QATAR UNIVERSITY

COLLEGE OF ENGINEERING

MINIMIZING NUMBER OF SENSORS IN WIRELESS SENSOR NETWORKS FOR STRUCTURE HEALTH MONITORING SYSTEMS

BY

FARAH ABDULMUTALEB ALQAWASMA

A Thesis Submitted to the Faculty of

College of Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in

Computing program

June 2016

COMMITTEE PAGE

The members of the Committee approve the thesis of Farah Abdulmutaleb AlQawsma defended on [22nd May, 2016].

Dr. Tarek El-Fouly

Thesis/Dissertation Supervisor

Dr. Mohamed Hossam Ahmed

Thesis/Dissertation Supervisor

Dr. Sebti Foufou

Committee Chair

Dr. Yasser Morgan

Committee Member

Dr. Abbes Amira

Committee Member

Dr. Tamer Mohamed Samir

Committee Member

Approved:

Dr. Khalifa AlKhalifa, Dean, College of Engineering

ABSTRACT

Nowadays, wireless sensor networks (WSNs) are considered an essential candidate to apply structural health monitoring (SHM). An important problem in this area is sensor placement optimization. In many research works, solving this problem focuses only on the network properties and requirements such as energy consumption, network coverage, …etc., without considering the civil engineering requirements. However, there are other research works that consider network and civil requirements while optimizing the sensor placement. Unfortunately, although minimizing the number of sensors is important, it has never been addressed. This could be noticed from the limited literature found that addresses this problem while considering both the civil and the network requirements. As a result, in this thesis we study the problem of minimizing the number of sensors for SHM in WSNs. The idea behind this research is to reduce the network size, which can solve some problems such as the scalability, installation time and cost. Our contribution in this work is not limited to the mathematical model of the mentioned problem, but will extend to solve the problem using different methods: the exhaustive search, genetic algorithm (GA), and a heuristic algorithm that applies the binary search. The problem is then solved for different number of sensors as well as different placements in many conducted experiments. Finally, the time complexity is evaluated to compare between all the applied methods. The obtained results showed that minimizing the number of sensors becomes more significant with big structures. Furthermore, the binary search algorithm is the best to use to solve the problem for small buildings. But, For larger buildings, there is a trade-off between the performance, and time complexity, where binary search gives optimal solution, but genetic algorithm gives better time execution.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

B&B	Branch and bound
DAQ	Data Acquisition
EFI	Effective Independence method
EFI-DPR	Effective Independence-Driving Point Residue
EC	Evolutionary Computation
FIM	Fisher Information Matrix
FTSHM	Fault-Tolerant wireless sensor configuration method for the SHM
GA	Genetic Algorithm
GAG	Generalized GA
GNTVT	Guangzhou new TV tower
HN	High-End Node
IAG	Improved GA
LN	Low-End Node
MAC	Model Assurance Criterion
MSE-AGA	Modal Strain Energy – Adaptive Genetic Algorithm
mop-SPEM	Multi-Objectives SPEM
O( )	Big O notation
p-SPEM	Power Aware Sensor Placement using EFI method
WSN	Wireless Sensors Networks
RP	Repairing Points
RN	Redundant Node
SA	Simulated Annealing

SPEM	Sensor Placement using EFI Method
TPSP	Three Phase Sensor Placement

Acknowledgements

First of all, I would like to thank Allah for giving me the strength to come over all the stress and study pressure during my master’s study journey. Then I want to give special thanks to many precious people in my personal life, to my husband for his infinite love, patience and support all the time. To my mother for her continuous love, prayers and endless support. To my father for his support to continue my studies and for his great encouragement through my thesis work. To my lovely brothers, sister, my family in low and friends for their help, and prayers. I would say that without their full support and love, this work would not have been achieved.

In addition to that, I would like to acknowledge some valuable people in my academic life. My sincere gratitude to the supervisors of this thesis Dr. Tarek El-Fouly, and Dr. Mohamed Hossam for their guidance, help through my journey towards producing this thesis. Also, I take this opportunity to convey my sincere thanks to Prof. Ali Jaoua, the Master Coordinator, for guiding and helping in submitting the thesis in an organized, and professional manner. In addition to that, we would like to thank Qatar National Research Fund (QNRF) for funding the project[1]. I wish to extend my heartfelt gratitude to some of NPRP project team, to Dr. Mohamed Mahgoub, and Eng. Mohammed El-Sersy for their great help and collaboration in the project. I wish to extend my thanks to all my wonderful professors for their never-ending support and to my beloved university for everything that it gave me.

[1] This thesis is under the umbrella of one of the projects of National Priorities Research Program (NPRP- 6-150-2-059) funded by Qatar National Research Fund under its). Its contents are solely the responsibility of the authors and do not necessarily represent the official views of Qatar National Research Fund.

Chapter 1: Introduction

1.1 Overview

Nowadays, many applications use wireless sensor networks (WSN), such as wireless body area networks (WBAN) [1], surveillance systems [2], and structural health monitoring (SHM). Structural health monitoring is the science of applying damage detection to structures. WSNs are a promising candidate for SHM due to their inelegance in sensing technologies and good computing abilities [3].

SHM applications of WSNs are great for high-rise towers and infrastructure such as bridges [4]. SHM reduces economic loss, saves human lives, and decreases catastrophic failures [5]. Some examples that show the importance of SHM are the collapse of I-35W bridge in Minneapolis in 2007 and the breakdown of the I-5 Skagit River bridge in Seattle in 2012 [6]. Implementing WSNs for SHM has several benefits. WSNs will help ensure long-lasting structures for future SHM. Additionally, deploying WSNs for SHM decreases installation time and reduces costs when compared with deploying wired sensors network [7]. While there is a huge benefit to deploying WSNs for SHM, challenges are faced in both the computer science and civil engineering fields that need to be taken in consideration. For example, in WSN, network scalability is one of the major challenges in that field, and the related sequences are communication, fault tolerance (the network should be fail-safe), energy, and high installation time and cost [8]. In civil engineering, there are some specific requirements that should be considered, such as checking sensors' information-quality. Information quality means to measure how much the data obtained by the sensors in their specific positions is correct and accurate. In civil engineering area, the sensor placement quality can be measured based on the by a metric called fisher information matrix (FIM). If such civil requirement is not considered when choosing the sensors' location in the SHM network, it may lead to misleading information about the structure damage because in damage surveillance choosing the sensor locations is very sensitive.

As a result, optimizing sensor placement has become a very hot research area. Many researchers worked on optimizing sensor placement to satisfy WSN requirements, such as good area coverage, maximum network connectivity, maximum network lifetime, minimum number of sensors, as in [9], [10], [11], [12], and [13]. But the solutions used in the listed research are general and did not consider civil requirements. However, other research works do consider civil engineering requirements to optimize sensor placement in addition to network needs, as in [14], [15], [16], [17], [18], and others. The above-mentioned research greatly contributed to this field, but if investigated further, we notice that they never considered minimizing the number of sensors as part of the solution for the sensor placement problem. Many solutions assume that there are M candidate sensor locations and N available sensors to fit into some of the M candidate locations. We know that M can be a large number if the granularity resolution of the surface is defined [14], [15]. Having such a huge number of candidate sensors M means higher network size and more challenges in obtaining scalability, and its related sequences as limitations in power, computational capacities, time synchronization, coverage problems, huge cost and installation time, and quality of service (QoS) related to the SHM applications and civil field requirements [19].

1.2 Research Motivation

The first motivation to work in this field is the importance of sensor placement optimization for SHM in WSN and its impact on global quality of life. Another inspiration to engage in this research is, as mentioned in the overview, that the problem of minimizing the number of sensors is not discussed in the literature, especially in relation to computer science and the civil requirements of SHM in WSNs. Additionally, we need to consider the enormous effects of having such large M candidate sensor node locations. Minimizing the number of candidate sensor locations can reduce these challenges. Another motivation is that this study offers a theoretical point of view to help the designers of SHM in WSNs balance the number of sensors deployed and the information quality gained. In cases of failure, this will enable the designer to check if reducing the number of sensors in specific locations effected the lower bound of the required information quality. Furthermore, this research is a part of NPRP project[1] that aims to solve the following problem: Minimize energy consumption, and maximize placement information-quality using some constraints related to network requirements. In the mentioned problem, the sensors are installed in infrastructures where it is difficult to change the batteries, so the research team in the project decided to optimize energy consumption without sacrificing the quality of the information by deploying the minimal number of sensors. Another motivation related to the NPRP project which is the big umbrella for this research work, that this project research work will be used for monitoring the health of bridges existing in Qatar, where the main company in Qatar that is responsible about infrastructure, and buildings (ASHGAL) assign the task of SHM of one of the existing bridges in Qatar to the team of this NPRP project. So solving the problem included in this thesis will contribute in designing a better WSN with minimal number of sensors. These reasons all motivated me to do my master's thesis in this field.

1.3 Problem Statement

The thesis question is, "Can one find a minimal number of sensors for SHM in WSN that satisfies civil requirements by maintaining a certain required information quality and satisfies some WSN network requirements, such as assured communication and specific energy level?"

The studied problem concerns researchers working in the field of sensor placement optimization for SHM in WSN all over the world. The problem needs to be studied and solved at the system design level, when a designer of SHM in WSN networks want to place the sensors in their candidate locations. The designer should have parameters, such as the M sensors' candidate location coordinates and a civil field parameter related to the information quality obtained by the entire sensors network . The designer should specify the required information quality for the entire system, for example, an information quality can be less or more than 80%, depending on the study's and the designer's requirements. In addition, depending on the type of sensor used, the designer needs the minimum transmission rate R_c in the network and the initial energy provided by each sensor. When the designer has all of the stated parameters and requirements, he or she can start looking for a solution to the reported problem.

This research will discuss how the solution to the problem can be obtained by different methodologies and how it can be implemented into the sensor network to monitor structure health. Solving the problem will save time and money in big structures and reduce the challenges related to network scalability.

1.4 Research Aim and Objectives

This study aims to understand the reported problem and its corresponding parameters and requirements; it will illustrate the concept and solve the problem by minimizing the number of sensors for SHM in a WSN. To do this, the following steps should be achieved:

1. Model the system problem to clearly illustrate the objective (minimizing the number of sensors) and the related constraints to solve the problem's lower bound for information quality, upper bound for transmission range, and limitations for energy consumption.
2. Find methodologies for the optimal or near-optimal solution for the problem. There are many methods that can solve this problem. Three methods are applied which are:
   a. Exhaustive search which is the method of brute force all the possible solutions, and although it is known by its high complexity, it is very simple to run, and it is be considered a baseline to compare its results to other algorithms results.
   b. Genetic algorithm that is a heuristic method the mimics the evolution process, and it is used because it is known that it is faster, and because one of its well-known steps is binary encoding (the variable values are changed to binary values 0-1) and this type of encoding fits the studied problem in this thesis very much

c. Binary search which is a non-heuristic method, a heuristic application of binary search method is used to get the optimal solution in better time than classical method such as exhaustive search. It is known that this method is applied on sorted arrays, and this is the case in the input of this research work.

Some other methods that can be applied to solve this problem, but are not applied in this work;

a. Greedy method that is an optimization approach that reaches the solution by making a sequence of choices, each of which looks the best at the moment (locally optimal).with the hope that the global optimal solution will be obtained [20].

b. Branch-and-bound method which guarantees the optimal solution with less time complexity[21].

Those methods were not applied, because in the worst case, both of them can reach the time complexity of exhaustive search but with more algorithmic complication. On the other hand, those methods may be used in future work to compare their performance against the applied methods.

3. Implement the chosen algorithms and conduct experiments to show the relation between the candidate sensors' locations and the reduced number of sensors using different required lower bounds of information quality.
4. Compare the performance of the different implemented algorithms with the different sets of candidate sensor locations and compute the time complexity of each.

1.5 Scope of the Research

Delimitations:

The delimitation of the studied problem is that there is a specified objective function, which is to optimize (minimize) the number of sensors in a specific field, which is SHM in WSN. Furthermore, the constraints on solving the problem are not the typical ones used in WSNs without a specific application and don't include all the requirements of a WSN. Instead, a specific set is used to guarantee sensor node communication by setting an upper bound of R_c, and a WSN energy requirement is used so each sensor should have a lower bound of energy equal to the initial energy E_{init} saved in the sensor. The last constraint is the information quality metric that is the determinant of the Fisher information matrix (FIM), a requirement of the civil field. This research can be extended in future work by changing the mathematical formulation of the problem.

Limitations:

A limitation of the study is that the problem is solved for only one type of structure: towers and high-rise buildings. In the future, the work can be extended to bridges and other structures.

Finally, the study was conducted on different sets of sensors: 5 sensors, representing a 5-story building; 9 sensors, representing a 9-story building; and 30 sensors, representing a 2-bay, 9-story building. Though these sets of sensors are considered small, they illustrate the concept and demonstrate its validity. A future work is to conduct experiments on larger sets of sensors.

1.6 Significance of Research

The research is considered significant in more than one way. First this thesis solves for the first time the problem of minimizing the number of sensors for SHM in WSN while satisfying both civil and network requirements. Also significant is that minimizing the number of sensors in SHM can reduce installation time and cost for high buildings, with further cost reduction if using SHM from the beginning. This creates a good economic impact for the companies and countries that might use the solution proposed in this paper to minimize the number of sensors for SHM in WSN.

In addition, the proposed solution can be used before implementing a wireless sensor network in the field of SHM. If the designer used all candidate sensors, and if some sensors failed, he or she could measure the reduced information quality and compromise between the number of sensors and the information quality needed. Nevertheless, the NPRP project[1] team who are working on optimizing the energy consumption and information quality can use this study to deploy the minimal number of sensors.

1.7 Contributions of the Research

This thesis formulates the minimization problem of the number of sensors using a single objective function. The objective function is employed to minimize the number of sensors in WSN for SHM systems.

In this thesis, the contributions are summarized in the following points:

1. We propose a single, objective mathematical formulation to minimize the number of sensors for SHM in WSN.

2. The optimal solution is found using an exhaustive search and a heuristic that applies a binary search.
3. A sub-optimal solution is found using a genetic algorithm.
4. In the optimal solution, we confirmed the trade-off between number of mode shapes used and number of reduced sensors, where increasing the order of mode shapes (increasing the number of mode shapes) leads to using more sensors.
5. We compared the solutions obtained by three different methods. The numerical results show the binary search efficiency as a low complexity solution for small buildings. There is a trade-off between an optimal solution using binary search and better time complexity using a genetic algorithm for large towers.

1.8 Thesis Outline

The outline of the rest of this thesis is as follows: An overview of the stated problem’s main concepts, a literature analysis about SHM in WSN, and in Chapter 2, optimizing sensor placement and sensor number are presented. Chapter 3 provides the system model of the problem and demonstrates the different applied algorithms—exhaustive search, genetic algorithm, and binary search—and explains the reasons for choosing those methods to solve the problem. Chapter 4 describes in detail the implementation of the proposed methods and presents related flowcharts. Chapter 5 discusses the experiments and corresponding results and validation. The time complexity is then measured for the three applied methods with the validation of the computations. Finally, the conclusion, challenges, and future work are reported in Chapter 6.

Chapter 2: Background and Literature Survey

2.1 Structural Health Monitoring (SHM)

Structural health monitoring (SHM) is the procedure of applying a damage detection strategy for many fields such as aerospace, civil, and mechanical engineering structures. Damage can occur due to mismanagement in construction, lack of quality control, temperature, initiation of cracks caused by cyclic loading, or changes in the geometric properties or characteristics of a system that harmfully affect its current or future performance [22].

SHM is used in different fields, and some of the examples are mentioned in Table 1.

Table 1: SHM Applications

Field	**Structure to monitor its health**
Aerospace	Civil and military airplanes, space craft , and helicopters
Civil engineering	Buildings, bridges, dams, and tunnels
Transport	Automotive trains, and ships
Energy	Oil and gas installations and pipelines, wind turbines, nuclear plants, and tidal wave generators
Chemical installations	Piping and tanks

SHM is mainly used to replace schedule- driven maintenance with condition-based maintenance. It is important in insuring scalability in terms of monitoring many structures. It can also increase the structure’s longevity by detecting damage in the early stages to enable proactive maintenance. Furthermore, SHM has potential economic and life safety benefits [23].

In the SHM process, the system is monitored over time using an array of sensors. These sensors respond with periodically dynamic measurements. Then extraction of damage-sensitive features from these measurements is done. After that, a statistical analysis on those features is applied to define the current status of the structure’s health [24].

The output of long-term SHM can be used to check the condition of the structure, and to decide if it can perform its functions in light of the expected aging and degradation resulting from the operational environment. Moreover, after a dangerous occurrence like an earthquake, SHM can be used to provide reliable information regarding the integrity of the structure [22].

To be more precise, the researchers in [22] think that the SHM process is a pattern recognition problem that can be divided into four main parts, namely, operational Evaluation, data acquisition, feature extraction, and statistical model development for feature discrimination as shown in Figure 1.

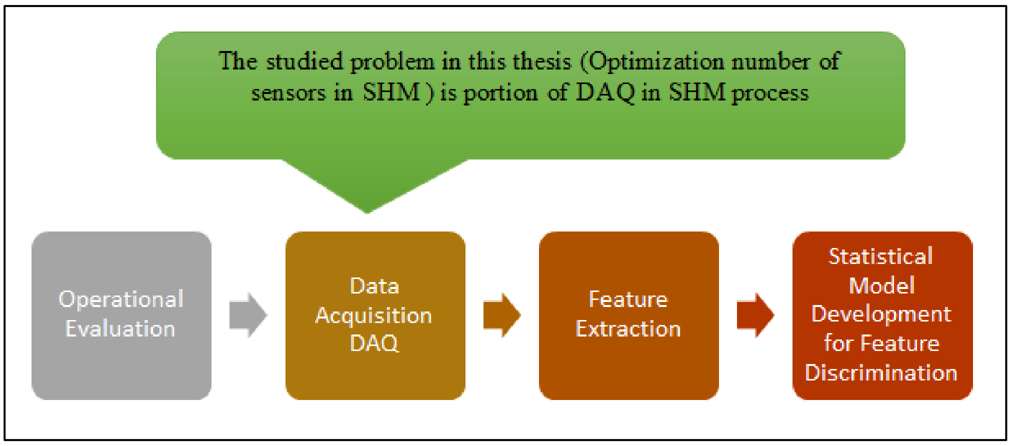

The studied problem in this thesis (Optimization number of sensors in SHM) is portion of DAQ in SHM process
Operational Evaluation
Data Acquisition DAQ
Feature Extraction
Statistical Model Development for Feature Discrimination

Figure 1: SHM Process

Operational evaluation is used to define what is the economic motivation or life safety motivation behind implementing the SHM process. Then it describes what are the damage types to be detected and under which operational and environmental conditions they are to be monitored. In addition to that, it shows the limitations of acquiring data in SHM [24].

The data acquisition (DAQ) part of the SHM process includes selecting the types of sensors to get the needed data, the number of sensors and the location where the sensors should be positioned, the bandwidth, and the data acquisition, storage, and transmittal hardware and equipment. The third main step in the SHM process is feature extraction, which gives the needed technical literature such as data normalization and processing techniques to recognize the damage-related information from the measured data. That means distinguishing the changes in sensor readings due to damage from those caused by varying operational and environmental conditions [24].

Finally, the last step in the in the SHM process is the development of statistical models to discriminate between features from the undamaged and damaged structures. In this step, The algorithms that operate on the extracted features are implemented to quantify the state of damage of the structure [25].

2.2 SHM: Development of Technologies

If a person wants to search about the development of using technologies in the SHM process, he/she finds that old conventional monitoring systems are categorized as having instrumentation points (sensors) wire-connected to the centralized DAQ system through coaxial cables and that the system is just used for monitoring. In addition to that, the sensors are independent and may not communicate with other sensors. The following problems result from using wired systems : (1) as the number of sensors increase, it becomes harder to install them; (2) the degree of sophistication in data processing becomes greater. (3) the cost of maintenance is higher [22].

Although wired network systems are still used in some SHM applications, wireless sensor network (WSN) systems are widely used for SHM nowadays due to their huge advantages [26]. One of the benefits of using WSN systems is to solve the recurring cabling problem of the conventional monitoring system. Furthermore, it is considered cost-effective compared with wired systems. On the other hand, there are also many constraints when WSN is deployed: scalability and the sequences of that: communication, fault tolerance, energy, and high installation time and cost. As a result, these constraints should be taken into consideration when the system is deployed [8].

Generally, in computer science, because of lack of knowledge of civil engineering, the sensor placement is often carried out randomly or uniformly to monitor an event such as an object or a target. But deploying the sensors randomly cannot be used in monitoring a structural event like a damage or crack because of the characteristics of SHM, such as strain and vibration. With the common methods used to implement WSN, effective SHM may not be possible because the spatial information to describe the dynamic behavior of a structure or sensitivity of an event (damage) is not sufficient at many locations, where choosing the sensor location is sensitive in monitoring a damage. As a result, sensor placement needs to be optimized during the DAQ step of the SHM process using the experience of civil engineers and computer scientists. [14].

2.3 Sensor Placement Problem

Sensor placement is an essential part of SHM applications, and optimizing sensor placement is very important in both civil engineering and computer science. To understand the optimization of sensor placement problem, assume that we have M possible locations for sensor deployment. M can become very large when the structure becomes bigger (e.g., feasible locations in high-rise buildings), and usually, there are a limited number of sensors ($N<M$). So to optimize the sensor placement, the N sensors need to be attached to some locations that satisfy an objective function or a multi objective function and some pre-specified constraints [14].

- **Sensor placement based on network requirements:**

In the networking community, the sensor placement optimization has been one of the important research topics on WSNs. There have been a lot of studies done on optimizing sensor placement in the WSNs framework. The researchers focus on satisfying the requirements of various applications using WSNs, such as network life time, area coverage, network connectivity, and data reliability, without taking into consideration the requirements of SHM. Here are some examples:

In [9], the authors estimated and evaluated sensor placement models that exploit different amounts of a priori information. The authors optimized the sensor placement by providing minimum energy consumption and maximum sensing coverage. Overall, the system requirements of WSNs, such as energy efficiency, sensing coverage, and operational lifetime, were enhanced by the authors' sensor placement. On the other hand, several estimations were used that make their work unfeasible when it comes to work with real structures.

In [10], the authors optimized the network life time and communication between sensors. To achieve the authors' target, the layout of the sensors was optimized using genetic algorithm (GA). The sensors were placed in the closest-possible distance in clusters using K-means clustering algorithm. In addition to that, the sensors could communicate with each other, and transmit their data to a high energy communication node (the cluster head) which acted as an interface between the data processing unit (sink) and sensors. The experiments showed improvement in the networks factors. Nonetheless, the number of computations in the used GA should be highly increased.

In some other research works on WSN, researchers are taking into consideration the minimization of a number of sensors as a part of network parameters. But again, the requirements of SHM are not satisfied.

An example of this is in [11]. The authors suggested an algorithm to satisfy a specific objective. The objective in that work was to optimize the sensor placement using smallest number of sensors to offer sufficient coverage of the sensor field. This minimum number of sensors was placed to transfer or report a minimum amount of sensed data. As a result a unique “minimalistic” view of the distributed sensor networks was achieved. Another algorithm was suggested to optimize the coverage. The objective function (coverage optimization) was studied under the constraints of imprecise detections and terrain properties. The suggested algorithm is a greedy algorithm that tries to accomplish the coverage goal through the smallest number of sensors. The method is iterative. One sensor is placed at the grid point with the least coverage in each iteration. The algorithm ends when the coverage objective is met or a bound on the sensor count is reached.

Another example is in [12]. Chen et al. studied an optimization problem with the objective of knowing the minimum number of sensors and their deployment that gives the network longest lifetime. An algorithm of two main steps was proposed to solve the problem. First, a fixed number of sensors was placed to gain maximum network lifetime. The authors defined this optimization as a multi-variant, nonlinear problem and solved it numerically. In the second step, the number of sensors was minimized, so the highest network lifetime per unit cost could be achieved. An analytically derived solution was used to solve the second step.

One more example is in [13]. The authors of [13] formulated the sensor placement problem as constrained multi objective optimization problem. The aim of this work is to place the sensors in such way that they maximize network coverage, minimize energy consumption, maximize network lifetime, and minimize the number of sensors to reduce cost and the payload of placement. To solve the problem, the authors divided the multi objective function into different single objective optimization problems and used a tree structure to keep the connectivity between the sensors and the sink. The authors compared their work with other works, and found that their work is better.

Sensor placement based on civil engineering and network requirements:

On the other hand, the optimal sensor placement problem for SHM using WSN is studied in many old research works where the parameters of SHM are considered alone or with the addition of network parameters as in the following published papers.

In civil engineering there are some traditional methods for optimizing the sensor placement in SHM that are reviewed in [8], [27], [28], and [29], such as the effective independence (EFI) method, and effective independence driving point residue (EFI-DPR) method . EFI is defined in [30] to be a sensor placement algorithm that starts with all possible sensor positions and reaches the wanted number of locations by gradually removing those that have the minimum contributions to the linearly independent manifestation of the fixture faults. While EFI-DPR is a composition of EFI method and an energetic approach, called the driving-point residue (DPR) [31]. A sensor with low energy can be selected in EFI method, and results with loss of information. EFI-DPR is used to avoid this weakness by using the DPR method that takes the sensor energy in consideration [32].

In [14] and [15], the authors discussed the sensor placement optimization problem for SHM in WSNs considering both network connectivity and civil engineering requirements such as the coverage of critical locations in the structure. The objective function studied is to maximize the Fisher information matrix (FIM) determinant that is a standard metric to identify the sensor placement quality in civil engineering [27], [14], [15]. In addition to that, maximizing the system life time that totally depends on the energy consumption. The authors suggested an algorithm based on EFI. The authors named their module (algorithm) sensor placement using the EFI method (SPEM). In SPEM, The possible locations of sensors are sorted according to the FIM results and excluding the nodes (locations) with the least quality and least contribution. The authors showed how data routing, topology control, and energy efficiency can be integrated with the SHM framework by introducing power aware SPEM (p-SPEM) algorithm. The authors did some experiment on the built-in Guangzhou New TV Tower, and the results on the sensor placement have validated the effectiveness of their methods. Furthermore, the authors' algorithm reduced the complexity of placement from $O(N^M)$ to $O(N^4M)$ [14],[15].

The authors of [33] added an improvement to SPEM, which is considering the amount of the energy consumption of a sensor node. In SPEM, the deployment of sensors is determined based on the determinant of the FIM. A new single objective function was proposed to be maximized, which is the determinant of FIM/ E_{max}, where E_{max} is the maximum energy used by a sensor in one round of data transmission. As a result, the energy consumed by a sensor node is minimized.

In another work, improvement was done to p-SPEM in [16]. Multi objective p-SPEM (mop-SPEM) algorithm for sensor deployment was suggested. The multi objective formulation gives the choice to specify the weights of the two objectives studied in the problem (energy consumption and information quality). As a result, the two objectives can be easily traded off.

Furthermore, in [17], the authors note that when the EFI method is used to have optimal sensor deployment, fault tolerance cannot be handled because EFI does not take into consideration the WSN parameters, so some data can be lost. As a result, the authors proposed a fault-tolerant wireless sensor configuration method for the SHM (FTSHM). FTSHM has two steps: the first one is to place the sensors using EFI, and then second, place some backup sensors called repairing points (RP) in a decentralized manner to ensure network connectivity, prolonged network lifetime, and reliable data delivery.

The authors in [26] and [34] designed a three-phase sensor placement approach (TPSP). The main objective in [26]and [34] was to find a high-quality sensor placement that could satisfy different system requirements while ensuring communication efficiency, low communication cost, and fault tolerance. In this research work, the sensor placement was addressed in heterogeneous WSN. Three kinds of sensors were used: high-end nodes(HNs) that are resource high, low-end nodes (LNs) that are resource limited, and redundant nodes (RNs) that have the same functionality of LNs. Redundant nodes were added to enable the fault tolerance ability in the network. The layout of these sensors was done based on three phases, the first phase to sub-optimally place HNs, the second phase to place LNs optimally, and the third one to place RNs to solve a sensor failure situation. The nodes deployment developed connectivity trees in such a way that the network connectivity is ensured. As a result, the structure health state or network maintenance after a sensor fault can be achieved in a distributed and decentralized manner. To validate the efficiency and effectiveness of TPSP, the authors ran extensive simulations. In addition to that, they implemented the algorithm on a real physical structure to prove the concept.

In some other work referenced in [35], the authors used a method called modal assurance criterion (MAC). MAC is used to check whether the sensors' locations are good enough. Let Φ be the matrix of target mode shapes where a mode shape is the shapes of the beam at different normal frequency. The MAC between model vector $\Phi_i$and Φ_j is defined as

$$\Psi_{i,j} = \frac{(\Phi_i^T \Phi_j)^2}{(\Phi_i^T \Phi_i)(\Phi_j^T \Phi_j)} \qquad (1)$$

where $\Phi_i : ith\ column\ vector\ of\ \Phi$

and $\Phi_j : jth\ column\ vector\ of\ \Phi$

When the $\Psi_{i,j}$ approaches 1, this means that the two model vectors are hard to distinguish, and that there is more correlation between Φ_i and Φ_j. A sensor position is chosen so the maximum off-diagonal terms of the MAC matrix are minimized. Some other authors of [36] extended the work in [35] by using the forward addition minMAC method together with the backward deletion minMAC method because it was noted that when the minMAC is used, when the number of sensors increases, the off-diagonal terms of the MAC matrix do not decrease monotonically.

Lately, some computational intelligence methods have been deployed to optimize sensor placement. One example is simulated annealing (SA). SA algorithm [37] is initialized by selecting a single random solution. Then to find better solution, the cost of one of the nearest neighbors of the selected solution is checked. If the neighbor has a better cost, this neighbor becomes the new selected solution. On the other hand, if the neighbor is with lower cost, then there is a probability whether to choose the neighbor as the new selected solution or not. The SA algorithm is mainly used when the best solutions have a tendency to be in one part of the structure space.

Another example of the computational intelligence algorithms is using genetic algorithm (GA), which is based on biological development. From the characteristics of GAs, the methods are not working with parameters directly, but the they are converted (coded) to another scheme. Usually, binary encoding is used. The coding has a discrete nature which makes the GAs a great option to solve discrete problems. The main operators of a genetic algorithm are selection, crossover, and mutation. First, some initial random solutions are selected using a specific selection algorithm . These initial solutions are combined and mutated to search for improved solutions through crossover and mutation steps.

In [38], the authors used GA instead of EFI and used the objective function to be about the determinant of FMI. In addition to that, the authors of [39] applied GA and SA to have sensor placement optimization. From the research work done, GA is a great method to try finding optimal solutions, but it can produce some invalid solutions because of randomness. As a result, in different research papers, the authors used GA with some modifications. For example in [40], the authors optimized the sensor deployment based on detecting structural damage using improved genetic algorithm (IGA). The modifications done on GA are the methods used to apply crossover and mutation where in IGA, crossover is based on identification code and mutation is based on two gene bits. The method used gives better optimization results than a simple GA. On the other hand, in [18], the authors used generalized GA (GGA) where the coding is dual structure based on the selection scheme, not binary based as in simple GAs. The authors demonstrated the effectiveness of the GGA suggested on the tallest building in the north of China. The GGA is compared with other GA algorithms, and it was shown that the GGA can improve the convergence of the algorithm and get the better placement scheme.

A hybrid optimization method called modal strain energy adaptive genetic algorithm (MSE-AGA) is suggested in [41]. The MSE-AGA provides multiple optimal indexes and has a short computation time. The MSE-AGA has three steps. First, mode shape orders are chosen carefully using the modal participation factor. After that, the MSE is used to get the initial sensor locations so that the location with high modal energy index becomes a candidate location. Finally, the AGA is used to minimize the number of sensors and their placement. The fitness function of the AGA is MAC, which is applied to guarantee minimized root mean square and the maximum of the off-diagonal elements are small.

A summary about sensor placement approaches is available in Table 2. From Table 2 one can notice that, the sensor placement optimization problem is very common in WSN where the deployment is achieved by ensuring optimizing one or some of the parameters of WSN such as network lifetime, energy, or coverage, and in some of them, the number of sensors used needed to be optimized. Nonetheless, the ways used to solve the problem in those research works could not fit the SHM field because of its special characteristic and parameters. On the other hand, in most of the research works where SHM parameters are considered, minimizing the number of sensors used is not part of the objectives when optimizing the sensor placement. And when it was achieved in [41], the minimization of the number of sensors was constrained by reducing the cost in the system. Based on all this, we introduce the problem of minimizing the number of sensors in WSN for SHM systems under some constraints related to WSN parameters such as connectivity and another important constraint related to SHM parameters which is to have a specific level of information quality represented by the FIM determinant.

Table 2: Summary off sensor placement approaches

Main Author	Algorithm Name	SHM Requirements	Optimization Objectives	Constraints	Computed Complexity
F. Oldewurtel [9]	-	None	Energy consumption and sensing coverage	-	-
M.Romoozi [10]	NSGA-II ,SPEA2, Clustering Fuzzy C-means	None	Network life time and communication	-	-
S.S. Dhilon [11]	MAX-MIN-COV, MAX-AVG-COV	None	Number of sensors	Minimalistic sensor network	O(A^2)[2]
Y. Chen [12]	-	None	Number of sensors and network lifetime	Coverage	-
S. Sengupta [13]	MOEA/D-DE	None	Area of coverage, net energy consumption, network lifetime, and number of deployed sensors	Connectivity for proper data transmission	-
B.Li [14], [15]	SPEM	Information quality	Sensor placement quality and system life time	Data delivery and connectivity	O(N^4M)
M. Najeeb [33]	-	Information quality	Sensor placement quality and the sensors lifetime		-
M. Elsersy [16]	p-SPEM	Information quality	Information quality and total energy consumption	Data delivery and limited energy consumed	
Z.A Bhuiyan [17]	FTSHM	Information quality	Ensure information quality and fault tolerance	-	O(n^2)[3]

[2] A is the grid points in the in the sensor field

M. Z. Z. Bhuiyan [26],[34]	TPSP	Information quality	Energy, cost, fault tolerance, and network life time	Connectivity, transmission load, data delivery	The input size is number of HN and LN[4]
T. Carne [35], C. Li [36]	MAC	MAC	maximum off-diagonal element of the MAC matrix are selected	-	-
L. Yao [38]	-	Information quality	Information quality	-	-
H. Y. Guo [40]	IGA	Information quality	Sensor deployment based on detecting structural damage	-	-
C. He [41]	MSE-AGA	MAC	Sensor optimal locations based on MSE.	-	-

[3] n is number of sensors per cluster, and this complexity is for backup sensor placement in FISHM..
[4] HN: high nodes, LN: low nodes. The detailed complexity of the different phase of TPSP are computed in [26], and [34].

Chapter 3: Problem Formulation and Methodology

In this chapter, the system model and mathematical formulation will be studied first. Then approaches used to solve the given problem will be introduced. Three methods were mainly used: exhaustive search, genetic algorithm, and heuristic algorithm, that is, using the bisection method.

3.1 System Model and Problem Formulation

This thesis formulates the minimization problem of the number of sensors using a single objective function. The objective function is employed to minimize the number of sensors in the WSN for SHM systems with some constraints. In this chapter, the details behind the problem's mathematical formulation and the system model are mentioned.

Preliminaries:

The mathematical formulation for the thesis problem depends on the system model in [14], [15], [16], and [42]. Consider that a location indicator S = $\{S_1, S_2, \ldots, S_M\}$, where if S_i equals 1, it means that the sensor node is selected, otherwise it is not, and M is the number of possible locations. In those research works, the researchers search for the optimal sensors placement by finding a location indicator S = S = $\{S_1, S_2, \ldots, S_N\}$, where N is a set of sensor locations selected from the feasible set of M total candidate locations that satisfy a certain objective function and some constraints. On the other hand, in this research work, minimizing the number of sensors is the goal of solving this thesis. In other words, we are trying to minimize the value of N based on some constraints.

The Euclidean distance considered in this research between sensor node i and sensor node j is given as follows:

$$d_{ij} = \sqrt{\left(c_u(i) - c_u(j)\right)^2 + \left(c_v(i) - c_v(j)\right)^2}, \quad \forall i, j \qquad (2)$$

where c_u *and* c_v are the two dimensions plane, and $c(i) = (c_u(i), c_v(i))$ are the Cartesian coordinates of a certain sensor i. So $C = \{c(1), c(2), \ldots, c(M)\}$ are the coordinates matrix of the M candidate nodes.

In this model, the WSN consisting of a number of sensors can be distributed in the sensing field with one sink node, wherein the data flow is generated at the source nodes and intended to the sink node. All sensors are assumed to have the same capabilities in signal processing and communication features. Each sensor is offered with a battery for power source. And the initial available energy node is set to be a constant value E_{init}.

Decision variables:

The decision variable for the mathematical model is the following: S_i is a binary indicator to indicate whether the location is selected or not in $S_i \in \{0, 1\}, \forall i$.

Objective function:

There is a single objective function in this formulation which is to minimize the number of sensors as follows:

$$\underset{S}{\text{Minimize}} \sum_{i=1}^{M} S_i \qquad (3)$$

Problem constraints:

There are three constraints in this problem. Two are related to WSN parameters connectivity and energy consumption. The third constraint is related to SHM parameters, that is, to have a specific level of information quality represented by the FIM determinant.

The first constraint is used to guarantee the data delivery where the distance between two sensors is not exceeding a maximum transmission range R_C:

$$d_{ij} I(x_{ij} > 0) \leq R_C \ i \neq j \ \forall i, j \qquad \text{C(1)}$$

where $I(x_{ij} > 0)$ is a binary indicator to know if the link $i-j$ is used. So it is imposed that $d_{ij} \leq R_C$ only if $I\big(x_{ij} > 0\big)$ is true.

The second constraint is related to energy consumption. The energy consumed between two sensors i and j is given as follows :

$$e_t(ij) = \left(\epsilon_t + \epsilon_{amp} d_{ij}^{\alpha}\right) n^b x_{ij} S_i S_j \qquad \forall i, j \qquad (4)$$

where the radio parameter ϵ_t is the energy cost for transmission, ϵ_{amp} is the power amplifier energy cost as in [42], α is the path loss exponent, and $\boldsymbol{n^b}$ is number of bits per packet [14]. In [42], x_{ij} is defined as the number of rounds the link i–j used. In this thesis work, since routing is not considered in the model system, x_{ij} ensures C(1), where if x_{ij} equals 1, it means that C(1) is satisfied, otherwise it equals infinity because C(1) is false.

Assuming that $E_t(i)$ is the energy consumed for each sensor node i during the transmission process, the transmission energy is computed as follows:

$$E_t(i) = \sum_{j=0}^{M} e_t(ij) \qquad (5)$$

On the other hand, in the reception process, the energy consumed for sensor node i and j is:

$$e_r(ji) = \epsilon_r n^b x_{ij} S_i S_j \qquad \forall i, j \qquad (6)$$

where ϵ_r is the reception energy cost. The total energy consumed in the reception process for sensor node i is calculed as follows:

$$E_r(i) = \sum_{j=1}^{M} e_r(ji) \qquad (7)$$

As a result, the total energy consumed in sensor node i through transmission and reception is given as follows:

$$E(i) = E_t(i) + E_r(i) \qquad (8)$$

To ensure that the consumed energy will not exceed the initial energy E_{init}, C(2) should be guaranteed.

$$E_i(S) \leq E_{init} \qquad \text{C(2)}$$

The third and last constraint is based on civil engineering requirements. In civil engineering, every mechanical structure has a certain pattern of vibration at a specific frequency. This is called mode shape. Mode shape also can be defined as in section 2.3 Sensor Placement Problem the shapes of the beam at different normal frequencies. In mathematics, the mode shapes of a certain structures form a mode shapes information matrix called Φ, and it is given below:

$$\Phi = [\Phi^1, \Phi^2, \dots, \Phi^K] = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1K} \\ \vdots & \vdots & \dots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MK} \end{bmatrix} \qquad (9)$$

where a column $\Phi^i = [a_{1i}, a_{2i}, \dots, a_{Mi}]'$ is considered the ith order mode shape, and a row $[a_{j1,} a_{j2}, \dots, a_{jK}]$represents the contribution of sensor node j in computing the mode shape measurement.

As mentioned in the literature in section 2.3 Sensor Placement Problem, the FIM determinant is a standard metric that measures the placement quality of sensors. The FIM determinant is computed as follows:

$|Q| = det[(\Phi)^T . R^{-1} . \Phi]$ (10)

where R is the noise covariance of the sensor measurements. R is a metric to show the dependency between objects, and variance is a measurement for the variability in the set of mode shape measured data. In addition to that, mathematically, it can be defined as the average squared deviation from the mean results.

Normalized $|Q|$ (L) can be defined mathematically as:

$$L(S) = \frac{|\boldsymbol{Q}|}{|\boldsymbol{Q}|_{\boldsymbol{max}}} * 100 \quad (11)$$

The last constraint is to set a lower bound for L that can be called L_{min}:

$L(S) \geq L_{min}$ C(3)

Problem formulation:

The authors could not confirm the convexity of the problem and the optimization problem is formulated as follows:

$$\underset{S}{\text{Minimize}} \quad \sum_{i=1}^{M} S_i$$

Subject to:

C(1) $d_{ij} \leq R_C \; i \neq j \; \forall i, j$

C(2) $E_i(S) \leq E_{init}$

C(3) $\boldsymbol{L(S) \geq L_{min}}$ (12)

3.2 Exhaustive Search

For any given optimization problem, there can be many ways to solve it. One of them is using brute-force search or exhaustive search, which is also known as generate and test.

Exhaustive search is a very general problem-solving methodology where all the possible solutions are enumerated and checked one by one to get the solution that satisfies the problem statement.

An easy example where brute-force method is used is in eight queens puzzle when examining all the possible combinations of eight queen pieces on the chessboard that has 64 squares, and, for each combination or arrangement, checking if any of the queen pieces can attack any other or if it is a safe arrangement (solution) of having all the right queens without having attacks.

The brute-force approach is known for its complexity and cost, where the cost is proportional to the number of candidate solutions. So if there are eight queens to arrange in a 64- square chessboard, it means that there can be $\binom{8^2}{8}$ different candidate solutions that need to be checked. And this is an indicator that whenever the problem size is increased, the cost behind solving the problem using brute-force method becomes higher [43].

Another example is the coin change, where a cashier has a group of coins of different denominations and is required to count out a sum of change using the smallest possible number of pieces[44].

The problem can be defined mathematically as follows:

Predefinitions:

N is the number of the pieces of coins.

$P = \{p_1, p_2 \ldots, p_n\}$ are the pieces of money (coins).

d_i is the denomination of p_i (e.g., if p_i, then d_i=10).

To count out a known sum of money A, we find the smallest subset of P, $S \subseteq P$, where $\sum_{p_i \in S} d_i = A$.

Decision variables:

$X = \{x_1, x_2 \ldots, x_n\}$ where $x_i = 1$ if $p_i \in S$, otherwise $x_i = 0$.

Problem objective function:

$$\text{Minimize } \sum_{i=1}^{n} x_i \qquad (13)$$

Constraints:

$$\sum_{i=1}^{n} x_i \, d_i = A \qquad (14)$$

To solve this problem using brute force, the user needs to find the best solution by checking all the possible values of X. For each value of X, the constraint in equation (14) is checked if satisfied or not. If yes, then the solution is considered a feasible solution. And the best solution for the problem is the feasible solution that minimizes the objective function of the problem in equation (13).

Because the value of x_i can be only zero or one, there are 2^npossible values for X. The execution time needed to decide whether a possible value of X is feasible is O(n), and the time needed to compute the objective function is as well O(n). As a result, the time complexity of the brute-force algorithm is in the order O(n2^n)[44].

And although of this cost, exhaustive search is still used in solving problems for different reasons. The first one is when simplicity of implementation is more important than speed. In addition to that, brute-force method can be used to prove a mathematical theorem, or it can be used as a baseline approach that gives the optimal solution and compares it with other methods solving the same problem. It can also be used when the problem size is manageable and limited.

In this thesis, the brute -force method was used due to its simplicity and to later on compare its results with the other methods used in this research work. The method is implemented in this thesis by running a code that brute force all the possible feasible solutions, and sort them to get the best solution depending on the objective function and constraints used.

3.3 Genetic Algorithm

The purpose behind developing genetic algorithms (GAs) was to study the phenomena of natural adaptation, then apply it somehow into computer systems and use the power of evolution to solve optimization problems. Genetic algorithms were introduced by John Holland, in the early 1960s at the university of Michigan. GA is an approximation heuristic search technique based on Darwinian's theory of survival of the fittest [45].

GA begins with solutions represented as one population of chromosomes that contains a number of genes (e.g., strings of alleles, ones and zeros, or “bits”). After that, it moves from that population to a new one using a type of natural selection in addition to some genetics inspired operators of crossover and mutation. The selection operator uses nature’s survival-of-the-fittest mechanism, where fitter chromosomes survive while weaker ones perish. Crossover exchanges subparts of two chromosomes, where some biological recombination between two single chromosome organisms. Mutation randomly changes the allele values of some locations in the chromosome. The mentioned process is repeated until some condition is satisfied.

The five main components in the GA process are encoding mechanism, fitness function, selection, crossover, and mutation.

3.3.1 Encoding Mechanism

It is considered an essential part of the GA structure to present the optimization problem’s variables and to transform the problem solution into chromosomes [46], and [47]. There are many encoding methods known through the published research work, and here are some of them:

a. Binary encoding: This is the most common used for encoding since it is very simple. The variable values are transformed into binary strings containing bits of 0s and 1s. Binary encoding provides several chromosomes even with a small number of alleles. On the other hand, this encoding is sometimes not natural for many problem variables, and some corrections should be done after crossover and/or mutation. An example of a problem where it is used is the knapsack problem. An example of binary encoding can be seen in Figure 2 chromosome A.

b. Permutation encoding: In this kind of encoding, each chromosome is a string of numbers that represent a position in a sequence. Permutation encoding is used in ordering or queuing problems like the traveling salesman problem, and similar to binary encoding, sometimes, crossover or mutation corrections should be done to leave the chromosome consistent with the same sequence in it. An example of permutation encoding can be seen in Figure 2 chromosome B.

c. Value encoding: Each chromosome in this encoding is a string of values. Values can be anything related to the problem, such as form numbers, characters, or complicated objects. Value encoding is used in problems in the neural networks field. On the other hand, it needs specially developed crossover and mutation techniques. An example of this scheme can be seen in Figure 2 chromosome C.

d. Tree encoding: This is used for developing programs or expressions and for genetic programming where every chromosome is a tree of some objects, such as functions or commands in programming language. An example of this encoding can be seen in Figure 2 chromosome D.

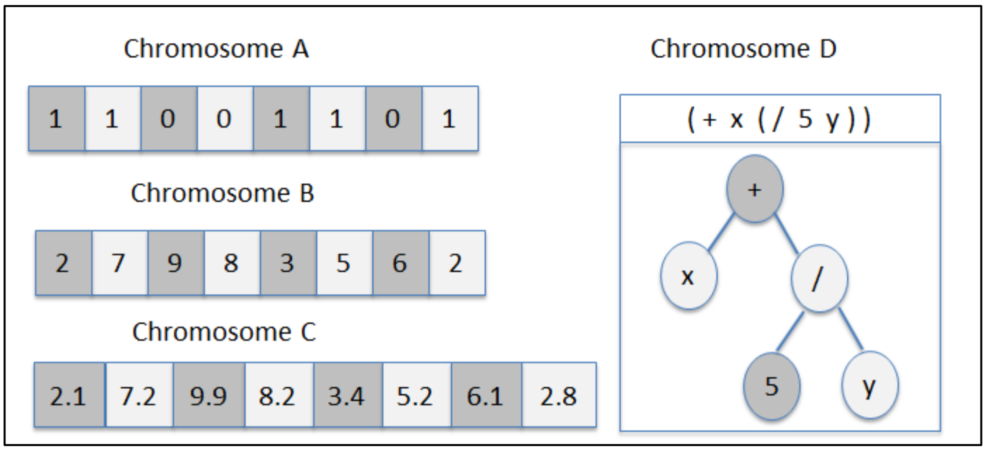


Figure 2: Encoding schemes in GA structure (A : binary encoding , B: permutation encoding, C: value encoding, D: tree encoding)

3.3.2 Fitness Function

It is the objective function to be optimized, whether minimized or maximized. And it is the way to score each string so it can be decided whether to choose it or not for the next generation. The ranges of fitness function values differ from problem to another, and sometimes, normalization can be used to uniform the output to a range of 0 to 1 and then feed the normalized fitness function values to the selection mechanism to evaluate the strings of the population [48].

3.3.3 Selection

The selection method is used in the GA process to choose the parents for the next generation based on the fitness of each individual from the population in the current generation. The main principle of the selection strategy is that if an individual is better than others, then it has a higher chance of being a parent. There are many algorithms used in the selection [49], [40], [46], [48], [50], and [51], and here are some of them:

1. Proportional selection (or roulette wheel selection): This method is very common for implementing fitness proportionate selection, where the chromosomes with better fitness have more chances to be selected in the next generation. An individual is assigned a portion of the circular roulette wheel, and the size of the portion is proportional to the individual's fitness. As a result, when the individual has better fitness, it will have a bigger slice in the roulette wheel than the others with smaller fitness. In this function, the sum of the fitness of all individuals in the population is calculated. Then a random number is generated from the given population interval to select one of the slices with a probability equal to its area.
2. Stochastic universal sampling: This is a way of roulette wheel selection that aims to reduce the risk behind premature convergence. In this method, each parent takes a part of the line with a length proportional to its fitness. The method goes through the line in steps of equal size, one step for each parent. In each step, the method places a parent from the part it lands on.
3. Tournament selection: This is a variant of rank-based selection methods. In this procedure, a set of k individuals are selected randomly, and then the individuals are ranked based on their fitness. The fittest individual is selected for reproduction. This process is repeated n times until the whole next generation is chosen.
4. Uniform selection: This selects individuals randomly from a uniform distribution using the expectations and number of parents. The result of this selection is an undirected search. This method is not a useful search strategy, but it can be used to test the genetic algorithm.

3.3.4 Crossover

In this step, recombination is done between two parents in the current generation to produce a new child (parent in the next generation). There are a lot of ways to do the crossover [47], [48], [46], and [51], and here are some:

1. Scattered crossover: In this recombination type, the parents exchange the corresponding genes to form a child. It uses a random binary vector. Then it selects the genes from the first parent when vector value is 1 and chooses the genes from the other parent where the vector's value is 0. An example on that, if parent1 = [a b c d e f g h], parent 2 = [1 2 3 4 5 6 7 8], and the random crossover vector is [11001000], then the new child after crossover is [a b 3 4 e 6 7 8].
2. Single point crossover: A recombination is done between two parents based on a point, where the new child's first genes come from the first parent, and genes after the randomly selected point come from the second parent. For example, using the same parents in scattered crossover, if the point selected is 3 then the new child is [a b c 4 5 6 7 8].
3. Two point crossover: In this crossover method, two points are randomly selected. In this, from the new child is created as follows: The first part of the first selected cross over point is copied from the first parent, and the second part till the second crossover point is copied from the second parent, and then the rest of the genes after the second selected crossover point are copied from the first parent. So if 3 and 5 were the selected crossover points, then the new child generated is [a b c 4 5 f g h].

4. Uniform crossover: In this scheme, the genes are randomly chosen from the two parents to create the new child.

3.3.5 Mutation

This method comes after the crossover in the reproduction process, where small random changes are done on the individuals in the population, which enable the GA to search a broader space. There are many ways to do mutation [46], [47], [48], and [51], and here are some:

1. Interchanging mutation: Two random positions of the individual are chosen, and the genes according to those positions are interchanged.
2. Reversing mutation: Tt can be used in a binary encoded chromosome. In reversing mutation, a random position is chosen and the bits next to that position are reversed, and child string is generated.
3. Uniform mutation: In this scheme, the value of the chosen gene is changed with the uniform random value selected between the specified upper and lower bound for that gene. It can be used in real and integer representation.
4. Adaptive feasible: In this method, the directions are randomly generated in such way that they are adaptive with respect to the last successful or unsuccessful generation. The length of the step depends on the satisfaction of the constraints and the bounds.

3.3.6 Why Use GA

Referring to [52], the block diagram of the presentation of the GA process is shown in Figure 3. More details on how GA works can be found in Appendix B: How GA Process Works.

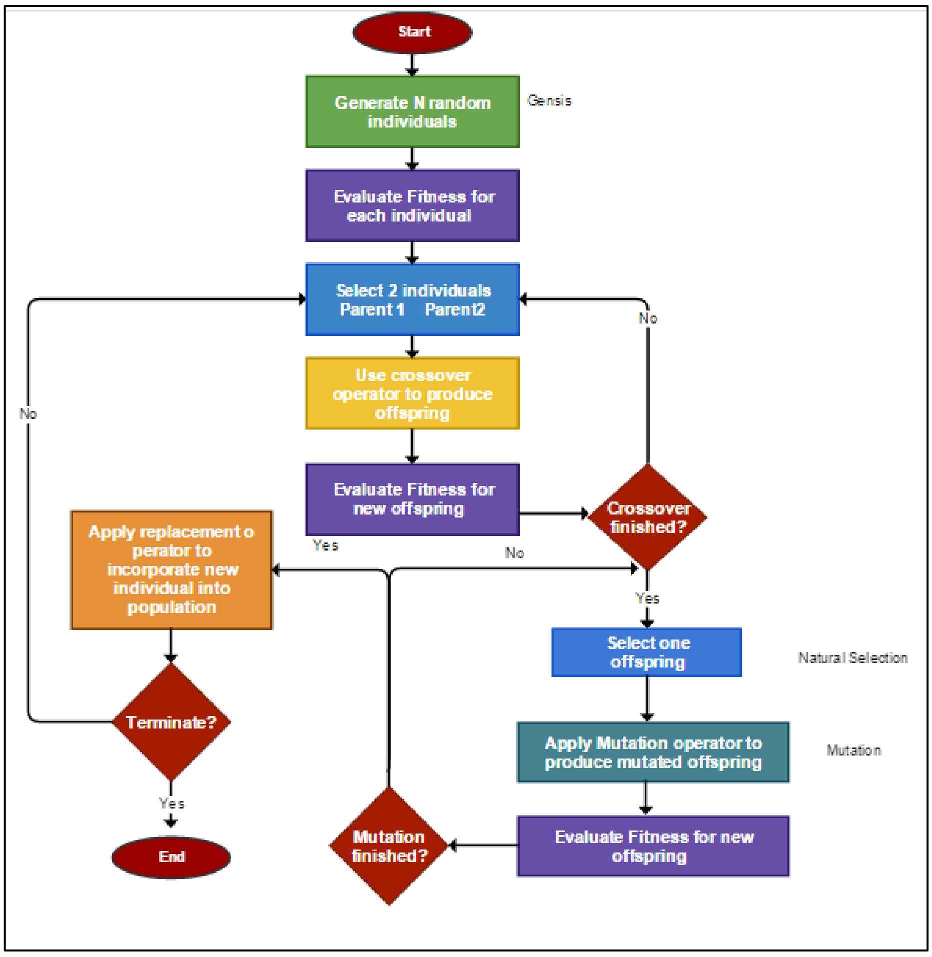

Start
Generate N random individuals
Gensis
Evaluate Fitness for each individual
Select 2 individuals Parent1 Parent2
No
Use crossover operator to produce offspring
No
Evaluate Fitness for new offspring
Crossover finished?
Apply replacement operator to incorporate new individual into population
Yes
No
Yes
Select one offspring
Natural Selection
Terminate?
Apply Mutation operator to produce mutated offspring
Mutation
Yes
End
Mutation finished?
Evaluate Fitness for new offspring

Figure 3: The block diagram presenting the GA process[52]

It is known general genetic algorithm is used to find a suboptimal solution (near-optimal solution) for the problem because the solution found depends on the set of some random variables generated as it has been seen in the process of the GA[53].

Although GA is used in general to find a suboptimal solution, here are some reasons that make it a good candidate for solving the problem studied in this thesis.

Referring to section 3.3.1 Encoding Mechanism, binary encoding is the most common encoding used. And the studied problem in this thesis fits very much to be binary encoded, where the sensor to be selected can be referred as 1; Otherwise, it is 0. This was a good motivation to use GA to solve the problem.

Moreover, we got the optimal solution for the problem using other classical algorithm such as exhaustive search that is known of its huge time complexity; and if one wants to compare classical algorithm and genetic algorithm, he/she will find that GA are better because it is faster and less likely to get stuck in a local extreme like other methods. Where in classical algorithms, a single point is generated at each iteration, and then the sequence of points will approach an optimal solution. On the other hand, in GA, a population of points is created at each iteration, and the best population approaches an optimal solution [54].

In addition to that, referring to Salvator Mangano Computer Design, May 1995 [55], “genetic algorithms are good at taking large, potentially huge search spaces and navigating the, looking for optimal combinations of things, solutions you might not otherwise find in a life time.”

Furthermore, because of its random nature, GA improves the chances of finding a global solution. It can solve unconstrained, bound-constrained, and general optimization problems, and continuous or discrete problems.

3.3.7 How Is GA Applied in This Research Work

It is good to know that there are different software or packages that can help to solve problems using GA [53], [56], [57], [58],and [59]. One of the tools that can be used is GPdotNET [59]. It is an artificial intelligence tool to apply GA and artificial neural networks in the modelling and optimization of different engineering problems. Another tool is open beagle [58]. It is a C++ evolutionary computation (EC) framework. It offers a high-level software environment to apply any kind of EC, with support for genetic algorithms. In addition to those tools, there is an optimization toolbox in MATLAB software that provides functions to find parameters that optimize (minimize or maximize) objectives while satisfying some introduced constraints. The toolbox includes many solvers that can be used in linear programming, quadratic programming, mixed-integer linear programming, nonlinear optimization, and nonlinear least squares. The solvers are used to find the best solutions to continuous and discrete problems The solver that is used is the GA solver [53] that applies the genetic algorithm.

In this thesis work, GA solver is chosen through the MATLAB software because dealing with MATLAB is easier. Moreover, the GA solver is very easy to use, and it can be used in two ways. One is through the optimization tool graphical user interface. The user can fill the parameters and change the options easily and see the running process. Or the user can write a small code to set the parameters and pass them to the function called GA and run the code to see the results.

It is important to mention here, that in the optimization tool box there are two functions that apply genetic algorithm which are *ga* [60], and *gamultiobj*[61]. And the reason behind using *ga* function and not *gamultiobj* is that the problem solved in this research is single objective function, and that *gamultiobj* is used to solve problems with multi objective functions, while *ga* is used for single objective problems to find the minimum of a function using genetic methodology.

The parameters that were passed to the function *ga*, are fitness function, the inputs that need to be optimized, their upper and lower bounds, and how many inputs are. The linear and nonlinear constraints if any. After feeding the parameters to the function *ga*, call it by running the code to find the solution to the problem. More details are discussed in Chapter 4: Implementation.

3.4 Binary Search Method

Binary search method is a classical non- heuristic method to solve a given problem. A heuristic application of binary search is used to solve the problem in this thesis to find the optimal solution. Heuristic method is a way for solving problems more quickly when other classical methods such as brute-force are too slow to solve the same problem or for finding an approximate solution when classical methods fail to find an exact solution. The aim of using such heuristic method is to get a solution in a good time frame that is reasonable enough for solving the problem at hand.

Binary search is called bisection method in some other references [62]. It is important to know that this algorithm can only be used for a sorted array in nondecreasing order. In this approach, if a person is searching for x in a sorted array, then the algorithm compares x with the middle item of the array. If they are equal, the solution is found. If x is smaller than the midpoint, then x, for sure, is in the first half of the array (if it exists within the array). And the algorithm repeats itself in the first half of the array until the solution is found. If x is larger than the midpoint, then the search will be repeated in the second half of the array. This way is repeated until x is found or that the algorithm stops and determine that x does not exist within the array[63].

The general algorithm for this method [63] can be found in Appendix A : Binary Search Algorithm.

As mentioned before, to solve this problem heuristically, many ways can be chosen. The binary selection is chosen because the input parameter to the function is considered a sorted array, where the number of sensors can be array of indexes from 1 to M. In addition to that the implementation and the analysis of such algorithm are considered easy and straightforward. Other methods may be used in future work, and compared with the current results of this work that can be seen in Chapter 5: Results, evaluation, and Validation. In this research work, the input is the array of available sensors A[1 .. M]. The algorithm will start by computing the middle item, and then the combinations of that computed number of sensors will be found. After that, the algorithm will start checking the feasibility of each combination until it finds a feasible solution. If a solution is found, then in the next iteration, the algorithm will search in the first half of the array to find a smaller number of sensors that can optimize the problem. On the other hand, if there is no solution, the algorithm will search in the second half of the array until a solution is found. The process will be repeated until the problem is solved and a global optimal solution is found, or to state that there is no solution. And it is known that using this searching algorithm will end up with a maximum number of comparisons that equals to $\log n + 1$, where n is the size of the input sorted array [20],[63]. More details about applying this algorithm and implementing it will be seen in Chapter 4: Implementation section of the binary search method.

Chapter 4: Implementation

This chapter provides a description on how the approaches mentioned in the chapter titled “Chapter 3: Problem Formulation and Methodology” have been applied or implemented to solve the minimization of number of sensors for SHM in WSN problem in equation (12).

In all the approaches, some common variables were observed which you can find in Table 3.

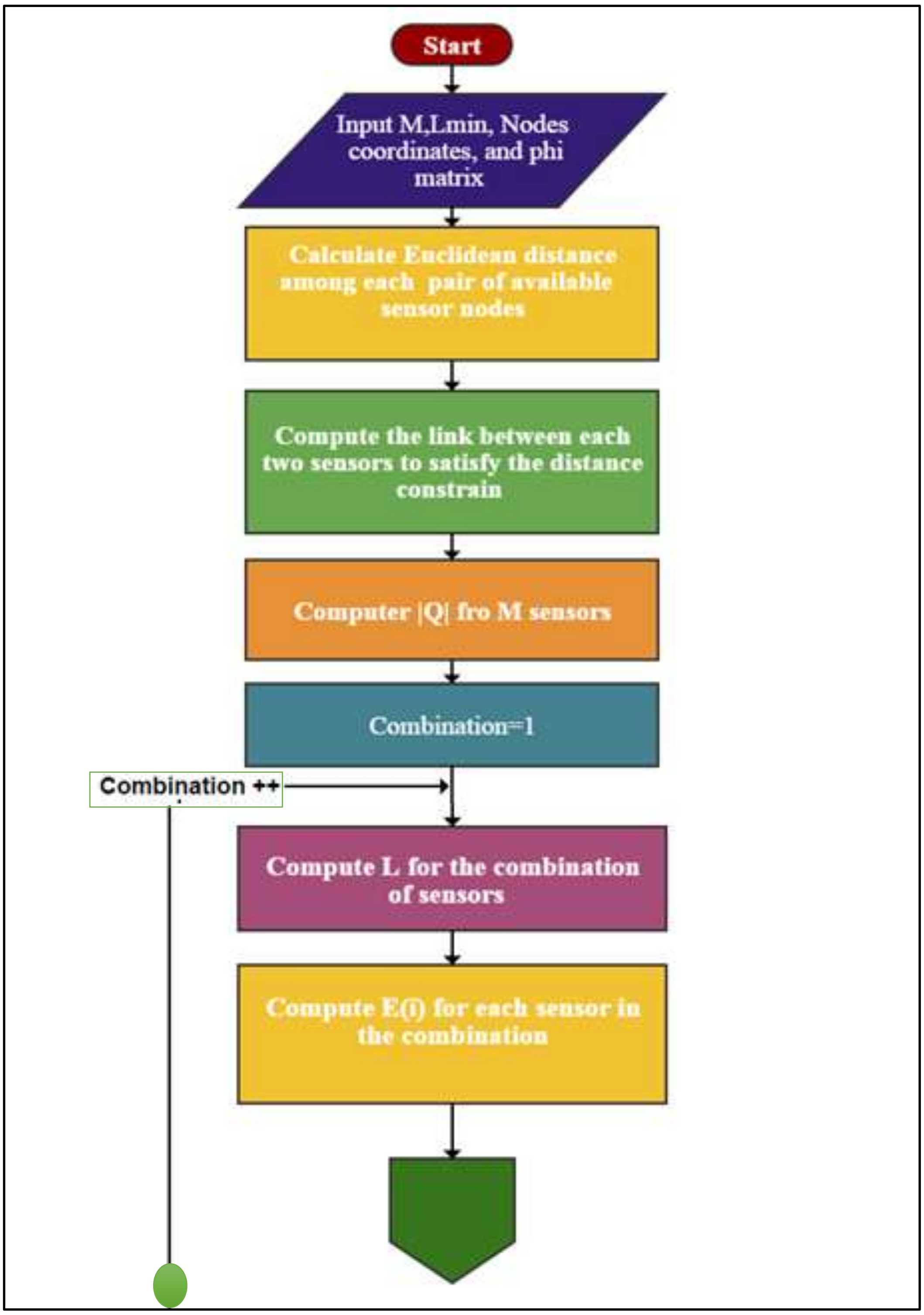
Table 3: Some problems' parameter descriptions

Variable	**Description**
M	Possible locations for sensor deployment
$\|Q\|$	Determinant of the Fisher information matrix
L	Normalized determinant of Fisher information matrix
L_{min}	Lower bound of the normalized $\|Q\|$ (L)
Node coordinates	Cartesian coordinates for all sensors M , and the sink sensor node
Φ	Matrix of target mode shapes for M sensors
x_{ij}	The number of rounds the link $i-j$ is used
d_{ij}	Euclidean distance between sensor node i and sensor node j

4.1 Implementation of an Exhaustive Search Method

As mentioned in section 3.2 Exhaustive Search, in this approach all the possible solutions are checked for their feasibility according to the information quality, distance and energy constraints (C1–C3) related to the problem. After that, all the feasible solutions are sorted according to their objective function stated in equation (3) evaluation, and the solution that best minimizes the number of candidate M sensor locations is chosen.

The flow chart presenting the brute- force algorithm is shown in Figure 4.


Start
Input M,Lmin, Nodes coordinates, and phi matrix
Calculate Euclidean distance among each pair of available sensor nodes
Compute the link between each two sensors to satisfy the distance constrain
Computer |Q| fro M sensors
Combination=1
Combination ++
Compute L for the combination of sensors
Compute E(i) for each sensor in the combination

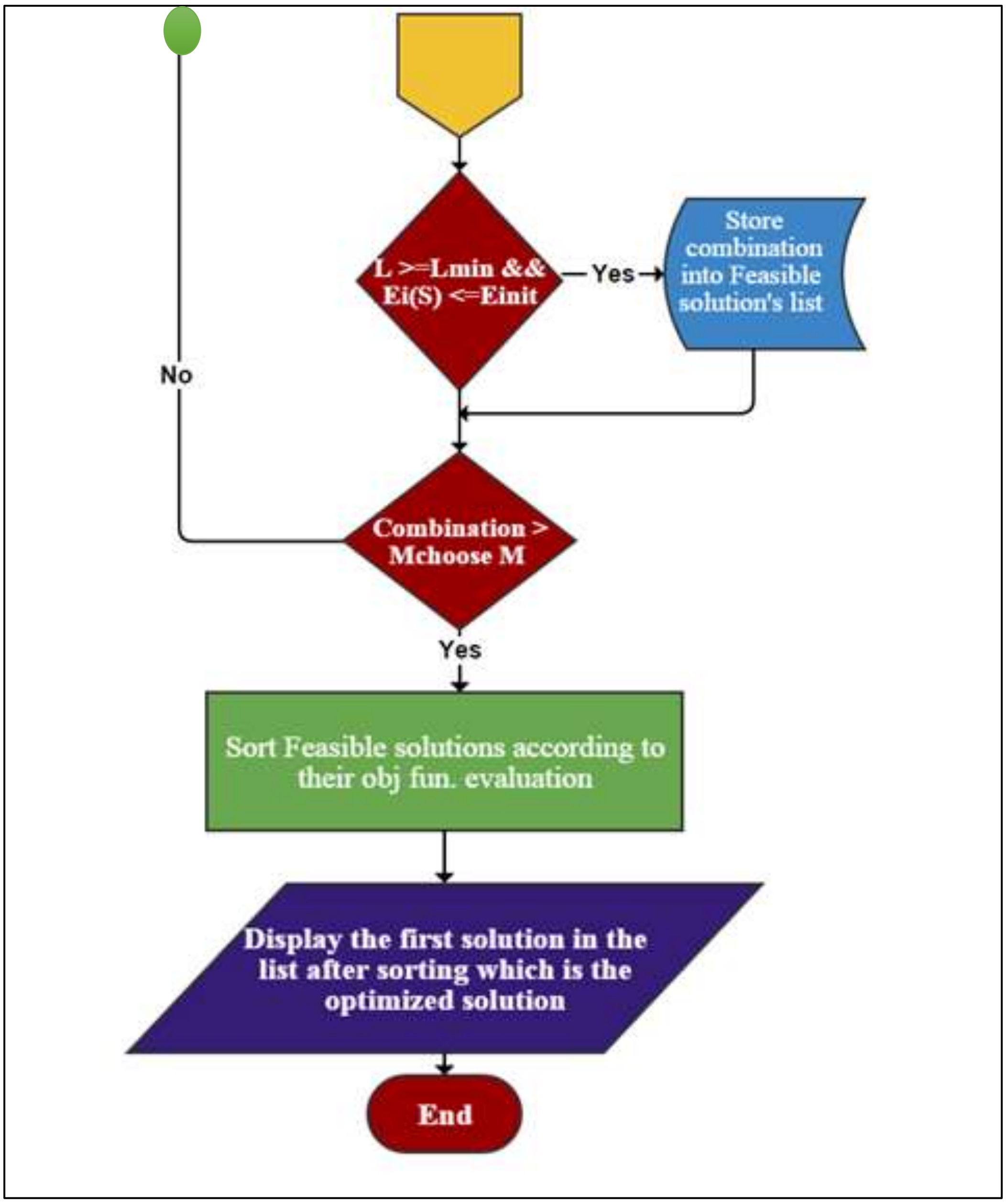

L >=Lmin &&
Ei(S) <=Einit
Yes
Store combination into Feasible solution's list
No
Combination > Mchoose M
Yes
Sort Feasible solutions according to their obj fun. evaluation
Display the first solution in the list after sorting which is the optimized solution
End

Figure 4: Flowchart of the implemented exhaustive search method

The explanation behind the flowchart in Figure 4 is found in the algorithm below.

Input: M, L_{min}, node coordinates, and Φ

Output: Optimized solution with an minimized number of sensors

Algorithm:

1. Calculate the distance $\mathrm{d_{ij}}$ $\forall \mathrm{i,j}$ where $\mathrm{i,j}$ belong to M and the sink using their given coordinates.
2. Compute the routing decision variable x_{ij} $\forall i,j$ where i,j belong to M and the sink node $\mathrm{M}+1.$ x_{ij} depends on the evaluation of distance constraint (C1) in equation (12). If (C1) is satisfied, and distance is limited within the transmission rang then we assume that a rout is established. So $x_{ij} = 1$; otherwise, $x_{ij} = 0$. In addition, in case $i = j$, or $i = the\ sink$, then $x_{ij} = 0$.
3. Evaluate determinant of FIM $|Q|$ as in equation (10) for M sensors when all M sensors are selected.
4. Compute the possible combinations of M sensors. Then start with combination 1.

5. Go through the combination of sensors, and then check the feasibility of the solution. To check the feasibility of a combination:

   a. Determinant of FIM $|Q|$ and $L(S)$ are computed for the combination of the selected sensors.

   b. The total energy consumed in sensor node i equation (8) is computed for each sensor i from the total selected nodes in the combination.

c. Check the satisfaction of the constrain related to information quality by comparing the results from step 5a with L_{min}.

d. Check the energy constrain satisfaction by comparing the results \ from step 5b for each sensor i a with E_{init}.

e. If any of the constraints is not satisfied, then the solution with that certain combination is not feasible. Otherwise, if all constraints are satisfied, then the solution is feasible.

f. Go to the next possible combination, and then go back to step 5 until all the possible combinations are checked.

6. Sort all the feasible solutions from step 5 in ascending order according to their objective function evaluation in equation (3) and then descending according to their normalized |Q| (L).
7. Choose the first solution in the list to be the optimized solution, where the combination of sensors has the least selected number of sensors with the highest possible $L \geq L_{min}$.

4.2 Implementation of the GA Method

As mentioned in section 3.3.7 How Is GA Applied in This Research Work, the GA solver (*ga* function) from the optimization toolbox in MATLAB is used to apply the genetic algorithm.

All the input arguments passed to this function, and all the different syntaxes that can be used to call the *ga* function are shown in [60].

The syntax I used to solve this problem is this:

```
ga(fitnessfcn,nvars,[],[],[],[],LB,UB,nonlcon,IntCon)
```

The input arguments that I used to solve the thesis problem is shown in Table 4.

Table 4: Input arguments for the ga function

Input argument	**Description**
`Fitnessfcn`	This parameter is a handler to the fitness function of the problem , which in the thesis problem is to minimize the number of sensors as in equation (3).
`Nvars`	This parameter stands for the number of design variables in the problem (M+1).
`A,b, Aeq,beq`	These parameters are used to set the linear equality and inequality constraints. And because there are no linear constraints in the thesis problem ,these parameters are substituted with null arguments [ ].
`LB`	The vector of lower bounds. In this problem, the decision variables that present the selection of the sensors in the solution are binary. So expressing the lower bound for choosing a certain sensor out of the M sensors, means that it is not selected . As a result, the lower bound is zero. But when one thinks about the sink node, he or she cannot eliminate it from the selection , so the lower bound of the sink node is 1.
`UB`	The vector of upper -bounds. Setting the higher bound for choosing a sensor, means that the sensor is selected. As a result, the UB is 1.
`Nonlcon`	This is the nonlinear constraints function handler [c,ceq] = nonlcon(x). Where GA tries to get $c \leq 0$ and $ceq = 0$, c stands for nonlinear inequality constraints, and ceq stands for nonlinear equality constraints. Both c and ceq can be used as row vectors in case of multiple constraints. The unused output can be set to null argument[ ]. In the thesis problem, we have M+2 nonlinear inequality constraints. (M+1) constraints to implement (C2) for each sensor in the combination used, in addition to one constraint that is applying (C3) in equation (12) . All the nonlinear inequality constraints in (C2) and (C3) are shown as follows: (C2) $\mathrm{E_i(S) - E_{init} \leq 0}$ (C3) $\mathrm{L_{min} - L(S) \leq 0}$
`IntCon`	This is the index vector of integer variables, which include all the M+1 sensors in this problem.

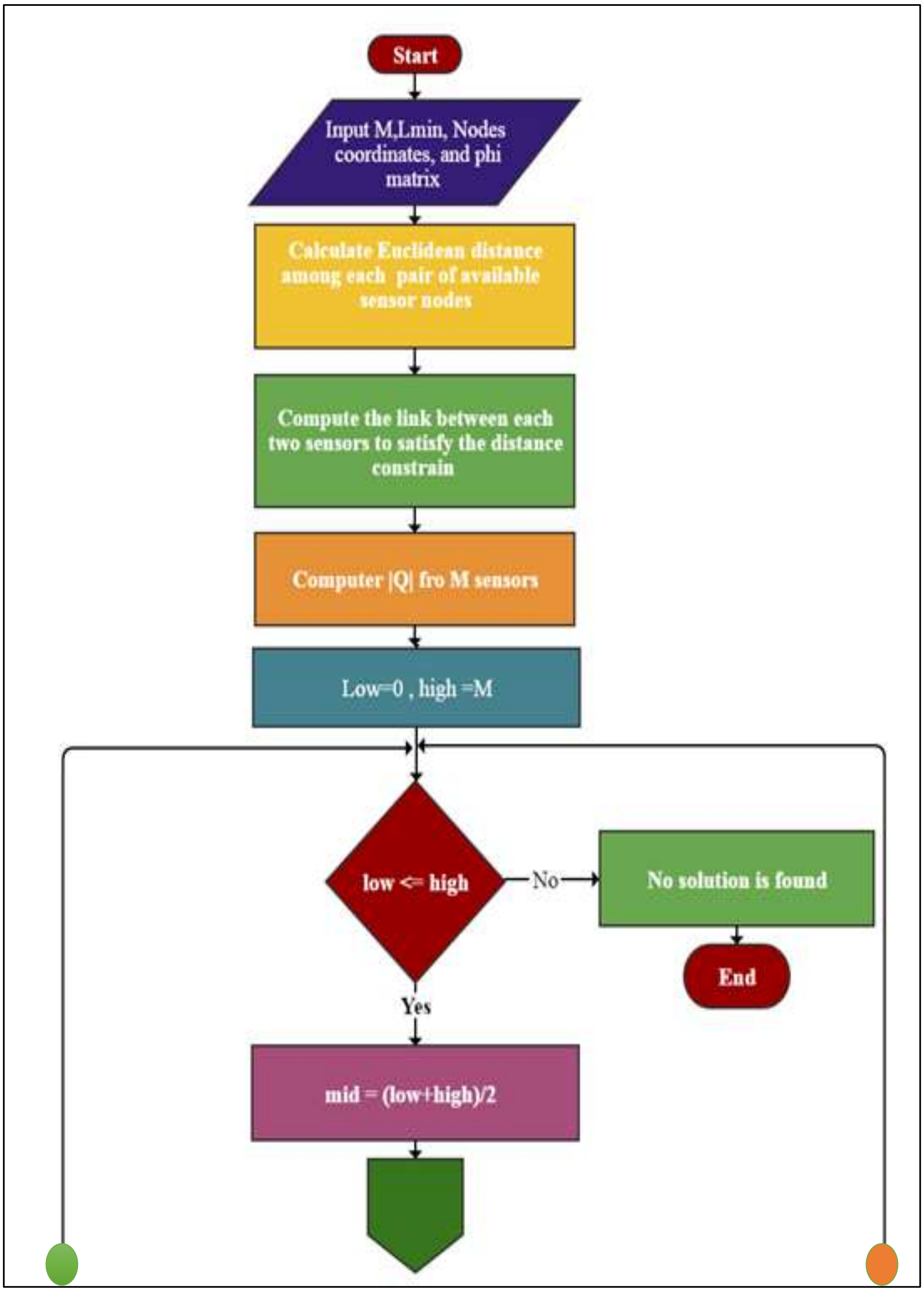
Another parameter that can be passed to the *ga* function is options [60]. In this thesis the default values set by the MATLAB optimization tool box are used to solve the problem, and one can see the important set of options chosen in Table 5, where the methods selected were explained before in sections 3.3.3 Selection, 3.3.4 Crossover, and3.3.5 Mutation.

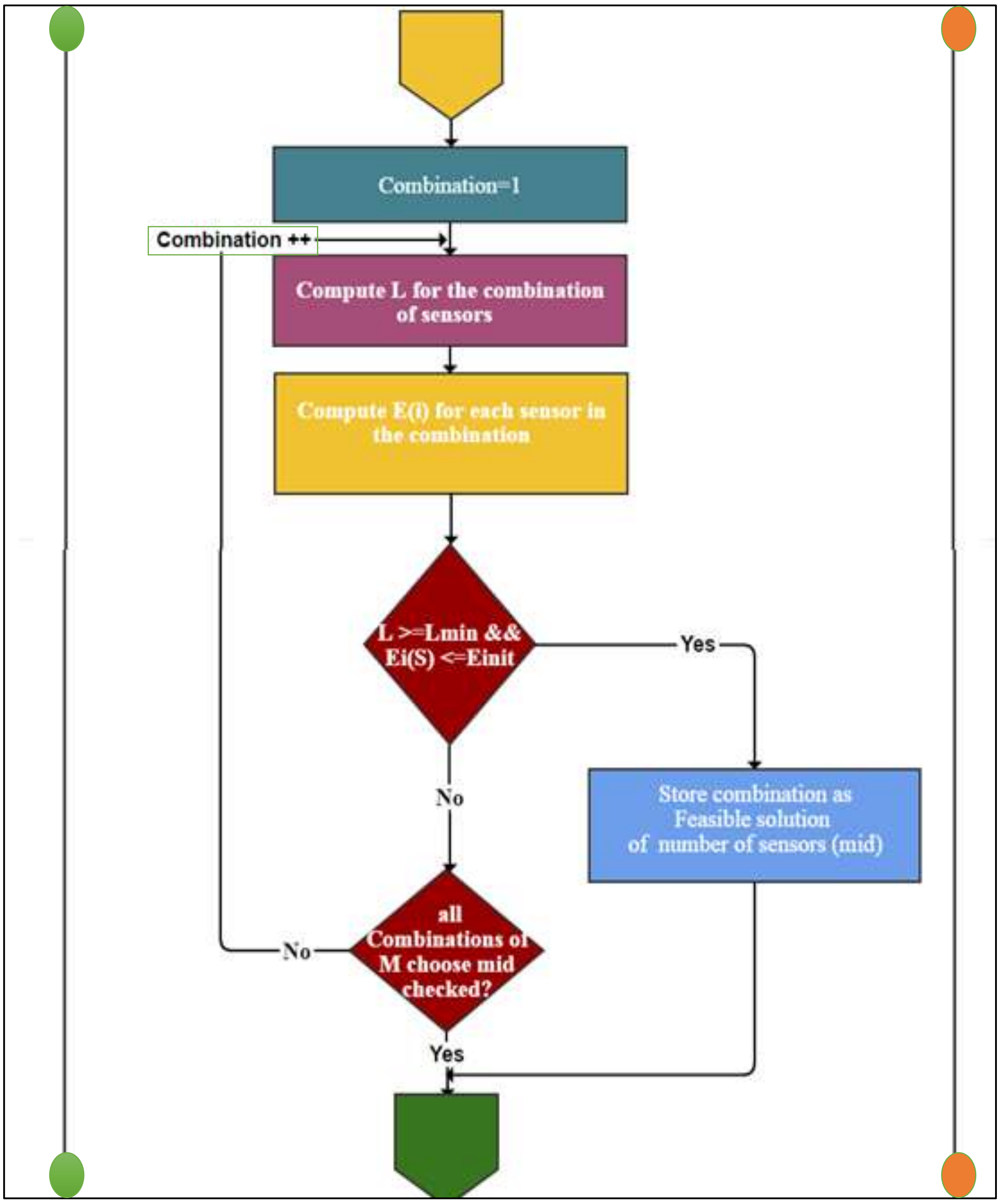
Table 5: Some of the most important default settings when calling ga function

Option	**Description**	**Method selected**
`SelectionFcn`	This option is used to choose the selection algorithm used in *ga*.	Stochastic uniform (stochastic universal sampling)
`CrossoverFcn`	This option is used to handle the crossover methodology.	Scattered crossover
`MutationFcn`	This option is used to express the mutation process.	Adaptive feasible

4.3 Implementation of the Binary Search Method

The flowchart of the implemented method is shown in Figure 5.


Start
Input M,Lmin, Nodes coordinates, and phi matrix
Calculate Euclidean distance among each pair of available sensor nodes
Compute the link between each two sensors to satisfy the distance constrain
Computer |Q| fro M sensors
Low=0 , high =M
low <= high
No
No solution is found
End
Yes
mid = (low+high)/2


Combination=1
Combination ++
Compute L for the combination of sensors
Compute E(i) for each sensor in the combination
L >=Lmin && Ei(S) <=Einit
Yes
No
Store combination as Feasible solution of number of sensors (mid)
all Combinations of M choose mid checked?
No
Yes

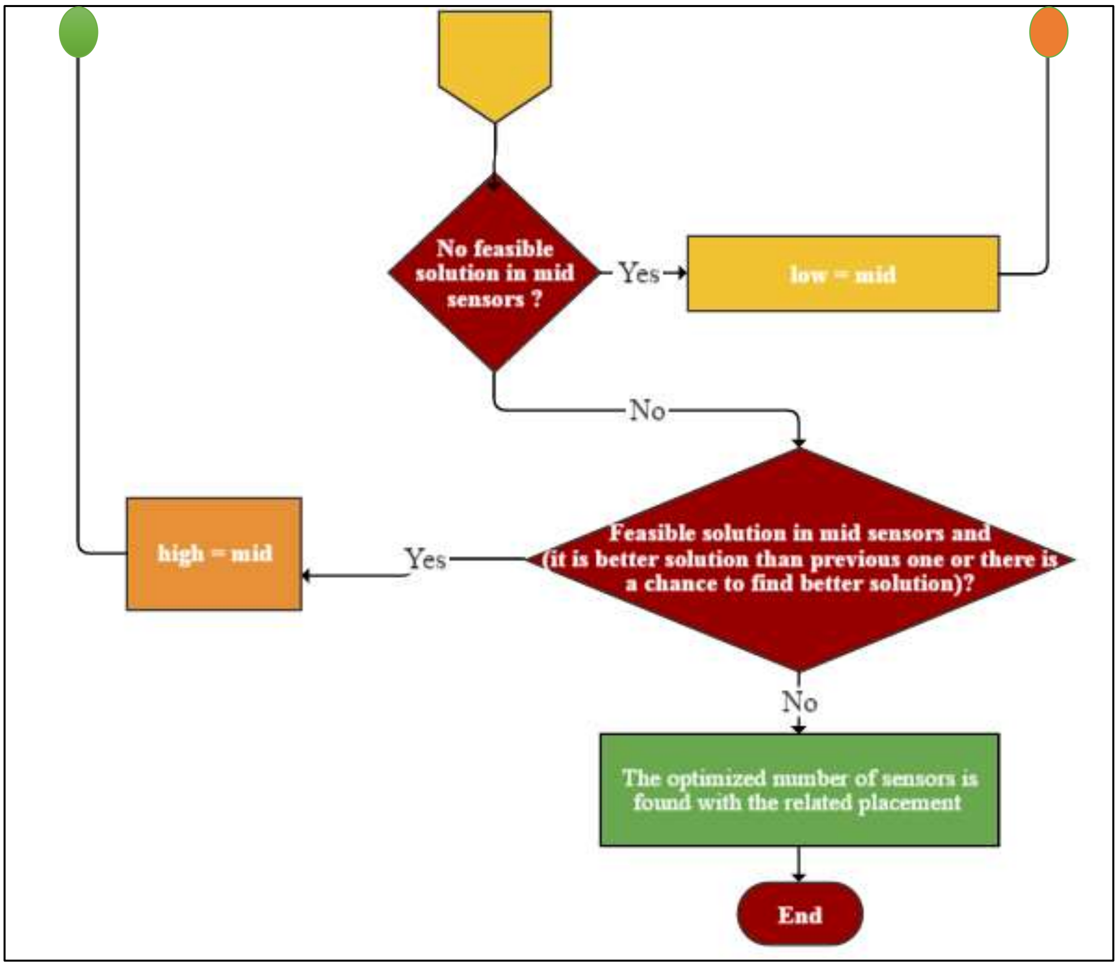

No feasible solution in mid sensors ?
Yes
low = mid
No
Feasible solution in mid sensors and (it is better solution than previous one or there is a chance to find better solution)?
Yes
high = mid
No
The optimized number of sensors is found with the related placement
End

Figure 5:Flowchart of the implemented binary search method

The algorithm used behind the flowchart is shown below:

Input: M, L_{min}, node coordinates, Φ , low (the lowest number of sensors can be used), and high (the highest number of sensors can be used).

Output: Optimized solution with an minimized number of sensors.

Algorithm:

The first three steps are the same first steps used in the implementation of the exhaustive search method:

1. Calculate $\mathrm{d_{ij}}\ \forall \mathrm{i,j}$ where $\mathrm{i,j}$ belong to M and the sink using their given coordinates.
2. Compute $x_{ij}\ \forall i,j$ where i,j belong to M and the sink node $\mathrm{M}+1$.
3. Calculate $|Q|$ when all M sensors are selected.

The next steps, from 4 to 10, show how the binary search method is implemented to find the best solution.

4. Check if $low \leq high$; if not, terminate the program. Otherwise, go to step 5.
5. Calculate the midpoint as (high + low) / 2.
6. Compute the different combinations of choosing midpoint sensors out of M sensors. Start with the first combination.
7. Go through the combination, and then check the feasibility of the solution.

To check the feasibility of a combination:

a. Detriment of |Q| and L(S) are computed for that combination of the selected sensors.
b. The total energy consumed by sensor i is computed for each sensor i from the total selected nodes in the combination.
c. Check the satisfaction of information quality constrain (C3) from equation (12) by comparing the results from step 7a with L_{min}.
d. Check the satisfaction of energy constrain (C2) from equation (12) by comparing the results from step 7b for each sensor i with E_{init}.
e. If all the constraints are satisfied, the solution of the combination of sensors is feasible. So stop looking through the other combinations to choose midpoint sensors out of M sensors. Then go to step 8.

Otherwise, the solution is not feasible, so go to the next combination and start again with step 7. Repeat step 7 until a feasible solution is found or all the combinations for choosing the midpoint out of M sensors are checked, and then go to step 8.

8. Check the validity of this statement: there is no feasible solution at midpoint sensors. If the statement is valid, it means that one needs to look for a feasible solution with a larger number of sensors. As a result, assign low to be the midpoint (low = mid), and then go to step 4. Otherwise, if the condition is not satisfied, go to step 9.
9. Otherwise, check if there is a feasible solution at midpoint sensors and whether that solution is better than the previous one. Or there is a chance of finding a better solution with a lower number. If any of the conditions holds, then assign high to be the midpoint (high =mid), and then go to step 4. Otherwise, go to step 10.
10. When this step is reached, it means that no better solution can be found and that the best solution has been found already. So display it out and terminate the program.

Chapter 5: Results, Evaluation, and Validation

This chapter shows the different results obtained from this research work and how the findings have been evaluated, tested, and finally validated. All the data and parameter values related to verification and testing will be presented with the adequate explanation.

5.1 Parameters and Platform

- **Parameters:**

The list of unified parameters used in all the different methods applied in this thesis with the needed description and values are presented in Table 6.

Table 6: The unified parameters used in all the methods applied in the thesis

Parameter	Description	Value
R_C	Maximum transmission range	30 m [5,6]
n^b	Number of bits per packet	2Kb [5,7,8]
A	The path loss exponent	2 to 6 [5,6]
E_{init}	Initial energy at a sensor node	1500mAhr [5,6]
ϵ_r	The reception energy cost	50nJ/bit [5,6]
ϵ_t	The transmission energy cost	50nJ/bit [5,6]
ϵ_{amp}	The power amplifier energy cost	10 pJ/bit/m^2 [5,6]

Platform specifications:

The device used in the implementation of all the experiments is a laptop for daily use. It has a ThinkPad T440s platform [64]. The specifications of the platform are shown in Table 7.

[5] This value is referred to the research work in [42].
[6] This value is referred to the research work in [16].
[7] This value is referred to the research work in [14].
[8] This value is referred to the research work in [15].

Table 7: Platform specifications[9]

Processor	4th generation Intel® Core™ i7 processor
Processor number	i7-4600U
Number of cores	2
CPU base frequency	2.10 GHz
CPU maximum turbo frequency	2.7 GHz
Installed memory	8.00 GB
System type	64-bit operating system
Software used	MATLAB

5.2 Experiments and results

In this thesis, exhaustive search, genetic algorithm, and binary search were applied using different numbers of sensors (five, nine, and 30) and one sink node. The setup and results for applying the different algorithms on the cases of five sensors , nine sensors, and 30 sensors are shown in this section.

5.2.1 Five story building

Although the number of sensors in this experiment is very small, this experiment is very important because it illustrates the concept and explains the idea of the studied problem, and it is used to validate the results and make sure that the algorithms used to solve the problem are implemented correctly since the results are

[9] You can get the platform by right-clicking on “My Computer”, then choose Properties.

considered small compared with the results of experiments with a larger number of sensors.

Setting up:

In this case, the building is assumed to be composed of five floors with no bays. Each floor is 3.65 meters high. It is also assumed that the five candidate sensors can be placed as one sensor per floor and that there is a one sink node used to collect the data out of other sensors in the WSN. The sink is placed 10 meters away from the first candidate sensor in the first floor. The sensor node coordinates are shown in Figure 6.

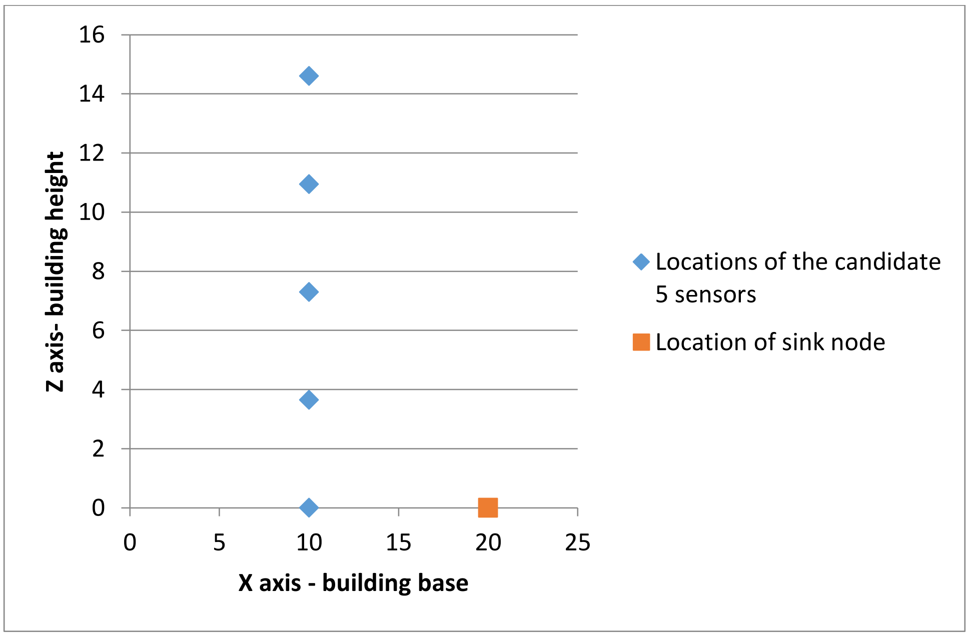


Figure 6: Coordinates of the five candidate sensors and the sink node

The mode shapes matrix (Φ) for the five sensors are shown below[65]:

$$\Phi = \begin{Bmatrix} 0.334 & -0.8954 & 1.173 & -1.078 & 0.641 \\ 0.641 & -1.173 & 0.334 & 0.895 & -1.078 \\ 0.895 & -0.641 & -1.078 & 0.334 & 1.173 \\ 1.078 & 0.334 & -0.641 & -1.173 & -0.895 \\ 1.173 & 1.078 & 0.895 & 0.641 & 0.334 \end{Bmatrix}$$

where each column represents a mode shape and each row represents the contribution of the candidate sensors in all mode shapes used. Another way to look at the mode shapes matrix is seen in Figure 7, which shows the natural modes of vibration of a five-story building.

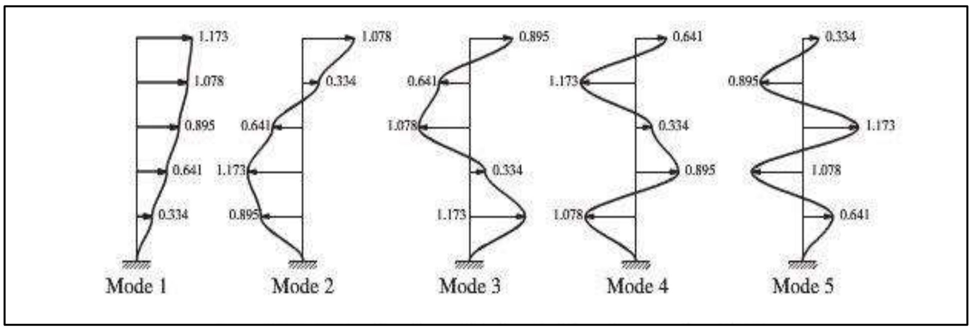


Figure 7:Five mode shapes used in a five-story building[65]

In all the experiments done, three mode shapes are used to represent the information quality matrix. Expect in one experiment where the relation between the number of mode shapes and number of sensors used in the solution is expressed and discussed.

Results of implementing brute force on a five-sensors story building :

The first part of the experiment is to implement brute force using five sensors. Using different runs, the minimum information quality required was changed from 0 to 100, and depending on that, each time different optimized solutions appear with different sensor placement is shown in Table 8 and Figure 8. Figure 8 shows the different sensor node placement based on the quality information required L_{min}. The sensor node placement showing the coordinates for each placed sensor using the candidate sensor node locations is shown in Figure 6. For example, when L_{min}= 20, the best solution is to place three sensors out of three in the first floor, third floor, and fifth floor where the coordinates of the first sensor in the first floor (10,0), the coordinates of the second sensor in the third floor (10,7.3), and the coordinates of the last sensor in the fifth floor is (10,14.6).

Table 8: Results of implementing exhaustive search on five-sensors story building

L_{min}	**minimized**	**Optimized Placement**					**time**
	# of sensors	**1st f**	**2nd f**	**3rd f**	**4th f**	**5th f**	**(s)[10]**
100	5	1	1	1	1	1	0.128264
80	5	1	1	1	1	1	0.134359
60	4	1	1	1	0	1	0.140677
40	4	1	1	1	0	1	0.145564
20	3	1	0	1	0	1	0.150977
0	0	0	0	0	0	0	0.001659

[10] The time is calculated using the functions tic, and toc in MATLAB software.

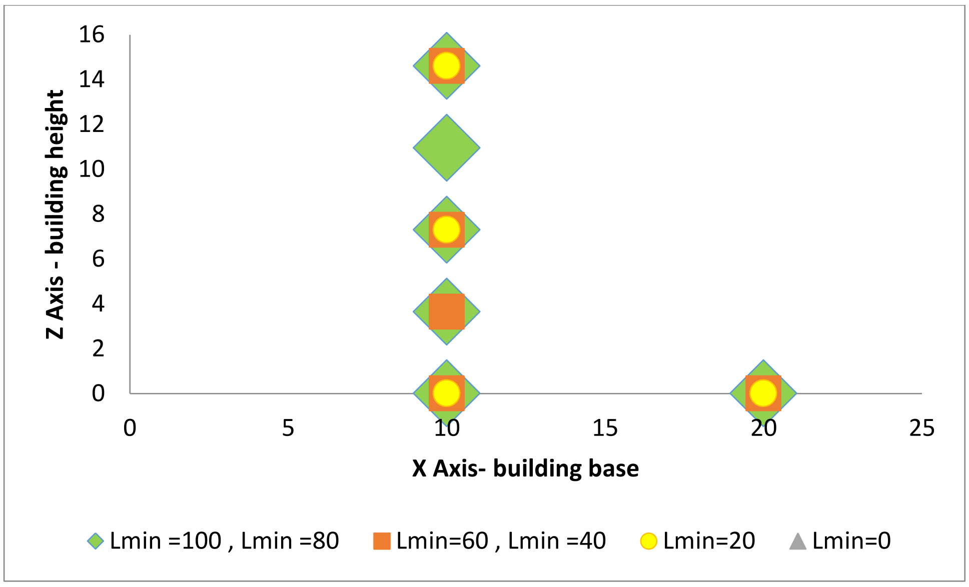


Figure 8: five-sensors story building placement based on different $\boldsymbol{L_{min}}$ *required*

In this experiment, the optimal number of sensors is found with the best information quality that can be reached. For example, when the L_{min} required is 20, the optimal solution is to place three sensors, one in the first floor, another in the third floor, and the last one in the fifth floor, and when one calculates the percentage of information quality L, he/she finds that L= 33.28. This quality information is the highest when three sensors are placed, and that happened because in the end of the algorithm the feasible solutions are sorted depending on the minimized number of sensors and then based on the best information quality.

One can see in Table 8 that whenever the quality information needed increases, more sensors are required to be placed. And this matches what is found in literature [14], [15], and [16].

Validation:

The results of this part of the experiment are validated by calculating the information quality for all the different combinations and manually checking the results based on L_{min} as shown in Table 9.. For example, when $\mathrm{L}_{\mathrm{min}}$=80 , the optimal solution using the brute force algorithm is choosing all the five sensors. And this matches the results in Table 9, where the least number of sensors that results with minimum L=80 is the combination of choosing all the five sensors where L =100. Another example when $\mathrm{L}_{\mathrm{min}}$ =5, the best solution is the combination of choosing the first, third, and fifth sensors out of the five sensors. This is validated manually by looking at Table 9. The combinations that gives $\mathrm{L}_{\mathrm{min}} = 5$ are [(1,2,5), (1,3,4), (1,3,5), (2,3,4), (2,3,5), (1,2,3,4), (1,2,3,5), (1,2,4,5), (1,3,4,5), (2,3,4,5), and (1,2,3,4,5)], and the best combination is (1,3,5) where it has the least number of sensors and best information quality among the combinations that place three sensors out of five.

Table 9:The computed information quality L for all the different combination of five sensors

Combination	**L**	**Combination**	**L**	**Combination**	**L**
1	7.99E-33	25	2.03E-15	235	22.991
2	3.83E-33	34	1.97E-15	245	0.1321
3	1.97E-32	35	4.07E-15	345	2.2055
4	-2.96E-33	45	8.78E-17	1234	30.463
5	0	123	0.258007	1235	65.583
12	-1.62E-16	124	4.494342	1245	15.55
13	7.13E-16	125	9.04963	1345	52.067
14	0	134	14.70369	2345	36.336
15	0	135	33.28403	12345	100
23	-5.42E-16	145	1.874299		
24	7.23E-16	234	11.00719		

Moreover, the optimal solutions are checked using another method that guarantee global optimization called branch and bound[11] using a solver called the BARON. The BARON solver[12] gives the same results we get from implementing exhaustive search algorithm. Finally, the run was repeated several times, and the same results always show up.

[11] For more details about the branch-and-bound method check reference [63].

[12] More information about the BARON solver can be found in http://archimedes.cheme.cmu.edu/?q=baron [21].

Results of implementing GA on a five-sensors story building :

The second part of the experiment is to implement genetic algorithm using five sensors. Using different runs, the minimum information quality required was changed from 0 to 100 , and depending on that, each time different optimized solutions appear with different sensor placement is shown in Table 10. In this part, the run is repeated 10 times for each required L_{min} . The different runs for a specific L_{min} always show the same optimized solution with the least number of possible sensors. On the other hand, different sensors placement is presented where the placements satisfy the problem's objective function and constraints. For example, when L_{min}=40 , in 10 different runs, we get the same minimized number of sensors (4). On the other hand, two different node placements appear among 10 different runs, which are the combinations [(1,2,3,5) and (1,3,4,5)] as shown in Table 10. In some other cases, we always get the same optimal solution of the same sensor locations as in L_{min} =80.

Table 10: Results of implementing genetic algorithm using a five-sensors story building

L_{min}	**Minimized # of sensors**	**Optimized Placement**	**Average time (s)**
		1 2 3 4 5	
100	5	1 1 1 1 1	1.085821
80	5	1 1 1 1 1	1.083507
60	4	1 1 1 0 1	1.088228
40	4	1 1 1 0 1	1.098744
	4	1 0 1 1 1	
20	3	1 0 1 0 1	1.098624
	3	0 1 1 0 1	
0	0	0 0 0 0 0	1.091493

Validation:

First, in genetic algorithm, there is randomness in selection, so it is normal to get different solutions or near-optimal solution. In this case, all the solutions were optimal.[13] This is validated by finding that for different required L_{min}, we get the same number of sensors as in exhaustive search implementation. The validation of the different placements we have in the different runs for a certain $L_{min}$is done using the values in Table 9. For instance, in Table 10, when $L_{min} = 20,$ the best solution is placing three sensors out of five. And from the different runs, there are two ways that

[13] Note that using genetic algorithm does not guarantee global optimality in other cases

satisfy this: using the combination (1,3,5) or the combination (2,3,5). If you look at Table 9, you can find that the combination (1,3,5) has quality information=33.28, and the combination (2,3,5) has quality information=22.991. This guarantees that those two solutions are true. Furthermore, if one asks, “Can we get other solutions with more different runs?” the answer in this case is no. The evidence on that is in Table 9, where you will not find any other solution that meet the constraints and the objective function of the problem other than (1,3,5) and (2,3,5).

Results of implementing binary search on a five-sensors story building :

The results are shown in Table 11, and Figure 8. One can see that we get the same number of minimized sensors as in implementing brute force method, but with less time since not all the possible combinations of five sensors are checked. This saves time, and ensures less complexity.

Validation:

The validation of the results is easy, because the results are only compared to the optimal solutions from implementing brute force. And the good thing is that the results are the same. Furthermore, the change in time is logical. For example, when L_{min}=60 and brute force is implemented, we need to check $2^5 - 1 = 31$ different combinations to end up with the best solution of using four sensors out of five. But when binary search is implemented, you start with low=0, and high=5. The first midpoint equals 3,and all the different combination of choosing three out of five are checked (which are 10 combinations), but the result is that there is no feasible solution with three sensors. As a result, low is assigned as 3, so the new midpoint equals 4. The second combination of four sensors (1,2,3,5) gives a feasible solution, so no more combinations of choosing four out of five will be checked. Using the implemented algorithm 4 sensors is the best solution to solve the problem when L_{min}=60, so the program will terminate. The total number of combinations checked in this example is 12 which is less than 31, and that for sure can reduce the time in solving the problem as shown in Table 11.

Table 11: Results of implementing binary search on a five-sensors story building

L_{min}	**Minimized**	**Optimized Placement**					**time (s)**
	# of sensors	**1st f**	**2nd f**	**3rd f**	**4th f**	**5th f**	
100	5	1	1	1	1	1	0.016758
80	5	1	1	1	1	1	0.034036
60	4	1	1	1	0	1	0.065281
40	4	1	1	1	0	1	0.053999
20	3	1	0	1	0	1	0.035904
0	0	0	0	0	0	0	0.002584

Summary:

As you can see, implementing the three methods on five sensors gives the same minimized number of sensors. But there are some differences from different point of views. For example, applying brute-force gives the optimal number of sensors with the best sensor placement that guarantees the largest information quality that can be reached. On the other hand, when this method is implemented, all the different combinations of five sensors are checked, and this results with longer execution time that applying the binary search method that gives better execution time and less complexity, and this is validated above. Moreover, in applying GA, one can see that different placements for the sensors can appear with different runs, and this is due to randomization in selection.

5.2.2 Nine-story building

In this experiment, the number of floors is increased to nine, and the candidate sensor locations are nine as well with the sink sensor node. This experiment shows better results in terms of minimizing the number of sensors since the candidate sensor locations are increased. Moreover, having more than one experiment with different M candidate sensors help in comparing between the three methods applied.

Setting up:

The sensor node coordinates are shown in Figure 9.

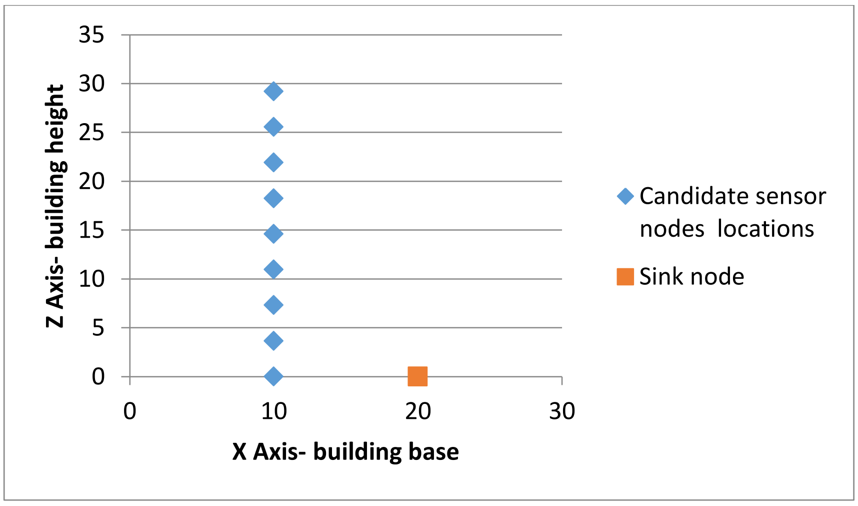


Figure 9: Nine Candidate sensor locations

The mode shapes matrix (Φ) for the nine sensors using three mode shapes are shown below[14]:

$$\Phi = \begin{Bmatrix} 0.26 & -0.69 & 1.12 \\ 0.51 & -1.25 & 1.68 \\ 0.75 & -1.58 & 1.39 \\ 1.04 & -1.60 & 0.08 \\ 1.29 & -1.22 & -1.29 \\ 1.50 & -0.52 & -1.79 \\ 1.73 & 0.72 & -0.75 \\ 1.87 & 1.69 & 1.04 \\ 1.91 & 2.01 & 1.79 \end{Bmatrix}$$

Results of implementing brute force on a 9-sensors story building :

In this part of the experiment, the optimal number of sensors is found with the best information quality that can be reached. The placement of the minimized number of sensors based on the required information quality is seen in Figure 10.

In addition to that, Table 12 shows the results of implementing exhaustive search on a nine-sensors story building. You can see that the minimization of the number of sensors is enhanced by comparing the solutions shown on implementing the same method on a five-sensors story building.

[14] The matrix is provided by Dr. Mohamed Mahgoub, and computed by Dr. Mostafa Elmorsi using Advanced SAP2000 v17.1.1 (an integrated solution for structural analysis and design).

Table 12: The results of implementing brute force on a nine-story building

L_{min}	minimized	Optimized Placement									time (s)
	#of sensors	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
100	9	1	1	1	1	1	1	1	1	1	0.175736
80	8	0	1	1	1	1	1	1	1	1	0.152554
60	7	0	1	1	0	1	1	1	1	1	0.157443
40	6	0	1	1	0	1	1	0	1	1	0.16647
20	5	0	1	1	0	1	1	0	0	1	0.151972
0	0	0	0	0	0	0	0	0	0	0	0.129678

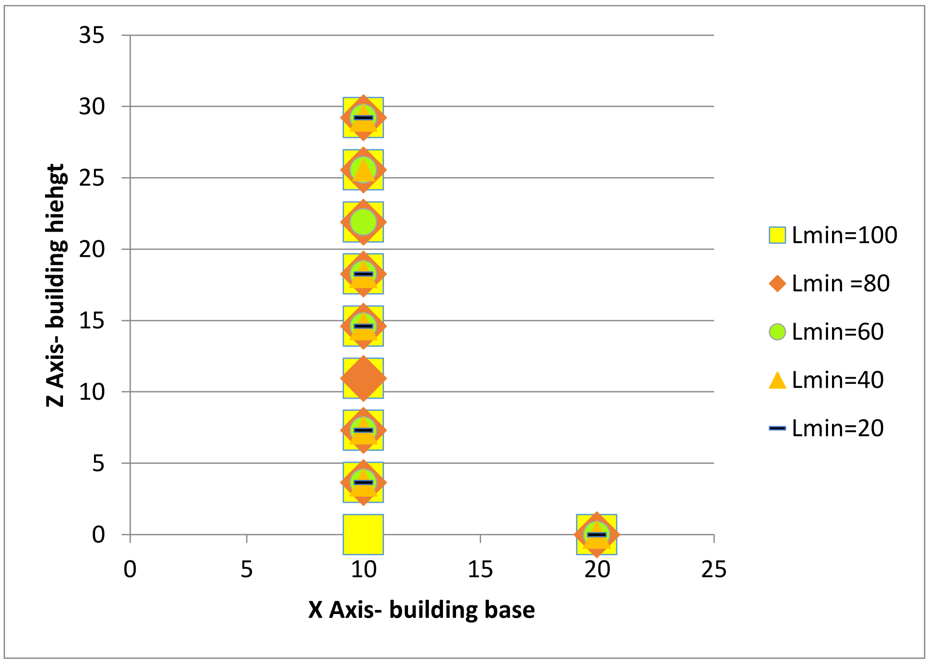


Figure 10: Sensor placement based on implementing brute force for different required information quality

Validation:

The validation of this part of the experiment follows the same way of implementing brute force on a five-sensors story building where the information quality for all the different combinations ($2^9 - 1 = 511\ combination$) are computed and then choosing some random results and checking them manually. For example there are two combinations out of nine sensors that satisfy the constraint $\mathrm{L_{min}}$=80. The combinations are [(1,2,3,4,5,6,7,8,9) and (2,3,4,5,6,7,8,9)] . But since the second combination satisfies the objective function, which is minimizing the number of sensors, then it is chosen to be the optimal solution.

Results of implementing GA on a nine-sensors story building :

Increasing the number of sensors from five to nine proves that the genetic algorithm can't guarantee optimal solutions, but it can find near-optimal solutions because of the randomization used in selection. This is shown when $L_{min} = 40$. In Table 13, and Figure 11, one can see that depending on 10 different runs, different solutions with different number of sensors resulted, where the first combination (2,3,5,6,8,9) show the optimized solution of placing six sensors out of nine, but the other two combinations (3,4,5,6,7,8,9) and (1,2,3,4,6,7,9) place seven out of nine candidate sensors. Moreover, from the results one can see different feasible solutions are outputted for the same required L_{min} in different runs (as shown before in the case of five sensors).

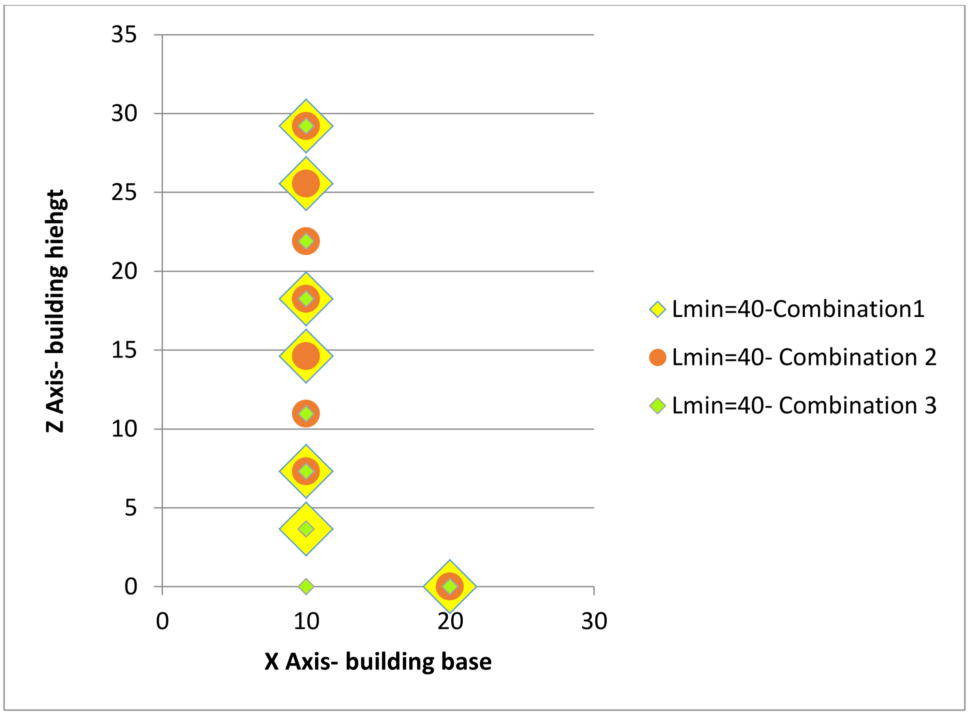


Figure 11:Different results shown when implementing genetic algorithm using Lmin=40

Table 13: Results of implementing the genetic algorithm on a nine-sensors story building

L_{min}	Minimized	Optimized Placement									Average
	#of sensors	1	2	3	4	5	6	7	8	9	time (s)
100	9	1	1	1	1	1	1	1	1	1	2.078938
80	8	0	1	1	1	1	1	1	1	1	2.099514
60	7	0	1	1	0	1	1	1	1	1	2.073879
	7	0	1	1	1	1	1	0	1	1	
40	6	0	1	1	0	1	1	0	1	1	2.032474
	7	0	0	1	1	1	1	1	1	1	
	7	1	1	1	1	0	1	1	0	1	
20	5	0	1	1	0	1	1	0	0	1	2.085858
	5	0	0	1	1	0	1	1	0	1	
	5	0	1	0	1	0	1	1	0	1	
	5	0	0	1	1	0	1	0	1	1	
	5	0	1	1	1	0	1	0	0	1	
0	0	0	0	0	0	0	0	0	0	0	2.032564

Validation:

First, the optimal results that we get from implementing brute force for each required L_{min} appear within the 10 runs for implementing the genetic algorithm. That is a good sign, but to validate the cases where different solutions appeared for the required L_{min}, random check is done. Information quality is computed for the combinations that did not show optimal solution from implementing exhaustive search. Then the combination is checked whether it is greater than L_{min}or not. If it is larger than L_{min}, then the solution meets the constraints and objective function, so it is feasible and correct. For instance, when L_{min}=60, the combination that does not show in implementing the brute force on nine sensors is (2,3,4,5,6,8,9). This solution is feasible because the information quality L equals to 63.451, which is larger than 60.

Results of implementing binary search on a nine-sensors story building :

The results of this part are shown in Figure 12 and Table 14. From the results, we can find that the optimal solutions with a minimal number of sensors appeared in implementing exhaustive search. On the other hand, it is not necessary to have the same sensor placement for the same required L_{min} in both methods. Actually, the reason behind this is to make the execution faster using the binary search method. In implementing the binary search method whenever there is a feasible combination of choosing midpoint sensors out of M sensors, the algorithm will break the loop ,and will not look for another solution of the same number of sensors.

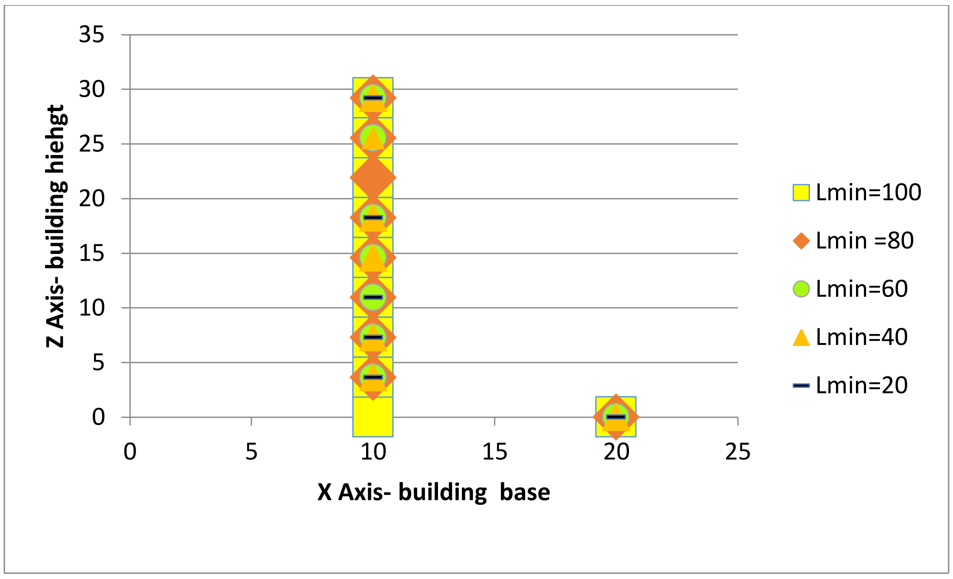


Figure 12: Sensor placement when implementng binary search on a nine-sensors story building

Table 14: Results of implementing binary search on a nine-sensors story building

L_{min}	Minimized	Optimized Placement									time (s)
	#of sensors	1	2	3	4	5	6	7	8	9	
100	9	1	1	1	1	1	1	1	1	1	0.020687
80	8	0	1	1	1	1	1	1	1	1	0.028136
60	7	0	1	1	1	1	1	0	1	1	0.035904
40	6	0	1	1	0	1	1	0	1	1	0.038616
20	5	0	1	1	1	0	1	0	0	1	0.035586
0	0	0	0	0	0	0	0	0	0	0	0.006462

Validation:

In this part when L_{min} equals 100, 80, 40, or 0, the results are correct and valid because they are the same as the result output from implementing brute-force. For the other cases when L_{min} equals 60 or 20, the number of candidate sensors to be used is the same, but the placement of the sensors is different, so the placements given are checked. When L_{min}=60, the combination chosen to be the solution (2,3,4,5,6,8,9) is 63.25, which is greater than 60. So the combination is valid as solution for this required information quality. The same when L_{min}=20, the information quality computed for the output combination (2,3,4,6,9) is 22.26. This makes this solution right.

Summary:

From the results of applying the three algorithms on nine sensors, more important details appear. For instance, the results prove that using GA can't provide 100% warranty of getting the optimal solution of the minimal number of sensors. Moreover, using the heuristic method of applying binary search guarantees to give the optimal solution within much shorter time than applying exhaustive search.

Relation between number of mode shapes and number of sensors used to solve the problem:

In this part of the experiment, we want to check the effect of increasing the number of mode shapes on the results that we get. For simplicity, we run the experiment on a five-story, and nine-story building for different required L_{min}. The experiment done on five-story building is checked for the first five mode shapes. And the experiment done on a nine-story building is checked for the first nine mode shapes.[15] The number of sensors obtained in the different number of mode shapes is shown in Figure 13 and Figure 14. One can see that increasing the number of mode shapes leads to hiring more sensors to solve the problem. This matches what is found in literature as in [15] and [14]. In this research work, our experiments are conducted using three mode shapes as used in literature [15] and [14]. In addition to that, three mode shapes are advised and recommended by people in civil engineering for a typical and normal tower[16]where they informed that the mostly used mode shapes are the first three out of 12.

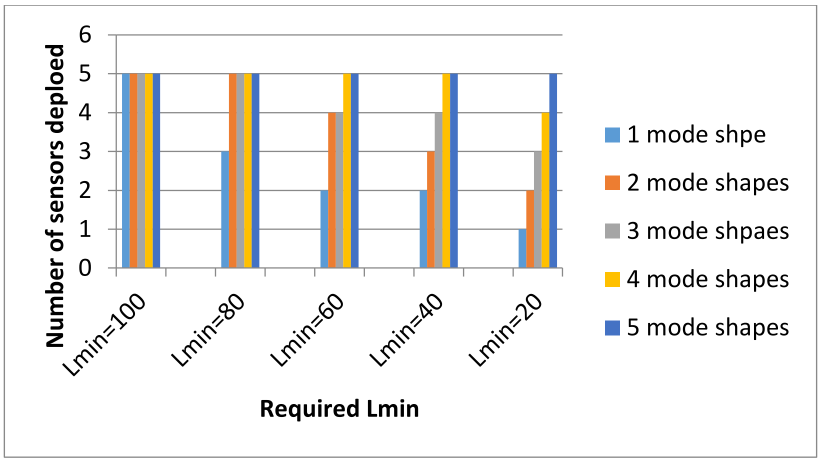


Figure 13: The effect of increasing the number of mode shapes on a five-story building

[15] The number of mode shapes K checked out is based on the available data we can get. For example, in the phi matrix for five-story building, we have the needed data for the first five modes only out of 12. In the phi matrix for nine-story building, we have the needed data for the first nine modes out of 12.

[16] The number of mode shapes is advised by Dr. Mohammed Mahgoub, an associate professor and program director of the concrete Industry Management (CIM) program at the John A. Reif Jr. Department of Civil and Environmental Engineering, Newark College of Engineering, New Jersey Institute of Technology, University Heights Newark, New Jersy

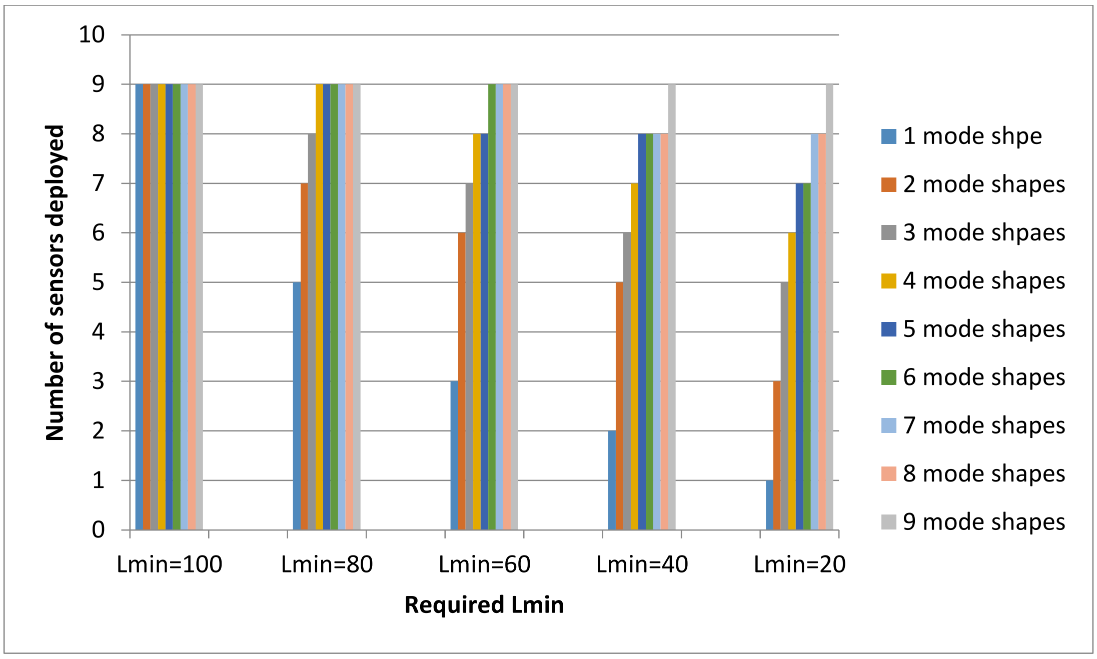


Figure 14:The effect of increasing the number of mode shapes on a nine-story building

5.2.3 Two-bay – nine-story building

In the following experiment, the number of floors is nine as in the last experiment, but the candidate sensors locations are increased to 30. Increasing the number of candidate sensors can better show the importance of the solved problem in this thesis. In addition to that, the first two experiments are conducted to validate the idea of the solved problem and make sure of the correctness of the values. This experiment ensure the same. Furthermore, it is more realistic in terms of having two bays in the simulated building. Like others, this experiments gives good insight on the implemented algorithms to solve the problem ,and shows the difference in the performance and complexity between them.

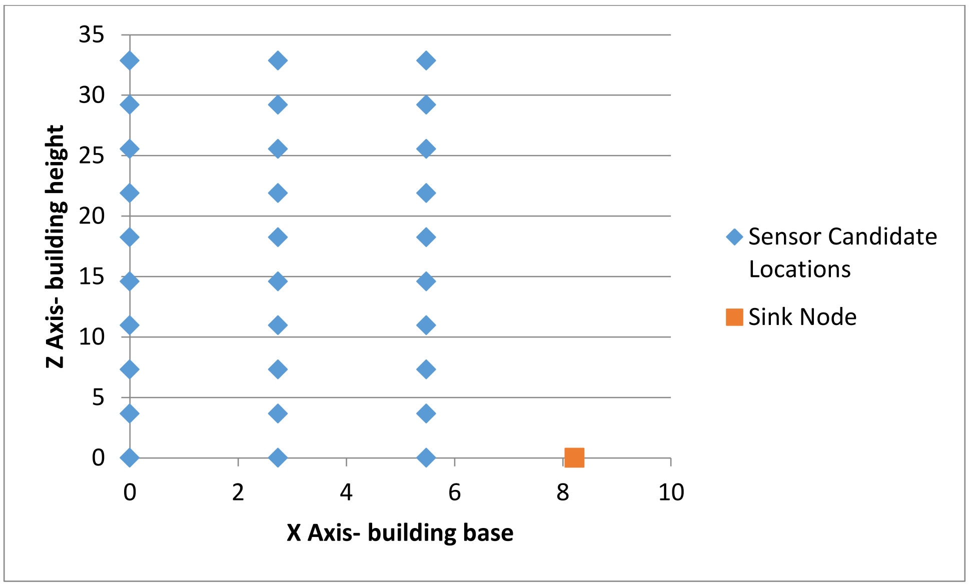


Figure 15: Candidate sensor locations in a two-bays nine-sensor story building

Setting up:

The sensor node coordination is seen in Figure 15. Moreover, the labels of the sensors are shown in Figure 16. For example, the candidate sensor locations in the first floor (not ground) are called 2, 12, and 22 and so on.

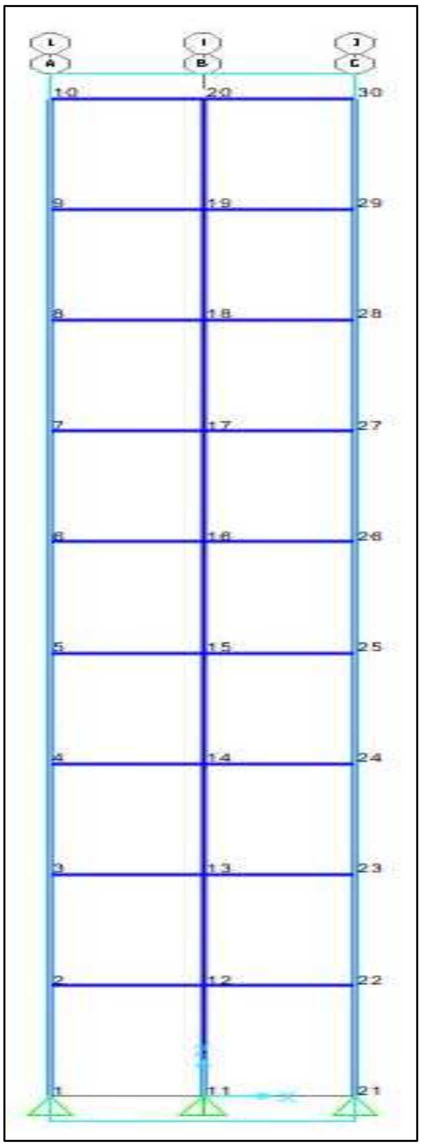


Figure 16: Labels of the candidate sensor locations in a two- bay nine-sensor story building

The target mode shapes matrix(Φ) for the two-bay nine-sensor story building using three mode shapes is shown below:

$$
\Phi = \begin{Bmatrix}
0 & 0 & 0 \\
-2.011978 & 6.838295 & 0.00016 \\
-4.007836 & 12.081435 & 0.000268 \\
-6.056818 & 14.740655 & 0.000281 \\
-8.165988 & 14.269124 & 0.000257 \\
-10.312154 & 10.691622 & 0.00022 \\
-12.459848 & 4.604105 & 0.000177 \\
-14.571959 & -2.953242 & 0.000127 \\
-16.616821 & -10.726328 & -0.000047 \\
-18.574111 & -17.544559 & -0.001823 \\
0 & 0 & 0 \\
-2.011959 & 6.842414 & 7.222E-09 \\
-4.007691 & 12.08817 & 1.183E-09 \\
-6.056809 & 14.748933 & -1.13E-09 \\
-8.166107 & 14.277179 & 5.307E-09 \\
-10.31237 & 10.697648 & -1.237E-08 \\
-12.460146 & 4.606623 & 9.348E-09 \\
-14.572332 & -2.955153 & 3.63E-09 \\
-16.617268 & -10.732907 & -8.068E-09 \\
-18.574541 & -17.554403 & 1.69E-09 \\
0 & 0 & 0 \\
-2.011978 & 6.838295 & -0.00016 \\
-4.007836 & 12.081435 & -0.000268 \\
-6.056818 & 14.740655 & -0.000281 \\
-8.165988 & 14.269124 & -0.000257 \\
-10.312154 & 10.691622 & -0.00022 \\
-12.459848 & 4.604105 & -0.000177 \\
-14.571959 & -2.953242 & -0.000127 \\
-16.616821 & -10.726328 & 0.000047 \\
-18.574111 & -17.544559 & 0.001823
\end{Bmatrix}
$$

Results of implementing brute force on a two-bay nine-sensor story building :

The results are shown in Table 15. The table shows the optimized number of sensors chosen to solve the problem with the corresponding L_{min} required. Additionally, the table shows the best sensor placement that satisfies the objective function and the constraint of the problem that can also be seen in Figure 17. Likewise in Table 15, you can see the total time needed to find each solution. This part of the experiment is the best among the experiments done in emphasizing the importance of solving the problem stated in this thesis. The minimization of the number of sensors is shown in a better way. For example when L_{min}=80, eight sensors are reduced out of 30 , and when L_{min}=60 , 13 sensors are reduced out of 30. This gives a huge expectation to increase minimizing the number of candidate sensors when the number of candidate sensors is increased. Moreover, this part of the experiment can show the disadvantages of using exhaustive search method, although it gives the optimal solutions. The disadvantage here is the long execution time. The average time to get the optimal solution for different required L_{min} is almost 18 hours. This is the result of checking all the different combinations out of 30 sensors and then sorting the feasible solutions to find the optimal solution.

Validation:

In this part of the experiment, the number of combinations out of 30 sensors is 1073741823. So it would be hard to get them all and check them one by one. As a result, to check the correctness of the results, some of the inputs are chosen randomly and checked corresponding to their satisfaction to the objective function and constraints. For example, when L_{min}=100, the best solution given is having 27 sensors out of 30. First, L is calculated for the given solution with its specific placement, and it gives 100. Moreover, the distance between the selected node locations is less than the transmission range given in this problem. In addition to that, the energy constraint is satisfied. Furthermore, to be more certain, the contribution of the three sensors that are not part of the solution is checked, and it is found that their contribution in the mode shape matrix is 0. All the evidence point that such result is true and correct.

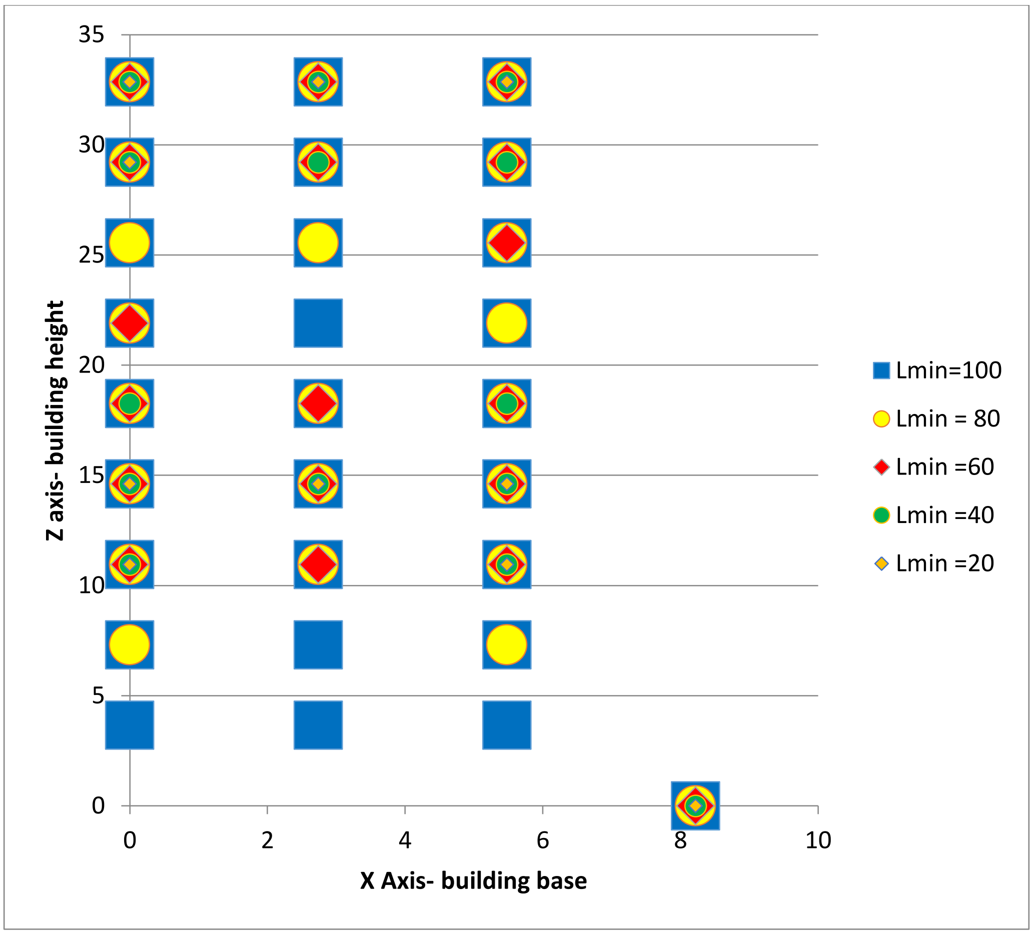


Figure 17: Sensor placement when applying brute-force on a two-bay nine-sensor story building

Table 15: Results of implementing exhaustive search on a two-bay nine-story building

L_{min}	**Optimized Placement**																														**Minimized # of sensors**	**time (s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**		
100	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	27	64464.51
80	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	1	1	0	0	1	1	1	1	1	1	1	1	22	64440.25
60	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	1	1	1	17	64129.87
40	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	13	64186.47
20	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	9	64215.24

Results of implementing genetic algorithm on two-bay nine-sensor story building:

The results of applying this algorithm are seen in Table 16. The different solutions from 10 different runs for each required L_{min} are stated. From the results, one can notice that although the execution time of implementing genetic algorithm is much less than the execution time of implementing exhaustive search, the genetic algorithm cannot guarantee outputting the optimal solution. For example, from the 10 different runs, when L_{min}=f20 or L_{min}=40, the solution given may be near optimal. Additionally, sometimes the execution time is finished by giving a non-feasible solution that provides the optimal number of sensors but does not satisfy the quality constraint. This happened when L_{min}=100, the second solution stated in Table 16. It gives the optimal number of sensors as found in exhaustive search implementation, but the information quality computed for the given sensor placement is 98.48, which is less than 100. That is why the solver gave a message without the output to indicate that the solution is not satisfying the constraints of the problem.

Validation:

Again, in this part, random results are checked, especially the suspicious ones. Near-optimal solutions are expected as the nature of genetic algorithm cannot be forced to have an optimal solution. One of the results that I checked is whenL_{min}=100. In one from the 10 runs, we got the following combinations (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29 ,30), it was given in the output that the solution is not satisfying the constraints. To validate that, the distance, energy, and information constraints are checked.

Table 16:Results of implementing genetic algorithm on a two-bay nine-sensor story building

L_{min}	**Optimized Placement**																														**Minimized**	**time (s)**
																															# of	
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**sensors**	
100	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	27	3.0124
	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	27	
80	0	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	22	3.0447
	0	0	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	22	
	0	0	1	1	1	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	22	
60	0	0	1	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	1	1	1	17	3.0313
	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	1	1	1	0	0	0	1	1	1	1	0	1	1	17	
40	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	13	2.983
	0	0	1	0	0	1	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1	0	1	0	1	1	1	15	
	0	0	0	0	1	0	1	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	1	0	1	1	14	
20	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	9	2.845
	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1`	0	0	0	1	1	9	
	0	0	0	0	1	1	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	1	1	10	

It is found that the quality of that combination is 98.48, which is actually not satisfying the information quality constraint, so the result given by the solver is true. Another random solution checked is (4, 5, 6, 9, 10, 14, 15, 16, 19, 20, 24, 25, 26, 27, 29, 30) This combination is found to be feasible with L= 60.1, which is larger than L=60. As a result, it is a valid solution when the required minimum L is 60.

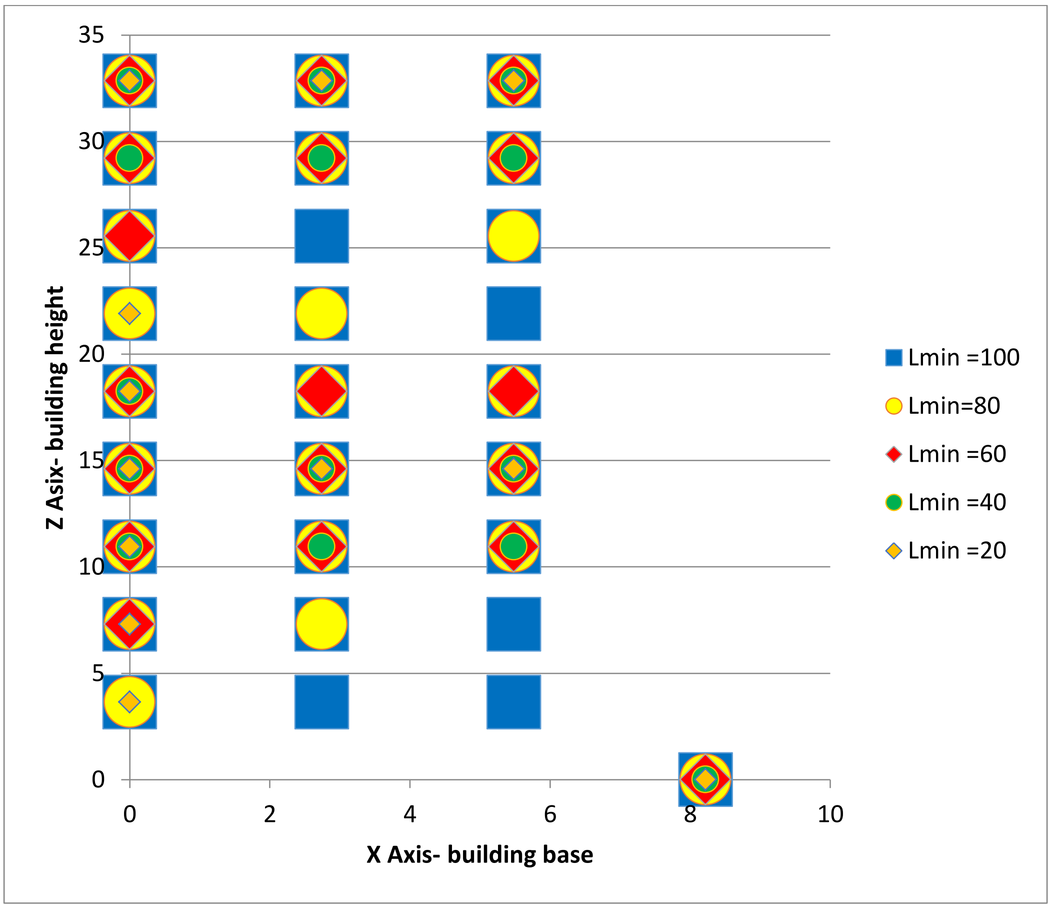


Figure 18: Sensor placement when implementing binary search on a two-bay nine-sensor story building

Results of implementing binary search algorithm on a two-bay nine-sensor story building:

The results of implementing binary search are seen in Figure 18, and Table 17. The results gives the same optimal solutions in terms of minimizing the number of sensors as shown before in implementing the brute force method. The sensor placement is different, but it meets the problem objective function and constraints.

Validation:

Since the number of sensors are the same in both cases (binary search and brute force) this is a good. To better validate the results, the constraints are checked for all the combinations that resulted in the different required $L_{\min}$.It is found that all of the results are feasible and optimal. Table 18 shows the computed L for each combination seen in the results in Table 17.

Summary:

This experiment makes the seen very clear. First this experiment gives the expectation that whenever a number of candidate sensors increase, the number of sensors that we can reduce increases as shown in Figure 19. This confirms that the thesis problem can be shown clearly with structures that need large number of sensors to monitor its health. As a result, this emphasizes the importance of solving the problem in those cases and optimizing the solution, which is the core of this research work. Furthermore, from this experiment, one can see that optimal solutions can be found using exhaustive search and binary search methods while genetic algorithm can find near or sub optimal solutions. In this experiment, one can also see that the time execution for all the applied algorithms increases since there are more sensors. The execution time of applying the genetic algorithm is the best in this experiment, where the average time to apply genetic algorithm on a two-bays , 9-sensor story building is 2.98 seconds. In contrast, there is no guarantee to find optimal solution. In addition, the worst execution time results from implementing exhaustive search, where the average time used when implementing this method is 64281.45 seconds, which equals to 17.86 hours. Solving the problem with the binary search method is the best among the algorithms used, as it guarantees optimal solution and, better time than exhaustive search, averaging time 4.4 hours.

To summarize, Figure 20 shows the minimized number of sensors in all the different combinations. One can see that binary search and exhaustive search always give the optimal solution. On the other hand, genetic algorithm cannot assure that, and this is further shown with the increase of candidate sensors. In addition, Figure 21 shows that binary search is the best method among the three applied methods in terms of execution time to use for building that needs a small number of sensors to monitor its health. On the other hand, for towers that need a larger number of sensors, there is a trade-off between the execution time and the performance. If execution time is preferred, then genetic algorithm should be applied as shown in Figure 22. But if performance is preferred, then binary search will be better to use than brute force.

Another way of comparing the methods used is by computing and evaluating the time complexity that is shown in the next section.

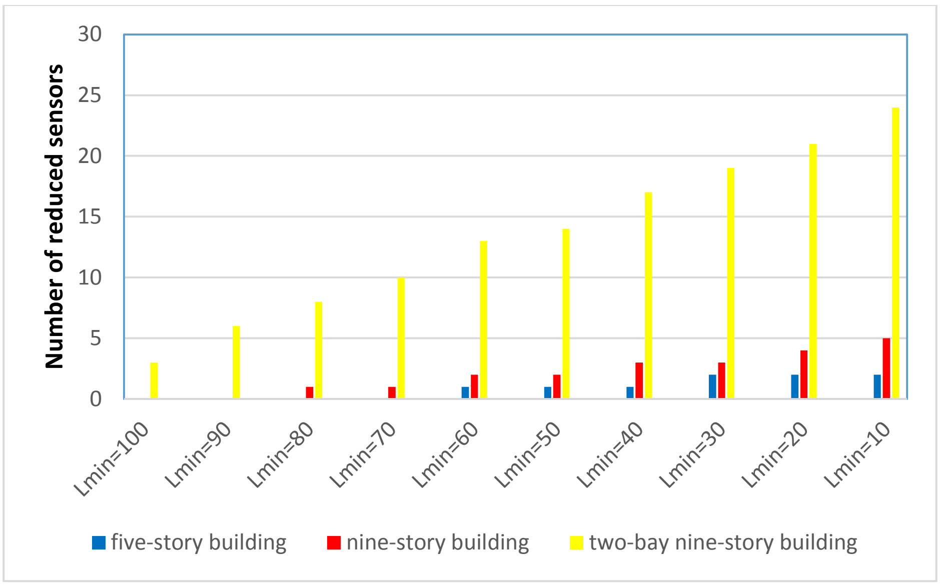


Figure 19: The number of sensors reduced in all the different experiments for different required $\boldsymbol{L_{min}}$ using brute force and binary search

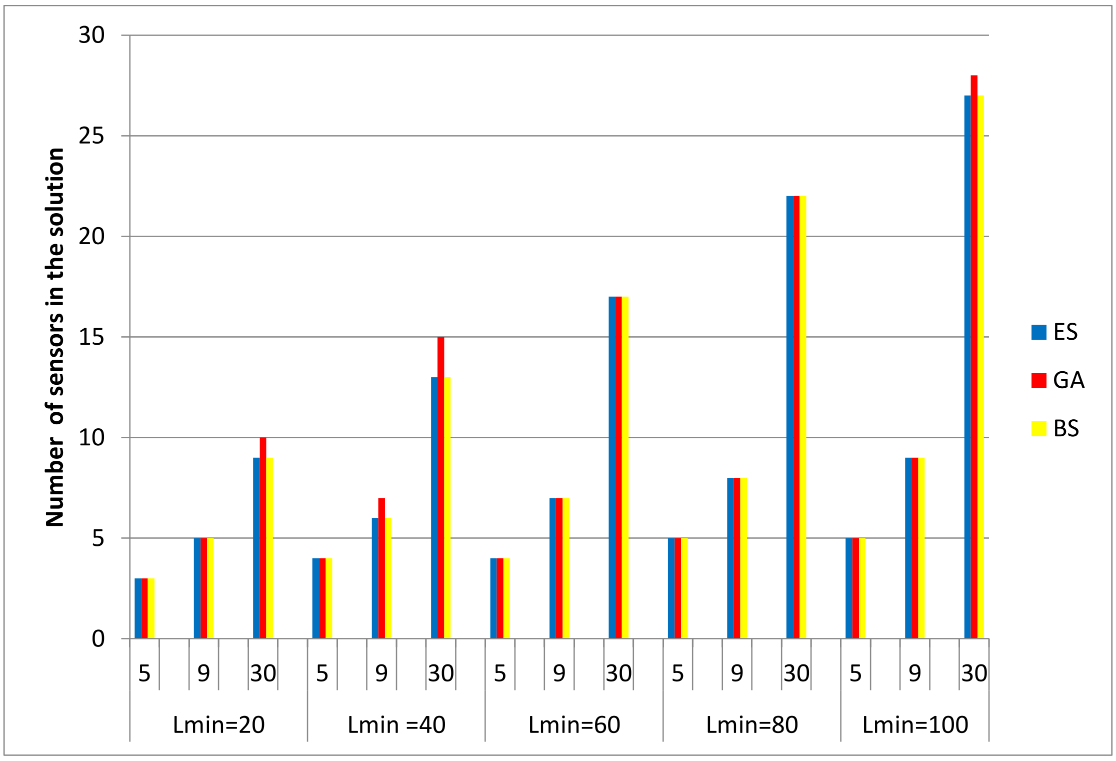


Figure 20: The number of sensors from implementing exhaustive search (ES), genetic algorithm (GA), and binary search (BS)

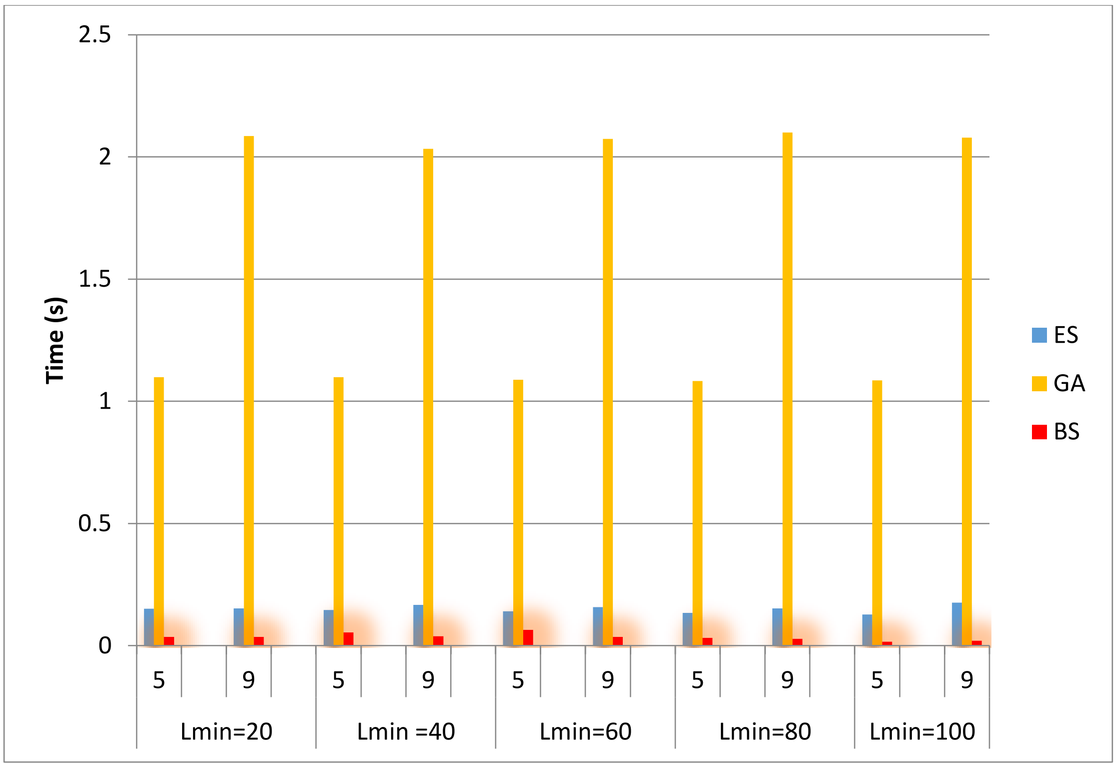


Figure 21: The execution time (s) of implementing exhaustive search (ES), genetic algorithm (GA), and binary search (BS) on a five-story building and a nine-story building

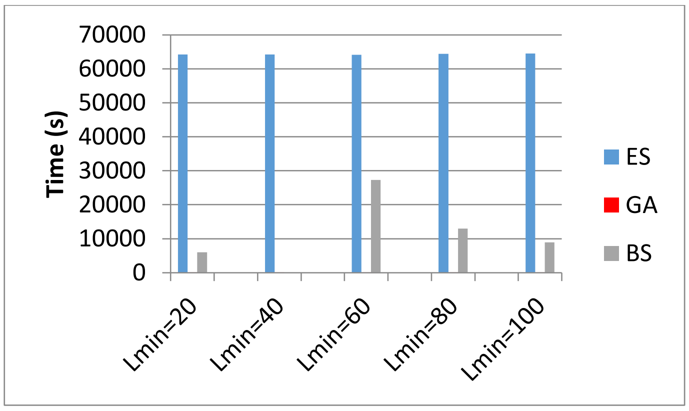


Figure 22:The execution time (s) of implementing exhaustive search (ES), genetic algorithm (GA,) and binary search (BS) on a two-bay nine-story building

Table 17: Results of implementing binary search on a two-bay nine- story building

L_{min}	**Optimized Placement**																														**Minimized # of sensors**	**time (s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**		
100	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	27	8956.969
80	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1	0	1	1	1	22	12982.67
60	0	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	1	1	17	27316.81
40	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	13	24158.92
20	0	0	0	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	9	6035.145

Table 18: Computed L for the combination that appeared when implementing binary search on a two-bay nine-story building

Optimized Placement																														L_{min}	**Computed L**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**		
0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	100	100
0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1	0	1	1	1	80	80.2158
0	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	1	1	60	60.0053
0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	40	40.2046
0	0	0	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	20	20.1232

5.3 Time Complexity Evaluation

To compute the time complexity of all the applied methods in this research work, Big O notation is used. Big O notation is used to classify methods by how they respond to changes in input size [63]. The input size used in all the applied methods is the number of candidate sensor locations. In genetic algorithm , the input size is affected by other things such as the number of generations, the population size, and the length of one solution.

5.3.1 Time complexity of exhaustive search

Time complexity of the method brute force is O($2^M.M^6$).The computation of the complexity according to the implemented algorithm is shown below.

1. The Euclidean distance is computed among all the sensors M used in the network. The worst case is in order of O(M^2). Euclidean distance is computed via a function called *pdist. Pdist* output is a vector of size (M+1(M)/2). After that, another function called *squareform* is used to convert the vector into a square symmetric format that takes the same complexity in order of O(M^2). So the overall complexity of this step is O(M^4).
2. Calculate to check whether the distance between each pair of sensors within transmission range R_c is in order of $O((M+1)^2)$. So one can say that to compute the distance constraint we need O(M^4) + $O((M+1)^2)$, which can be considered in order of O(M^4).

3. Evaluate the determinant of FIM ($|Q| = det[(\Phi)^T.R^{-1}.\Phi]$) for M sensors when all sensors M are selected. The complexity of computing the transpose of an input O(M^2) , the inverse of a matrix of size

M x M using Gaussian elimination, needs O(M^3). In addition, the determinant function is in order of O(M^3). In total the determinant in this case needs O(M^3). (O(M^3)+ O(M^2)) = O(M^6).

4. Compute the different possible combinations of M sensors that need $O\left(\sum_{j=1}^{M}\binom{M}{j}\right)$.
5. Go through all combinations of sensors needs $\sum_{j=1}^{M}\binom{M}{j}$ iterations and check the feasibility of the solution in each iteration. To check the feasibility of a combination,
   a. one needs to check the information quality constraint that needs the evaluation of determinant of FIM $|Q|$ ($|Q| = det[(\Phi)^T . R^{-1} . \Phi]$. This gives complexity in order of O(M^6).
   b. one needs to check the energy constraint by computing the total consumed energy for each sensor i in the combination, and this is in order of O(M^2). As a result, since $\sum_{j=1}^{M}\binom{M}{j} \ll 2^M$ then this step needs O($2^M . (M^6 + M^2$)) = O($2^M . M^6$).
6. Sort all the feasible solutions in ascending order according to their objective function evaluation in equation (3) and then in descending order according to their normalized |Q| (L) and finally get the optimal solution. In the worst case, all the M solutions are feasible, so sorting in that case has a time complexity O($M \log M$).

So the complexity of the brute force is O(M^4) $+$ O(M^6) + O($2^M . M^6$) that ends up with complexity of order O($2^M . M^6$).

5.3.2 Time complexity of genetic algorithm

In this part, the default setting was used to set the different parameters and algorithms of GA operators. The most important default parameters and algorithms are shown in Table 19.

Table 19: Some of the parameters input into ga function

Parameter/*ga* operator	**Value /Type of algorithm**
Population size (N_{ind})	20
Number of generations (g)	100
Creation function for population	Based on constraints
Selection	Stochastic Universal Sampling (SUS)
Crossover	Scattered (with fraction =0.8)
Mutation	Constraint dependent (Adaptive feasible)

In general, the time complexity of GA algorithm (without the evaluation of objective function and constraints complexity) is in order of O($N_{ind}.L_{ind}$), where N_{ind} is the population size, and L_{ind} is the length of an individual [66]. It is also known that the complexity of GA can be different from implementation to another, where the parameters and operator algorithms can be changed, and generally, it depends on the following:

1. Fitness function and constraints (application dependent)
2. Selection operator

3. Cross over operator
4. Mutation operator

The complexity of the implemented GA is calculated to be O(g. N_{ind}. L_{ind}. M^9).The following are the details behind the calculations:

1. Fitness function: The time complexity of the fitness function depends on the number of decision variables in the problem which are M decision variables. As a result, time complexity equals O(M).
2. Constraints :
   a. Distance: As discussed before in exhaustive search, the distance constraint has a time complexity in order of O(M^4).
   b. Energy: As discussed before in exhaustive search, the distance constraint has a time complexity in order of O(M^2).
   c. Quality: O(M^6) is computed in brute-force analysis.
3. Selection operator: According to the literature review, the time complexity of SUS algorithm is O(N_{ind})[67]. And since the fitness function is used as a parameter in the selection, the final time complexity is O(M . N_{ind}).
4. Crossover operator: According to the definition of scattered crossover in section (3.3.4 Crossover), the time complexity is computed as follows: In the worst case, $\frac{N_{ind}}{2}$ iterations are generated to guarantee that crossover is applied in all the parent pairs in the population. Each time a crossover between a pair of parents is done, a random binary vector of individual's length is generated, and L_{ind} comparisons are done to generate the new

parent. This means to finish the crossover operator, $\frac{N_{ind}}{2}.L_{ind}$ comparisons are done, which result in time complexity of order O($N_{ind}.L_{ind}$).

5. Mutation operator: As shown in Table 19 the mutation algorithm is adaptive feasible because the problem has constrains. Adaptive feasible means that the decision of applying mutation depends on the constraints of the problem , and in the worst-case, all the population individuals are mutated, so the time complexity for the mutation operator is in order O(($N_{ind}.L_{ind}.$ O(M^2) $+ O(M^2) + O(M^9)$.

Therefore, the time complexity for GA for the number of generations g is

O(g(M $.N_{ind} + N_{ind}.L_{ind} + N_{ind}.L_{ind}.\ M^9$) = O(g. $N_{ind}.L_{ind}.\ M^9$) = O ($M.\ M^9$) = O(M^{10}), where, in this research work, g=20, population size N_{ind} = 100, and individual size L_{ind} = M .

5.3.3 Time complexity of binary search

The steps used to compute the complexity of applying binary search algorithm are as follows:

1. Compute the complexity needed to evaluate the distance constraint which is the same computation done as in applying brute force. This needs complexity of order O(M^4).
2. Compute |Q| for all selected M sensors that take time complexity of O(M^6).
3. Then in the binary search start to look for the optimal solution., where midpoint is computed, and then search for a solution for midpoint sensors.

To do this, the problem constraints related to the information quality and energy should be checked, and to apply this for one single solution, the time complexity is in order $O(M^6)$.

4. In the worst case, there is no solution at midpoint sensors, which means that different $\binom{M}{midpoint}$ combinations are checked.
5. After that, depending on the result of looking for a solution using midpoint sensors, a series of three nested IF conditions need to be checked to know whether we need to look for a solution with a smaller number of sensors or a higher number of sensors, or just to determine that an optimal solution is found. In the worst case, all the conditions need to be checked.
6. The loop used to find the optimal solution can take $\log M + 1$ iterations in the worst case.

From the different steps above, one can see that the total time complexity in worst case needed to apply the binary search is in order O($(M^4) + (M^6) + (3.\sum_{j=1}^{logM+1} \binom{M}{midpoint}.(M^6)))$ which can be presented in the order of O($(logM + 1 \binom{M}{logM+1}).(M^6)$)where choosing logM+1 sensors out of M candidate sensors gives the highest number of combinations.

As you can see the time complexity is computed in the worst case. But of course, we can get better results in the average case and best case.

Chapter 6: Conclusion and Future Work

This chapter summarizes the research work, its related results and findings. Then the challenges faced while working on this research problem. Finally, the future research directions are reported.

6.1 Conclusion

This thesis discussed the problem of minimizing the number of sensors for SHM in WSN systems. This problem was never discussed in the literature. We presented a new mathematical formulation for the problem that addresses WSN requirements such as communication and energy consumption without ignoring the civil requirements such as the information quality computed using determinant of the FIM. Reducing the network size solves the problems of scalability, installation time, and cost. In addition to, solving the problem, from a theoretical point of view, shows the trade-off between the number of sensors and the information quality when the designer chooses to use all the candidate sensors to place. Based on the conclusion of this study, designers can know how much information quality will be reduced if they will use fewer nodes. In cases of node failure, the designer will know how much information quality will remain if certain nodes fail.

Three methods were implemented to solve the problem. The methods are exhaustive search, genetic algorithm, and binary search. The experiments applied the three methods on different configurations of sensors. The first configuration of sensors represents a building with 5 floors with 5 candidate sensor locations , the second configurations represents a 9 story building with 9 candidate sensor locations, and the last configuration represents a 2-bay 9 story building with 30 candidate sensor locations. We studied the performance of utilizing each algorithm on the different configurations for different required lower bound for the normalized determinant FIM (L_{min}). Additional experiments studied the relationship between the number of mode shapes versus the number of required sensors. Finally, the time complexity of the three applied methods was calculated to compare the three algorithms and to validate the results of the conducted experiments.

The obtained results showed that minimizing the number of sensors becomes more significant with big structures that require more sensors. Whenever the number of candidate sensors increased, the number of reduced sensors increased. Minimizing the number of sensors will result in a minimized network size, and lower cost and time for installation. Furthermore, we found that the binary search algorithm is best for small buildings because it gives the optimal solution with the best execution time. On the other hand, in larger buildings, there will be a trade-off between the performance (getting the optimal solution for the problem) and the execution time to get the results. If the designer wants to raise performance the binary search is best. Otherwise, the genetic algorithm is a better choice to find the near-optimal solution in a much less time. These results were confirmed with the evaluation of the time complexity for the three applied methods. Exhaustive search has the highest time complexity for large buildings in the order of O($2^M . M^6$). It is reduced in the other two algorithms. The time complexity for the genetic algorithm is O(M^{10}) , and in our own heuristic that applies the binary search, the worst-case time complexity is in the order of O($logM + 1 \binom{M}{logM+1}) . (M^6)$), where M is the number of candidate locations.

6.2 Challenges

Collecting mode shape information matrixes for large, tall structures was difficult. We tried to get the mode shape information matrix for the famous Guangzhou New TV Tower (GNTVT) [14], but the research team who worked on it did not reply. Additionally, the computation of the mode information matrix takes a long time and needs study that is beyond the scope of this research. As a result, our research team asked some researchers from the civil engineering field to compute the matrixes of a 9-story building and a 2-bays, 9-story building, while we got the mode shape information matrix of 5-story building from [65], so we can use them in this research work.

The second challenge was obtaining the optimal solution from applying the exhaustive search on a 2-bays 9-story building. The algorithm code needs to run for 18 hours. To solve this problem, I conducted the experiments with the different required L_{min} on different devices to quickly collect the results.

Understanding how a genetic algorithm works and measuring its time complexity was another challenge that was faced. In this research work, the method using MATLAB's GA solver from the optimization tool box was applied. The solver made implementing the algorithm very easy because it handles the details of the method. However, finding the time complexity was challenging as the details behind the running function needed to be understood to accurately compute the time complexity.

The last challenge I faced is related to finding the optimal solution. In addition to using the exhaustive search to find the optimal solution, we had planned to use the branch-and-bound (B&B) algorithm through the BARON solver [21] which guarantees the optimal solution with less time complexity in the average-case, and best-case. I modeled the problem for a 5-story building and passed it to the BARON solver. The results were promising. But when I modeled the problem for 9-story building, and for a 2-bay, 9-story building, the solver gave some results which differ from this research's results. A lot of time was spent on working by this solver and it was realized that it doesn't use a pure branch-and-bound algorithm; But there are some modifications to the algorithm that lead to a local optimal solutions, but not a global optimal solution when the number of candidate sensors increase. No more investigations were hold due to time limitation, and running the exhaustive search eventually gave the optimal results, which can be compared with results from the other algorithms. One of the future work directions is inspired by this challenge.

6.3 Future Work

This research work suggests many directions for future work, we selected some of them in the following list:

1. Conduct the experiments for larger sensor networks by collecting and utilizing the mode shapes information matrixes for higher buildings with more bays. The methods could also be applied to other type of structures, such as bridges or stadiums.
2. Measure the effect of varying the initial total energy for sensors, and changing the value of the transmission range.
3. Improve the problem formulation by adding more WSN constraints (e.g. add constraint related to routing).

4. Apply the B&B, greedy algorithm, or other algorithms and compare their performance with this research work applied algorithms used in this research.

References

[1] M. Segovia, E. Grampín, and J. Baliosian, “Analysis of the Applicability of Wireless Sensor Networks Attacks to Body Area Networks,” Proc. 8th Int. Conf. Body Area Networks, vol. 1, pp. 509–512, 2013.

[2] T. He, S. Krishnamurthy, J. a. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, “Energy-efficient surveillance system using wireless sensor networks,” pp. 270–283, 2004.

[3] X. Xie, J. Guo, H. Zhang, T. Jiang, R. Bie, and Y. Sun, “Neural-network based structural health monitoring with wireless sensor networks,” Nat. Comput. (ICNC), 2013 Ninth Int. Conf., no. 61171014, pp. 163–167, 2013.

[4] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, “A wireless sensor network For structural monitoring,” Proc. 2nd Int. Conf. Embed. networked Sens. Syst. - SenSys ’04, p. 13, 2004.

[5] S. Wijetunge, U. Gunawardana, and R. Liyanapathirana, “Wireless Sensor Networks for Structural Health Monitoring: Considerations for communication protocol design,” 2010 17th Int. Conf. Telecommun., pp. 694–699, 2010.

[6] S. Beskhyroun, L. D. Wegner, and B. F. Sparling, “New methodology for the application of vibration-based damage detection techniques,” Struct. Control Heal. Monit., no. May, 2011.

[7] F. P. Pentaris, J. Stonham, and J. P. Makris, "A review of the state-of-the-art of wireless SHM systems and an experimental set-up towards an improved design," IEEE EuroCon 2013, no. July, pp. 275–282, 2013.

[8] G. Movva, Y. Wan, S. Fu, and H. F. Wu, "Optimal sensor placement for structural health monitoring: a comparative study between the control engineering and civil engineering approaches," vol. 8694, p. 86941K, 2013.

[9] F. Oldewurtel and P. Mahonen, "Analysis of Enhanced Deployment Models for Sensor Networks," Veh. Technol. Conf. (VTC 2010-Spring), 2010 IEEE 71st, pp. 1–5, 2010.

[10] M. Romoozi, M. Vahidipour, M. Romoozi, and S. Maghsoodi, "Genetic Algorithm for Energy Efficient and Coverage-Preserved Positioning in Wireless Sensor Networks," 2010 Int. Conf. Intell. Comput. Cogn. Informatics, pp. 22–25, 2010.

[11] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," 2003 IEEE Wirel. Commun. Networking, 2003. WCNC 2003., vol. 3, no. C, pp. 1609–1614, 2003.

[12] Y. Chen, C.-N. Chuah, and Q. Zhao, "Sensor Placement for Maximizing Lifetime per Unit Cost in Wireless Sensor Networks," MILCOM 2005 - 2005 IEEE Mil. Commun. Conf., pp. 1–6, 2005.

[13] S. Sengupta, S. Das, M. D. Nasir, and B. K. Panigrahi, "Multi-objective node deployment in WSNs: In search of an optimal trade-off among coverage,

lifetime, energy consumption, and connectivity," Eng. Appl. Artif. Intell., vol. 26, no. 1, pp. 405–416, 2013.

[14] B. Li, D. Wang, F. Wang, and Y. Q. Ni, "High quality sensor placement for SHM systems: Refocusing on application demands," Proc. - IEEE INFOCOM, pp. 1–9, 2010.

[15] B. Li, "Demo : On the High Quality Sensor Placement for Structural Health Monitoring," pp. 2–3.

[16] A. A. Elsersy, Mohamed, Hossam, Mohamed, Elfouly, Tarek, "Multi-Objective Sensor Placement using the Effective Independence Model (SPEM) for Wireless Sensor Networks in Structural Health Monitoring," pp. 576–580, 2015.

[17] Z. A. Bhuiyan, J. Cao, and G. Wang, "Deploying Wireless Sensor Networks with Fault Tolerance for Structural Health Monitoring," vol. 64, no. 2, pp. 382–395, 2012.

[18] T.-H. Yi, H.-N. Li, and M. Gu, "Optimal Sensor Placement for Health Monitoring of High-Rise Structure Based on Genetic Algorithm," Math. Probl. Eng., vol. 2011, pp. 1–12, 2011.

[19] G. Fan, S. Jin, and D. Processing, "Coverage Problem in Wireless Sensor Network : A Survey," J. Networks, vol. 5, no. 9, pp. 1033–1040, 2010.

[20] K. Neapolitan, Richar, Naimipour, Foundations of algorithms, Fourth edi. Jones & Bartlett Learning, 2011.

[21] “BARON Software | Sahinidis.” [Online]. Available: http://archimedes.cheme.cmu.edu/?q=baron. [Accessed: 27-Mar-2016].

[22] D. D. S. Hoon Sohn, Charles R. Farrar, Francois M. Hemez and and J. J. C. Daniel W. Stinemates, Brett R. Nadler, “A Review of Structural Health Monitoring Literature : 1996 – 2001,” Los Alamos National Laboratory Report, 2004.

[23] C. R. Farrar and K. Worden, “An introduction to structural health monitoring.,” Philos. Trans. R. Soc., vol. 365, no. 1851, pp. 303–15, 2007.

[24] K. Worden, C. R. Farrar, G. Manson, and G. Park, “The fundamental axioms of structural health monitoring,” Proc. R. Soc. A Math. Phys. Eng. Sci., vol. 463, no. 2082, pp. 1639–1664, 2007.

[25] C. R. Farrar and K. Worden, Structural Health Monitoring: A Machine Learning Perspective. UK: John Wiley & Sons, Ltd, 2013.

[26] M. Z. A. Bhuiyan, G. Wang, and J. Cao, “Sensor Placement with Multiple Objectives for Structural Health Monitoring in WSNs,” 2012 IEEE 14th Int. Conf. High Perform. Comput. Commun. 2012 IEEE 9th Int. Conf. Embed. Softw. Syst., pp. 699–706, 2012.

[27] M. Meo and G. Zumpano, “On the optimal sensor placement techniques for a bridge structure,” Eng. Struct., vol. 27, no. 10, pp. 1488–1497, 2005.

[28] G. W. van der Linden, A. Emami-Naeini, R. L. Kosut, H. Sedarat, and J. P. Lynch, “Optimal sensor placement for health monitoring of civil structures,” Am. Control Conf. (ACC), 2011, pp. 3116–3121, 2011.

[29] P. G. Bakir, “Evaluation of Optimal Sensor Placement Techniques for Parameter Identification in Buidlings,” Math. Comput. Appl., vol. 16, no. 2, pp. 456–466, 2011.

[30] J. a. Camelio, S. J. Hu, and H. Yim, “Sensor Placement for Effective Diagnosis of Multiple Faults in Fixturing of Compliant Parts,” J. Manuf. Sci. Eng., vol. 127, no. 1, p. 68, 2005.

[31] S.-F. Lung and C.-G. Pak, “Updating the Finite Element Model of the Aerostructures Test Wing Using Ground Vibration Test Data,” 50th AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf., 2009.

[32] A. Joorabchian and A. A. Golafshani, “Study of pattern of sensor installation for analysis behavior of offshore jacket structure,” pp. 1677–1686.

[33] M. Najeeb and V. Gupta, “Energy Efficient Sensor Placement for Monitoring Structural Health,” Int. Electron. Conf. Sensors Appl., pp. 1–6, 2014.

[34] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Sensor Placement with Multiple Objectives for Structural Health Monitoring," ACM Trans. Sens. Networks, vol. 10, no. 4, pp. 1–45, 2014.

[35] T. Carne and C. Dohrmann, "A modal test design strategy for model correlation," Proceedings-Spie …, no. NOVEMBER 1994, pp. 927–933, 1995.

[36] D. Li, C. Fritzen, H. Li, C. Engineering, L. Rd, P. Str, and D.- Siegen, "Extended MinMAC algorithm and comparison of sensor placement methods."

[37] L. CanXing, Z. Ping, and D. Jing, "A Review on Optimal Sensor Placement for Health Monitoring," 2007 8th Int. Conf. Electron. Meas. Instruments, pp. 4–170–4–173, 2007.

[38] L. Yao, W. a. Sethares, and D. C. Kammer, "Sensor placement for on-orbit modal identification via a genetic algorithm," AIAA Journal, vol. 31. pp. 1922–1928, 1993.

[39] K. Worden and a. P. Burrows, "Optimal sensor placement for fault detection," Eng. Struct., vol. 23, no. 8, pp. 885–901, 2001.

[40] H. Y. Guo, L. Zhang, L. L. Zhang, and J. X. Zhou, "Optimal placement of sensors for structural health monitoring using improved genetic algorithms," Smart Mater. Struct., vol. 13, no. 3, pp. 528–534, 2004.

[41] C. He, J. Xing, J. Li, Q. Yang, R. Wang, and X. Zhang, "A Combined Optimal Sensor Placement Strategy for the Structural Health Monitoring of Bridge Structures," Int. J. Distrib. Sens. Networks, vol. 2013, pp. 1–9, 2013.

[42] M. Elsersy, "Joint Optimal Placement and Routing for Wireless Sensor Networks in Structural Health Monitoring."

[43] "Eight Queens Problem." [Online]. Available: http://www.datagenetics.com/blog/august42012/. [Accessed: 08-Mar-2016].

[44] "Data Structures and Algorithms with Object-Oriented Design Patterns in C++." .

[45] M. Melanie, An Introduction to Genetic Algorithms. MIT press, 1998.

[46] R. Malhotra, N. Singh, and Y. Singh, "Genetic Algorithms : Concepts , Design for Optimization of Process Controllers," vol. 4, no. 2, pp. 39–54, 2011.

[47] P. G. Aggarwal, Shaifali, Richa Garg, "A Review Paper on Different Encoding Schemes used in Genetic Algorithms," Adv. Res. Comput. Sci. Softw. Eng., vol. 4, no. 1, pp. 596–600, 2014.

[48] M. Srinivas and L. M. Patnaik, "Genetic Algorithms : A Survey," Computer (Long. Beach. Calif)., vol. 27, no. 6, pp. 17–26, 1994.

[49] S. Jin, M. Zhou, and A. S. Wu, "Sensor Network Optimization Using a Genetic Algorithm," 7th World Multiconference Syst. Cybern. Informatics., pp. 109–116, 2003.

[50] “Selection.” [Online]. Available: https://courses.cs.washington.edu/courses/cse473/06sp/GeneticAlgDemo/selection.html. [Accessed: 28-Jan-2016].

[51] “Genetic Algorithm Options - MATLAB & Simulink.” [Online]. Available: http://www.mathworks.com/help/gads/genetic-algorithm-options.html#f6633. [Accessed: 01-Feb-2016].

[52] “Genetic algorithms.” [Online]. Available: http://www.slideshare.net/unalozgur/genetic-algorithms-16439115. [Accessed: 08-Mar-2016].

[53] “Genetic Algorithm - MATLAB & Simulink.” [Online]. Available: http://www.mathworks.com/help/gads/genetic-algorithm.html. [Accessed: 01-Feb-2016].

[54] “What Is the Genetic Algorithm? - MATLAB & Simulink.” [Online]. Available: http://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html. [Accessed: 01-Feb-2016].

[55] “Research.” [Online]. Available: http://bees2006.awardspace.com/Research.htm. [Accessed: 01-Feb-2016].

[56] “JGAP: Java Genetic Algorithms Package.” [Online]. Available: http://jgap.sourceforge.net/#documentation. [Accessed: 02-Feb-2016].

[57] "ECJ." [Online]. Available: http://cs.gmu.edu/~eclab/projects/ecj/. [Accessed: 02-Feb-2016].

[58] "Google Code Archive - Long-term storage for Google Code Project Hosting." [Online]. Available: https://code.google.com/archive/p/beagle/. [Accessed: 02-Feb-2016].

[59] "Function optimization with Genetic Algorithm by using GPdotNET | Bahrudin Hrnjica Blog on WordPress.com." [Online]. Available: http://bhrnjica.net/2013/03/26/function-optimization-with-genetic-algorithm-by-using-gpdotnet/. [Accessed: 02-Feb-2016].

[60] "Find minimum of function using genetic algorithm - MATLAB ga." [Online]. Available: http://www.mathworks.com/help/gads/ga.html?refresh=true. [Accessed: 12-Mar-2016].

[61] "Find minima of multiple functions using genetic algorithm - MATLAB gamultiobj." [Online]. Available: http://www.mathworks.com/help/gads/gamultiobj.html. [Accessed: 13-Mar-2016].

[62] "Binary search." [Online]. Available: https://www.math.ust.hk/~mamu/courses/231/Slides/ch02_1.pdf. [Accessed: 08-Mar-2016].

[63] M. H. Alsuwaiyel, Algorithms Design Techniques and Analysis, 13th ed. World Scientific, 1999.

[64] “ThinkPad T440 Platform Specifications.” [Online]. Available: http://www.lenovo.com/psref/pdf/tabook_WE.pdf. [Accessed: 28-Mar-2016].

[65] Anil K. Chopra, Dynamics of Structures: International Edition. Pearson, 2015.

[66] C. Chipperfield, A., Fleming, P., Pohlheim, H., & Fonseca, “Genetic algorithm toolbox for use with MATLAB,” 1994.

[67] Belew, Richard K. Foundations of Genetic Algorithms-FOGA 4: Proceedings. Eds. Richard K. Belew, and Michael D. Vose. Morgan Kaufmann, 1997, 1997.

Appendix A : Binary Search Algorithm

In this appendix, the general algorithm of the binary search method [63] is presented:

Input: An array A[1..n] of sorted n elements in non-decreasing order and element x.

Output: The index of x j where x = A[j] where $1 \le j \le n$. Otherwise it will be zero.

Algorithm:

```
1: low =1 , high =n, j=0
2: while ( low ≤ high), and (j=0)
3: mid = (low + high )/2
4: if x=A[mid]m, then j = mid, and stop.
5: else if x <A[mid] m, then high = mid-1.
6: else low= mid+1
7: End while
8 : Return j
```

Appendix B: How GA Process Works

Algorithmically, the general genetic GA is applied as below [46]:

1. Start by generating a random population of solution “chromosomes.”
2. Evaluate the fitness function of each chromosome in the population.
3. Create a new population using the following steps, and repeat them until the new population is complete, and the solution is found.
   a. Select two parent chromosomes from a population based on their fitness where if the fitness is better, then there is a bigger chance to be selected as parent.
   b. Cross over the parents to form a new offspring, that is, children. If no crossover was performed, the offspring is the exact copy of parents.
   c. Mutate new offspring at some locations in the chromosome.
   d. Place new offspring in the new population.
4. Use the generated population for a additional run of the algorithm.
5. Check the end condition. If it is satisfied, stop and return the optimal solution in the current population.
6. Otherwise, loop and go to step 2.

An example of a problem that can be solved using genetic algorithm is the eight- queens problem mentioned in section 3.2 Exhaustive Search section. Here are steps that can be used to solve it using GA:

1. Start by generating a population 'P' of strings with '8' row positions, the row position generated randomly for each column, representing a configuration of queens on the board. For example, '6 3 1 7 4 8 5 2'

is a string of size '8' belonging to population 'P'. Create 'P' such strings.

2. As a result, an initial population Pi is ready to be selected.
3. Evaluate the fitness value for each solution, then choose randomly some strings to be in next generations depending on their scores in the fitness function.
4. Apply crossover to some chosen strings and generate one new string S. For example:

String 1: '6 3 1 7 4| 8 5 2'

String 2: ' 1 4 3 25 |7 6 8'

New string S based on crossover: '6 3 1 7 4 7 6 8'.

5. With a small probability, apply mutation to string S. Otherwise leave it as it is.

6. Apply steps 2 to 5 until a solution string (string with maximum fitness value) representing a correct solution, for example '6 3 1 8 5 2 9 7 4'.