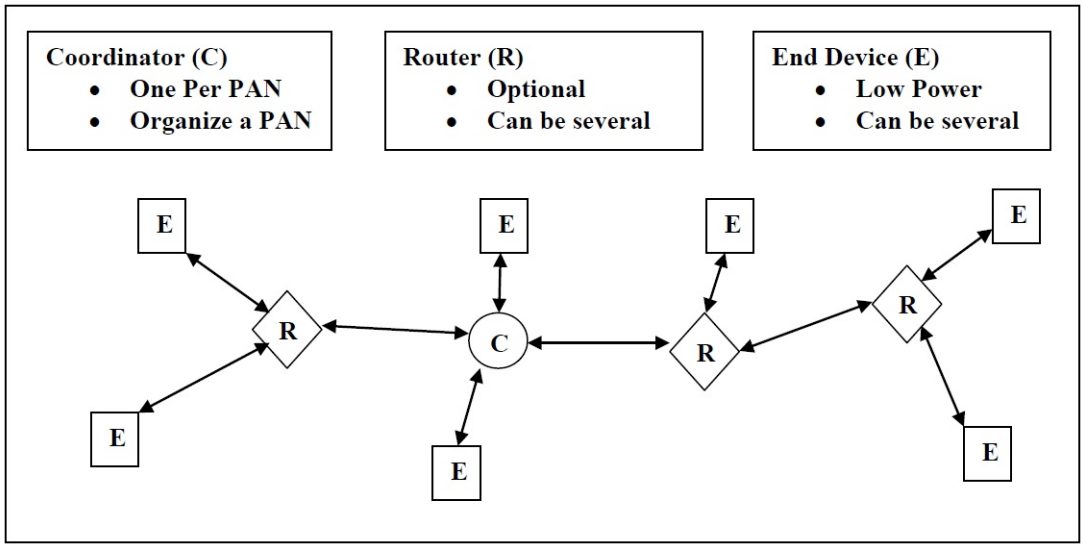

Figure 2.2: A Typical ZigBee Network Configuration [13]

2.1.3.1 Coordinator

The coordinator's main duty is to form the network. It controls channel by the energy scan method and valid PAN ID through checking the region. PAN ID may be both 64-bit and 16-bit. PAN ID and operating channel are published for the entire network. After the coordinator establishes the network and determines connection parameters and rules, it publishes permission rules for routers and end-devices to join the network. As a helper, coordinator assists in routing packets and buffers data for destination node which may be in sleep state. The coordinator has to be main-powered to manage network strictly and efficiently. The PAN may contain only one coordinator.

2.1.3.2 Router

Router's main duty is to route data from a source to a destination. The router checks embedded PAN ID, then searches region for joining a valid network. If the parameters of the router are appropriate for current network, then the coordinator allows joining.

After getting rules, routers may also allow other devices to join the network. This may be used main-powered or battery-powered depending on usage area. Routers also have buffers to store data for sleepy destination nodes. A PAN may contain lots of routers.

2.1.3.3 End-Device

End-devices have to join a valid ZigBee PAN in order to transmit or receive data. End-devices talks with only their parents. If an end-device loses its parent, then it has to find another one to continue its communication in PAN. End-devices may be the source of data or one destination of data but they cannot route data or allow another nodes to join the network. They may be considered as slave nodes. End-devices provide low-power modes to minimize energy consumption. End-devices are mostly battery-powered. A PAN may contain lots of end-devices.

2.1.4 XBee-ZigBee Addressing

All ZigBee devices support two types of addressing, 64-bit device address and 16-bit network address.

2.1.4.1 64-Bit Device Addressing

This address is unique for each XBee-ZigBee device. This address is embedded into devices during manufacturing process and never change. This address may be used as extended address when network grows.

2.1.4.2 16-Bit Network Addressing

Coordinator network address is assigned to zero. However, coordinator randomly generates network address and assigns this network address for each device which join the network. Each device's network address is unique for current network. If

problem of network address such as conflicts occur, then the coordinator determines a new address and changes the device's network address.

2.1.5 Data Transmission

Zigbee sends data packets using unicast or broadcast transmission. Although, unicast transmission is routed data directly from source to destination, using broadcast transmission packet are sent to all devices in the network.

2.1.5.1 Unicast Transmission

Packets are transmitted from source node and received from destination node by following determined path which contains only necessary devices instead of all devices. Unicast transmission uses 16-bit network address to form a path from source to destination. Unicast transmission includes handshake structure due to MAC layer which is CSMA/CA. Before transmitting a packet, the node gets acknowledgement from destination node and determines efficient path towards the destination.

2.1.5.2 Broadcast Transmission

Packets are transmitted from source and received from all devices in the current PAN. After all devices are received broadcast packets, they also try to retransmit packets 3 times as a MAC layer characteristic. Each device holds maximum 8 entry information about packets for 8 seconds to make controllable retransmissions.

2.1.6 Data Rate and Range

XBee-ZigBee operates in three unlicensed bands, 2.4 GHz, 928 MHz and 868 MHz. Its theoretic RF data rate limit is 250 Kbps. On the other hand, data throughput theoretic limit is 35 Kbps. Serial interface communication (between XBee-ZigBee and platform) data rate is up to 1 Mbps which can be adjustable by software. The

communication range is between 40-120m depending on operating environment such as indoor and outdoor.

2.2 Hardware of Testbed

In this section, we mention about the components, the architecture and the communication of XBee-ZigBee.

2.2.1 Hardware Components

Components of testbed include Digi XBee-ZigBee Series 2 wireless module, USB and RS-232 converter module, Digi Coordinator AT firmware (0x20XX), Digi Router AT firmware (0x22XX) and laptop with Linux operating system.

2.2.2 Digi XBee-ZigBee Series 2 Wireless Module

Digi XBee-ZigBee series 2 as shown in Figure 2.3 wireless modules are developed by Digi Company to meet a need of low energy consumption and low cost for WSN. The module operates at 2.4 GHz.

2.2.3 Host and XBee-ZigBee Series 2 Module Communication

A host can send data to the XBee-Zigbee module through RS-232 serial port. The host puts data into "serial receive buffer" of XBee-ZigBee module. On the other hand, host gets data from "serial transmit buffer" of XBee-ZigBee module as shown in Figure 2.4.

"Serial Receiver Buffer" may become full and possibly it may overflow due to continuous transmission. "Serial Transmit Buffer" may become full and result in dropped packets due to traffic load or flow control. Figure 2.5 shows Xbee module architecture.

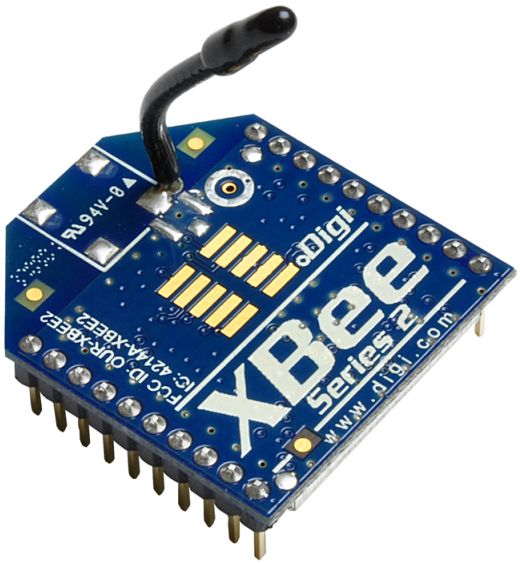

Figure 2.3: XBee Hardware Device [17]

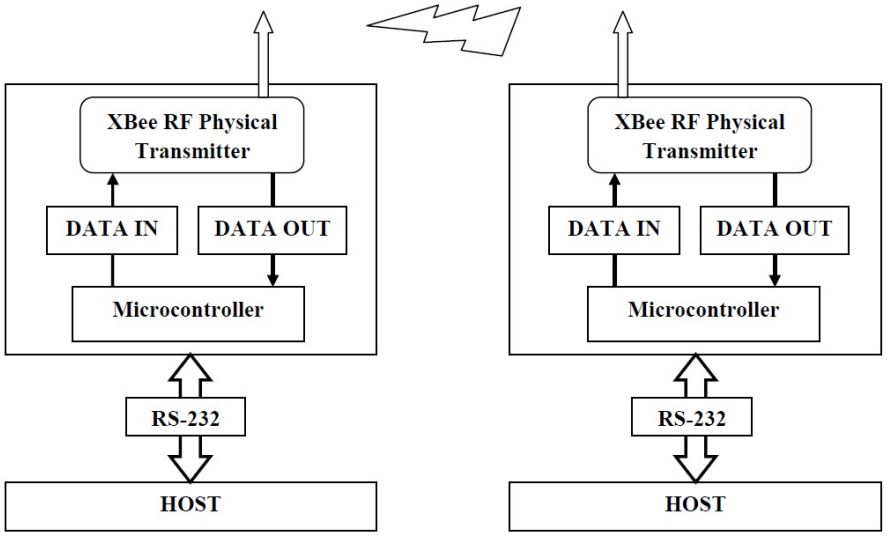

Figure 2.4: XBee-ZigBee Communication [17]

2.2.4 Serial Interface Protocol

Hosts send data through DIN pin and data are queued up for wireless transmission. When a packet is received, then the data are acquired by host through DOUT pin. Data are buffered in the serial receive buffer until occurrence of conditions such as packetization timeout, command mode request with (GT+CC+GT) specific combina-

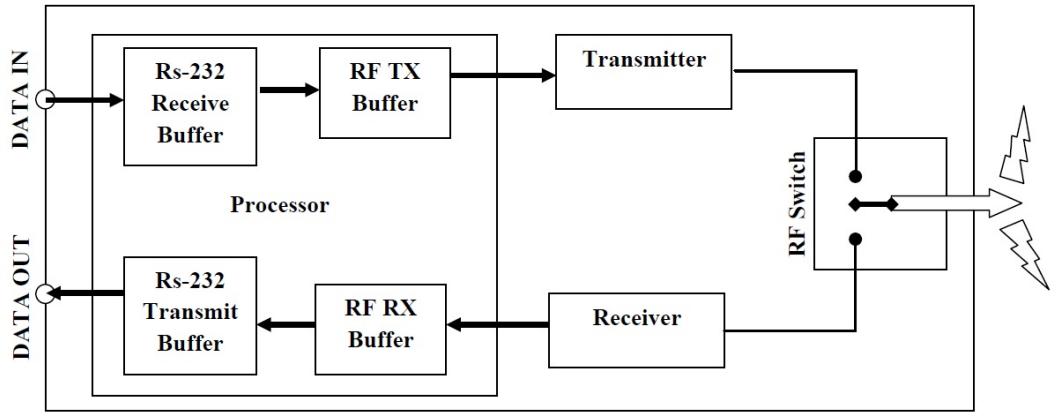

Figure 2.5: XBee-ZigBee Transmitter and Receiver Buffers [17]

tion and the maximum characters that fit in a packet.

2.2.5 Module Modes of Operations

XBee-ZigBee has five different modes which are "Idle", "Transmit", "Receive", "Command" and "Sleep".

2.2.5.1 Idle Mode

XBee-ZigBee put itself into idle mode, when it carries out no action such receiving or transmitting. While it is waiting in idle mode, it may switch itself to another mode such as transmit, receive, sleep or command.

2.2.5.2 Transmit Mode

After getting data from host via serial communication and becoming ready for packetization, the module decides to switch to transmit mode. Before transmission, the module makes sure about destination path's validation. If there is no determined path, ZigBee discovers path before transmission.

2.2.5.3 Receive Mode

When a packet is received, then the data are copied into the serial transmit buffer.

2.2.5.4 Command Mode

The module enters the command mode to read and modify desired parameters. In order to enter this mode, special four-step sequence has to be applied.

- Host does not send any character in pre-determined guard time (GT)
- Host sends special characters such as "+++".
- Host does not send any character in pre-determined guard time (GT)
- Host receives OK message from the module.
- Command mode is ready for reading and modification.

2.2.5.5 Sleep Mode

The module uses sleep mode for low power consumption. In sleep mode, module does not listen to the channel. Module may enter into sleep mode, when it is free for action.

2.2.6 Platform

The Xbee sensors are connected to a notebook through the serial port. It harvests information from sensors, applies algorithms and prepares for communication. Serial communication parameters as follows: 115200 baud rate, 8bit data, no parity and no flow control.

2.3 Firmware, Platform Operating System and Software Information

In this section, we present the features of compatible Xbee module firmware which is provided from ZigBee, the effect of platform operating system on communication and graphical user interface of Xbee module namely "X-CTU".

2.3.1 Firmware of XBee-ZigBee

XBee-ZigBee presents “AT” router firmware and coordinator firmware to provide control interface to developers. The XBee-ZigBee “AT” firmware commands are follows:

- Providing direct communication with network layer through by-passing application framework layer.
- As a good feature, ZigBee protocol handles only forming a network and managing the network without modifying any data.
- Providing maximum payload size (84 Bytes / packet)
- Operating in one of the determined channel which is between from 11 to 26. In order to select the best channel, the module checks each channel’s energy state in an efficient way peculiar to its improved protocol.
- Packetization timeout is a controllable parameter to adjust wireless packet size.

2.3.2 Platform Operating System

Linux Ubuntu open source operating system is selected due to these following reasons:

- The sleep resolution of Linux is approximately 1.25ms comparing with windows7 which is 10 ms.
- Developer environment is better due to being open source.
- The algorithms use multi-thread architecture; therefore, Linux is more efficient.

2.3.3 Software

In order to program firmware of XBee-ZigBee and change parameters, X-CTU graphic user interface of Digi is used. Reading and modifying parameters through X-CTU provides an easy interface. As a programming language, Python is used due its support for multi-cross environment.

2.4 General Information About FIT IoT-LAB

In this section, we present FIT IoT-Lab, node deployment, hardware and software details are given.

2.4.1 What is FIT IoT-LAB?

FIT IoT-LAB [34] is a large scale Internet of Things (IoT) heterogeneous communicating testbed (node) infrastructure which is designed to provide any academic or industrial developer [20] an environment where they are able to conduct real-time and fully-controlled experiments. This platform presents an invaluable opportunity to make faster development on network and application layers of wireless sensor networking technologies. Web-based and command-line interfaces are supported for the management of system. Using one of these interfaces, any developer can schedule an experiment and deploy different firmware for selected nodes. After running the experiment, developers can monitor and store nodes' data such as energy consumption, sensor state or debugging information in real-time. Various real-life use cases can be performed through different node types, topology designs and different sites.

IoT-LAB system has a total of 2728 wireless sensor nodes deployed expanding over eight different locations across Central Europe. In addition to this large size of sensor nodes implemented; there are numerous types of mobile nodes included within the networks with either controllable or uncontrollable motion. The infrastructure has been built with large-scale bandwidth allocation, connectivity and power consumption in mind. Sensors used in creating these topologies offer a variety of different types in the sense that some of them differ in the architecture of their processors or

even their physical wireless chips. This kind of variation is preferred when scientific studies regarding WSNs are conducted due to results being more like that of real life events. Researches are also provided with full access to these nodes via the gateways in which the nodes are connected. Users can directly connect to a selected testbed and seamlessly control these nodes, setting up an experiment via the interface without much of an interruption.

2.4.2 Topology

Due to the full capacity of IoT-LAB expanding over numerous sites [35], application developers can instrument the network infrastructure in Inria Grenoble (928 nodes), Inria Lille (640 nodes), ICube Strasbourg (400 nodes), Inria Rocquencourt (344 nodes), Inria Rennes(256 nodes) and Institute Mines-Telecom (160 nodes), CITI Lab (41 nodes) and Freie Universitat (50 nodes).

We have mentioned the mobility capacities of a part of these nodes in order to facilitate real life applications where sensors needed to be non-stationary. The portability of these sensors are administered using robots; where each node is embedded on a Turtlebot2 or Wifibot; with either has fixed trajectory or can be controlled by the network. IoT-LAB offers various scenarios where the user can engage on these robots in order to determine their path on various levels. In the uncontrollable course case, there are a number of ways that can be preferred by the user, these can be either predictable or non-predictable. The bot can either adopt a pre-defined circuit, or it can randomly select a way point inside the testbed and can repeatedly manoeuvre to the currently set way point until it arrives to the destination. The bot can also instrument an another type of mobility dubbed "Manhattan", where it moves along horizontal and vertical lines and selects which way to go randomly at every stop, making it a candidate in order to study traffic .

The last of the non-stationary applications implemented in IoT-LAB system is of course, user controlled mobility. Control over the bots is left entirely on the user, where the designer of the experiment can upload a mobility model just like the way they can adjust the other parameters of the simulation. This kind of flexibility over the trajectories that the bots can take paves way to such applications that investigates

the behaviours of swarm or fleet.

2.4.3 Node Hardware

The IoT-LAB network contains roughly 512 WSN430 nodes that run on 800MhZ, 632 WSN430 nodes that run on 2.4GhZ, 944 M3 nodes, 536 A8 nodes and 108 open host nodes. WSN430 nodes are based on low-power MSP430F1611 MCU and can communicate via a 802.15.4 PHY layer that can either run on 800MhZ or 2.4 Ghz. M3 on the other hand, is based on STM32F103REY MCU and communicates with 802.15.4 PHY Layer that operates on 2.4 Ghz. A8 node is based on TI SITARA AM3505 (ARM Cortex A8) which can run Linux. A M3 hardware node is embedded on the A8. Hardware configurations are various as were aforementioned; which leads to a cross-platform experiment medium that can produce more realistic solutions. In order to further increase the diversity of hardware; custom boards can be added externally that can be also used as nodes. Among these boards are Atmel SAM R21; based on ARM Cortex architecture (2.4 Ghz operating frequency), Arduino Zero and Zolertia RE-Mote.

The hardware infrastructure [37] is basically a set of IoT-LAB nodes that can communicate with each other and is set up by the user remotely in accordance to the current experiment needs, while this whole time powered by a backbone provider which also serves as a foundation for implementing seamless connectivity. The standard network consists of three main components; open node, gateway and control node. Open node is fully customisable by the user, meaning they can be load and run any software and can be used for any purpose. Gateways act as a passage between the open node and the rest of the network while the control node is used to reach out to open nodes and to control them.

2.4.4 Embedded Software Development

IoT-LAB supports software development [36] on a great extent where users can build their applications both via an operating system that is run on the node or on hardware itself.

2.4.4.1 Architecture

In IoT-LAB environment; we can talk about three layers of API. These consist of drivers, operating systems and communication libraries that is installed upon the hardware. While drivers work as a mediator between the hardware and software; users can develop their applications on top of supported operating systems or onto the driver itself.

2.4.4.2 Drivers

IoT-LAB provides low-level APIs for WSN430 and M3 nodes that is used to operate and to communicate with the available hardware components within these nodes, like ambient sensor lights, temperature sensors, accelerometers and so on.

2.4.4.3 Operating Systems

As mentioned above, the hardware that is used as nodes in IoT-LAB environment supports a number of operating system; albeit simple nodes like WSN430 and M3 can be run with no operating system at all. A8 nodes are more heavyweight in terms of software performance and can even run Linux. Other supported OS include FreeRTOS, Contiki, TinyOS and Riot.

2.4.4.4 Libraries

Communication protocol between 802.15.4 chips used on the open nodes are carried out with wireless communication libraries like CSMA, TDMA or 6LoWPAN.

2.4.4.5 Software in Open Source

The software used in IoT-LAB infrastructure is licensed under a CeCILL License. Users are encouraged to contribute the research community with the results of their inquiries regarding wireless sensor networks.

2.4.5 System Platform Tools

IoT-LAB system supports numerous tools that are provided in order to give the user a wide range on the topic of deploying, running and manage experiments within the network. The platform is built on three layers with the REST API on the bottom. Web portal offers simplicity in tasks such as managing experiments and editing the elements, which ssh front ends cater for the communication between nodes and offer CLI tools.

2.4.5.1 Web-Based Tools

Web-based tools are included within the interface to accomplish regular tasks such as checking platform status, allocation of nodes to selected experiments and so on.

2.4.5.2 CLI-Command Tools

Command line tools are also included in order to build control interfaces of the open nodes. CLI tools leverage the REST API and their capabilities vary from querying experiments to managing node configurations.

CHAPTER 3

NETWORK CHANNEL CAPACITY

The bandwidth packing problem incorporates both controlled and uncontrolled parameters which have an impact on the system itself [43]. One of the important factors is the wireless system channel capacity denoted by C. The bandwidth capacity has a correlation with the number of sensors, shown by N, the environmental conditions and the distances between the sensors. The bandwidth capacity of the system is determined to find the solution for the bandwidth packing problem at hand. Therefore, in this chapter, we primarily concentrate on determining the bandwidth capacity through controlled experiments. We will also elaborate on these experiments' details including the topology of the sensors and the types of hardware used in those sensors. Lastly we will present the network capacity measurement results.

3.1 Capacity Measurement on ZigBee

In this experiment, we have used five XBee Series-2 with integrated ZigBee protocol. Data packets are provided from the personal computer as a host system. The coordinator is located in the center position and four sensor nodes are placed around at the periphary of the coordinator with a distance of 2 meters as seen Figure 3.1. Modules are adjusted with minimum internal delay and maximum packet payload. Moreover, the $ATMode$ of ZigBee, which disables the ZigBee control layer, is selected in order to reach maximum throughput. In other words, $ATMode$ is designed to send accurate data in relatively short time. We have used the communication protocol called TDMA for the task of sending data from host to ZigBee module.

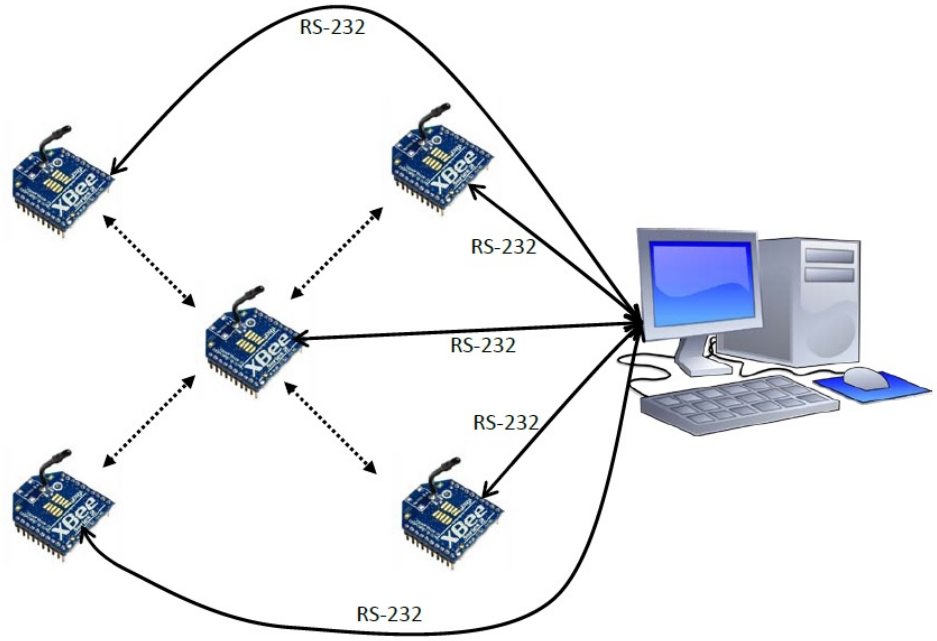

Figure 3.1: ZigBee Network Configuration

The host sends and receives a payload of packets to and from the ZigBee modules prepared in advance, which has a size of 84 bytes, via RS-232 data communication protocol. The ZigBee module then gets the payload data, and starts the encapsulation process. Even if IEEE 802.15.4 supports 127 byte packet structure, ZigBee modules use 43 bytes of it for header information. Theoretically, in order to reach maximum throughput, platform should be sending packets sequentially without a delay. However, ZigBee multi-level infrastructure makes it impossible to make the transmission sequentially without any guard time since the processes of sending a payload to the ZigBee module, making packets of it inside the module, transferring this data between these modules and acquiring the payload from the received packet and sending the payload to the host take reasonable time. Figure 3.2. Moreover, ZigBee protocol in coordinator continuously checks network configuration such as node acceptance.

In the experiments, initially coordinator begins to establish network configuration. After that, coordinator accepts other ZigBee nodes which have appropriate configuration settings. All nodes have connection only with the coordinator. All nodes have waited a beacon message from coordinator to obtain current slot configuration. The coordinator randomly assign slots to nodes and shares slot configuration with nodes every 2 second for experiment purpose. This system runs 20 times for each lasts 10

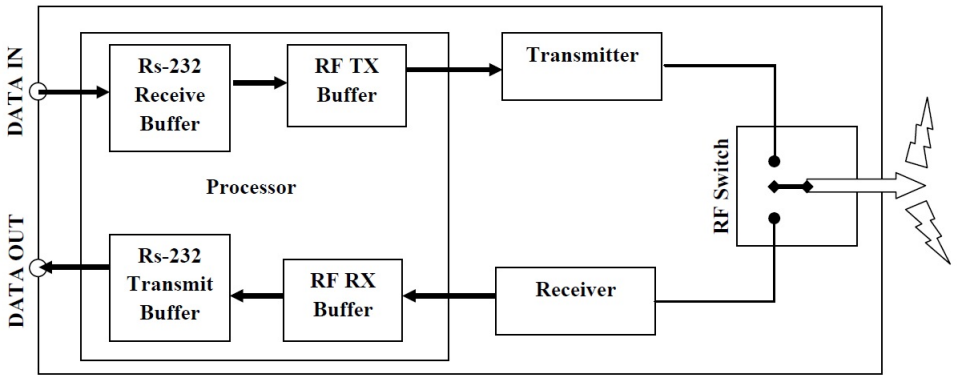

Figure 3.2: ZigBee Packet Transmission Infrastructure [17]

minutes period. These conditions are preserved for each slot size from 4 ms to 60 ms with 2 ms interval.

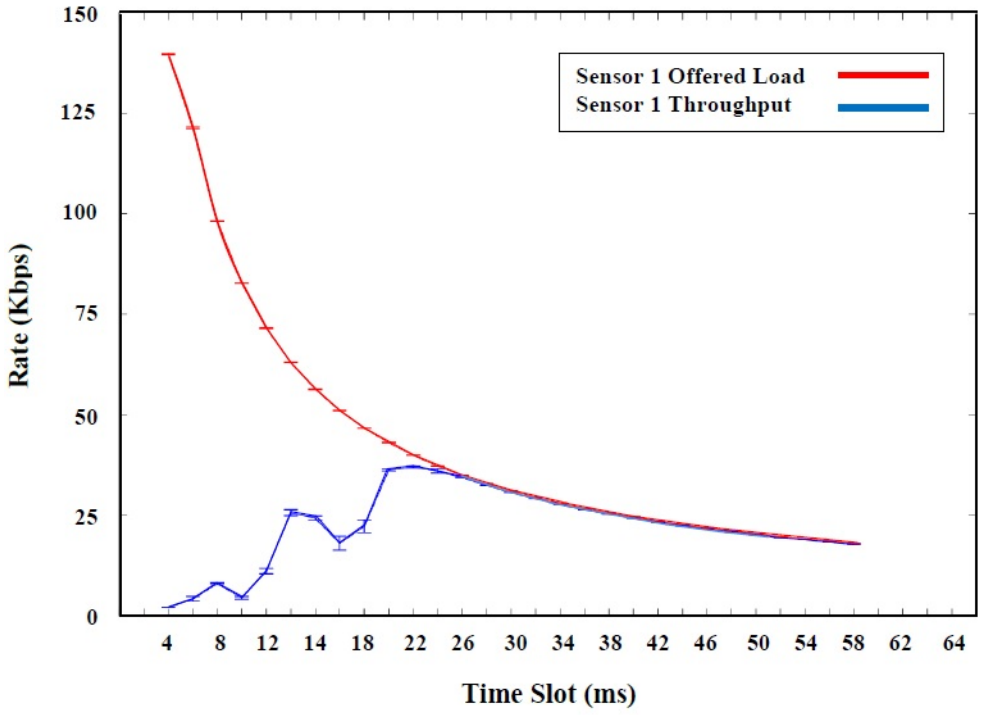

Figure 3.3: Offered and Successful Rates Depending on Slot Size

As shown in Figure 3.3, TDMA slot size starts at 4 ms and increases 4 ms for each measurement. XBee-ZigBee tries to send packets for each slot size. Before 24 ms slot size, offered load and throughput is not correlated. However, after 24 ms slot size, offered load and throughput produce same results. Figure 3.3 indicates that 26 ms slot size is the best rate performance for current XBee-ZigBee network configuration.

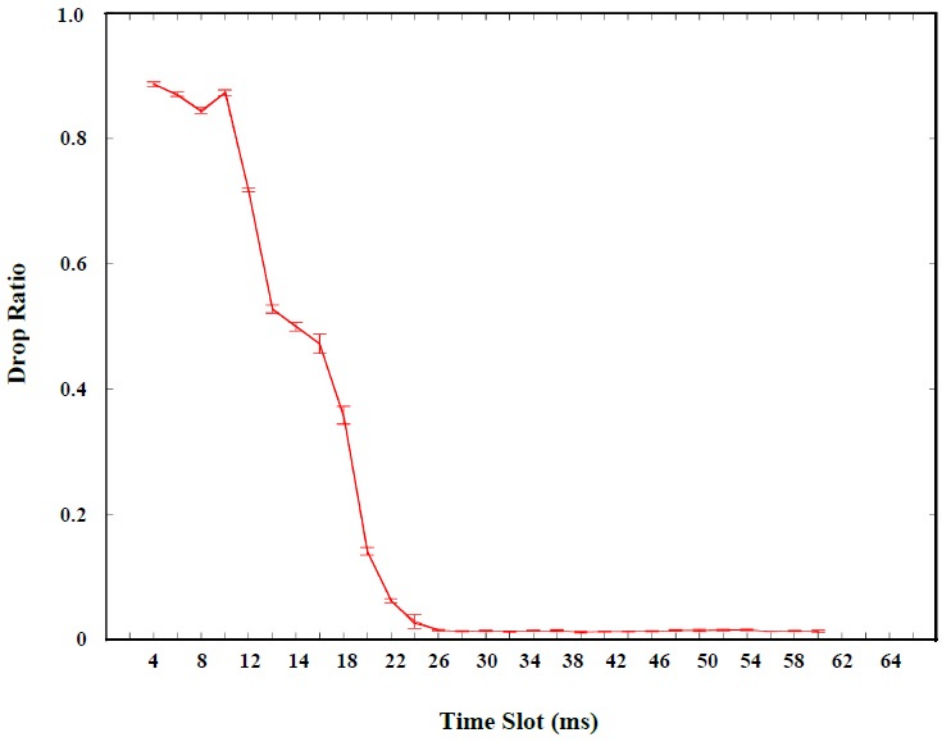

Figure 3.4: Sensor Packet Drop Ratio Depending on Slot Size

On the other hand, packet drop ratio as shown in Figure 3.4 verifies measurement result of Figure 3.3. Depending on these results, XBee-ZigBee maximum bandwidth capacity could be considered 35.4 Kbps as shown in Figure 3.3.

3.2 Capacity Measurement on IoT-Lab

In this section, IoT-Lab experiment environment and capacity measurement details are given.

3.2.1 Experimentation Environment

In this work, we concentrate on QoS and therefore consider TDMA as the medium access control protocol.

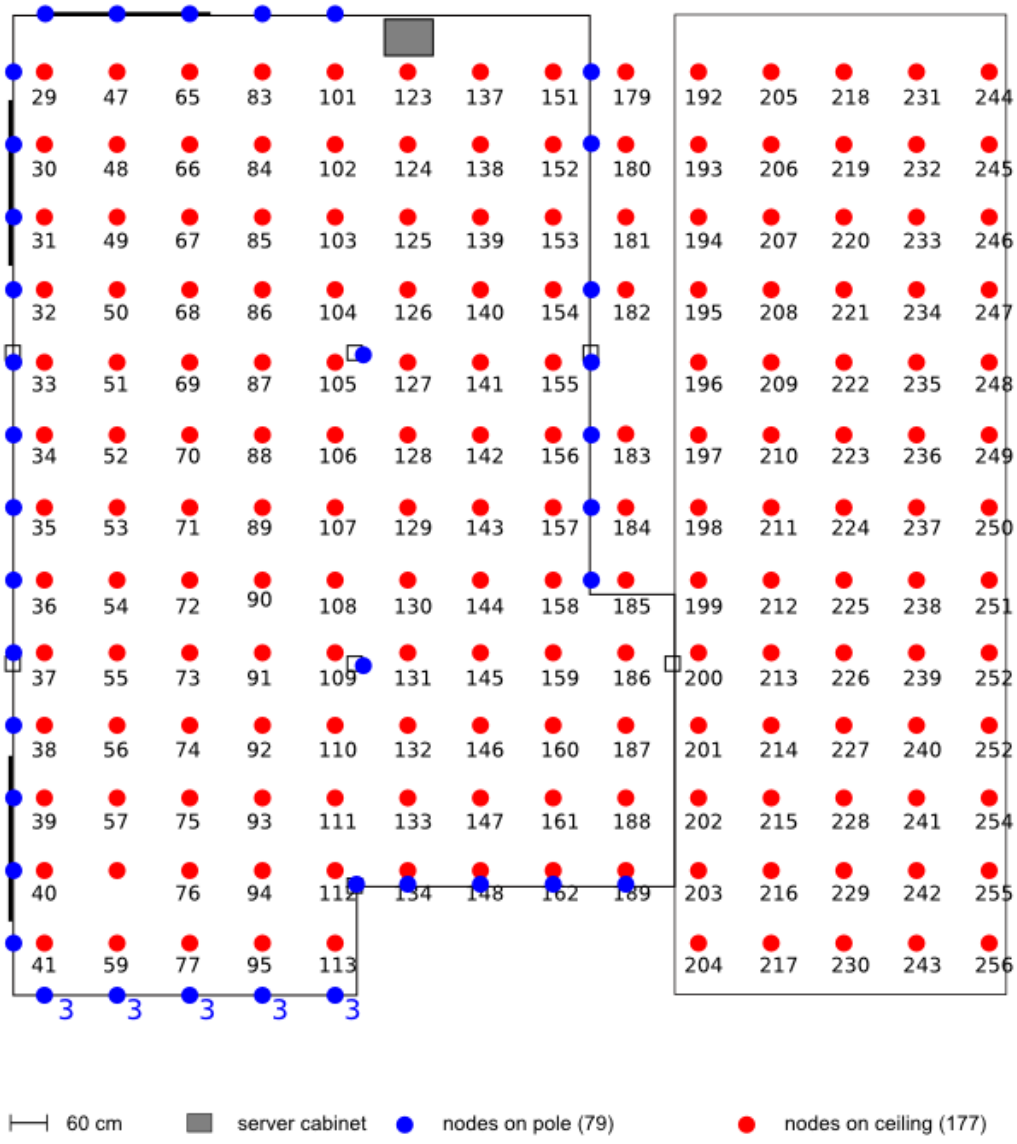

Figure 3.5: The Topology of the IoT-Lab Lille Laboratory [38]

We have chosen Lille laboratory because of equidistant placement of sensors as shown in Figure 3.5. In this design, the coordinator is placed at the epicentre and the nodes are positioned in a star topology. The coordinator of the network may assign slots to sensors after solving the bandwidth packing problem.

In order to acquire realistic results, we use the M3 hardware [37] node as shown in Figure 3.6. This device supports the OpenWSN TDMA module which of details is shown Figure 3.7 as the basis of our solution. The TDMA channel is represented as superframes, which consists of frames that are comprised of variable number of slots. In this scheme, every 2 second a beacon is broadcasted from coordinator to synchronise all connected sensors. A frame contains 20 slots. Two slots are reserved for management. First management slot is used for collecting the traffic requirements of the sensors and second one is required to broadcast scheduling configuration. The guard period between consecutive slots is 1300 microseconds and all the listening nodes start receiving 500 microseconds before the start of each subsequent slot. Packets are 127 Bytes in size.

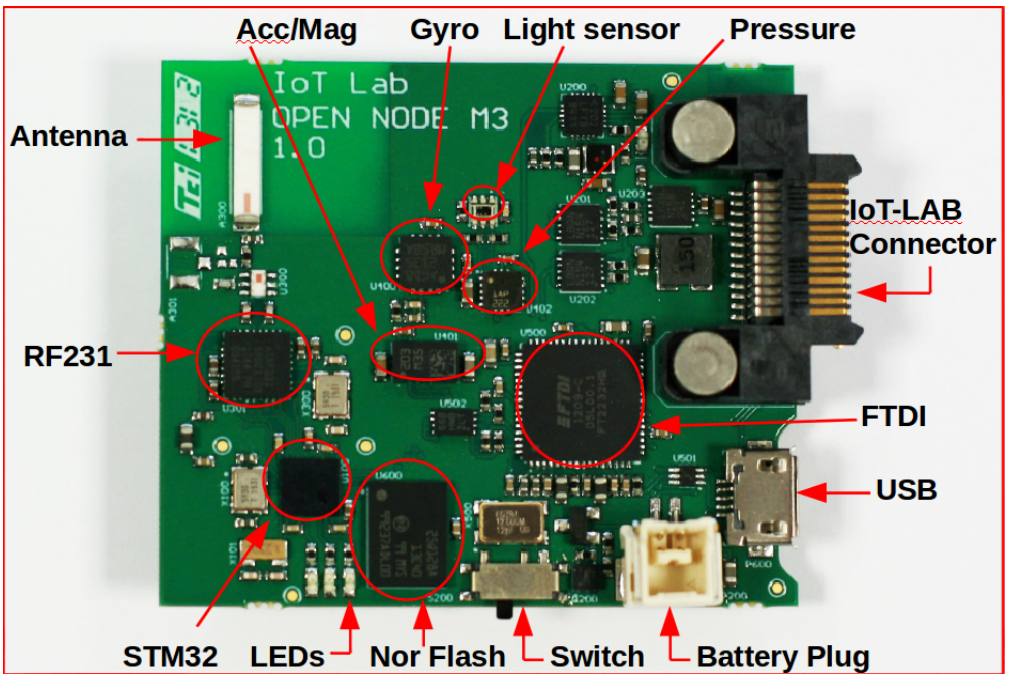

Figure 3.6: M3 Node Structure [39]

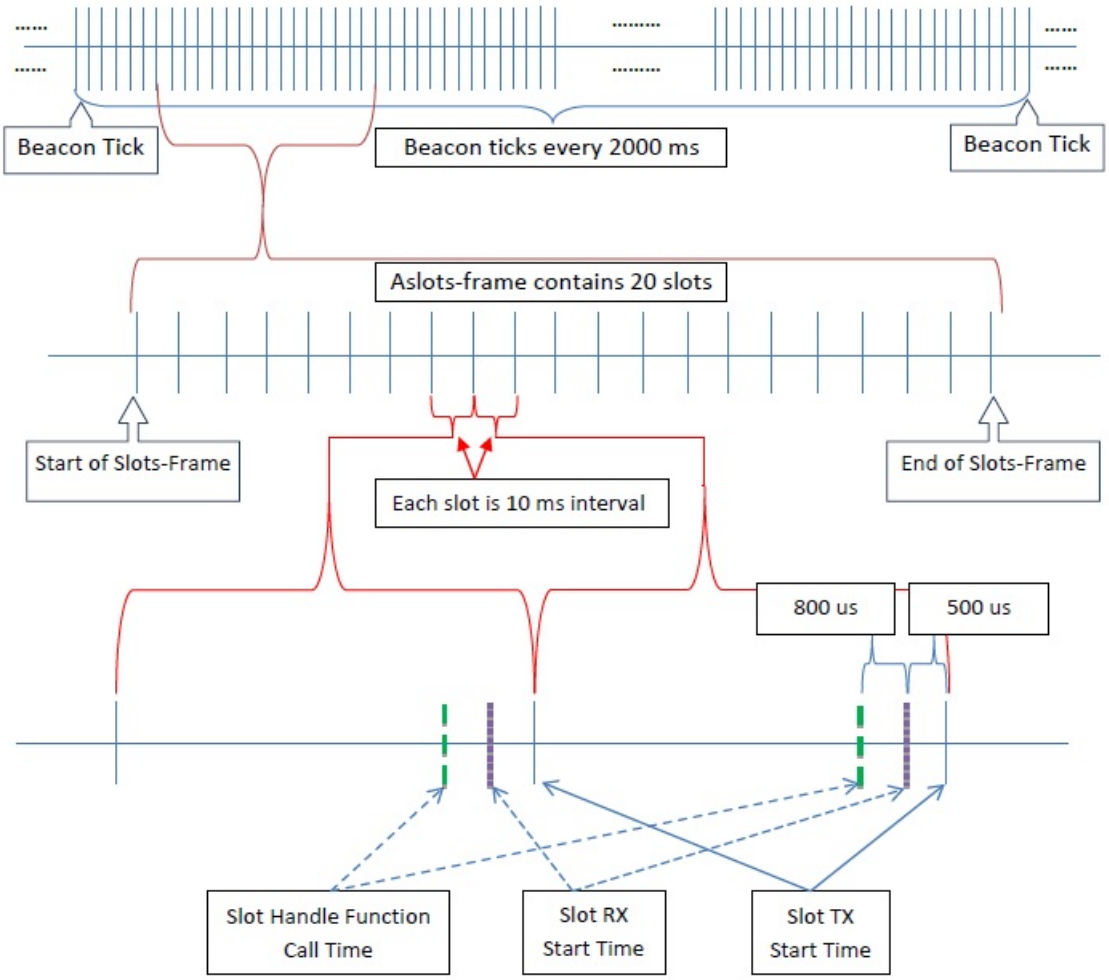

Figure 3.7: IoT-Lab M3 Node TDMA Scheme

3.2.2 Capacity Measurement

The effective capacity of the channel is determined by the TDMA superframe structure, management slots and the guard period. The rate of the phy is 250 Kbps and packets are 127 Bytes yielding a transmission duration of 4.06 ms per packet. 2 slots are reserved for management. Depending on the slot length, the number of management frames change because the number of slots in a frame is fixed to 20. Furthermore, the guard period also chokes up the channel capacity. A simple back-of-the-envelop computation will yield a theoretic effective capacity shown in Figure 3.8 without considering the errors on the channel. We calculate theoretic effective capacity using TDMA structure as shown in Figure 3.7.

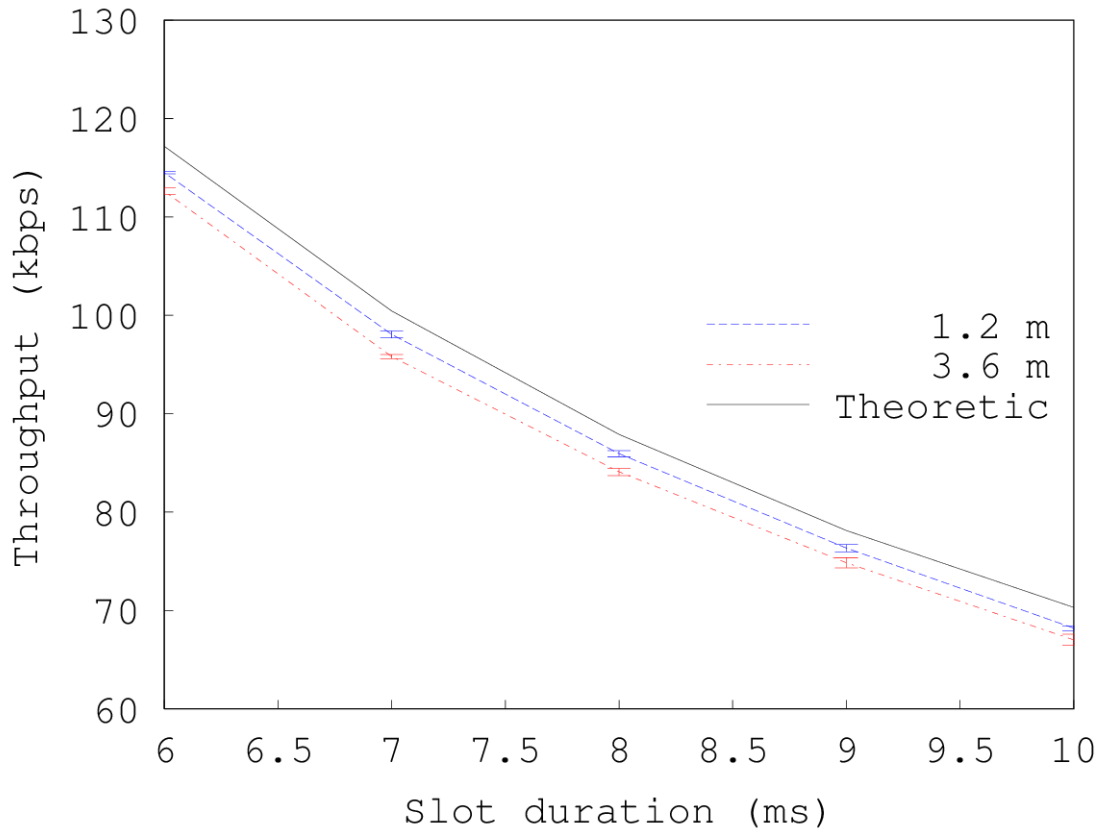

Figure 3.8: IoT-Lab M3 Nodes Network Capacity in Lille Laboratory

We present the test-bed measurement results in Figure 3.8 for various star topologies where the distance to the sink is 1.2 m and 3.6 m. We employ one coordinator and four nodes in the experiments. The coordinator is put center position and nodes are located

equidistant to the coordinator and each other. All nodes have connection only with the coordinator. The coordinator randomly assign slots to nodes. Each experiment is started with warm-up period. Then, these last 3 minutes. Results are averages of 20 runs.

As shown in Figure 3.8, the theoretic capacity and measured capacity results are almost the same. When the slot duration is less than 6 ms, the effective throughput decreases because packets in consecutive slots start colliding with each other. This phenomenon does not reveal itself since errors are not considered in our simple theoretic computations. If it is large, we do not utilize the channel because of very large slot durations in comparison to actual transmission times. In these scenarios, we employed the smallest transmission power levels (-17 dBm) and have not observed a significant bit error rate. Depending on the channel conditions, we may have to periodically probe the channel capacity to be able to solve the bandwidth packing problem.

CHAPTER 4

BIO-INSPIRED SOLUTIONS

In this section; we are going to go over numerous optimization solutions which are derived from the nature itself. These formations proved to be the nature's solution for complicated problems [71] that resembles what we deal with in optimizing WMSNs and allocating the resources. Swarm intelligence-based techniques [78] in algorithms that are used for optimization share a certain characteristic as they've all originated from real life behaviour of animals.

4.1 Methodology

The process begins by generating a sensor list with four different rates and priorities assigned to each sensor type as shown in Chapter 1. From 6 to 18 sensor configurations are constructed by selecting a sensor randomly from this list. Regardless of which sensor configuration is selected, all sensors are assumed to be active during the experiment. Using the exact algorithm, optimal solution values are found for each sensor configuration. Before the final comparison, various parameters of the algorithms are analysed in order to run the algorithms in the best performance possible. Each parameter's initial value is selected from the previous implementations of the algorithm. Then, the imminent neighbourhood of the parameters' values are checked. Each experiment is conducted 500 times in order to ensure the results. Using the optimal parameters, the solving time of the bandwidth packing problem and the deviation percentage from the optimal solution are taken into consideration; and the selected algorithms are compared in Chapter 5.

4.2 Artificial Bee Colony

The Artificial Bee Colony (ABC) [42] algorithm has become one of the standards in swarm intelligence-based optimization problems due to its high accuracy and relatively low-cost. We will introduce in this section a chosen few of these swarm intelligence-based algorithms, along with Ant Colony Optimization and Particle Swarm Optimization in order to try and compare how do they fare if implemented in Wireless Sensor Networks. In the ABC case of course, the algorithm tries to mimic a bee colony. In order to achieve that, the algorithm puts to use three active classes of bees in a colony; employed bees, onlooker bees and scouts. Fictionally speaking, these three groups all have their respective tasks that work in some way to optimize the solution for the problem at hand. In order to further understand how the optimization process works, we have to once again turn to the real life example of how a bee colony functions and distinguish the related tasks of these mentioned classes. First, we have the employed bees phase; these bees are tasked to search for new and more valuable resources of food in order to sustain the growing need of resources in the colony proportional to the increase in population. The algorithm conducts this exact "work versus food" rhetoric by introducing a reward system that fortifies a worker bee's fitness level whenever a bee finds a new source of food. This incentive serves as a way to optimize the way the elements work to gather resources, as the "fittest" element continues to live on to the next cycle. Moving on to the subsequent phase; we have the onlooker bees, which is an alegory for the caste system that is present in bee colonies. All employed bees nominee to be an onlooker bee, but there's a catch: the more fitness levels gained in the previous phase by a certain employed bee will accentuate its chance to be selected as one. The algorithm presents a reward for each cycle of this phase whenever a new member gains their stripes.

The last phase of the ABC algorithm is the scout bees phase, which serves as a way to cut any loose employed bee elements that can't produce rewards or "food" through evolving into an onlooker bee after a predetermined number of cycles. These forementioned phases are ran in the algorithm as many times as necessary in order to meet the predetermined optimal solution.

ABC has three parameters which affect solution performance. These parameters are

population size (PS), generation size (GS) and comparison probability of the distance between the new food position and current selected bee (R) as shown in Table 4.1. In order to analyse R parameter effect, the 18 sensor network configuration case is selected and other parameters are fixed. Experiments are run for each generation size from the start.

Table 4.1: ABC Algorithm Parameters

Name	**Value**	**Parameter Explanation**
PS	90	Population Size
GS	7	Generation Size
R	0.3	Probability of the distance between the new food position and a bee

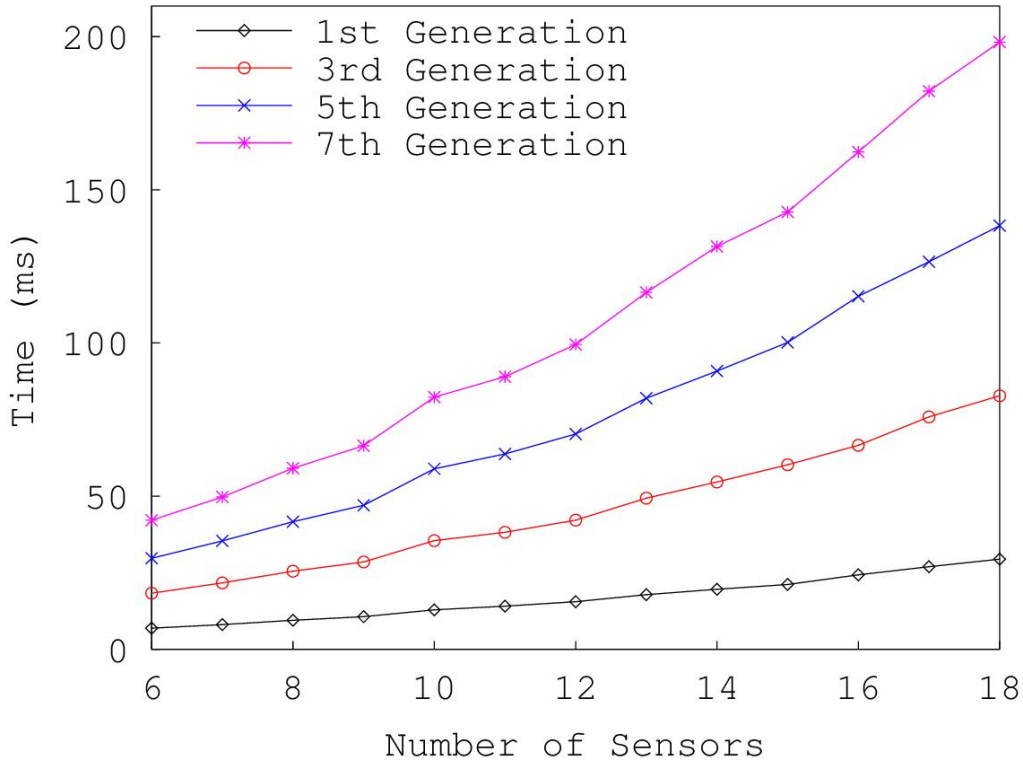

Figure 4.1: Time-to-solve the Bandwidth Packing Problem for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.1, we can acknowledge that the generation size in fact affects the solution time considerably. For the 6 sensors network configuration case, the solution takes

about 10 ms, 22 ms, 36 ms, 48 ms for each consecutive generation respectively; yet in the graph we have shown only the 1st, the 3rd, the 5th and the 7th generations in order to provide some manner of clarity. The elapsed time difference between generation sizes are approximately 12 miliseconds for the 6 sensor case. On the other hand we should note that, for the 18 sensors network configuration, solution takes about 30 ms, 75 ms, 125 ms, 185 ms for selected generation sizes, almost double the amount of time elapsed in 6 sensor case. In other words, as the number of sensors increase, the time difference between generations also increases. Therefore, the generation size should be preferred as small as possible for the real life network applications.

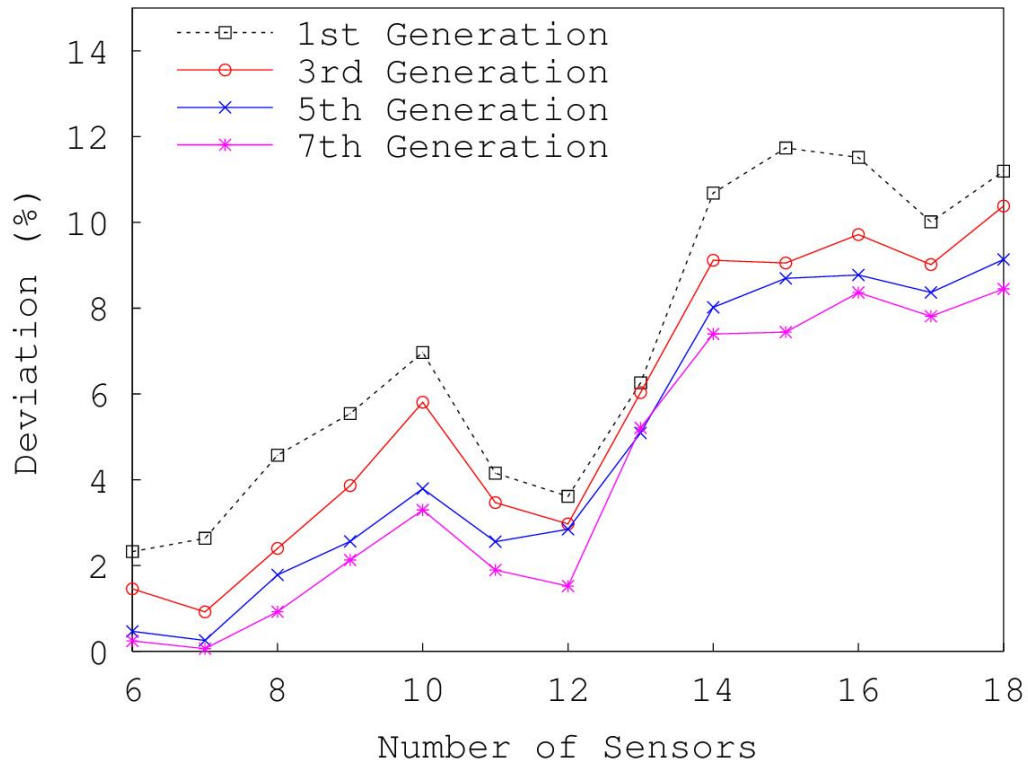

Figure 4.2: Deviation from the Optimal Solution for Various Generation Sizes Depending on the Number of Sensors

Figure 4.2 shows that the deviation is not directly related with the number of sensor nor the generation size. For example, a 12-sensor network configuration presents better deviation results for each generation size when compared with the 10 sensor network configuration, while the opposite is also true. We can also note that; while the deviation percentage can be seen as decreasing while moving towards higher gen-

eration sizes at each and every number of sensor configuration, the variance is not linear and therefore irrelevant.

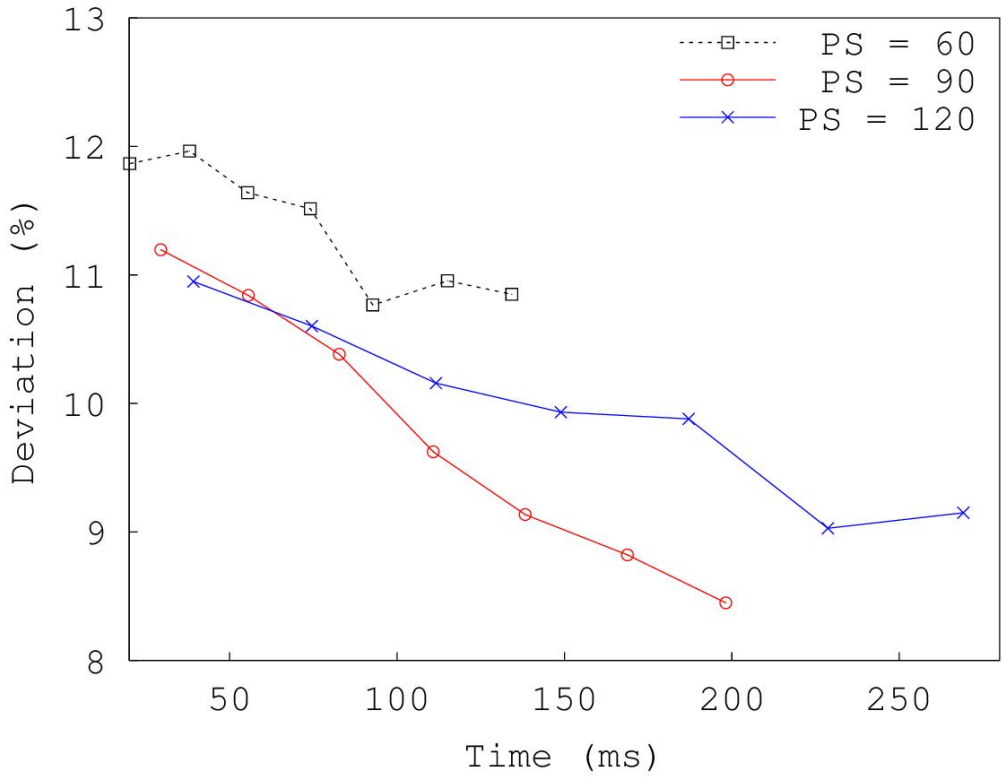

Figure 4.3: Deviation from the Optimal Solution in Time Depending on the Population Sizes

As the x axis of Figure 4.3 shows the solution time versus the deviation percentage of the 18 sensor configuration depending on various population sizes (PS). In population sizes of 90 and 120; the results are considered to be better when compared to population size of 60, due to the low deviation rate. Moreover, even if population size of 120 may seem better for a while, population size of 90 shows better performance after 75 ms. According to these data, a population size of 90 gives the better solutions in this case.

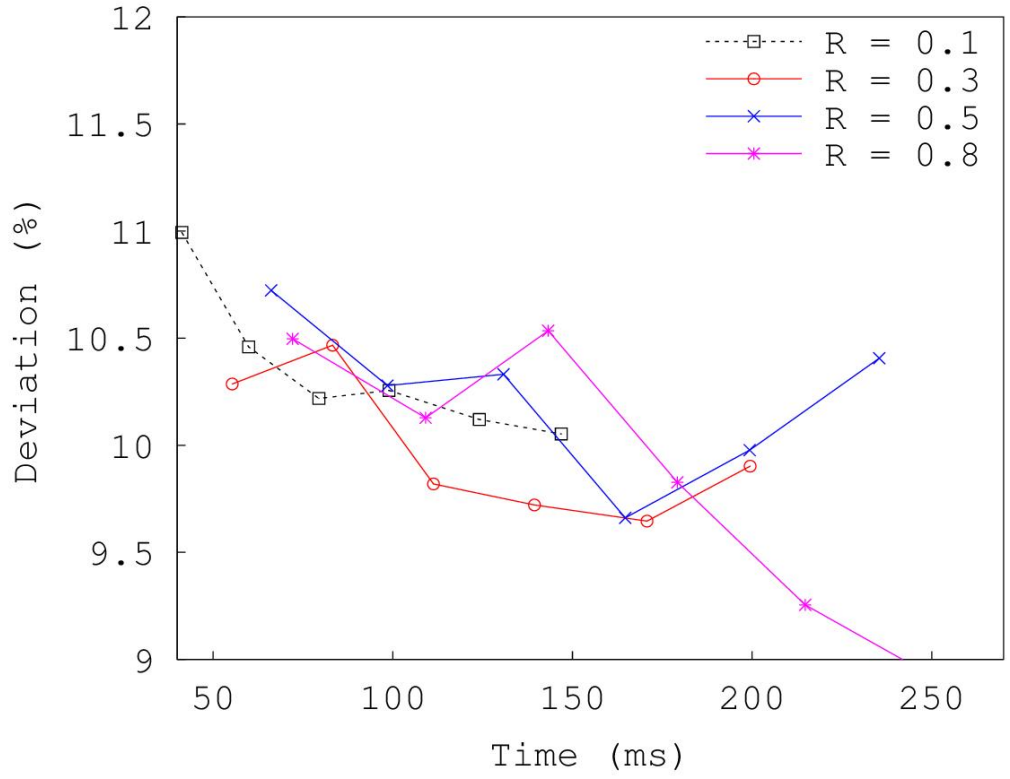

Figure 4.4: Deviation from the Optimal Solution in Time Depending on the Values of R

Figure 4.4 shows the characteristic of R that is defined as the effect of probability of the distance between new food positions and a search agent, in this case a bee; on the graph regarding the deviation percentage over the solution time. $R = 0.3$ has performed generally better in terms of deviation under 175 ms. The time elapsed for finding the first solution relatively increases depending on the R parameter. Deviation percentage can be considered to be somewhat trivial compared to the time axis. Under these conditions we are to select the optimal value of R as 0.3 for experiment run-time lower than 200 ms.

4.3 Ant Colony Optimization

The Ant Colony Optimization (ACO) [18] is yet another one of the most popular algorithms in swarm intelligence-based category, as we've mentioned earlier. As the name suggests, this technique has also been deduced from a real-life example; in this case however, the subject in hand are ants. Ants, like bees; are also commonly known for how their society functions, a rather interesting layered-class system where the transitions between the classes are limited. Ants have forged this complex mechanism for a very simple reason of course; in order to reproduce and grow their population. And to increase population, the colony is in constant need of food. The way the ants locate food sources in a limited space is particularly interesting. Each member starts to search for food by moving rather freely, at random locations. This blind search continues on until a food source is found. In order to retrace their steps and signal their peers, ants release pheromones. Eventually with more ants locating the food source and transmitting it piece by piece back to their colony, the amount of pheromones on the path between the food source and the colony multiplies, thus strenghtening the bond leading to more members of the colony eventually finding their ways to the located food source. Meta-heuristic solution derived from this optimization method can be thought of in a similar manner. In a cycle, all members move in a random direction; then update their pheoromone values. A "trail" is then formed with these pheromone values, between the starting point and the next node. The member then selects their new path according to the accumulated pheromone between the arrival point and the next possible node. This process is iterated until a solution is found. This manner of soft computational method used for solving discrete optimization problems have certain advantages and disadvantages over other algorithms [9]. The ACO algorithm is random at first and the decisions are probabilistic, meaning the time of convergence is uncertain; an undesirable situation in some cases. But in enough generations, a solution is bound to be found.

ACO has five parameters which affect solution performance. These parameters are population size (PS), generation size (GS), relative importance of pheromone (β), maximum controlling level pheromone update (ρ) and minimum controlling level pheromone update (ϵ) as shown in Table 4.2. In order to analyse PS, β, ρ and ϵ parameter effect, the 18 sensor network configuration case is selected and other parameters are fixed. Experiments are run for each generation size from the start.

Table 4.2: ACO Algorithm Parameters

Name	**Value**	**Parameter Explanation**
PS	90	Population Size
GS	7	Generation Size
β	25	Relative Importance of the Pheromone
ρ	0.98	Control Maximum How the Pheromone is Updated
ϵ	0.005	Control Minimum How the Pheromone is Updated

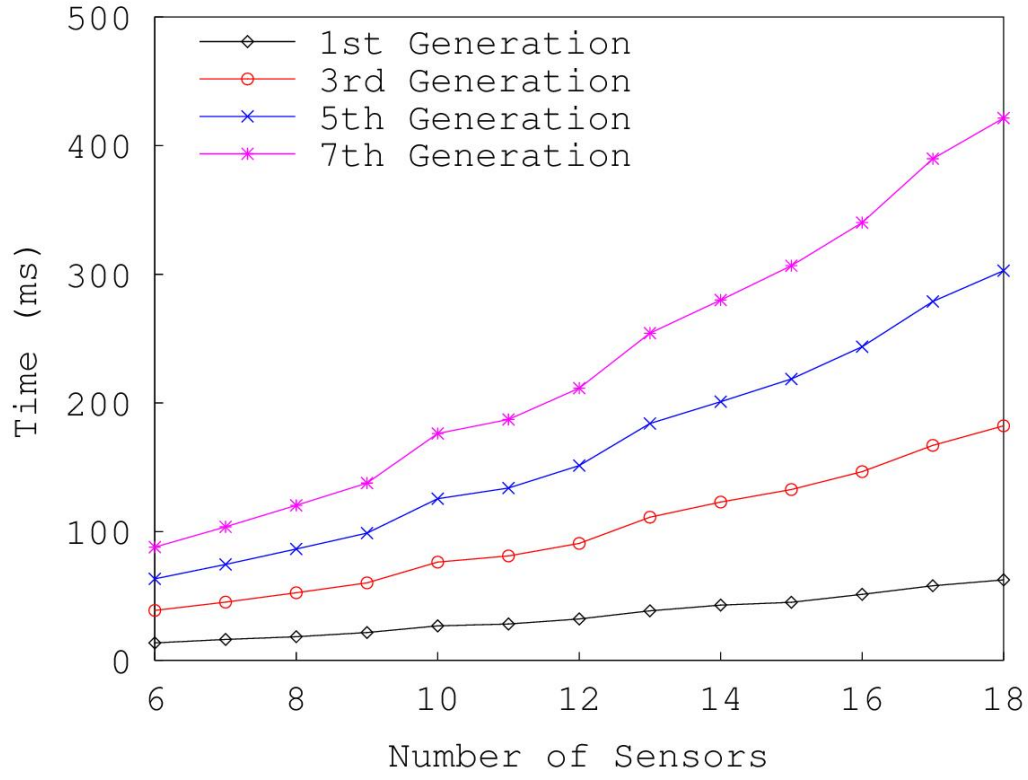

Figure 4.5: Time-to-solve the Bandwidth Packing Problem for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.5, we can acknowledge that the generation size in fact affects the solution time considerably. For the 6 sensors network configuration case, the solution takes about 20 ms, 40 ms, 60 ms, 90 ms for each consecutive generation respectively; yet in the graph we have shown only the 1st, the 3rd, the 5th and the 7th generations in order to provide some manner of clarity. The elapsed time difference between generation sizes are approximately 20 miliseconds for the 6 sensor case. On the other hand we should note that, for the 18 sensors network configuration, solution takes about 50 ms, 180 ms, 300 ms, 420 ms for selected generation sizes, almost 4 times the amount of time elapsed in 6 sensor case. In other words, as the number of sensors increase, the time difference between the generations also increases. Therefore, the generation size should be preferred as small as possible for the real life network applications.

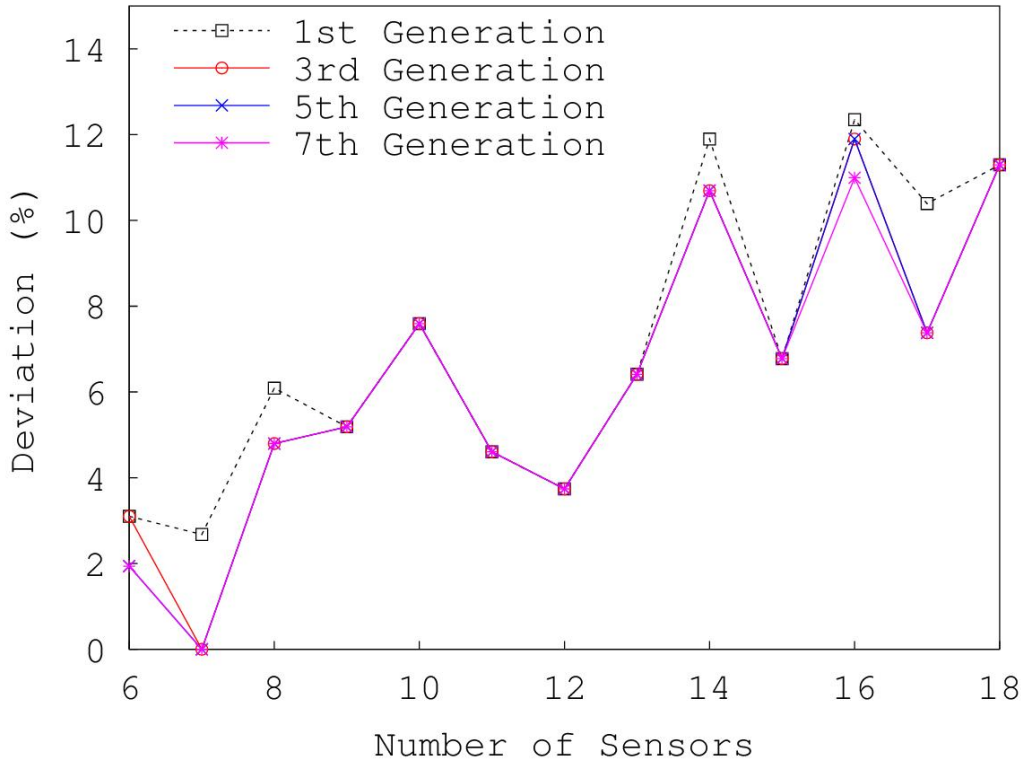

Figure 4.6: Deviation from the Optimal Solution for Various Generation Sizes Depending on the Number of Sensors

Figure 4.6 shows that the deviation is irrelevant with the number of sensors or the generation size. For example, a 12-sensor network configuration presents same deviation results for each generation size while a 14-sensor network configuration behaves

differently. We can also note that; while the deviation percentage can be seen as decreasing while moving towards higher generation sizes at each and every number of sensors configuration, the variance is not linear and therefore can be thought of as irrelevant.

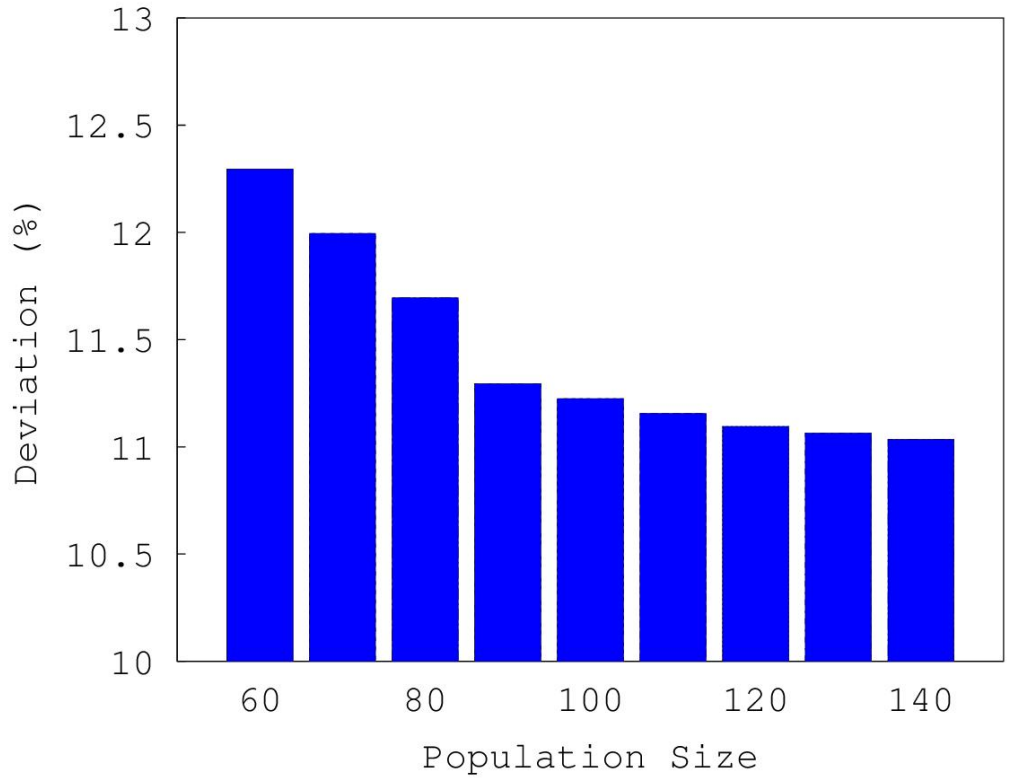

Figure 4.7: Deviation from the Optimal Solution in Time Depending on the Population Sizes

In Figure 4.7, the Ant Colony Optimisation algorithm case, we have yet again different population sizes to test in order to find an optimal value; meanwhile, the 18 sensor network configuration case is selected and other parameters are fixed. As the graph suggests, the deviation rate lowers upon increasing the population size, yet the elapsed total time also rises. We have selected a PS value of 90 in this experiment; since it has the best of both worlds.

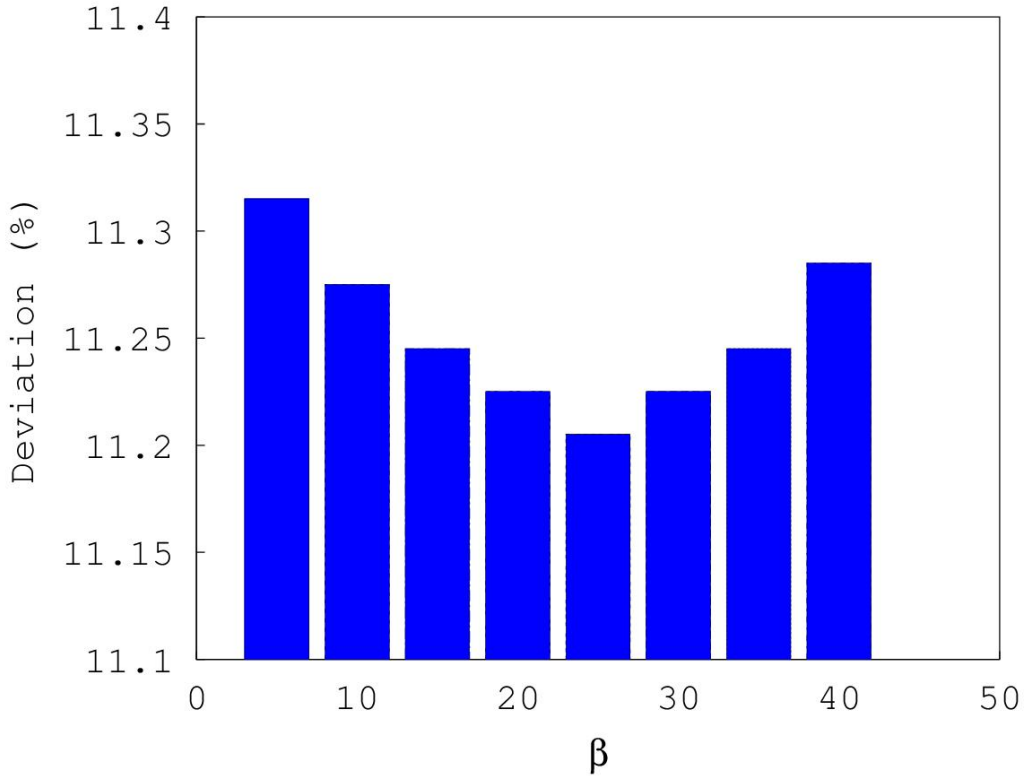

Figure 4.8: Deviation from the Optimal Solution in Time Depending on the Values of β

In Figure 4.8, we are obliged to find the optimal value for each related parameter of the algorithm in order to solve for the best results. Therefore, a β level of 25 can be acknowledged as the best case for this scenario.

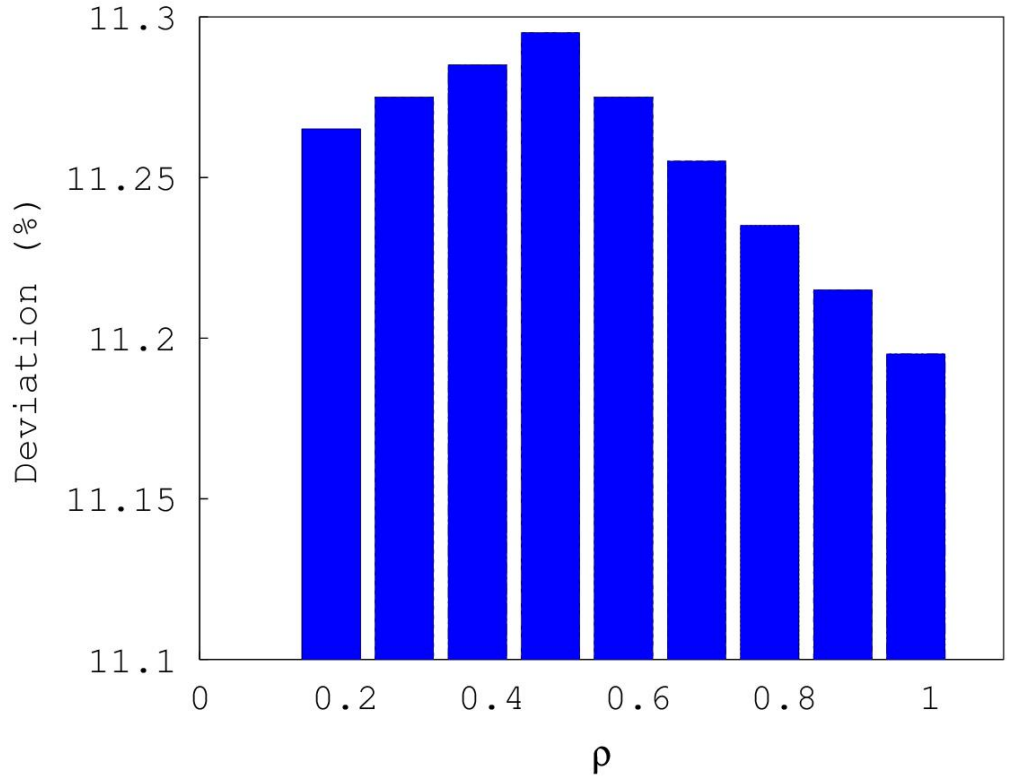

Figure 4.9: Deviation from the Optimal Solution in Time Depending on the Values of ρ

We have selected in Figure 4.9 a ρ value of 0.98 from the data we have gathered from this graph; since it provides us the lowest deviation rate among these samples and the time needed is same on all selections.

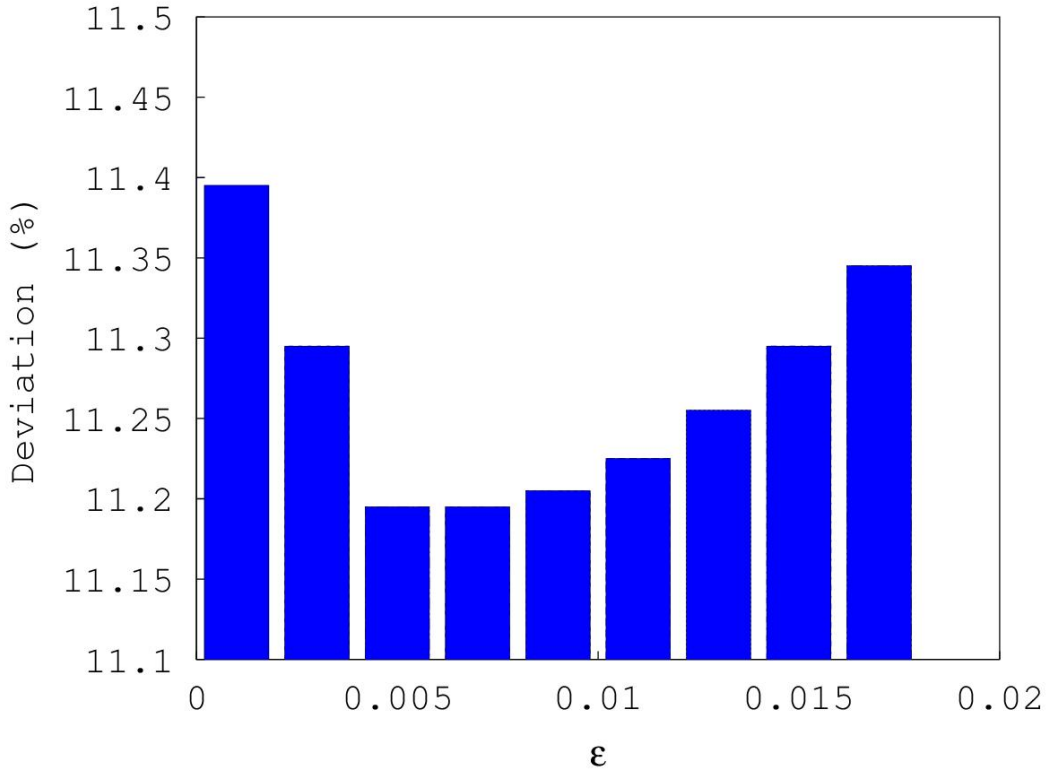

Figure 4.10: Deviation from the Optimal Solution in Time Depending on the Values of ϵ

In the Figure 4.10, we have observed the behaviour of the algorithm regarding the deviation rate, and at ϵ value of 0.005 we can observe a slightly improved deviation percentage.

4.4 Binary Bat Algorithm

Binary Bat Algorithm (BBA) [64] refers to a rather recently developed meta-heuristic computational technique that is inspired from, in this case, bats and the way they locate prey. Just like other optimization algorithms that was reproduced from natural systems, BBA also utilizes a hunter-prey logic but with an important difference; BBA does not offer an evolution mechanic in order to promote producing solutions. On the contrary, binary bat algorithm relies on something called "animal echolocation", the process where certain animals, bats among them, use their voices as some kind of a sonar in order to locate obstacles, prey or one another. Echolocation functions as a tool for the bat, who are otherwise blind, to determine their location by calculating the distance from themselves to obstacles, walls, prey or even other bats. The main feat of this meta-heuristic compared to the others is the fact that the individual, in this case the bat, can change their search space in a continuous manner by altering their cries' amplitude and frequency. This enables some form of flexibility as different positions in the search space may require larger or smaller steps in order to reach to a solution in an efficient manner. This kind of alteration plays a vital role in scenarios that require a faster rate of convergence and strict resource management.

The attributes mentioned above are what made Bat Algorithm rather popular amongst population-based solutions. In time, it has been applied to all sorts of problems, one of them is discrete binary optimisation problem. For the purposes of developing a variant of the BA for discrete solutions, since it was originally not applicable to binary problems directly, a binary version of BA was introduced to solve feature selection problems. Hence BBA was acquainted as a high-performance meta-heuristic that offers relative ease in solution finding in discrete problems. The mainframe of the algorithm remains relatively the same while some minor key changes are applied. Feature selection can be regarded as an optimisation problem because of the fact that it generally requires a smarter approach contrary to impractical exhaustive search. Swarm intelligence based algorithms are proven to function well in these problems; and BBA is easily one of the most popular among them recently. The inception of the algorithm remains the same yet in the feature selection part, the bat's position is represented by binary vectors; and it moves in the search space now remodelled as a

boolean lattice.

BBA has seven parameters which affect solution performance. These parameters are population size (PS), generation size (GS), minimum frequency (f_{min}), maximum frequency (f_{max}), velocity (V), rate of pulse (R) and loudness (A) as shown in Table 4.3. In order to analyse PS parameter effect, the 18 sensor network configuration case is selected and other parameters are fixed. Experiments are run for each generation size from the start. This algorithm's parameters are selected randomly in the run-time except of population size (PS) and generation size (GS) which are examined in determined range values.

Table 4.3: BBA Algorithm Parameters

Name	**Value**	**Parameter Explanation**
PS	90	Population Size
GS	7	Generation Size
f_{min}	[0,1)	Minimum Frequency (Uniform Real Distribution using MT19937 generator)
f_{max}	[0,1)	Maximum Frequency (Uniform Real Distribution using MT19937 generator)
V	[-1,1)	Velocity (Uniform Real Distribution using MT19937 generator)
R	[0,1)	Rate of Pulse (Uniform Real Distribution using MT19937 generator)
A	[0,1)	Loudness (Uniform Real Distribution using MT19937 generator)

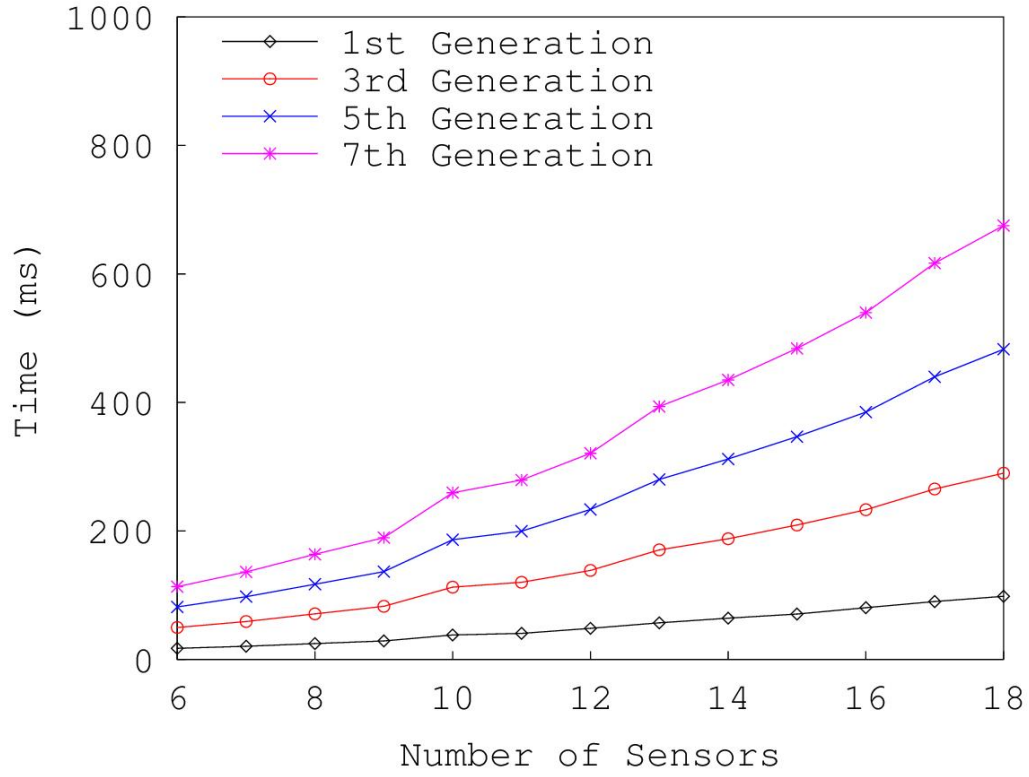

Figure 4.11: Time-to-solve the Bandwidth Packing Problem for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.11 for the Binary Bat algorithm implementation, we have once again sampled the algorithm in various generation sizes and compared how they fare against each other in scenarios where 6 to 18 sensors are implemented within the network. Yet again, we have come to a conclusion that the difference between each generation of solutions increase with respect to the number of sensors implemented within the system. We have graphed the 1st, the 3rd, the 5th and the 7th of these generations in order to further emphasize this difference in time.

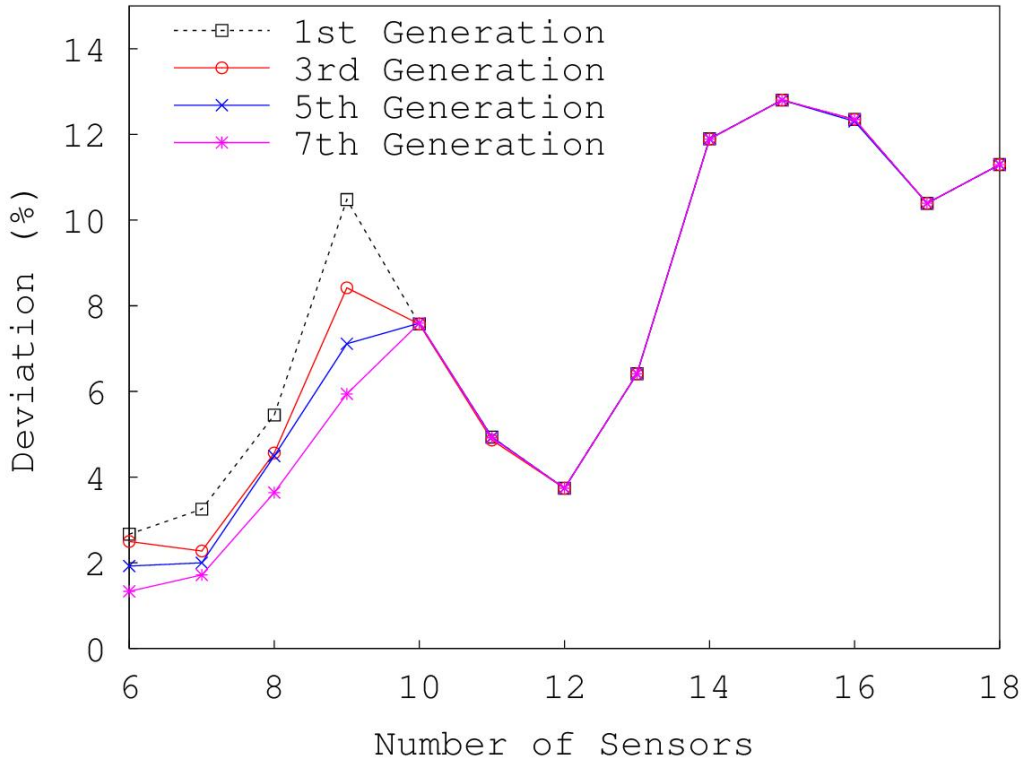

Figure 4.12: Deviation from the Optimal Solution for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.12 shows that the deviation is irrelevant with the number of sensors or the generation size. For example, right side of the 10-sensor network configuration presents same deviation results for each generation size while left side shows deviation percentage variation. We realize that comparing of 10-sensor, 12-sensor and 14-sensor network configurations shows that deviation does not have any relevance with the number of sensors in the current network configuration.

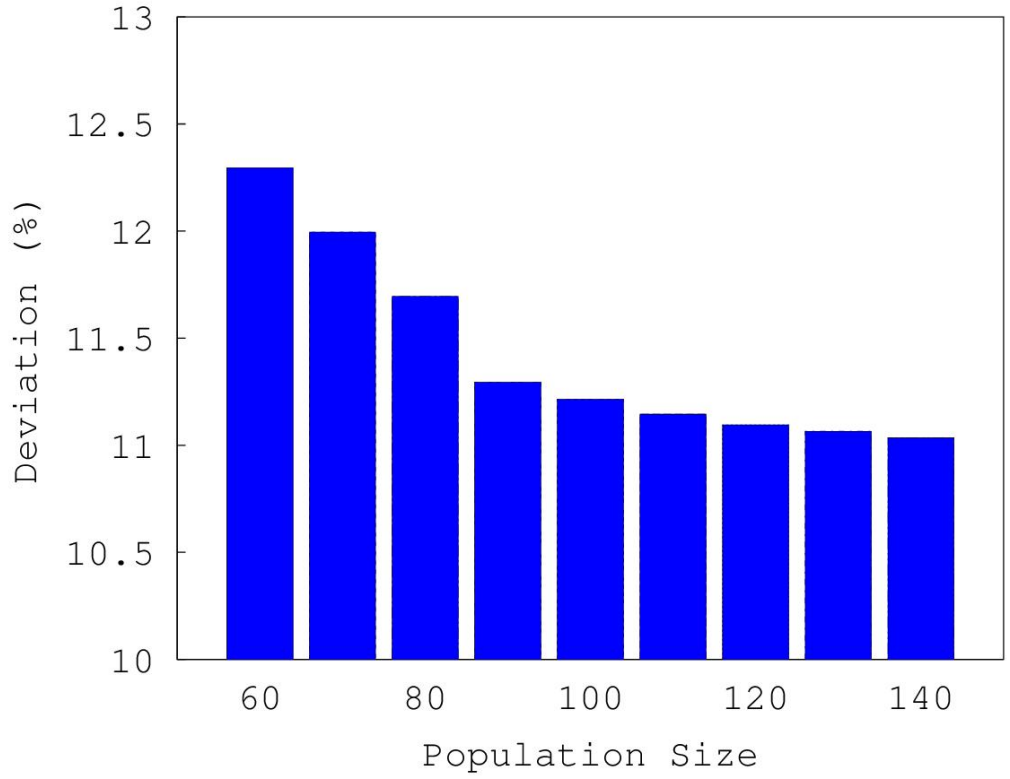

Figure 4.13: Deviation from the Optimal Solution in Time Depending on the Population Sizes

In the beginning of simulation, the 18 sensor network configuration case is selected and other parameters are fixed. In Figure 4.13, a population size of 90 could be considered as optimal value from the data we have gathered from this graph; since we observe that there is a slightly improved rate in deviation percentage.

4.5 Criss-Cross Optimization

If we were to explain what Criss-Cross Optimisation (CSO) [53] stands for in one word; it would be moderation. Inspired by the golden mean philosophy, the doctrine the middle way between two extremes is considered to be the wisest; CSO instruments the crossover operator originated from genetic algorithms into a dual search mechanism where both horizontal and vertical crossovers are supported in order to produce dissimilar offspring with the aim to secure more moderate solutions.

The individuals within the search space evolves into those of delivering increasingly more optimal solutions after each generation, where after every generation, fitter offspring takes place of the parent members. Therefore we can say that utilising a multi-dimensional crossover ability in CSO should have allowed us to achieve faster convergence rates, since it reduces the unseen areas in the search space by going in both directions. Thus by scanning the peripheral of every node that come up in the search space we'd lessen the probability of premature convergence since in most cases this was caused by not taking the necessary steps in both axes within the proximity of a node. In this scenario, when reaching a stagnant point where no progress is made after several generations, a vertical crossover can be utilised in order to jump out of the local minima of that stagnation region then a proper convergence can be achieved by once again making use of once again horizontal crossovers. This kind of criss-cross notion applied through the search space allows us to exert some kind of superiority over traditional meta-heuristic methods.

Contrary to other swarm-intelligence based meta-heuristics, CSO uses its advantage of multi-axis crossover operators by applying it after each evolutionary step twice. The updated populations are then selected by the competitive operator, which functions as a way of comparing parent and offspring individuals and decides whom to terminate.

COA has four parameters which affect solution performance. These parameters are population size (PS), generation size (GS), horizontal crossover probabilities (HP) and vertical crossover probabilities (VP) as shown in Table 4.4. In order to analyse PS, HP and VP parameters effect, the 18 sensor network configuration case is selected

and other parameters are fixed. Experiments are run for each generation size from the start.

Table 4.4: COA Algorithm Parameters

Name	Value	Parameter Explanation
PS	90	Population Size
GS	7	Generation Size
HP	1	Horizontal Crossover Probabilities
VP	0.5	Vertical Crossover Probabilities

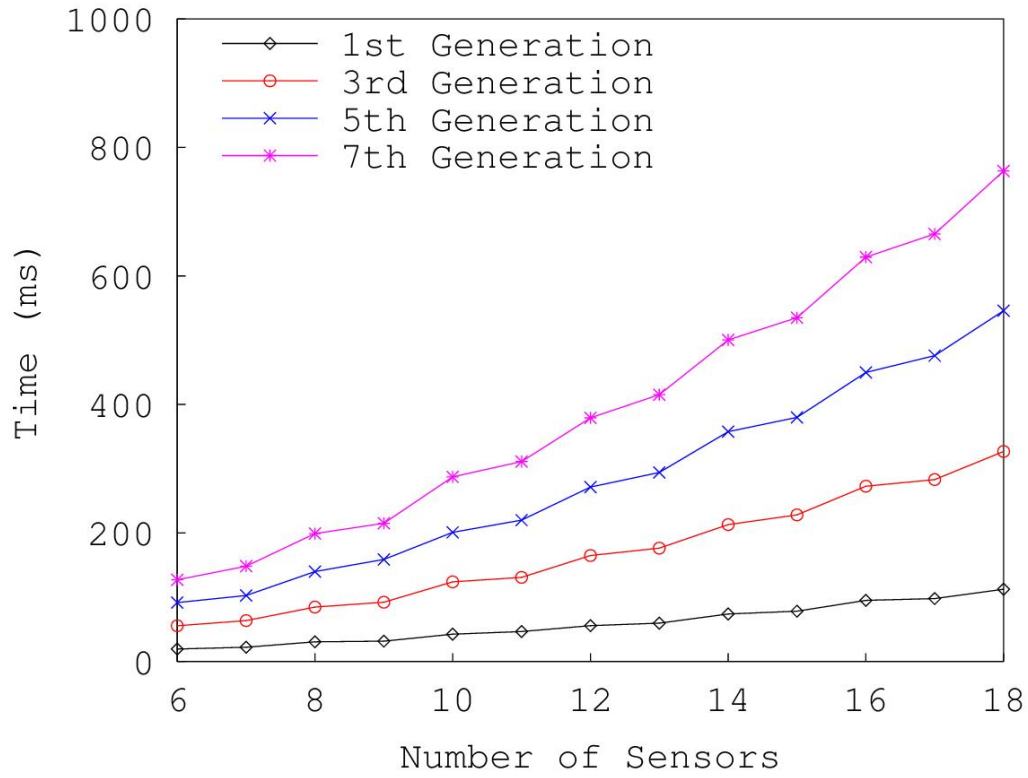

Figure 4.14: Time-to-solve the Bandwidth Packing Problem for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.14, for the Criss-Cross algorithm implementation, we have sampled the algorithm in various generation sizes and compared between implemented scenarios where 6 to 18 sensors. As a result, the difference between each generation of solu-

tions increase with respect to the number of sensors implemented within the system. Through evaluating of the 18-sensor network configuration, solution time takes about 800 ms which could be considered barely applicable for real life systems.

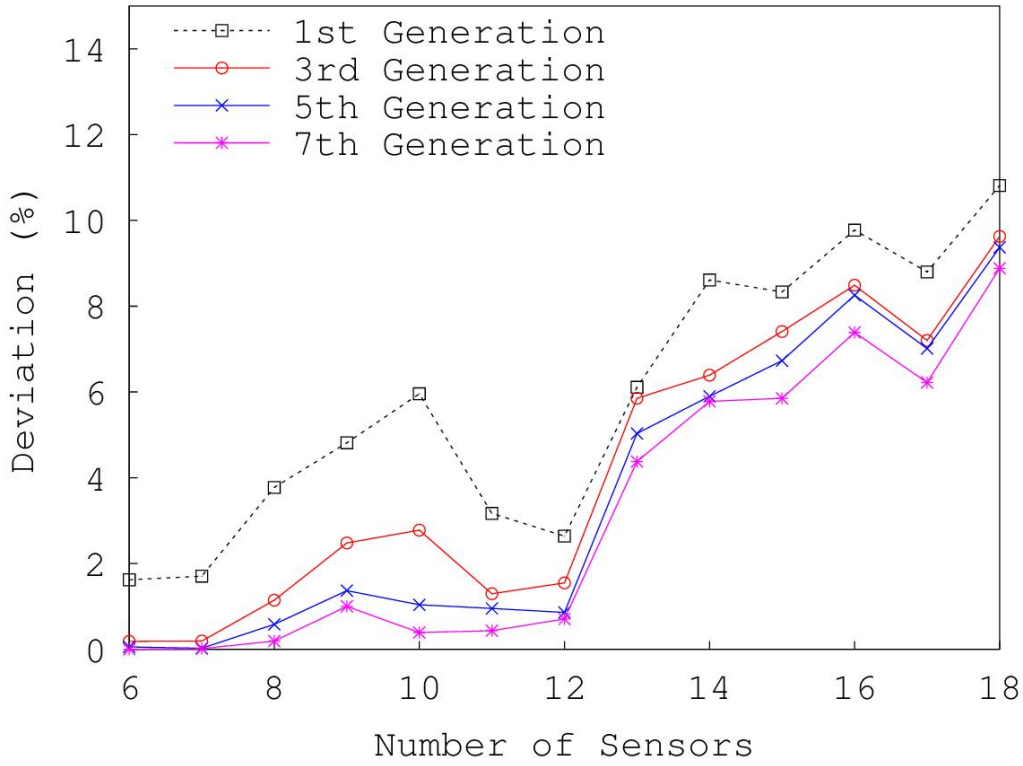

Figure 4.15: Deviation from the Optimal Solution for Various Generation Sizes Depending on the Number of Sensors

Figure 4.15 shows that the deviation could be considered as irrelevant with the number of sensor nor the generation size. For example, a 10-sensor network configuration presents better deviation results for each generation size when compared with the 12-sensor network configuration. We can also note that; while the deviation percentage can be seen as decreasing while moving towards higher generation sizes at each and every number of sensor configuration, the variance is not linear and therefore does not have relevance.

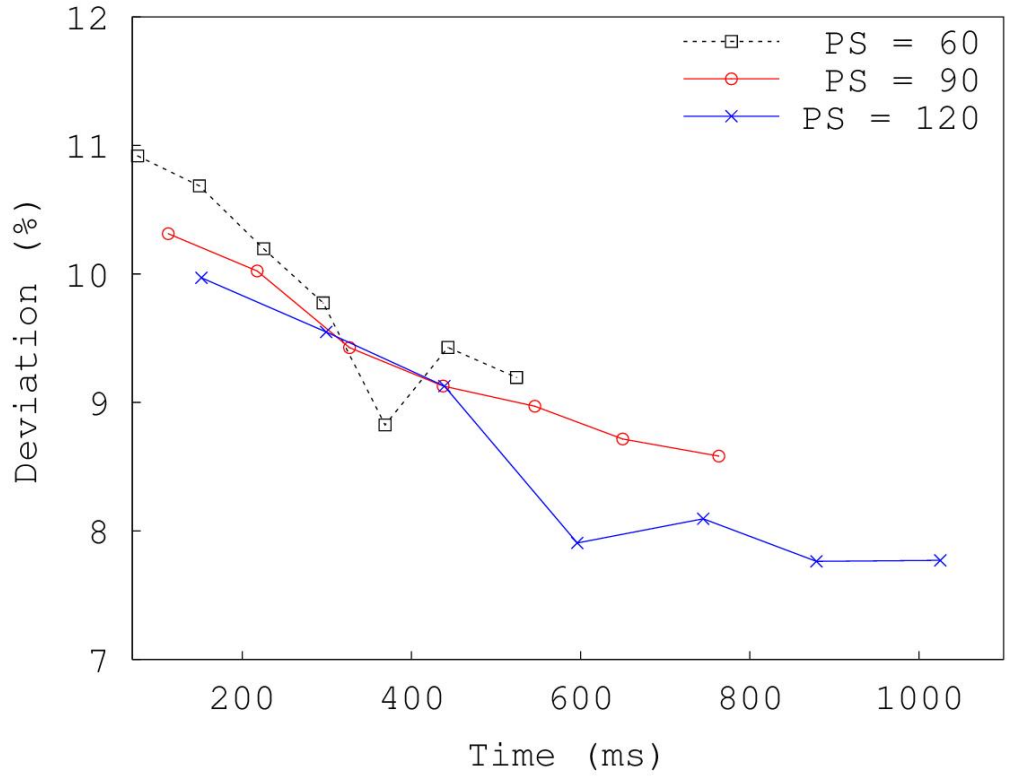

Figure 4.16: Deviation from the Optimal Solution in Time Depending on the Population Sizes

In Figure 4.16, as we can expect; COA meta-heuristic also behaves the same as with the ACO case where increased population size improves the deviation rate to some extent; meanwhile taking significantly longer to process.

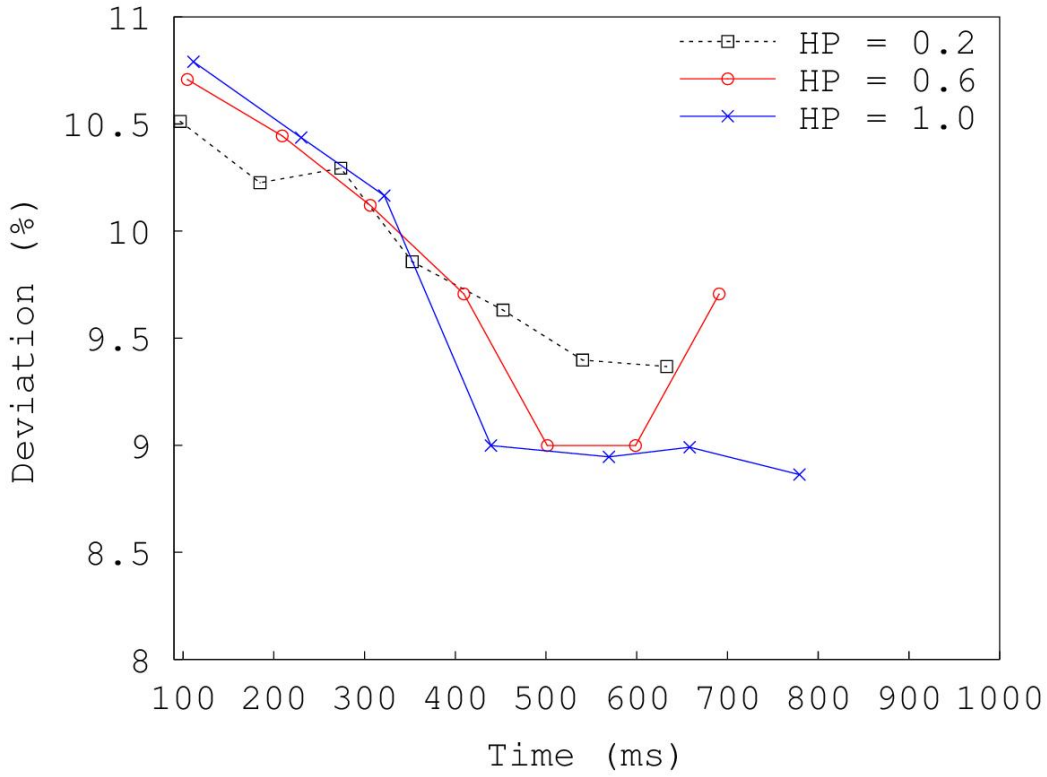

Figure 4.17: Deviation from the Optimal Solution in Time Depending on the Horizontal Probabilities

In Figure 4.17, we realize that Criss-Cross algorithm does not show any characteristic behavior according to the HP parameter. For example, deviation percentage is higher in some HP values while it is lower in others. Therefore, we cannot say that it is directly correlated. Thus a horizontal probability value of 0.6 was selected taking both deviation percentage and the time needed to concern.

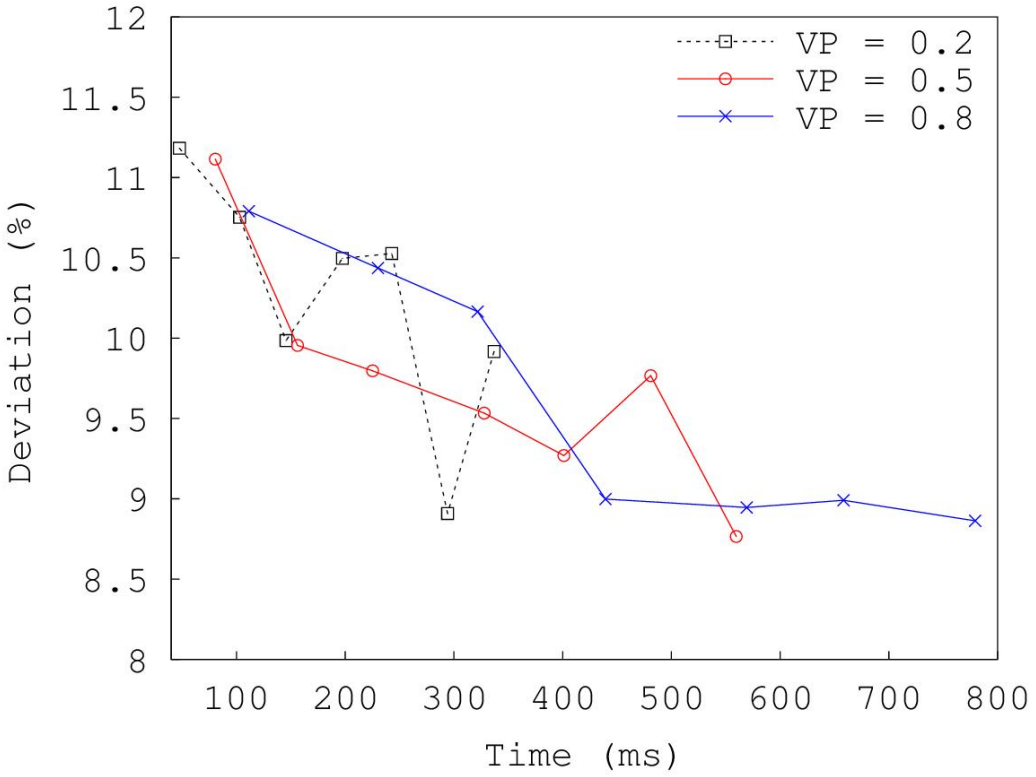

Figure 4.18: Deviation from the Optimal Solution in Time Depending on the Vertical Probabilities

In Figure 4.18, we realize that Criss-Cross algorithm does not show any characteristic behaviour according to the VP parameter. For example, deviation percentage is higher in some VP values while it is lower in others. Therefore we cannot say that it is directly correlated. Thus a 0.5 of vertical probability was selected taking both deviation percentage and the time needed to concern.

4.6 Genetic Algorithm

Lastly we have a genetic algorithm (GA) [15] based heuristic developed specifically for the multidimensional knapsack problem (MKP for short). If we were to elaborate on how these terms hold any significance, it should be noted that genetic algorithms have been proven to present efficient solutions to MKP time and time again. Genetic algorithms are series of processes in which the possible members that would be eligible to a solution to an optimization problem are "evolved" into more viable individuals as the system progresses and eventually converges. This kind of iterative behaviour is key on almost all heuristic and meta-heuristic optimization algorithms as we have expatiated on previous sections. Another common trait between these algorithms is the "survival-of-the-fittest" rationale, whereafter each generation, individuals are evaluated by their perseverance, meaning how they have had fared in the last evolutionary cycle towards the solution. Members are then "rewarded" by their performance in terms of updating their "fitness" levels, which in turn increases the chance to reach into an actual solution. Thus in each generation, members develop and adapt to the task at hand, which is a desired situation for iterative solutions. After initialization and representation problems are dealt with, one may put use the primary operators of GA: selection, crossover and mutation operators. In the parent selection phase, the logic is to give priority to more successful individuals, allowing them to continue on to the next phase as candidates of crossover operations. Whether the individual is worthy of crossover is resolved by their fitness levels, i.e. how well they fared in reaching a solution. Members that have proven to be more efficient than others are chosen from the population via the selection operator and delivered onto the next phase, crossover operation. In this part of the algorithm, individuals with high fitness levels are selected two by two, and a crossover site amongst their bit strings is randomly chosen. Selected bit sections are then transfered between the two strings in order to form a genetic offspring. This process concludes the crossover. Then there's the mutation operator, which is a probabilistic process in order to maintain diversity among fitter individuals and can be applied with selection operators in order to make use of random walks through the search space. This serves as a way to balance between deterministic and probabilistic attitudes in order to achieve a fast and efficient way to solve discrete problems. Genetic algorithms for solving MKP problems

have been proven to be profitable due to its optimal use of selection, crossover and mutation operators. The processes have been designed to deliver inevitable optimal solutions for many problems.

GA has four parameters which affect solution performance. These parameters are population size (PS), generation size (GS), parent pool size (PPS) and mutation probability (MP) as shown in Table 4.5. In order to analyse PS, PPS and MP parameters effect, the 18 sensor network configuration case is selected and other parameters are fixed. Experiments are run for each generation size from the start.

Table 4.5: GA Algorithm Parameters

Name	**Value**	**Parameter Explanation**
PS	60	Population Size
GS	7	Generation Size
PPS	8	Parent Pool Size
MP	0.5	Mutation Probability

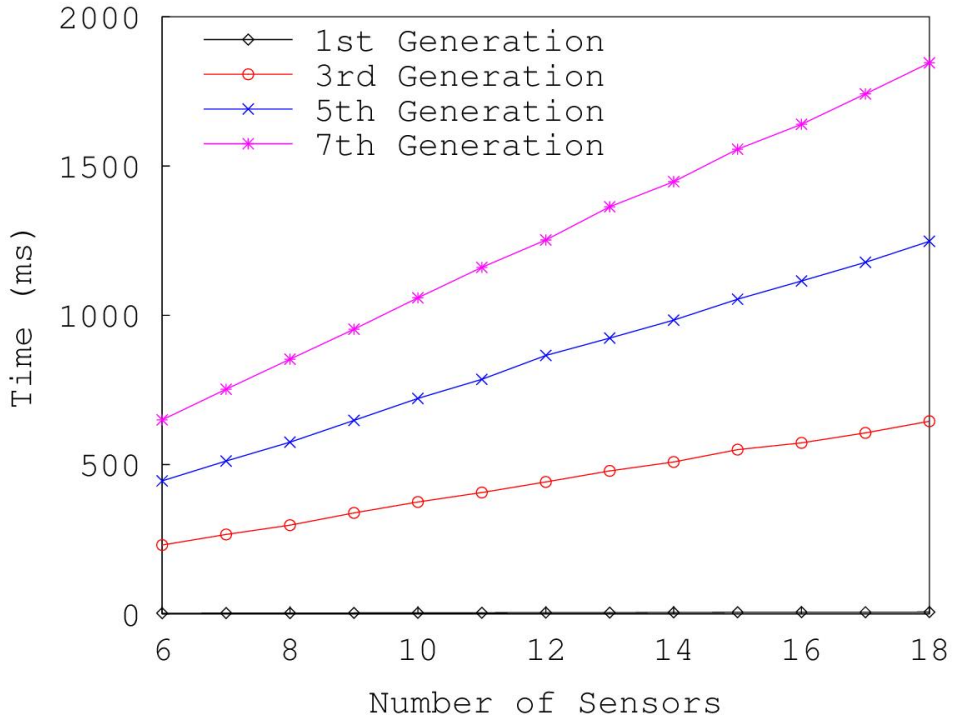

Figure 4.19: Time-to-solve the Bandwidth Packing Problem for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.19, the algorithm is sampled in various generation sizes and compared between 6 to 18 number of sensors. As a result, the difference between each generation of solutions increase with respect to the number of sensors implemented within the system. Through evaluating of the 18-sensor network configuration, solution time takes more than 500 ms which could be considered hardly applicable for real life systems.

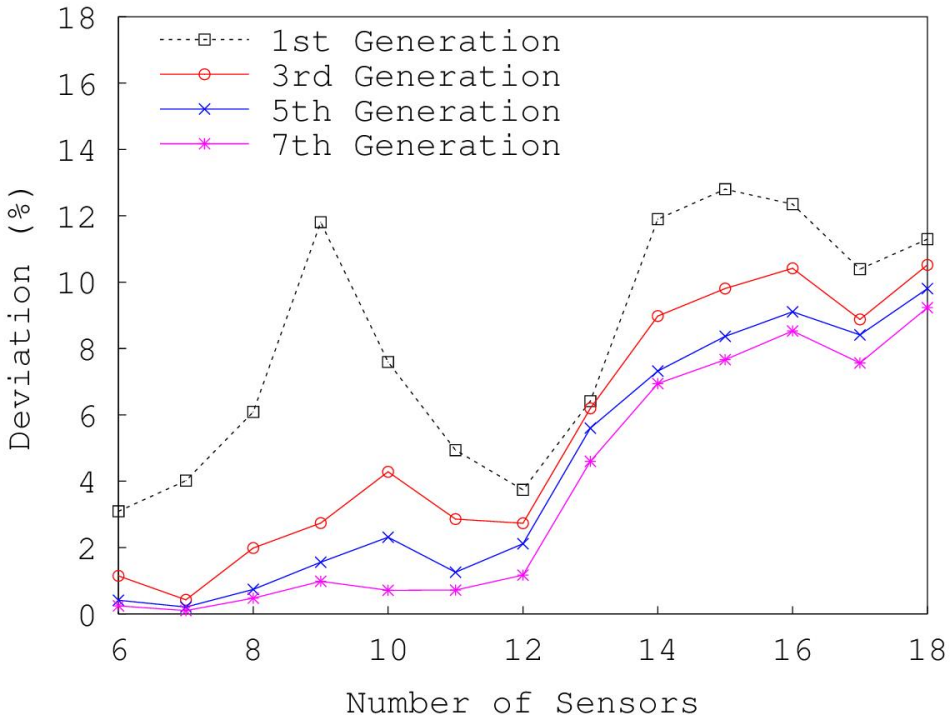

Figure 4.20: Deviation from the Optimal Solution for Various Generation Sizes Depending on the Number of Sensors

In genetic algorithm case Figure 4.20; we can see sharp increases in deviation rate on almost all numbers of sensors on the first iteration; while the graph becomes smoother after each generation. The same general manner of the graph is relatively the same on all iterations though, deviation rate inevitably becomes a growing concern to the algorithm as the number of sensors increase in the implementation of the network.

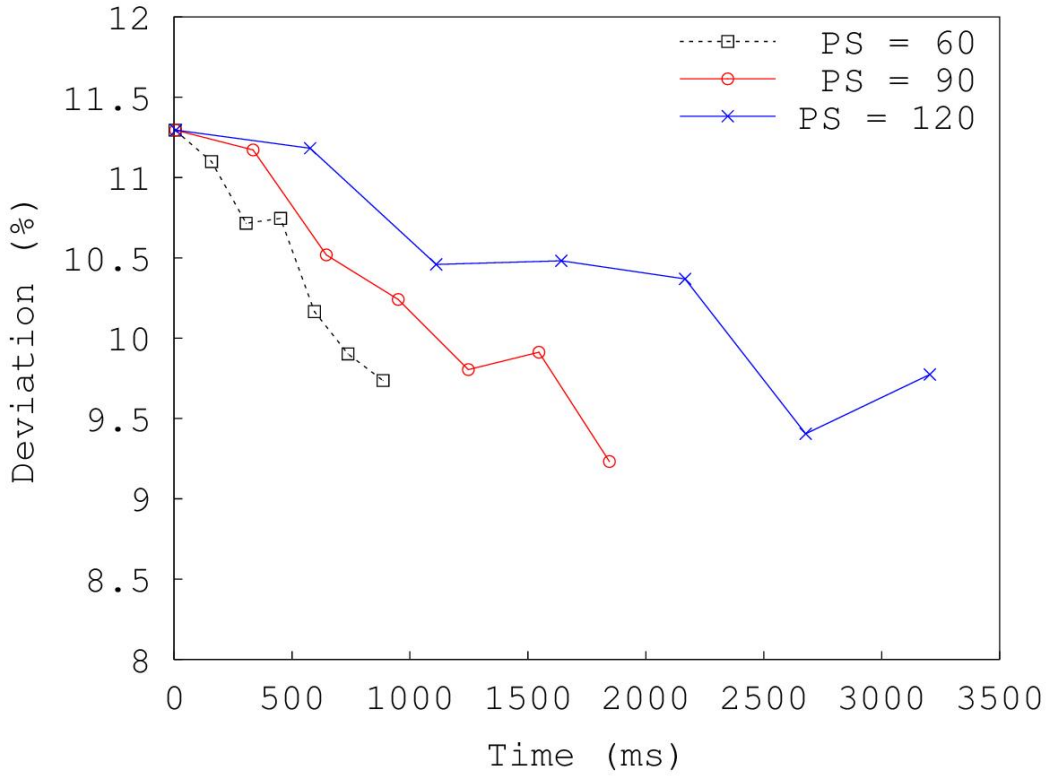

Figure 4.21: Deviation from the Optimal Solution in Time Depending on the Population Sizes

According to Figure 4.21, genetic algorithm population size parameter shows better performance at value of 60 compared with the population sizes of 90 and 120.

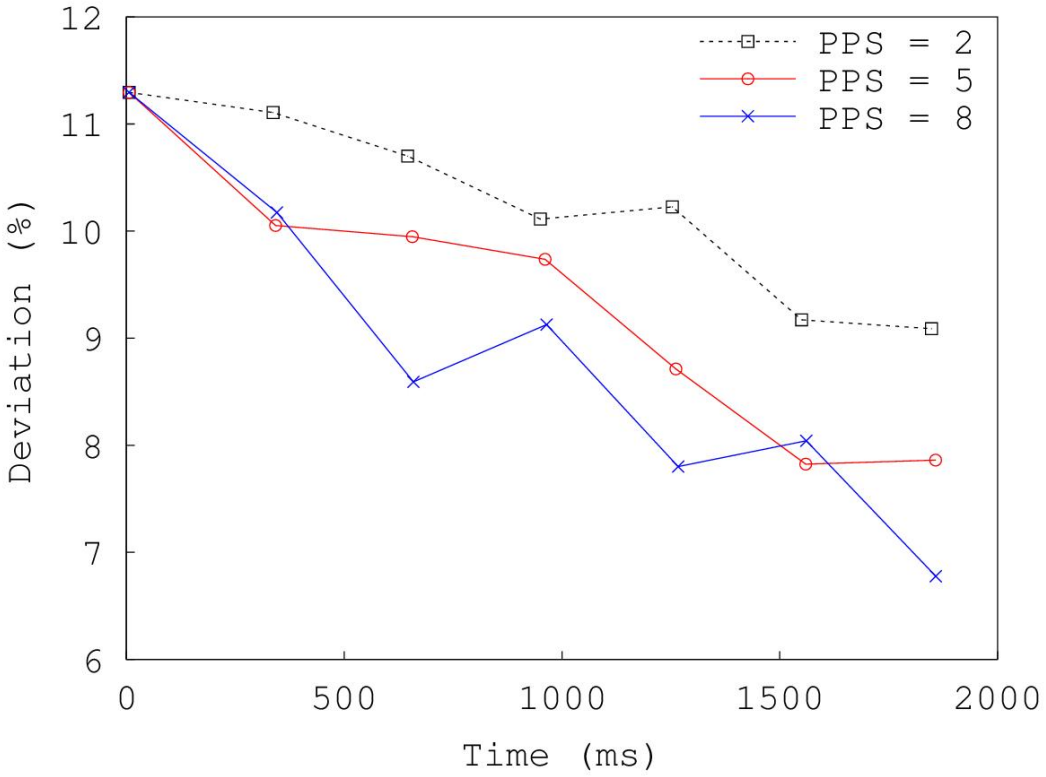

Figure 4.22: Deviation from the Optimal Solution in Time Depending on the Parent Pool Sizes

In Figure 4.22 parent pool size plays an important role in genetic algorithms as can be seen from the graph, it directly effects the deviation percentage; in which the decrease from 11 percent to 7 percent indicates an improvement of almost a half from the starting point.

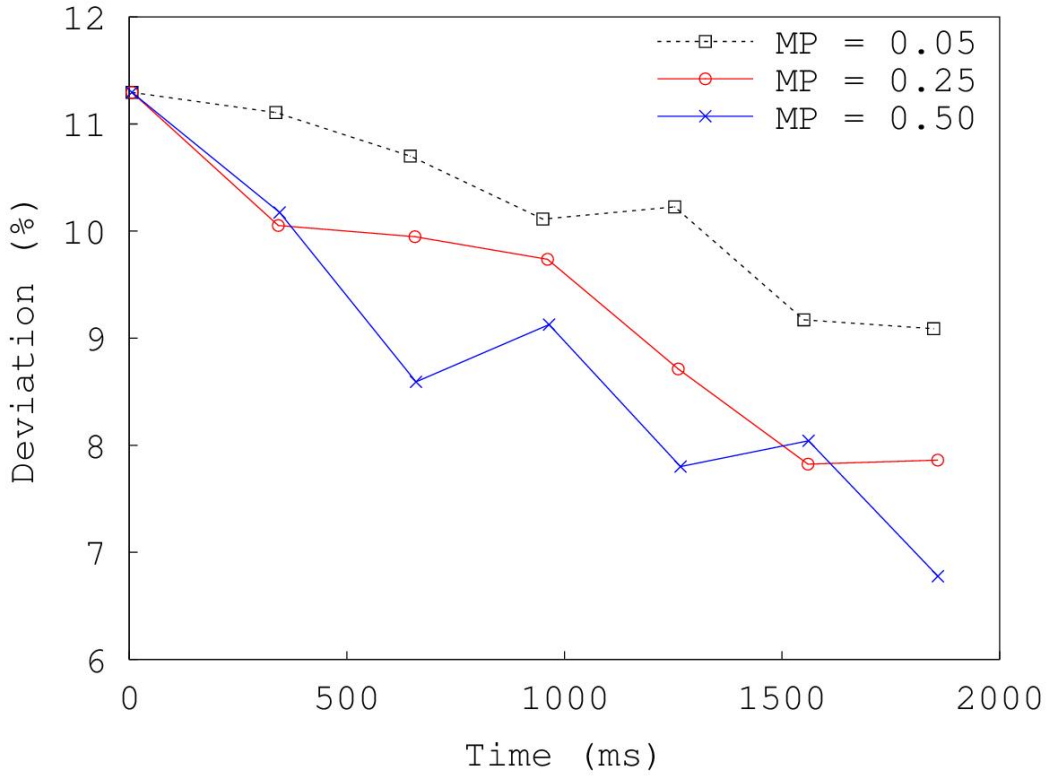

Figure 4.23: Deviation from the Optimal Solution in Time Depending on the Mutation Probabilities

In Figure 4.23, the mutation probability (MP) indicates, as the name suggests; the coefficient that decides the probability of a mutation after each evolutionary step. The optimal value for mutation probability is selected as 0.50 in this case since it behaves the most consistent along the timeline even though mutation probability of 0.25 line may appear better in some cases.

4.7 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) [29], has to be one of the most commonly used optimization application in the field of swarm-intelligence based algorithms. It was instigated from swarm or flock behaviours in nature, when Russell Eberhart and James Kennedy started simulating bird flocks' murmuration in computer environment. This algorithm was later discovered to be effective in finding optimal solutions in search spaces, as many swarm-intelligence algorithms are. There are many PSO variants developed since then, but many of them has the same structure as the original. If we were to elaborate, the string of processes consists of basically manipulating the trajectories of individuals searching for solutions in the search space along the way in correlation with the current location and heading of the member who is the closest to the solution at any given time. In this case, the individuals are called "particles". The movement of a swarming particle along the search space can be defined by a stochastic component and a deterministic component. This balance in randomization and definition tends to function well in solving for the optimal. Each particle determines their next move in the search space by comparing its location with the current location of the member that is closest to the solution, as in the flock example; where birds chirp louder if they are near to a food source and the rest of the flock swirls around it. The goal is to improve this location to an optimal solution until no more progression can't seem to be made after certain number of generations. This behaviour inevitably brings the swarm closer and closer to the target, as one can expect. PSO is a relatively simple algorithm due to the fact that there is no need for encoding or decoding into binary strings, unlike Genetic Algorithms due to the fact that it can also work with real numbers. It can also be implemented in both continuous and discrete cases; because positions and velocities can be discretized. There are studies that show that PSO algorithms can outperform genetic algorithms in solving many optimization problems, mainly because of its ability to update their members towards a common goal regularly and globally.

PSO has three parameters which affect solution performance. These parameters are population size (PS), generation size (GS) and learning factor (LF) as shown in Table 4.6. In order to analyse PS and LF parameters effect, the 18 sensor network configuration case is selected and other parameters are fixed. Experiments are run for each generation size from the start.

Table 4.6: PSO Algorithm Parameters

Name	Value	Parameter Explanation
PS	90	Population Size
GS	7	Generation Size
LF	2	Learning Factor

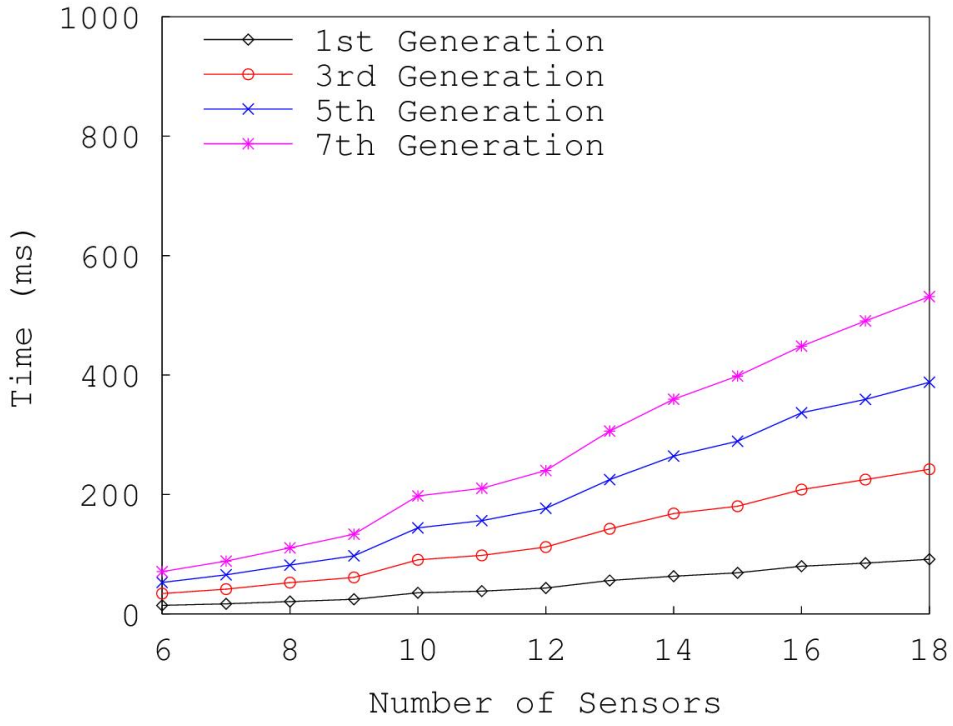

Figure 4.24: Time-to-solve the Bandwidth Packing Problem for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.24, PSO algorithm has behaved similar to their counterparts; we can observe that, like the ones before, the difference between the elapsed times of each

generation increases as more sensors are added to the network.

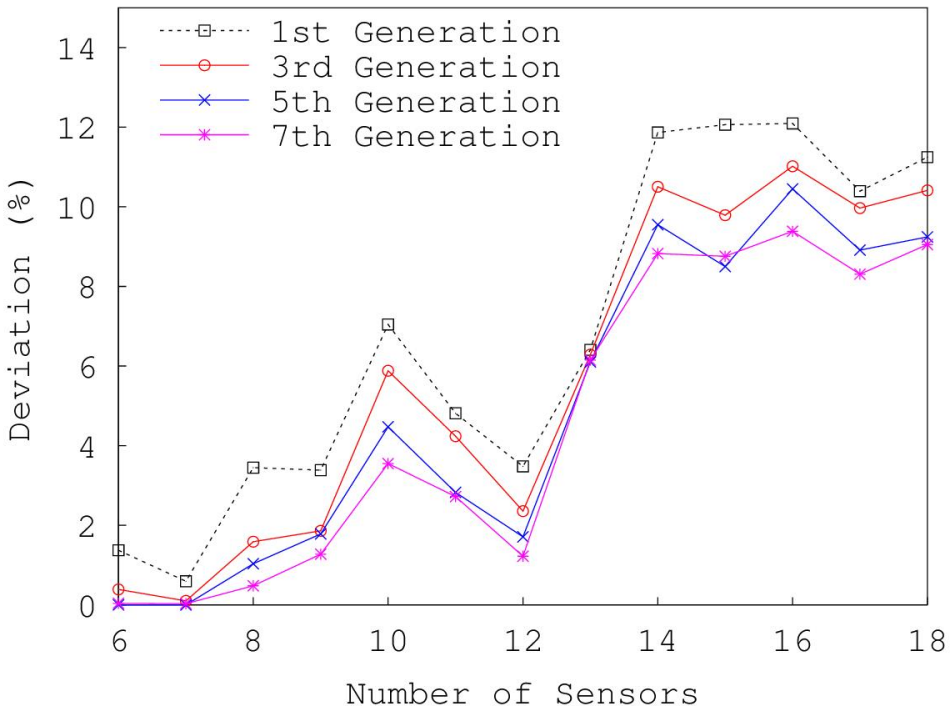

Figure 4.25: Deviation from the Optimal Solution for Various Generation Sizes Depending on the Number of Sensors

In Figure 4.25, the deviation rate decrease after each iteration in PSO, as expected. There are instances where the lower generation produces better results, but these can be discarded as the general trend is higher the iteration, better the results.

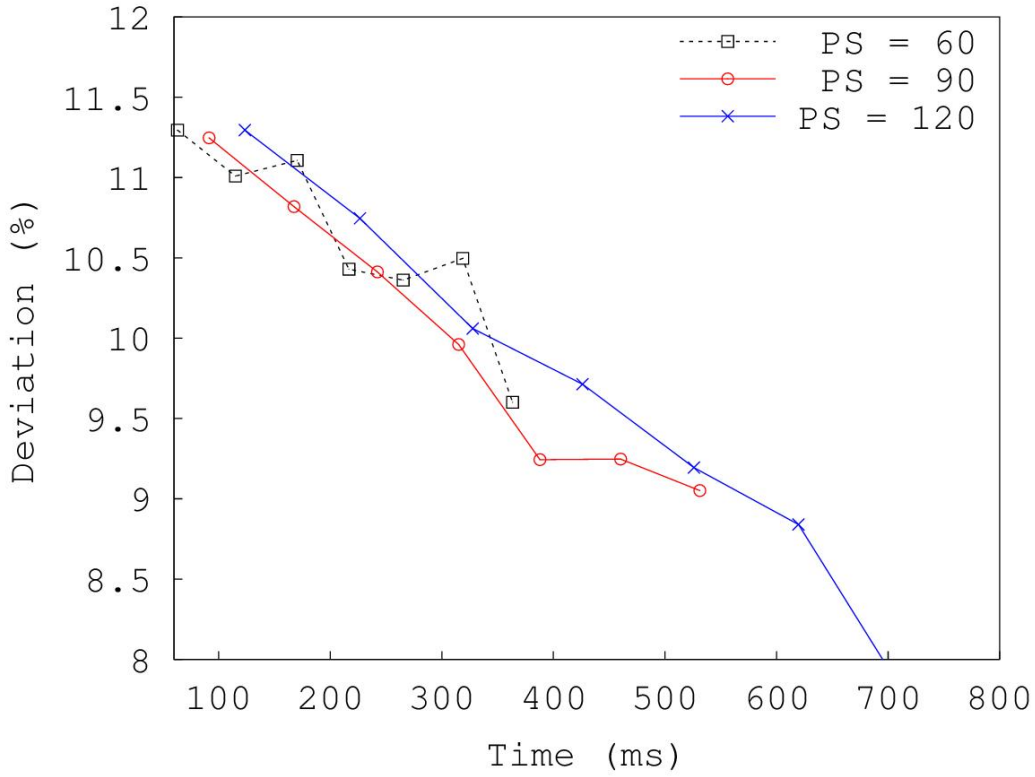

Figure 4.26: Deviation from the Optimal Solution in Time Depending on the Population Sizes

In Figure 4.26, as with the other algorithms, population size acts the same the to general process of solution finding on almost all of the bio-inspired mechanisms. The optimal point is proven to be a population size of 90 from this graph because it provides an optimal spot for our needs regarding lower deviation rates and excellent solution time. During the simulation, the 18 sensor network configuration case is selected and other parameters are fixed.

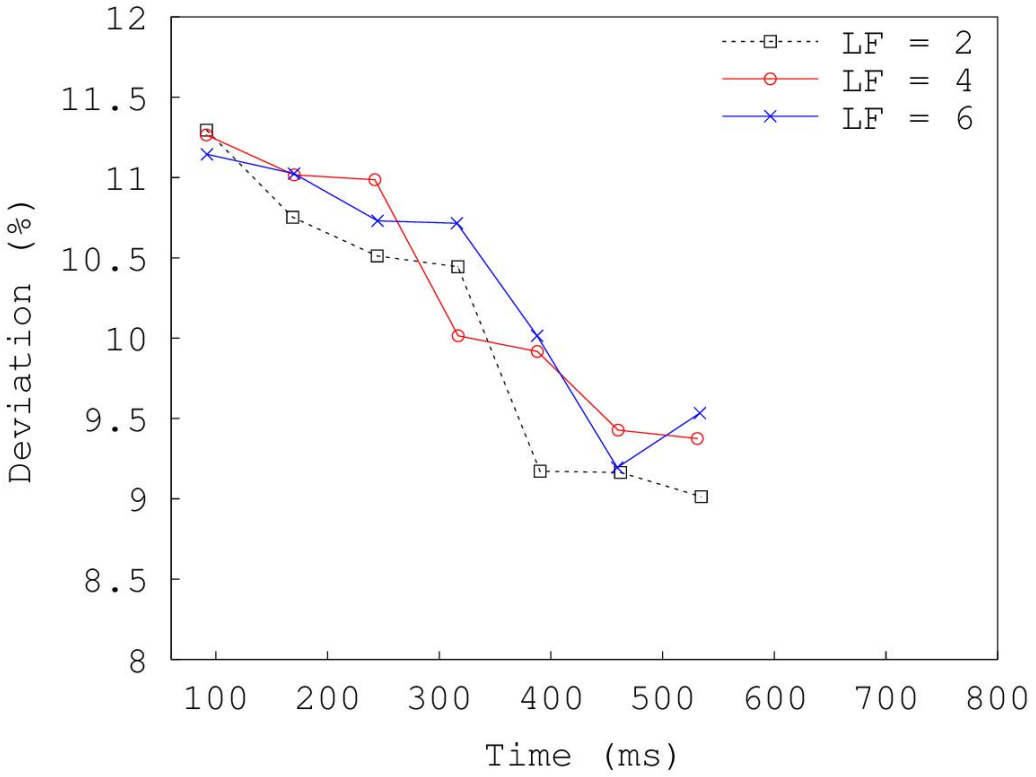

Figure 4.27: Deviation from the Optimal Solution in Time Depending on the Learning Factors

Figure 4.27 shows that in algorithms that encompasses different selected learning factor values should be examined in parts; mainly because there are some regions of time where lower values of learning factor procures better results as can be seen in between 250 ms to almost 350 ms; the overall rate of deviation is lower in learning factor of 2. During the simulation, the 18 sensor network configuration case is selected and other parameters are fixed.

CHAPTER 5

RESULTS AND DISCUSSION

Meta-heuristic and exact algorithms are implemented in C++. Data set is constructed using the sensor information table in Chapter 1 for network configurations with the implementation of 6 to 18 sensors. Each sensor has four different discrete codec settings. Initially, we have ran a brute force mechanism in order to find the optimal priority value, the elapsed time and the number of deviation for that run for each data set constructed. Afterwards, each data set is run for 500 times, while selecting the bandwidth capacity randomly between 20 Kbps and 110 Kbps, which was determined in network capacity measurement section. In order to solve for a near optimal configuration cases in relatively short time, generation size of meta-heuristic algorithms is limited to seven times. Elapsed time and the number of deviations are calculated for each generation of the data set for the current number of sensors. Meta-heuristic algorithm parameters are selected optimally and was fixed during the entire run time.

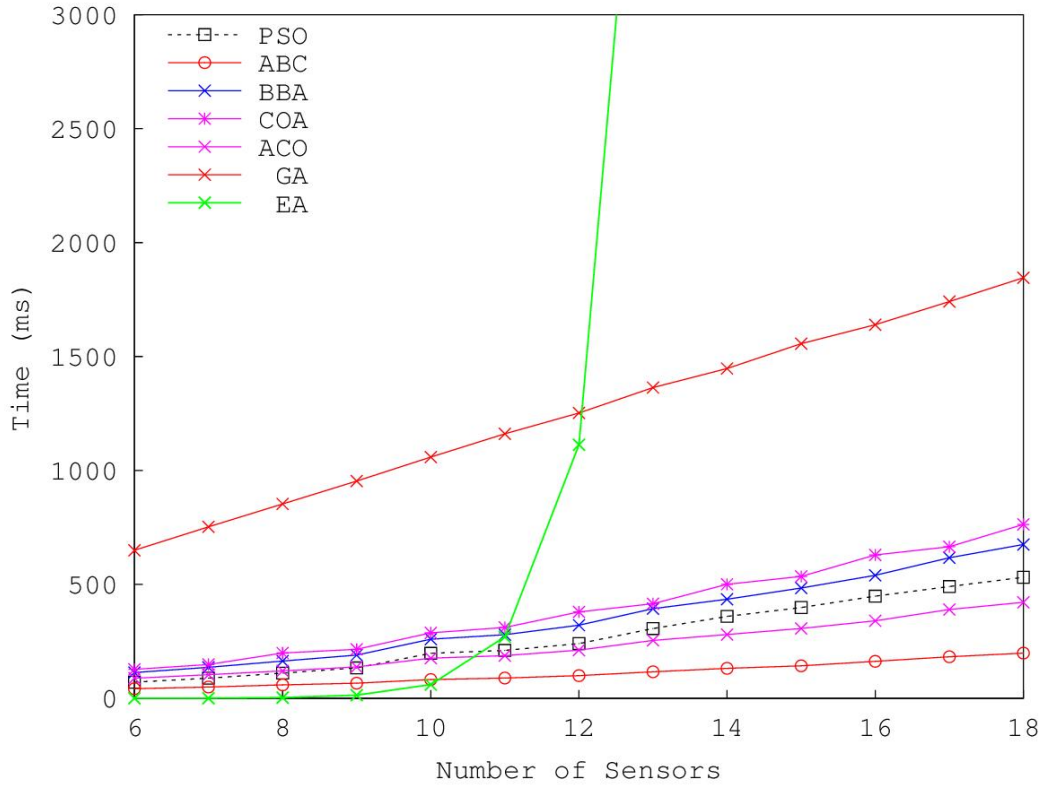

Figure 5.1: Time-to-solve the Bandwidth Packing Problem For Each Algorithm Depending on the Number of Sensors

Figure 5.1 are generated with using a 18-sensors network configuration and seven times generated solution. These show that the exact algorithm works perfectly with zero deviation until we reach a network configuration constructed with 10-sensors configuration. Moreover, the exact algorithm could be evaluated more efficiently due to zero deviation and reasonable time elapsed for each generation at 11-sensors configuration. However, after 12-sensors configuration, the exact algorithm takes a large amount of time to find out the optimal solution, therefore it cannot be applicable for real-time network systems. Meta-heuristic algorithms present much better performance after 12-sensors configuration compared to the exact algorithm. On the other hand, ABC is the quickest algorithm in terms of solve time.

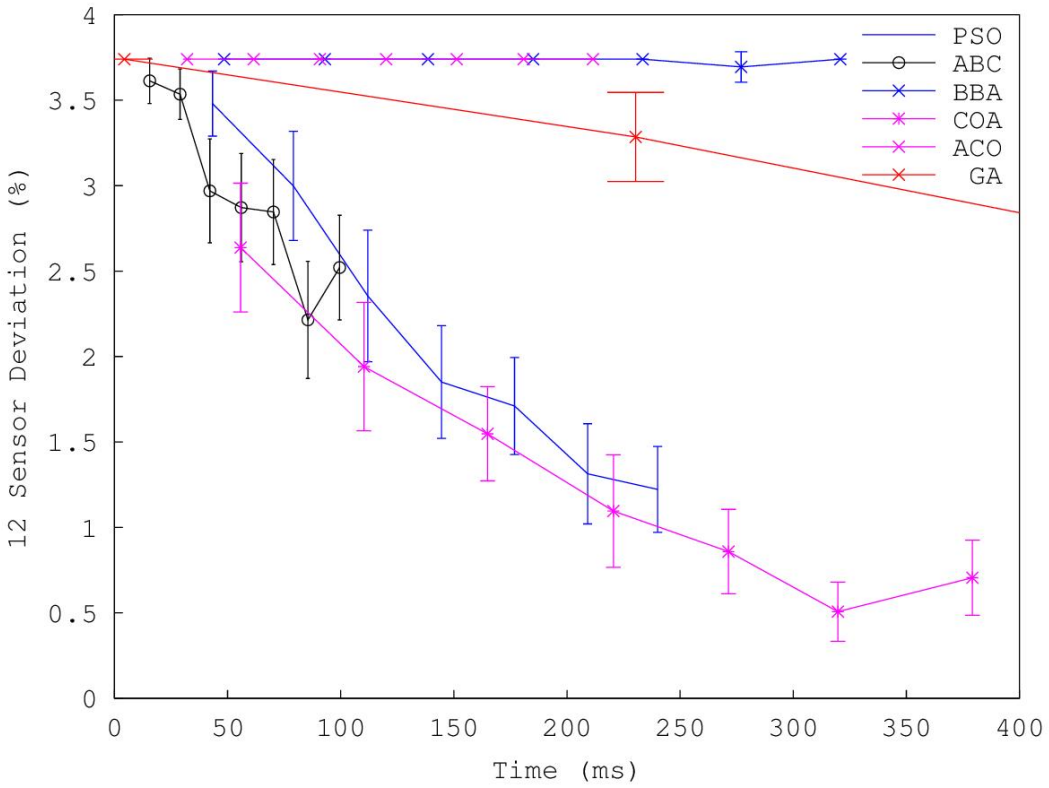

Figure 5.2: Deviation from the Optimal Solution in Time for the 12-Sensors Configuration

As we can observe from the graphs at Figure 5.2, the Artificial Bee Colony, Criss-Cross Optimisation and the Particle Swarm Optimisation are observed to have similar rates of convergence when compared to Binary Bat Algorithm, Ant Colony Algorithm, and the Genetic Algorithm in this scenario. In some cases, we may have observed results in COA that have deviated less from the exact solution then their counterparts; this algorithm is rather slow to acquire the near optimal results when compared to the ABC. In the process of result evaluation, we have observed that the deviation amount from the solution changes rather slowly in large intervals of time. When we take into consideration all the deviation-calculated time results, we have come to a conclusion that the deviation rate, when compared to the time samples used in this scenario can be regarded as a trivial parameter.

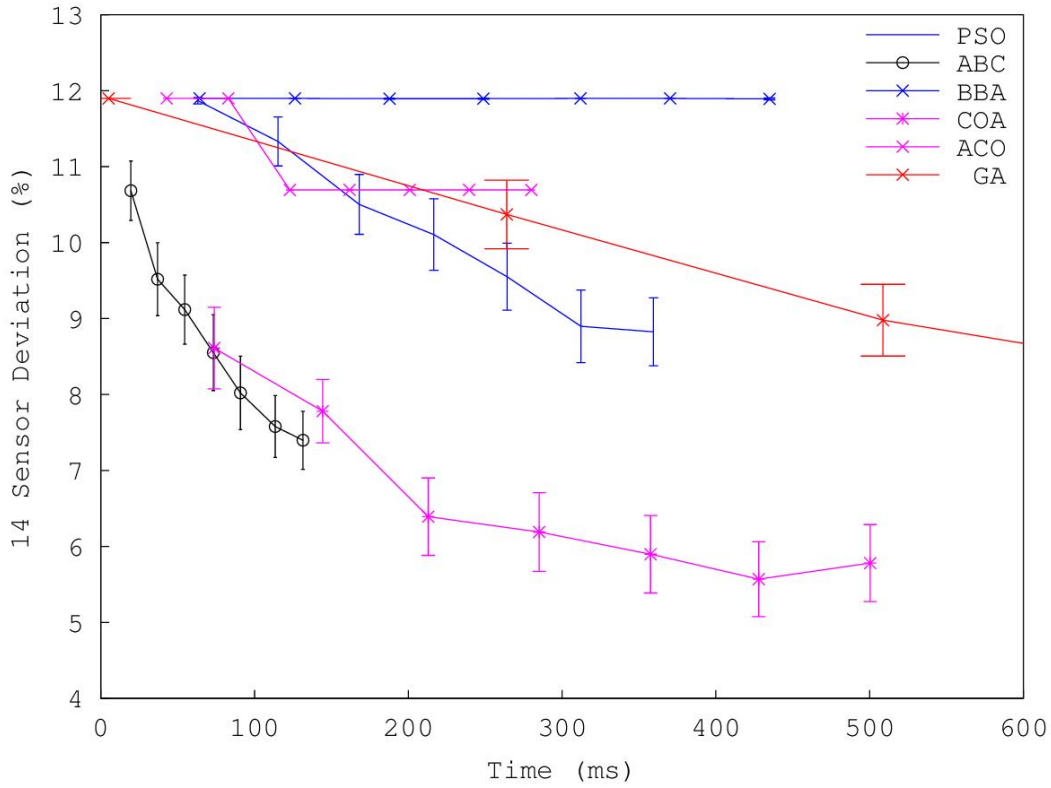

Figure 5.3: Deviation from the Optimal Solution in Time for the 14-Sensors Configuration

Figure 5.3 shows us that the Artificial Bee Colony and the Criss-Cross Optimisation has a much higher rate of convergence when compared to Binary Bat Algorithm, Ant Colony Optimisation, Particle Swarm Optimisation and Genetic Algorithm in this scenario. In some cases, we may have observed results in COA that have deviated less from the exact solution then the ABC; this algorithm is rather slow to acquire the near optimal results when compared to the latter. In the process of result evaluation, we have observed that when compared to 12 sensor configuration case, the ABC and COA's performance thrives in terms of the deviation amounts from the solution. When we take into consideration all of the deviation-calculated time results, we have come to a conclusion that the deviation rate, when compared to the time samples used in this scenario can be regarded as a trivial parameter.

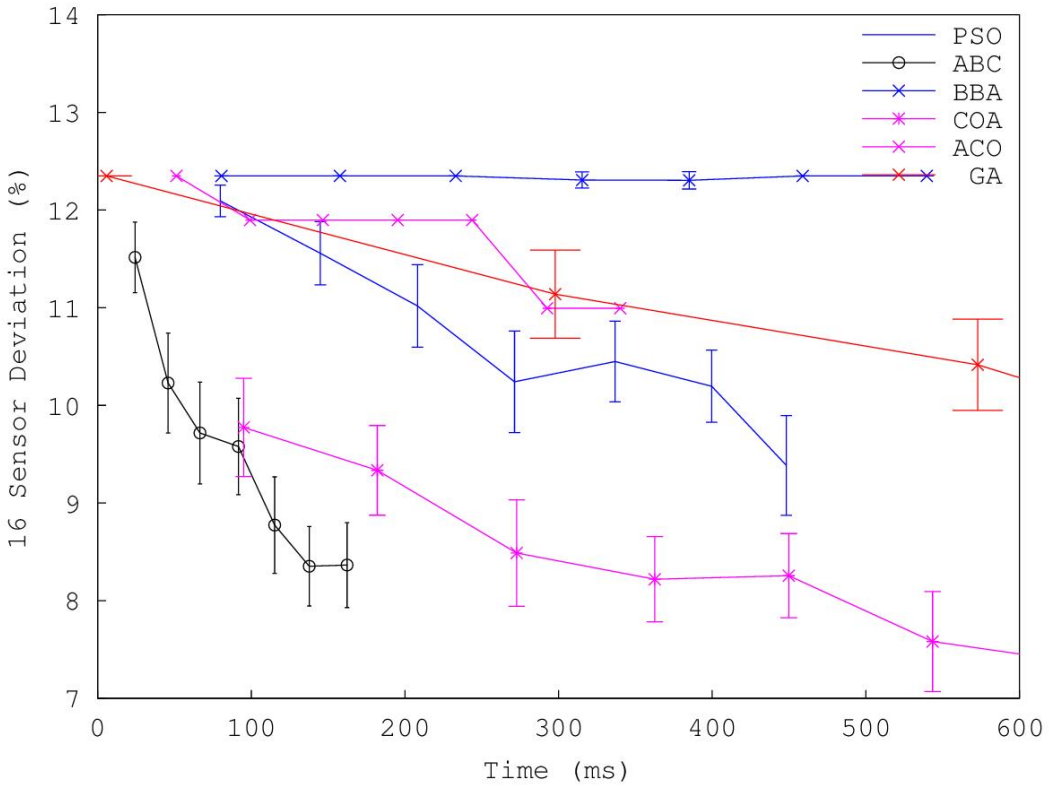

Figure 5.4: Deviation from the Optimal Solution in Time for the 16-Sensors Configuration

In Figure 5.4, the Artificial Bee Colony starts to perform relatively better in terms of solution deliverance time and convergence rate when compared to Binary Bat Algorithm, Criss-Cross Optimisation, Particle Swarm Optimisation, Ant Colony Algorithm and Genetic Algorithm the more we increase the number of sensors in the scenario. It may have been observed in smaller network sizes that COA competes with ABC in deviation rates and solution times; this algorithm is rather slow to acquire the near optimal results when compared to the ABC.

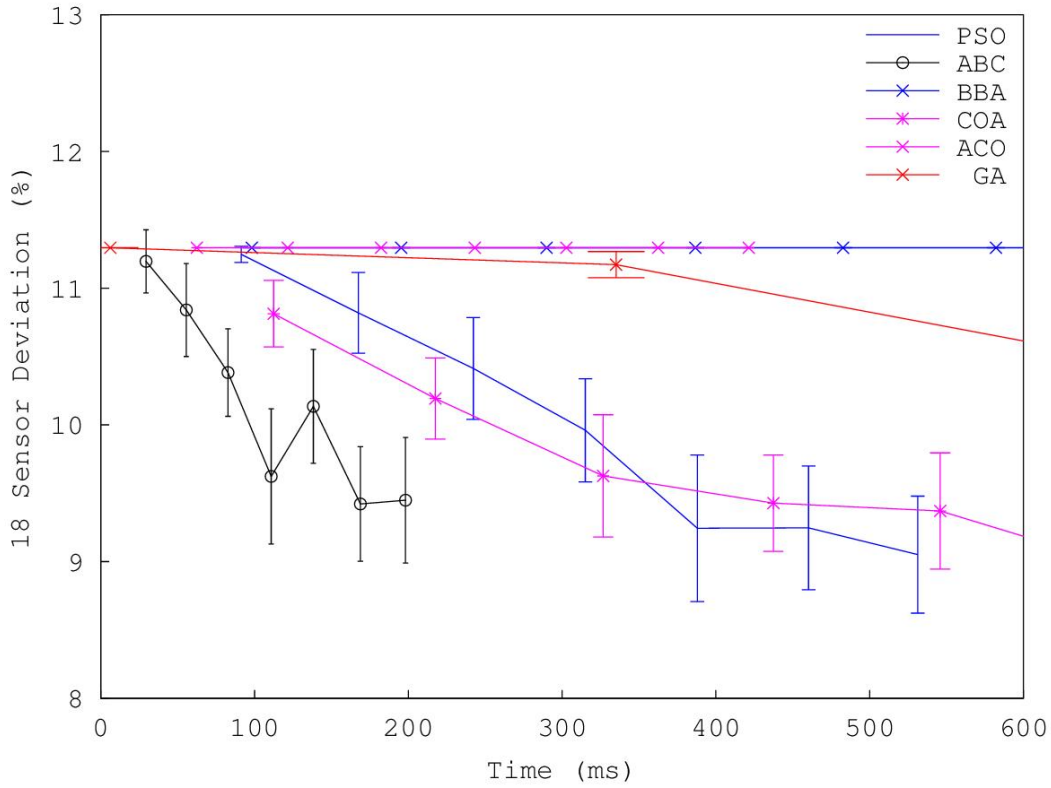

Figure 5.5: Deviation from the Optimal Solution in Time for the 18-Sensors Configuration

As we can observe from the graphs at Figure 5.5, the Artificial Bee Colony clearly outperforms others in convergence rate when compared to Binary Bat Algorithm, Criss-Cross Optimisation, Ant Colony Optimisation, Particle Swarm Optimisation and Genetic Algorithm in this scenario. When we take into consideration the time management of these algorithms, the ABC remains the best option. Thus; as the number of sensors employed within the wireless sensor network rises, other algorithms pale in comparison even though they perform similarly in smaller network sizes. Due to its performance as number of sensors increase, ABC proves to be the best choice among these algorithms in terms of its applicability in real life implementations of WMSNs when taking into account its relatively faster convergence.

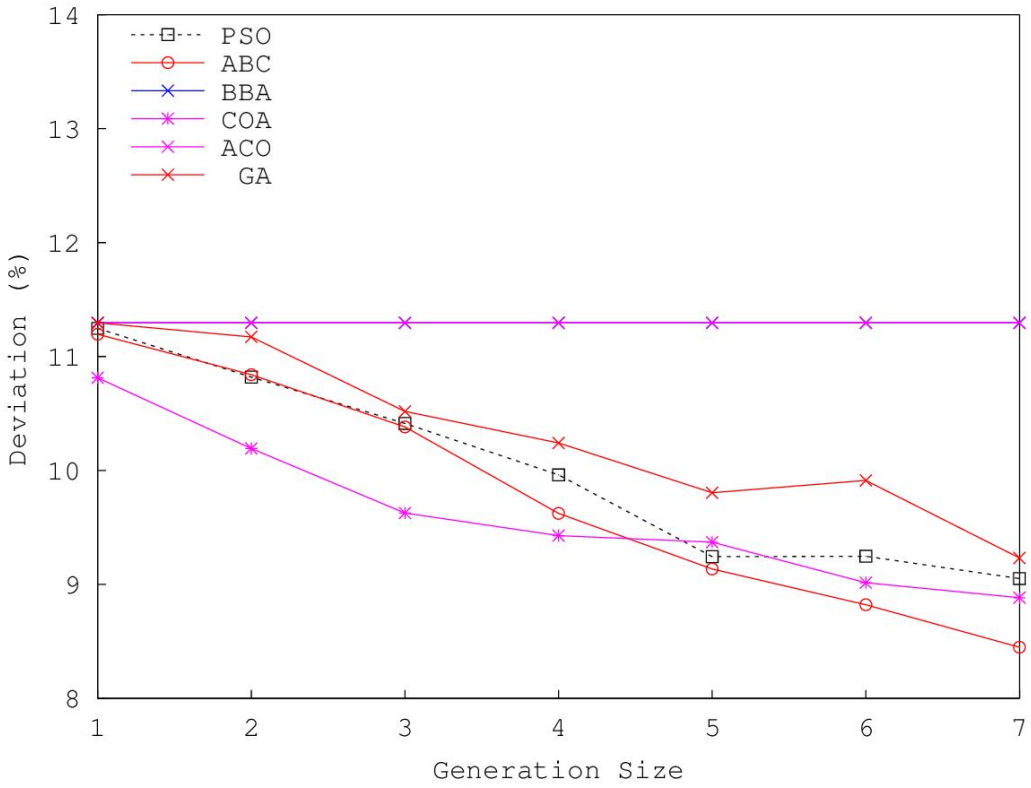

Figure 5.6: Deviation from the Optimal Solution Depending on Various Generation Sizes for the 18-Sensors Configuration

In Figure 5.6 shows all meta-heuristic algorithms' deviation rates for each generation at 18 sensors network configuration. At a first glance at this graph; the COA may appear to be converging much better when compared to others until the fourth generation mark; if we were to take the elapsed time between each generation into consideration, this algorithm yet still falls behind in convergance rate then ABC.

Upon reaching a conclusion in our research of finding the meta-heuristic algorithm to solve for parameters such as network capacity, sensor rate and sensor priority values in wireless sensor network implementation problem presented, we have experimented on different bio-inspired solutions and tried to compare them in terms of convergence rate; deviation from the exact solution, time elapsed for each solutions and so on. After running each algorithm on these problems that have been produced; we have observed that in these terms; the Artificial Bee Colony Algorithm provides the best of both worlds; faster but mature convergence rate with relatively less time requirement for each generation in multi sensor cases.

CHAPTER 6

CONCLUSION

In today's world, homes are becoming increasingly sentient with the help of smart-systems that are designed to fulfill the expectation of humans. These technologies have been adapted to living spaces in order to achieve maximum yield in life quality. Modern smart home applications have only been possible because of the fact that wireless sensor networks and multimedia networks that have become increasingly easy and cheap to facilitate. These networks have to be used in immediate moderation, however; due to prominent use of bandwidth by the multimedia sensors integrated within these smart homes such as those that provide real-time video and audio streaming. For multi-element sensor networks in such scenarios, this optimization problem consists of a number of ever-changing dilemmas that has to be solved in meta-heuristic procedures rather than brute force methods.

The algorithms that have been implemented in order to solve this wireless multimedia sensor network deployment problem have a number of subsections in which we can further elaborate. There is the matter of resource allocation that first comes to mind; in our case we have considered to implementation of large sensor networks with centralised multiple node systems, where all nodes report to a coordinator center, this coordinator is then tasked to constantly adjust the existing resources in order to submit efficient results to the user. This presents a problem which the coordinator overcomes by instrumenting logic in processing the data received, for example in our case video broadcasting from camera sensors are restricted to discrete transmission rates that are adjusted in conformity with the current status of the network traffic.

Having a centralised single-hop wireless sensor network rather than a multi-hop one

proved to be a smart move on our part for a number of reasons, partly because all sensor nodes answering to a center coordinator, and the coordinator doing the most of the traffic management helps with the issue of coping with multiple data types that are trafficked along the network. This kind of almost hierarchical approach in designing the network has also helped with the issue of balancing the traffic and data flow; meaning it gives the coordinator of the sensor network more authority to decide on the time devoted to the communication of sensor nodes, and to shorten or lengthen it according to the immediate need. If we had not built the architecture in this manner or in other words if each data packet was appointed a fixed time slot to be delivered; then there would be huge periods of time where only a small amount of data is delivered to the coordinator hence causing a major deficit in already limited resources.

A follow up on the advantages of implementing a single-hop architecture in the wireless sensor networks is the fact that it has also proved to be relatively easier to run meta-heuristic algorithms in such structured networks. In all population-based bio-inspired meta-heuristic solutions that we've acquired, we had treated the varying parameters like transmission rates and packet sizes between sensor nodes as data that is later utilised for finding an optimal solution by the coordinator instrumenting a deterministic approach but also using probabilistic notions. This kind of weighed demeanor is key to processing NP-hard problems, and most nature inspired algorithms have employed this as the main objective.

In our research, we have implemented several of the aforementioned bio-inspired solutions to try and compare how well they fare against the presented bandwidth packing problem. Each meta-heuristic that have been explained briefly in Chapter 4 has their advantages and superiorities against each other in the objective of finding the optimal solutions. We've examined and compared them by their convergence rates, whether they give prematurely converged solutions or not, the accuracy of the solutions acquired, the applicability of these algorithms to the proposed problem.

The experimentation over these algorithms has shown us; after examining the data sets each of them have produced thoroughly, we are convinced that the Artificial Bee Colony algorithm, among of these meta-heuristics, have provided the best near optimal solutions when taking into account its relatively faster convergence and when

taking its ease of applicability to the real life wireless multimedia sensor network scenarios.

As a future work, we will use the optimum meta-heuristic algorithm as presented in this research in order to design a new type of medium access layer that is much appropriate for real life usage.

REFERENCES

[1] Ahmad Naseem Alvi, Safdar Hussain Bouk, Syed Hassan Ahmed, Muhammad Azfar Yaqub, Mahasweta Sarkar, and Houbing Song. BEST-MAC: Bitmap-assisted efficient and scalable TDMA-based WSN MAC protocol for smart cities. *IEEE Access*, 4:312–322, 2016.

[2] Abdul Wahid Ansari, Mannika Garg, Sushabhan Choudhury, and Rajesh Singh. Arm based real time video streaming using XBee for perimeter control in defense application. In *Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2014 International Conference on*, pages 943–947. IEEE, 2014.

[3] Mohammad Arifuzzaman, Mitsuji Matsumoto, and Takuro Sato. An intelligent hybrid mac with traffic-differentiation-based QoS for wireless sensor networks. *IEEE Sensors Journal*, 13(6):2391–2399, 2013.

[4] Md Abul Kalam Azad, Amina Khatun, and Md Abdur Rahman. A slotted-sense streaming MAC for real-time multimedia data transmission in industrial wireless sensor networks. *International Journal of Advanced Engineering Research and Science (ISSN: 2349-6495 (P)| 2456-1908 (O))*, 4(3):236–244, 2017.

[5] Mehdi Azarmi and Bharat Bhargava. An end-to-end dynamic trust framework for service-oriented architecture. In *Cloud Computing (CLOUD), Honolulu, USA, 2017 IEEE 10th International Conference on*, pages 568–575. IEEE, 2017.

[6] Jaypal Baviskar, Afshan Mulla, Manish Upadhye, Jeet Desai, and Aniket Bhovad. Performance analysis of ZigBee based real time home automation system. In *Communication, Information & Computing Technology (ICCICT), Mumbai, India, 2015 International Conference on*, pages 1–6. IEEE, 2015.

[7] Charles Bell. *Beginning sensor networks with Arduino and Raspberry Pi*. Apress, 2014.

[8] Bilal Erman Bilgin and VC Gungor. Performance evaluations of ZigBee in different smart grid environments. *Computer Networks*, 56(8):2196–2205, 2012.

[9] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[10] Vongsagon Boonsawat, Jurarat Ekchamanonta, Kulwadee Bumrungkhet, and Somsak Kittipiyakul. XBee wireless sensor networks for temperature moni-

toring. In *The Second Conference on Application Research and Development (ECTI-CARD 2010), Chon Buri, Thailand*, 2010.

[11] Vincent Boyer, Moussa Elkihel, and Didier El Baz. Heuristics for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 199(3):658–664, 2009.

[12] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, Boulder, Colorado, USA*, pages 307–320. ACM, 2006.

[13] Hong-Yi Chang. A connectivity-increasing mechanism of ZigBee-based iot devices for wireless multimedia sensor networks. *Multimedia Tools and Applications*, pages 1–18, 2017.

[14] Jin Soo Choi and Meng Chu Zhou. Design issues in ZigBee-based sensor network for healthcare applications. In *Networking, Sensing and Control (ICNSC), 2012 9th IEEE International Conference on, Beijing, China*, pages 238–243. IEEE, 2012.

[15] Paul C Chu and John E Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.

[16] Sarah J Darby. Smart technology in the home: time for more clarity. *Building Research & Information*, pages 1–8, 2017.

[17] DIGI, Digi International 11001 Bren Road East Minnetonka, MN 55343 US. *Xbee/Xbee-PRO ZigBee RF modules*, March 2015.

[18] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[19] Dushyanta Dutta, Arindam Karmakar, and Dilip Kr Saikia. Determining duty cycle and beacon interval with energy efficiency and QoS for low traffic IEEE 802.15.4/ZigBee wireless sensor networks. In *Advanced Computing, Networking and Informatics-Volume 2*, pages 75–84. Springer, 2014.

[20] Alperen Eroğlu, Ertan Onur, and Halit Oğuztüzün. Estimating density of wireless networks in practice. In *Personal, Indoor, and Mobile Radio Communications (PIMRC), Hong Kong, China, 2015 IEEE 26th Annual International Symposium on*, pages 1476–1480. IEEE, 2015.

[21] Shuo Feng, Peyman Setoodeh, and Simon Haykin. Smart Home: Cognitive interactive people-centric internet of things. *IEEE Communications Magazine*, 55(2):34–39, 2017.

[22] Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133–137, 1981.

[23] M Gayathri and I Harish. Smart home power management system using ZigBee. *International Journal of Emerging Trends in Engineering and Development*, 2(5), 2015.

[24] Vangelis Gazis, Nikos Houssos, Nancy Alonistioti, and Lazaros Merakos. On the complexity of "Always Best Connected" in 4g mobile networks. In *Vehicular Technology Conference, Orlando, FL, USA 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 4, pages 2312–2316. IEEE, 2003.

[25] Lei Guo, Zhaolong Ning, Qingyang Song, Lu Zhang, and Abbas Jamalipour. A QoS-oriented high-efficiency resource allocation scheme in wireless multimedia sensor networks. *IEEE Sensors Journal*, 17(5):1538–1548, 2017.

[26] Guangjie Han, Jinfang Jiang, Mohsen Guizani, and Joel JP C Rodrigues. Green routing protocols for wireless multimedia sensor networks. *IEEE Wireless Communications*, 23(6):140–146, 2016.

[27] Mohammed Zaki Hasan, Hussain Al-Rizzo, and Fadi Al-Turjman. A survey on multipath routing protocols for qos assurances in real-time wireless multimedia sensor networks. *IEEE Communications Surveys & Tutorials*, 2017.

[28] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, Honululu, USA, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[29] Fernanda Hembecker, Heitor S Lopes, and Walter Godoy Jr. Particle swarm optimization for the multidimensional knapsack problem. In *Proc. of the International Conference on Adaptive and Natural Computing Algorithms, Warsaw, Poland*, pages 358–365. Springer, April 11-14, 2007.

[30] Raymond R Hill and Chaitr S Hiremath. Generation methods for multidimensional knapsack problems and their implications. *Journal of Systems, Cybernetics, and Informatics (JSCI)*, 5(5):59–64, 2007.

[31] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, 21(2):277–292, 1974.

[32] Goran Horvat, Drago Zagar, and Tomislav Matic. Analysis of QoS parameters for multimedia streaming in wireless sensor networks. In *in Proceedings of the 55th International Symposium (ELMAR '13), Zadar, Croatia*, pages 279–282. IEEE, September 2013.

[33] Toshihide Ibaraki, Toshiharu Hasegawa, Katsumi Teranaka, and Jiro Iwase. The multiple choice knapsack problem. *J. Oper. Res. Soc. Japan*, 21:59–94, 1978.

[34] IoT-Lab. Fit iot-lab. `https://www.iot-lab.info/`. Accessed: 2017-10-15.

[35] IoT-Lab. Fit iot-lab deployment. `https://www.iot-lab.info/deployment/`. Accessed: 2017-10-15.

[36] IoT-Lab. Fit iot-lab development center. `https://www.iot-lab.info/dev-center/`. Accessed: 2017-10-15.

[37] IoT-Lab. Fit iot-lab hardware. `https://www.iot-lab.info/hardware/`. Accessed: 2017-10-15.

[38] IoT-Lab. Fit iot-lab lille deployment. `https://www.iot-lab.info/deployment/lille/`. Accessed: 2017-10-15.

[39] IoT-Lab. Fit iot-lab m3 node structure. `https://www.iot-lab.info/hardware/m3/`. Accessed: 2017-10-15.

[40] Pankaj Jadhav, Amit Chaudhari, and Swapnil Vavale. Home automation using ZigBee protocol. *International Journal of Computer Science & Information Technologies*, 5(2), 2014.

[41] Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. Computational intelligence based QoS-aware web service composition: A systematic literature review. *IEEE Transactions on Services Computing*, 2015.

[42] Dongli Jia, Xintao Duan, and Muhammad Khurram Khan. Binary Artificial Bee Colony optimization using bitwise operation. *Computers & Industrial Engineering*, 76:360–365, 2014.

[43] Mohamed Amine Kafi, Djamel Djenouri, Jalel Ben-Othman, and Nadjib Badache. Congestion control protocols in wireless sensor networks: a survey. *IEEE communications Surveys & Tutorials*, 16(3):1369–1390, 2014.

[44] Arpan Kumar Kar. Bio inspired computing-A review of algorithms and scope of applications. *Expert Systems with Applications*, 59:20–32, 2016.

[45] Shahadat Khan, Kin F Li, Eric G Manning, and Md Mostofa Akbar. Solving the knapsack problem for adaptive multimedia systems. *Stud. Inform. Univ.*, 2(1):157–178, 2002.

[46] Talha Koruk and Ertan Onur. Bio-inspired bandwidth packing. In *Consumer Communications & Networking Conference (CCNC), Las Vegas, USA, 2017 14th IEEE Annual*, pages 1–4. IEEE, 2017.

[47] Jingang Lai, Hong Zhou, Wenshan Hu, Dongguo Zhou, and Liang Zhong. Smart demand response based on smart homes. *Mathematical Problems in Engineering*, 2015, 2015.

[48] Adam N Letchford and Juan-José Salazar González. The capacitated vehicle routing problem: Stronger bounds in pseudo-polynomial time. 2017.

[49] Yanjun Li, Chung Shue Chen, Ye-Qiong Song, Zhi Wang, and Youxian Sun. Enhancing real-time delivery in wireless sensor networks with two-hop information. *IEEE Transactions on Industrial Informatics*, 5(2):113–122, 2009.

[50] Yang Liu, Itamar Elhanany, and Hairong Qi. An energy-efficient QoS-aware media access control protocol for wireless sensor networks. In *Mobile Adhoc and Sensor Systems Conference, Washington, DC, USA, 2005. IEEE International Conference on*, pages 3–pp. IEEE, 2005.

[51] Guisselle A García Llinás and Rakesh Nagi. Network and QoS-based selection of complementary services. *IEEE Transactions on Services Computing*, 8(1):79–91, 2015.

[52] Michael J Magazine and Maw-Sheng Chern. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247, 1984.

[53] An-bo Meng, Yu-cheng Chen, Hao Yin, and Si-zhe Chen. Crisscross optimization algorithm and its application. *Knowledge-Based Systems*, 67:218–229, 2014.

[54] Naoyuki Morimoto. Energy-on-demand system based on combinatorial optimization of appliance power consumptions. *Journal of Information Processing*, 25:268–276, 2017.

[55] A Narmada and P Sudhakara Rao. Average end-to-end delay of customised ZigBee stack. In *Parallel Computing Technologies (PARCOMPTECH), 2017 National Conference on*, pages 1–7. IEEE, 2017.

[56] Jian Ni, Bo Tan, and Rayadurgam Srikant. Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transactions on Networking*, 20(3):825–836, 2012.

[57] Kathleen Nichols and Van Jacobson. Controlling queue delay. *Communications of the ACM*, 55(7):42–50, 2012.

[58] Adam Noel, Abderrazak Abdaoui, Ahmed Badawy, Tarek Elfouly, Mohamed Ahmed, and Mohamed Shehata. Structural health monitoring using wireless sensor networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2017.

[59] Rajeev Piyare and Seong-ro Lee. Performance analysis of XBee ZB module based wireless sensor networks. *International Journal of Scientific & Engineering Research*, 4(4):1615–1621, 2013.

[60] Florian Polzlbauer, Iain Bate, and Eugen Brenner. Optimized frame packing for embedded systems. *IEEE Embedded Systems Letters*, 4(3):65–68, 2012.

[61] S Rajeswari and Y Venkataramani. Congestion control and QOS improvement for AEERG protocol in MANET. *International Journal on AdHoc Networking Systems (IJANS) Vol*, 2:13–21, 2012.

[62] Bushra Rashid and Mubashir Husain Rehmani. Applications of wireless sensor networks for urban areas: A survey. *Journal of Network and Computer Applications*, 60:192–219, 2016.

[63] Priyanka Rawat, Kamal Deep Singh, Hakima Chaouchi, and Jean Marie Bonnin. Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48, 2014.

[64] Sara Sabba and Salim Chikhi. A discrete binary version of bat algorithm for multidimensional knapsack problem. *International Journal of Bio-Inspired Computation*, 6(2):140–152, 2014.

[65] Anirudha Sahoo and Shanti Chilukuri. DGRAM: A delay guaranteed routing and mac protocol for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(10):1407–1423, 2010.

[66] Kristian Sandstrom, C Norstom, and Magnus Ahlmark. Frame packing in real-time communication. In *Real-Time Computing Systems and Applications, Cheju Island, 2000. Proceedings. Seventh International Conference on*, pages 399–403. IEEE, 2000.

[67] Chinyang Henry Tseng. Coordinator traffic diffusion for data-intensive ZigBee transmission in real-time electrocardiography monitoring. *IEEE Transactions on Biomedical Engineering*, 60(12):3340–3346, 2013.

[68] Muhammad Usman, Ning Yang, Mian Ahmad Jan, Xiangjian He, Min Xu, and KM Lam. A joint framework for QoS and QoE for video transmission over wireless multimedia sensor networks. *IEEE Transactions on Mobile Computing*, 2017.

[69] Navneet Vidyarthi, Sachin Jayaswal, Vikranth Babu Tirumala Chetty, et al. *Exact solution to bandwidth packing problem with queuing delays*. Indian Institute of Management, 2013.

[70] Naimah Yaakob, Ibrahim Khalil, and Jiankun Hu. Performance analysis of optimal packet size for congestion control in wireless sensor networks. In *In Proc.*

of. Network Computing and Applications (NCA), pages 210–213. IEEE, 15-17 July 2010.

[71] Xin-She Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

[72] Lan Yao, Fuxiang Gao, Ge Yu, and Jian Wang. A Multi-QoS constraint routing protocol for multimedia wireless sensor networks. In *In proc. of IMSCI 2009, July 10th-13th, Orlando, USA, 2009*, 2009.

[73] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM, New York, USA, 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576. IEEE, 2002.

[74] Mohamed Younis, Kemal Akkaya, and Moustafa Youssef. Handling QoS traffic in wireless sensor networks. In *Encyclopedia On Ad Hoc And Ubiquitous Computing: Theory and Design of Wireless Ad Hoc, Sensor, and Mesh Networks*, pages 257–279. World Scientific, 2010.

[75] Tao Zhong and Rhonda Young. Multiple choice knapsack problem: Example of planning choice in transportation. *Evaluation and program planning*, 33(2):128–137, 2010.

[76] Shuguo Zhuo, Ye-Qiong Song, Zhi Wang, and Zhibo Wang. Queue-MAC: A queue-length aware hybrid CSMA/TDMA MAC protocol for providing dynamic adaptation to traffic and duty-cycle variation in wireless sensor networks. In *9th IEEE International Workshop on Factory Communication Systems (WFCS2012), Lemgo, NRW, Germany*, pages 105–114. IEEE, May 21-24, 2012.

[77] Shuguo Zhuo, Zhi Wang, Ye-Qiong Song, Zhibo Wang, and Luis Almeida. IQUEUE-MAC: A traffic adaptive duty-cycled MAC protocol with dynamic slot allocation. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), New Orleans, USA, 2013 10th Annual IEEE Communications Society Conference on*, pages 95–103. IEEE, 2013.

[78] Kenneth Zyma. Multiple choice multi dimensional knapsack heuristics. `https://github.com/kzyma/MMKP_Heuristics`. Accessed: 2016-07-15.

[79] Kenneth Zyma, Yun Lu, and Francis J Vasko. Teacher training enhances the teaching-learning-based optimisation metaheuristic when used to solve multiple-choice multidimensional knapsack problems. *International Journal of Metaheuristics*, 4(3-4):268–293, 2015.