Republic of Iraq

Ministry of Higher Education & Scientific Research

University of Babylon

College of Education for Pure Sciences

Department of Mathematics


# *Improvement the Reliability of Shutdown System by Using Hybrid Meta-Heuristic Algorithms*

A Dissertation

Submitted to the Council of College of Education for Pure Sciences,

University of Babylon in Partial Fulfillment of the Requirements for

the Degree of Doctor of Philosophy in Education / Mathematics.

By

**Roaa Aziz Fadhil Mohsin**

Supervised by

**Prof.Dr. Zahir Abdul Haddi Hassan**

**2023 A.D.** **1445 A.H.**

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

وَيَرَى الَّذِينَ أُوتُوا الْعِلْمَ الَّذِي أُنزِلَ
إِلَيْكَ مِن رَّبِّكَ هُوَ الْحَقَّ وَيَهْدِي إِلَىٰ
صِرَاطِ الْعَزِيزِ الْحَمِيدِ (6)

صدق اللَّهُ العظيم

سورة سبأ

# *Dedication*

To those who have lived in hearts .... And who dwelt in heaven

For every national martyr .... Previous and subsequent ones

And to the soul of my beloved father .... May he finally see the light

Mohammed ... The God of the Immaculate

May Allah's prayers and peace be upon them all

I raise this modest product ...

Roaa A.

# *Acknowledgements*

In the name of Allah, the Most Gracious, the Most Merciful First and above all, Praise be to Allah who gave me the health and patience to conduct this work. And then, I would like to thank prophet Muhammad and Ahl Al-Bayt (peace be upon them). I am especially grateful for the help provided by my wonderful and respected supervisor, Dr.Zahir Abdul Haddi Hassan, to encourage, guide and support me during this work. Deep appreciation and love to all my family and everyone who gave me a hand of support.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

Table 1: Symbols and Acronyms

| **Symbol** | **Description** |
|---|---|
| Pr | Probability |
| *pdf* | Probability density function |
| $R(t)$ | Reliability function |
| $F(t)$ | Failure function |
| *cdf* | Cumulative distribution function |
| $H(t)$ | Hazard rate |
| $G$ | Graph |
| $x_i$ | $i-th$ edge (component) |
| $v_i$ | $i-th$ vertex |
| $C$ | Connection matrix |
| $A$ | Adjacency matrix |
| $I$ | Incidence matrix |
| $N(v)$ | Neighborhood of a vertex |
| TPR | Two Terminal or Terminal Pair Reliability |
| $s, t$ | Terminal pair nodes (source and sink) |
| $RBD$ | Reliability block diagram |
| $\phi(x)$ | Structure function |
| $E(X)$ | Mathematical Expectation of X |
| PN | Petri net |
| RPS | Reactor Protection System |
| MP | Minimal path |
| MC | Minimal cut |
| MCV | Minimal cut vertices |
| $f(x)$ | The Objective function |
| $Minf(x)$ | The Minimum of Objective Function |
| $Maxf(x)$ | The Maximum of Objective Function |
| $\coprod_{i=1}^{n}(x_i)$ | Parallel structure |
| $\prod_{i=1}^{n}(x_i)$ | Series structure |

| | |
|---|---|
| $g(x)$ | The Inequality Constraint |
| $h(x)$ | The Equality Constraint |
| $LP$ | Linear Programming or (Problem) |
| $NLP$ | Non-Linear Programming (problem) |
| $du$, $df$ and $y^{'}$ | First Order Derivatives |
| $d^2u$, $d^2f$ and $y^{''}$ | Second Order Derivative |
| $\partial u$ | Partial Differential |
| $\nabla f$ | Gradient the function |
| $\nabla^2 f$ | Hessian the function |
| NM | Nelder-Mead Simplex Method |
| $PF(x)$ | Penalty Function |
| $\rho$ | Reflection factor |
| $\mu$ | Expansion factor |
| $\sigma$ | Contraction factor |
| $Avg$ | Average |
| $Std$ | Standard deviations |
| $BQS$ | The candidate solution |
| $I$ | Definining intensity |
| $\alpha$ | Density factor |
| $\beta$ | Ability of the honey badger to get food |
| $G_\alpha$ | Alpha group |
| $peep$ | the alpha female's vocalization |
| $sm$ | The sleeping mound |
| $\vec{M}$ | The vector movement of the mongoose |
| $\omega$ | The weight vector |
| $\kappa$,$\vartheta$ | The penalty factor |
| P-HBA | Penalty method with HBA |
| P-HBNMA | Penalty method with HBNMA |
| P-DMOA | Penalty method with DMOA |
| P-DMONMA | Penalty method with DMONMA |

# Abstract

This Dissertation presents new techniques that aim to increase the reliability of the shutdown system located inside a nuclear reactor using hybrid algorithms consisting of heuristic and meta-heuristic algorithms. The system was created by converting the Petri net of the operating shutdown system into a complex network.

To find the minimum path sets, we introduced five techniques: Rai and Aggarwal's algorithm, multiplication of matrices, path set enumeration, child node matrix algorithm and two end node algorithm. Three techniques were also used: nodes and edges algorithm, Shall algorithm, and a new technique called Roaa's technique to find the minimum cut sets of. To calculate the fire network polynomial, we will use sum of disjoint products and the minimum cut techniques.

To study improving and increasing the reliability of the shutdown system, we first work on creating two mathematical models for optimization were obtained by combining meta-heuristic algorithms with the Nelder-Mead method, where the honey badger algorithm (HBA) was hybridized with the Nelder-Mead method (NM), where the results of the hybridization were the algorithm Hybrid HBNMA, and by hybridizing the Dwarf Mongoose algorithm (DMOA) with the Nelder-Mead method, we obtained the hybrid algorithm DMONMA.

To verify the robustness and effectiveness of the hybrid algorithms, twenty-three test functions were used, and the results of these functions were verified using statistical evaluation of the mean, standard deviation, and execution time of the algorithms. We found that the hybrid algorithms improved most of the functions compared to the original algorithms. The next step to improve and increase the reliability of the given system will be by hybridizing the Penalty method with the P-HBA, P-HBNMA,

P-DMOA, and P-DMONMA algorithms, and we will provide a comparison between the results of the algorithms that use hybridization technique with the results of the algorithms that do not use this technique.

To obtain Petri solutions for solving nonlinear multi-objective optimization problems for the shutdown network, the weight sum method was used to transform a multi-objective function problem into a single objective function, while studying three mathematical models of to design a mathematical model for multi-objective optimization to increase reliability and reduce cost. The hybrid algorithms result for P-HBA, P-HBNMA, P-DMOA and P-DMONMA, showed that it is possible to find the best value for reliability and cost lowest compared to other algorithms. We conclude that the combination of HBA and DMOA with the Penalty method and the Nelder-Mead method gives the best solutions in addition to reducing the execution time compared to the execution time of other algorithms.

# CHAPTER 1

# INTRODUCTION

## 1.1. General Introduction

The history of the reliability field may be traced back to the early 1930 , problems with the production of electric power were resolved using probability ideas. In order to improve the dependability of their V1 and V2 rockets, the Germans are reported to have adopted the reliability concept for the first time during World War II. The American department of defense formed an *ad hoc* reliability committee in 1950. It was established permanently as the advisory committee on the reliability of electronic equipment in 1952 [32], almost all of us rely on a variety of technical goods and services in our daily lives. The results of a failed product, piece of equipment, or service can occasionally be disastrous. Customer discontent and expenses for the provider due to warranty fees and product recalls are more frequently caused by product faults and service interruptions. Reliability has become crucial to many suppliers' existence. It takes time to develop a reputation for reliability but it may take a short time to lose this reputation. The main drivers for high reliability are listed in **Figure** (1.1) [67].


Figure 1.1: Main drivers for high reliability.

The accepted definition of "*reliability*" in the engineering community is a system's ability to operate as intended, without malfunction, in an operational environment, for a defined period of time, a more comprehensive definition to reliability is as the science to forecast, evaluate, avoid and mitigate failures across time [38]. A network is any system

which can be thought of and graphically represented as a collection of small circles (nodes) interconnected by lines (edges). Networks is a vast field which deals with a wide spectrum of real-life systems in industry, communication, software engineering,... etc [18]. Network design with system reliability objectives or restrictions has received a lot of attention. These designs are applicable to computer networking, telecommunications, and related industries like gas and electricity. The overall network reliability, often referred to as all-terminal network reliability is a measure of the likelihood that each node in the network is connected to each other node, the more common source-and-sink (s-t) network, where dependability depends on how likely it is that the source and sink are connected. When components with known specifications are available, it is an NP-hard (NP) combination challenge to design networks while taking into account all-terminal dependability and additional goals or limits like cost or distance. The size of the network and the number of available components make the search space more difficult because it takes a lot of processing resources to establish the reliability of all-terminal networks. It is quite difficult to optimize network architecture for all-terminal reliability when these two criteria are combined, especially for small networks [42]. Optimization algorithms are essential tools in the design processes used in engineering and science. Numerous disciplines, such as engineering, physics, economics, sociology, and many others that deal with decision-making are using these extensively and are incorporating them into their core practices. An easy test could reveal the value of optimization, the tasks or problems include optimizing (either minimization or maximization) of an objective. The basic steps in the optimization procedure for each problem are shown in **Figure** (1.2). The problem is viewed as a physical problem requiring inputs or resources, which are essential bits of knowledge. These inputs are used to mimic the physical problem. The constraints and objective of the physical situation are used to formulate the problem. The output, which results from the application of optimization techniques to the replies is next examined. From the output, the best collection of solutions is (are) ultimately discovered. The output is

finally used to determine the optimal set of solutions. Here, that claimed optimization is the action of attaining the best result attainable in a specific circumstance. To put it another way, optimization is a technique that helps us to maximize the utilization of the resources at our disposal [112].

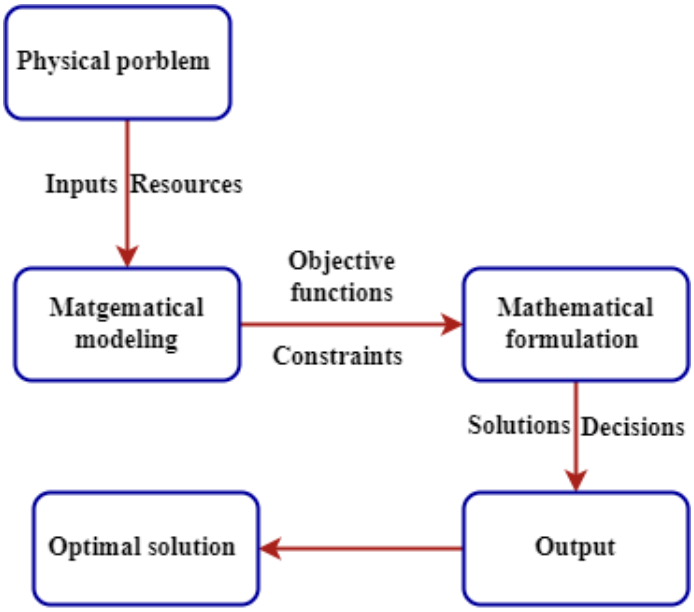



Figure 1.2: Flowchart of modeling physical the problem to get an optimal solution.

The classification of optimization, particularly in relation to the use of certain terms. Here, Only employees the terminology that are frequently used. However, sought not to classify classify things precisely, instead, All relevant concepts will be presented, quickly and effectively. According to a general definition, categorization can be done in terms of the quantity of objectives, restrictions, function forms, landscape of the objective functions, kind of design variables, uncertainty in values, and computing effort see **Figure** (1.3). If attempted to categories optimization issues based on the quantity of targets, there are two categories both single-objective and multi-objective approaches. Multi-objective optimization is also known in the literature as multi-criteria optimization or even multi-attributes optimization. Multi-objective optimization issues are the norm in real-world situations. For instance, while constructing an automobile engine, we want to improve fuel efficiency and reduce carbon dioxide emissions. Similarly, Optimization classes are either

a problem with constraints or one without any constraints. Problems with constraints on equality turn into problems with constraints on inequality [147].

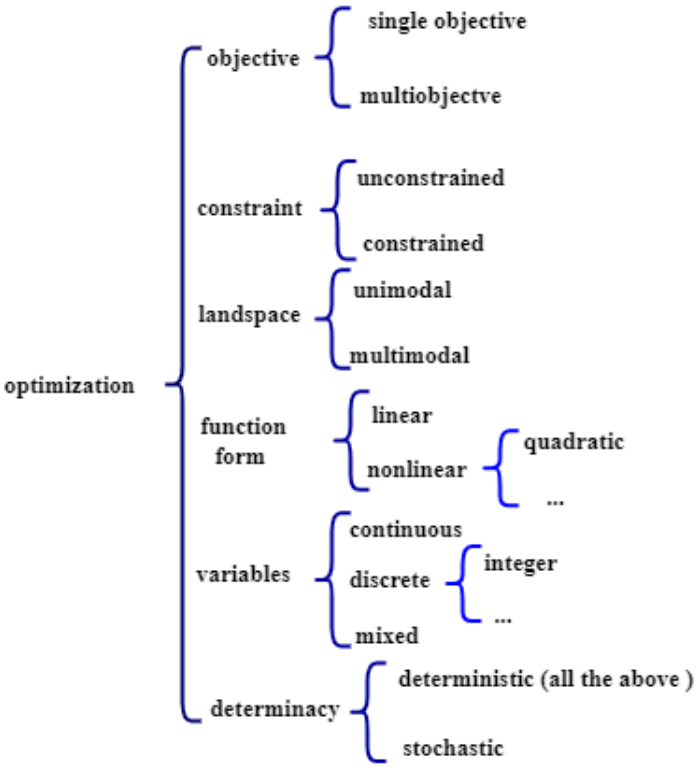



Figure 1.3: Classification of optimization problems.

The using of heuristics in optimization theory is a relatively recent concept. Although there are some early examples from the 1950 or so, the massive increase in computing power only recently has made these methods practically applicable. The core idea of heuristics can be summed up as seeking approximate solutions to precise issues. Heuristics try to provide accurate and quick approximations to ideal solutions. It has been demonstrated that heuristics are effective for issues that classical methods are entirely unable to solve. Heuristics are methods for learning and solving problems that are based on experience. It may not be the best solution, but it provides a good one in a reasonable length of time. The design of certain heuristics is problem-specific and limited, with the purpose of resolving a specific issue. Use of a rule of thumb, an educated estimate, an instinctual judgement, or even common sense are examples of this

approach. Heuristics are used in many algorithms, whether they be exact or approximation algorithms. We define the term "optimization heuristic" using the following criteria:

- The approach should provide a "good" stochastic approximation of the genuine optimum, "goodness" can be gauged by the amount of time needed to compute and the accuracy of the result.
- The approach should be resilient to changes in the goal function, constraints, and scale of the provided challenge. Additionally, outcomes shouldn't be overly variable.
- The technique should be simple to apply and not need the use of subjective judgments, either during application or while modifying the heuristic's parameter settings.

Further development of heuristic algorithms is the so-called meta-heuristics algorithms [49] [55]. Both of the phrases meta and heuristic have their roots in ancient Greek, meta means "upper level", and heuristic refers to the practice of developing novel tactics. It is important to note that there are no consensus definitions of heuristics and meta-heuristics in the literature. Heuristics and meta-heuristics are sometimes used interchangeably. On the other hand, a current tendency is to name all stochastic algorithms, such as simulated annealing (SA) and genetic algorithms (GAs) ,... etc.
Utilizing local search and randomization as meta-heuristics. To transition from local to global search, randomization offers a useful solution. In light of this, practically all meta-heuristics algorithms aim to be suitable for global optimization [84] difference is that meta-heuristics are intended to extend the capabilities of heuristics by combining one or more heuristic methods, *properties of meta-heuristics*, as follows [23] [31]:

- meta-heuristics are methods for directing the search process.
- On the objective function's mathematical features, such as continuity and derivability, they offer no hypotheses. The only prerequisite is that it be possible

to compute $f(x)$ for every $x$.

- The goal is to effectively explore the search space for (near) optimal solutions.
- meta-heuristics algorithms use a variety of techniques, from straightforward local search techniques to intricate learning procedures.
- To direct the exploration, they make use of a few criteria. The quality of the solutions found and the rate of convergence depend on the values of these factors. The best values for the parameters, however, are typically unknown and are either chosen empirically, based on prior knowledge, or through a learning process.
- It is necessary to designate a search's starting place. The initial solution is frequently, but not always, picked at random.
- The search must also incorporate a stopping condition. This is typically dependent on CPU usage, the quantity of assessments, or when the fitness has plateaued after a certain number of repeats.
- meta-heuristics algorithms are typically non-deterministic and approximate.
- They may have features that prevent them from becoming caught in small spaces in the search area.
- meta-heuristics' fundamental ideas enable an abstract level description.
- meta-heuristics do not have a problem-specific focus.
- meta-heuristics may use heuristics that are governed by the higher level strategy to leverage domain-specific information.
- In most cases, they can be efficiently parallelized and are simple to implement.

In the realm of optimization, interest in hybrid meta-heuristics has increased significantly. For many real-world or traditional optimization issues, hybrid algorithms

currently available provide the best results, trend is the development of hybrid methods, which aims to capitalise on the unique benefits of several methodologies by fusing them, different kinds of combinations are [136]

- Using many (complementary) meta-heuristics in combination.
- Using precise approaches from mathematical programming in conjunction with meta-heuristics methods primarily employed in operations research.
- By combining constraint programming techniques developed in the artificial intelligence community with meta-heuristics.
- Using meta-heuristics in conjunction with data mining and machine learning methods.

## 1.2. Objectives of the Dissertation

The goal of the dissertation is to find new mathematical methods for increasing shutdown network reliability, and to achieve this goal we will present two mathematical optimization models that merged. Namely, hybrid algorithms are HBNMA and DMONMA from combining a meta-heuristics algorithm honey badger algorithm (HBA) with the Nelder-Mead technique and Dwarf Mongoose Optimization algorithm (DMOA) with the Nelder Mead technique respectively. The Penalty method was applied for hybrid algorithms and the original algorithms are P-HBA, P-HBNMA, P-DMOA and P-DMONMA and compare it with algorithms are HBA, HBNMA, DMOA and DMONMA to solve multi-objectives function problem to improve the network reliability. The results calculated by using MATLAB R2018b using PC an Intel Core i7, 1.8 GHz CPU and 16 GB of RAM.

## 1.3. Contributions

The dissertation included a set of contributions that can be summarized as follows:

1. We suggested a two new a hybrid algorithms are HBNMA and the DMONMA.
2. Utilizing statistical analysis with the average and standard deviation, the effectiveness was examined the hybrid algorithms.
3. The shutdown network has been created by converting Petri net.
4. We extracted all minimal paths and cut sets of the shutdown network.
5. We found the reliability of this network by using two methods and we suggest an approach that depends on the minimal path vertices.
6. We have used meta-heuristics and hybrid meta-heuristics algorithms to solve the problem of multiple objectives of the closure network to obtain the best increase in shutdown network reliability at the lowest possible cost.
7. Utilizing sum of weight methods to solve multi-objective function problem.
8. Using the penalty method with the HBA, HBNMA ,DMOA, and DMONMA to solve the problem of constraint.

## 1.4. Dissertation Outlines

This dissertation consists of six chapters.

1. The first chapter contains the introduction, objectives, contributions of dissertation and related works.
2. The second chapter contains some definitions and basic concepts.

3. Third chapter includes techniques for calculating the reliability networks. We extracted all minimal path sets by Rai, Aggarwal's algorithm, matrix multiplication, path set enumeration, node-child matrix algorithm, and two terminal nodes algorithm , found minimal cut sets using the nodes and edges algorithm, Shal algorithm and Roaa techniques of a complex network that has been obtained from the transform Petri net of the shutdown system. Finding a polynomial function of reliability by twot techniques were Sum of Disjoint Products techniques and Minimal Cut techniques.

4. The fourth chapter discusses hybrid meta-heuristics algorithms, using the honey badger algorithm and dwarf mongoose optimization algorithm. And suggested hybrid Meta heuristics consisting of the honey badger algorithm with Nelder-Mead method, and dwarf mongoose optimization with Nelder-Mead method. The performance of the proposed algorithms was verified using statistical results and implemented on a set of test functions.

5. Chapter five discusses optimization reliability shutdown network by using are the HBA, HBNMA, DMOA, and DMONMA, once with penalty method are P-HBA, P-HBNMA, P-DMOA and P-DMONMA, and once without penalty method for solve multi-objective optimization problem and the importance of network components was determined using the values produced by the algorithms.

6. The Chapter six consists of conclusions and future works.

## 1.5. Related Works

### 1.5.1 Network Reliability and Reliability Optimization

All-terminal network reliability, known as *uniform* or *overall network reliability*, is the likelihood that every pair of nodes can interact with one another. The main design

challenge is deciding which set of links to use for a given collection of nodes in order to either maximize dependability or minimize cost. Design of reliable networks is an NP-hard design problem [43]. Calculating a network's reliability is an NP-hard task. The evaluation of network reliability [14, 91] are using a range of tools and techniques. Because networks are based on graph theory, of approaches used to assess their dependability use graph-based algorithms that use either minimal paths (MP) or minimal cuts (MC) [66, 123]. The development of reliability allocation methods has advanced significantly over the past few decades, considering the knowledge currently available regarding systems and subsystems. In 1956 a common allocation technique was established by the advisory group on reliability of electronic equipment (AGREE). This strategy gives the complexity and criticality of units and subsystems a greater weight than failure rates. The AGREE technique has the benefit of being extremely thorough, but it can only be used for systems in series and in a later stage of the design process [32]. Aeronautical Radio Inc, in contrast to AGREE, presented the ARINC approach, which prioritizes unit or subsystem failure rates. In 1975 methods for mathematical programming have been employed, such as the implicit enumeration technique and the integer programming [48]. In 1979 the Generalized Lagrangian Function (GLF) technique was used by Hwang et [70] In 1983 any nonlinear reliability optimization problem may be solved using the Generalized Reduced Gradient (GRG) method. Two reliability issues are resolved using the sequential unconstrained minimization technique (SUMT) and the Luus and Jaakola (LJ) method, and the outcomes are compared [124]. The first problem maximizes system reliability while minimizing system cost, while the second problem minimizes system cost while minimizing system reliability [124]. Performance indicators, optimization methods, and reliability-enhancing choices. A general strategy for distributing network reliability that may be used on any network with identical or mismatched components was suggested by Aggarwal and Shashwati. In 1988 the Mil-hdbk-338B standard for military reliability design, which is included in the reliability design Handbook, contains the

feasibility-of-objectives (FOO) technique. With this method, a thorough reliability allocation process for mechanical-electrical systems is provided [61]. In 1993 the idea is built on the numerical analytical method [6]. In 1996 the reliability design problems can be resolved using adaptive penalty-guided genetic search, which has shown to be effective and resilient for issues involving huge search areas and challenging restrictions [34] and the issue of limited redundancy allocation in a series system with interval-valued component reliability. The problem is stated as an optimization problem for increasing the total system reliability with restricted resource restrictions. Unconstrained integer programming issue solved by an advanced GA using penalty function and interval coefficients [59]. In 1997 Karmiol links the mission objectives to the system's criticality, operating profile, complexity, and state-of-the-art. Although Karmiol's allocation approach is quite thorough, it can only be used for systems that are connected in series and is subject to some analyst subjectivity [39], and in 2000 the dynamic programming and mixed-integer programming [145]. In 2001 the suggested multi-objective genetic algorithm GA approach then apply it to a realistically complex system, design issue aimed at determining the best system configuration and components with regard to dependability and financial cost objective this provides the decision-maker with the whole range of optimal options with regard to the different targets [24]. In 2004 Smedley used this allocation technique for a low energy booster ring magnet power system in a superconducting super collider, which is a mechanical-electrical system, the FOO approach is a fairly straightforward methodology, however it can only be used for systems that are connected in series and in the first stages of design [131] and the real-world engineering systems almost always have many competing objectives, hence multi-objective optimization is becoming increasingly popular, leading some providers to create a set of ideal solutions known as the Pareto optimum set [99]. In 2005 The optimization problem is generally referred to as a many-objective optimization problem when there are several objectives [51]. In 2007 the reliability optimization challenge of series-parallel systems has been addressed by the

multi-objective ant colony system (ACS) meta-heuristics. This category of issues involves choosing components with a range of options and levels of redundancy that maximize benefits while still being subject to weight and cost restrictions at the system level. When compared to other optimization techniques, the multi objective ACS algorithm has clear advantages for these issues and may be used to solve problems of a wider range of sizes and types. The multi-objective ACSRAP, which combines probabilistic search, multi-objective formulation of local moves, and the dynamic penalty method, enables us to swiftly and more frequently arrive at an optimal design solution than with some other methods [152]. In 2013 only systems connected in series and the early design stages can use ARINC [26]. In 2011 similar to this Bracha presented a system that considers four variables operation duration, environmental circumstances, sub level complexity as indicated by the number of pieces, and state-of-the-art., the Bracha method's biggest drawback is that it doesn't consider component criticality, to assign reliability to each [95]. In 2014 Due to its intrinsic benefits, including derivative-freeness, ease of implementation, and resilience, meta-heuristics algorithms have gained popularity as a method for solving multi-objective optimization problems. A non dominated sorting genetic algorithm (NSGA II) was presented and designed. An object-oriented technique was introduced to map each individual to each corresponding object and a population to an array of objects. The Pareto operation was implemented based on the ranks and congestion degrees of the individuals. The suggested method was used multi-objective optimization method for system reliability allocation [94]. In 2019 order to find the best system design under a number of limitations and maximize system reliability submitted a work examining the topic of intensive reliability and redundancy assignment [3]. In 2020 discussed reliability of the same system and calculated reliability allocation and optimization for a complex network using GA algorithm, PSO algorithm, Ant Colony Algorithm (ACA), and Bee's Colony optimization, with a comparison between these algorithms to choose the best algorithm that gives the highest reliability and lowest

cost [2]. In 2021 multi-objective reliability-based design optimization (MORBDO) method considering the maximum allowable deviation range of design variables is proposed for the reducer housing of electric vehicles. The structural parameters of the RBF are optimized using the heuristic global optimization ability of the particle swarm optimization algorithm. Sequential quadratic programming and non-dominated sorting genetic algorithm II are used to perform the MORBDO [144]. In 2019 Multi-objective system reliability optimization is provided using an adaptive particle swarm optimization (ADAP-PSO). For the purpose of avoiding local optima and ensuring diversity in the search space exploration, the method employs a Levy flight for some of the swarm's constituent particles. The a multi-objective problem is reduced to a single-objective problem, and a penalty function is put in place to handle the restrictions [103]. In 2022 the performance of wireless sensor networks is always significantly influenced by the deployment of the sensor nodes, create the competitive multi-objective marine predators algorithm, a swarm-based multi-objective optimization method, which provides a thorough understanding of the trade-off between heterogeneous WSNs deployment costs and reliability. This work specifically attempts to give the best deployment to increase coverage degree and connection degree, while at the same time minimizing total deployment cost [29]. In 2023 present a comparison between the results of Bat algorithm and Grey wolf optimization to optimize the reliability of complex networks [1].

### 1.5.2 Hybrid Nelder-Mead with meta-heuristics

Nelder and Mead simple research method is a derivative-free method to locate local search results in 1965 [113]. This method remeasures the single information on locating a local minimum of a function by using four fundamental operators. In 1991 the NM simplex algorithm is the most popular derivative-free and frequently used Nelder-Mead algorithm (NM) for optimizing meta-heuristics. It is also considered one of the most popular direct search methods for addressing unconstrained nonlinear optimization

problems. A method for accelerating the search and overcoming the algorithm's sluggish convergence is condensation. Additionally, hybrid algorithms are utilized to combine local and global search techniques since they can provide balance between exploration and exploitation that is necessary [17]. In 2005 the utilizing a hybrid approach Nelder-Mead and genetic algorithms to solve ordinary differential equations (ODEs) boundary value problems with the collocation Method [100]. In 2009 three engineering design problems were solved using a hybrid Nelder-Mead simplex search (NM) method and particle swarm optimization(PSO) technique known as NM-PSO [150]. In 2015 the optimal reactive power dispatch (ORPD) challenges of the power system are addressed using a hybrid algorithm that combines the Firefly Algorithm (FA) and the Nelder-Mead technique. With continuous and discrete control variables, ORPD is a highly nonlinear, non-convex optimization problem. It is observed that the proposed method has better convergence characteristics and robustness compared to the original version of FA [119]. In 2016 hybridization of the Cuckoo Search (CS) and Nelder-Mead technique, order to solve integer programming and minimax problems [9]. In 2018 a hybrid method combining tabu search (TS) and simplex Nelder-Mead (NM) has been proposed for solving global continuous optimization problems [153]. In 2019 the hybrid optimization algorithm (HWOANM) combining Nelder-Mead algorithm (NM) and whale optimization algorithm (WOA) and hybridization is intended to speed up the whale algorithm's global convergence rate for resolving production optimization issues [149], hybrid optimization algorithm that is based on the Dragonfly algorithm (DA)and improved Nelder-Mead algorithm (INM) algorithm [143]. In 2020 the hybrid optimization based on Grey Wolf optimizer and Nelder- Mead Method for solving multi-objective for scientific workflow scheduling (SWS) problems [107].

CHAPTER 2

# BASIC AND GENERAL CONCEPTS

## 2.1. Introduction

It is very necessary to start our study by including and clarifying some of the mathematical definitions and concepts that we need. Therefore, in this chapter, we will explain some concepts of probability theory such as properties of probability, independent events and probability density function,... etc. In addition, we discuss the graph theory and some of its concepts. Also, we will explain the structure function, types and its properties. Moreover, we will covers the fundamental definition of reliability, as well as specific types, characteristics, and reliability systems. We also discussed some concepts Petri net and the concept of optimization.

## 2.2. Some Basic Definitions of Probability

**Definition 2.2.1.** [106] The probability that a continuous random variable will be between limits a and b is given by an integral

$$Pr[a \leq X \leq b] = \int_a^b f(x)dx \tag{2.1}$$

The function $f(x)$ in equation ( 2.1 ) is called a probability density function ($pdf$).

**Definition 2.2.2.** [114] The cumulative distribution function($cfd$) for continuous random variables is expressed by

$$F(t) = \int_{-\infty}^{t} f(x)dx \tag{2.2}$$

Where $t$ is time, $f(x)$ is probability density function ($pdf$) and $F(t)$ is the cumulative distribution function, a relationship between $pdf$ and $cdf$ as follows:

$$f(x) = \frac{dF(x)}{dx} \tag{2.3}$$

**Definition 2.2.3.** [146] The reliability function, denoted $R(t)$, also called the survival function, is often interpreted as the population fraction surviving time $t$,$R(t)$ is the probability of success, which is the complement of $F(t)$. It can be written as

$$R(t) = Pr(T \geq t) = 1 - F(t) = \int_t^{\infty} f(t)dt \tag{2.4}$$

**Definition 2.2.4.** [71] [146] Hazard rate is the probability that a product will fail in the upcoming brief period of time is measured by the hazard function, abbreviated $H(t)$ and frequently referred to as the failure rate. This can be written as

$$H(t) = \lim_{\Delta t \to 0} \frac{P\{t < T \leq t + \Delta t, T > t\}}{\Delta t} = -\frac{1}{R(t)} \times \frac{dR(t)}{dt} = \frac{f(t)}{R(t)}$$

## 2.3. Basics Definitions of Graph Theory

**Definition 2.3.1.** [75] [130] A graph $G$ is a pair $(V, E)$, where $V$ and $E$ are finite sets. An element $v$ of $V$ is called vertex or nodes and an element $e$ or $e = \{u, v\}$of $E$ is called edge, we denote this simply by $uv$ and by convention the notation $e_{uv}$ is used for an edge between the vertices $u$ and $v$ $(u, v \in V)$.

***Example*** 2.3.1. In the graph $G$ in **Figure** (2.1) has five vertices $\{u, w, x, y, z\}$ and edges are $\{e_1, e_2, e_3, e_4, e_5\}$

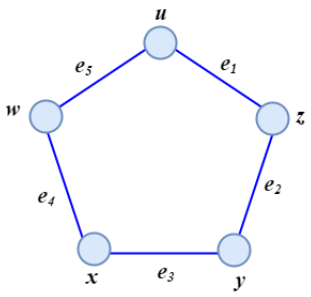



Figure 2.1: Graph G.

**Definition 2.3.2.** [80] A graph $G$ is bipartite when the vertices sets can be divided into the two sets $A$ and $B$, every edge has one endpoint in $A$ and the other in $B$, for see in **Figure** (2.2).

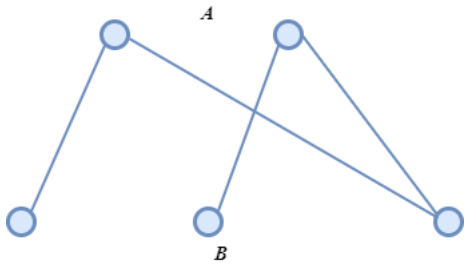



Figure 2.2: Two bipartite graphs.

**Definition 2.3.3.** [50] Child node if $(j, i)$ occurs in a network, node $i$ is considered a child of node $j$. The term "branch node" refers to a node with more than one child, for example in **Figure** (2.3) we get the nodes ( $v_2, v_3, v_4$) are child nodes for $v_1$.

**Definition 2.3.4.** [50] Node-child matrix is matrix $B_{(n\times p)}$, $n$ number of nodes, $p$ is the greatest out-degree of nodes, each network node is represented by a row. Nonzero entries in each row of $B$ are equal to the children's numbers of its corresponding node, for example have the Node-child matrix in **Figure** (2.9) .

$$M = \begin{array}{c} s \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ t \end{array} \begin{bmatrix} v_1 & v_2 \\ v_3 & v_4 \\ v_3 & v_4 \\ t & 0 \\ t & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.5}$$

**Definition 2.3.5.** [15] [63] The vertices $u$ and $v$ are said to be adjacent. If $uv \in E$. In this case, $u$ and $v$ are said to be the end vertices of the edge $uv$. If $uv \notin E$, then $u$ and $v$

are nonadjacent.

**Definition 2.3.6.** [46] [63] A edge $e$ has a vertices $v$ and $u$ as end vertex , the edge $e$ is said to be incident to the vertices $u$ and $v$.

**Definition 2.3.7.** [63] The neighborhood (or open neighborhood) of a vertex $v$, denoted by $N(v)$, is the set of vertices adjacent to $v$.

$$N(v) = \{x \in V \mid vx \in E\} \tag{2.6}$$

**Definition 2.3.8.** [130] Edge $e = (u, v)$ from initial point $u$ of $e$ to the terminal point $v$, is called a directed edge in in graph $G$.

**Definition 2.3.9.** [44] A directed graph or digraph $G$ if each edge of the graph $G$ has a direction show in **Figure** (2.3).

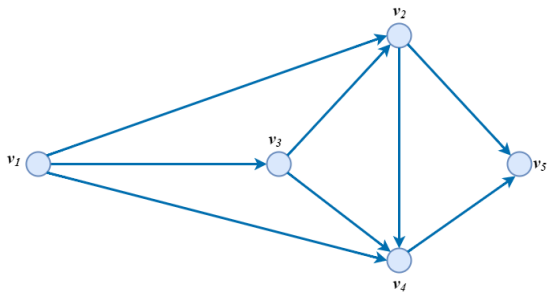



Figure 2.3: Directed digraph.

**Definition 2.3.10.** [140] An edge associated to a set $\{v, u\} \in E$ is called undirected edges if the edges can be traversed in both directions.

**Definition 2.3.11.** [141] Undirected graph each edge of the graph $G$ has no direction then the graph is called undirected graph show in **Figure** (2.4).

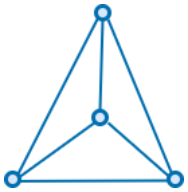


Figure 2.4: Undirected digraph.

**Definition 2.3.12.** [57] A graph $G$ with both directed and undirected edges is called a mixed graph, as an example in **Figure** (2.5).

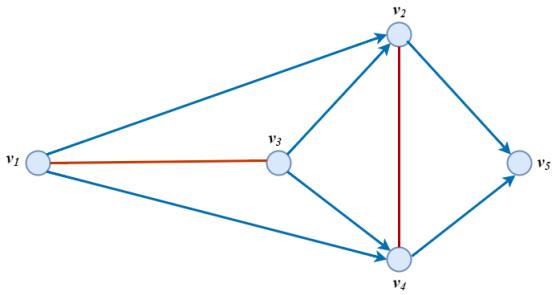



Figure 2.5: Mixed digraph.

**Definition 2.3.13.** [140] A graph $G$ is said to be a connected if every pair of vertices in $G$ are connected. Otherwise, $G$ is called a disconnected graph. Two vertices in $G$ are said to be connected if there is at least one path from one vertex to the other. In other words, a graph $G$ is said to be connected if there is at least one path between every two vertices in $G$ and disconnected if $G$ has at least one pair of vertices between which there is no path. A graph is connected if we can reach any vertex from any other vertex by traveling along the edges and disconnected otherwise, for example see **Figure** (2.6).

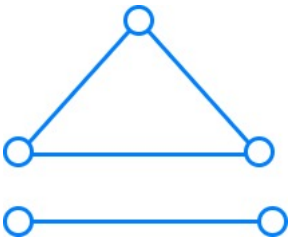


Figure 2.6: Unconnected graph.

**Definition 2.3.14.** [63] [15] Let $G$ be a graph with vertices $v_1, v_2, \ldots, v_n$. The adjacency matrix of $G$ is the $n \times n$ matrix $A$ whose $(i, j)$ entry, denoted by $[A]$ is defined by

$$A = \begin{cases} 1 & v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{if otherwise} \end{cases} \tag{2.7}$$

the graph in **Figure** (2.7) has four vertices. Its adjacency matrix $A$ is

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

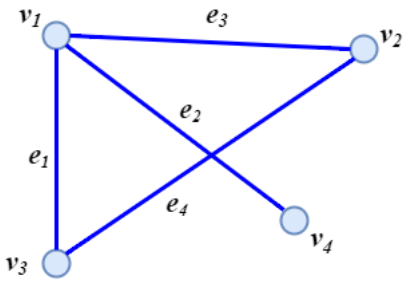



Figure 2.7: Graph with four vertices.

**Definition 2.3.15.** [117] [62] let $G$ be a graph with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and the edge set $E = \{e_1, e_2, .., e_m\}$. The incidence matrix $I$ of $G$ is an $n \times m$ whose $(i, j)$ entry is defined by

$$I = \begin{cases} 1 & \text{if the vertex } v_i \text{ is adjacent to the edge } e_j \\ 0 & \text{if otherwise} \end{cases} \tag{2.8}$$

illustrates the incidence matrix of the simple graph in **Figure** (2.8)

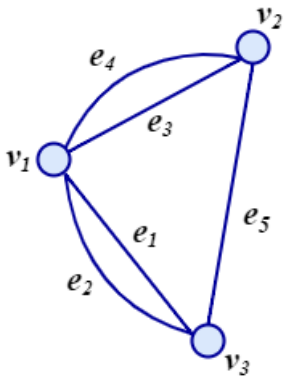



Figure 2.8: Graph with three vertices.

$$I = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

**Definition 2.3.16.** [86] [129] Connection matrix of graph $G$ with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$, used to represent adjacent matrix is defined by

$$C = \begin{cases} e & \textit{if the vertex } v_i \textit{ and } \ v_j \textit{ are adjacent} \\ 0 & \textit{if otherwise} \end{cases} \tag{2.9}$$

for example the connection matrix of the simple graph in **Figure** (2.7)

$$C = \begin{pmatrix} 0 & e_3 & e_1 & e_2 \\ e_3 & 0 & e_4 & 0 \\ e_1 & e_4 & 0 & 0 \\ e_2 & 0 & 0 & 0 \end{pmatrix}$$

## 2.4. Reliability Block Diagram (RBD)

In reliability analysis, we frequently use graphical systems modeling. This gives a visual representation of the parts and their configuration as a system. Functional blocks that are represented as rectangles or squares and connected by lines make up the diagram. There is just one starting point and one ending point in the reliability block diagram as shown in Figures (2.11),(2.12), a working state and a failing state are the two possible states for each functional block. An item or a particular function of an item may be represented by a functional block. A reliability block diagram which is not a physical layout design but rather shows the logical requirements for the system's operation [60] [121].

**Definition 2.4.1.** [53] Network $N$ is a triple $N = (V, E, T)$, where $E$ is a set of edges, $|E| = m$ , $V$ is a set of nodes or vertices, $|V| = n$, and $T$ is a set of special nodes called terminals, $T \subseteq V$ , $|T| = k$, for example the **Figure** (2.9) $|T| = 2$ and nodes $s, t$ are terminals.

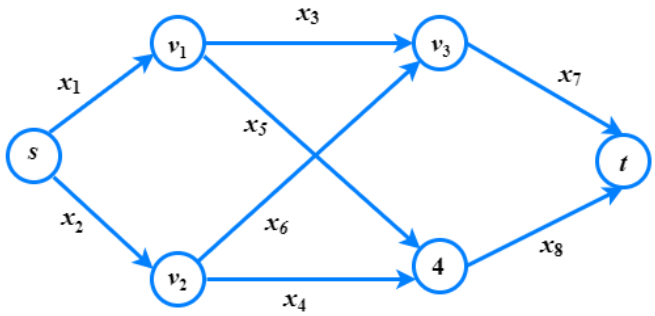



Figure 2.9: Network with two terminal.

**Definition 2.4.2.** [56] [69] Two terminal or terminal pair reliability (TPR) there are two special headers called terminals which are source ($s$) and terminal ($t$), the system functions if and only if in a network there is a path connecting the source node and the terminal node.

**Definition 2.4.3.** [77] The networks are considered operational if there is a path between a pair of nodes usually labeled source and sink, then the probability of a

message successfully reaching the sink node from the source node is termed two-terminal reliability, as **Figure** (2.9).

**Definition 2.4.4.** [89] complex network is a network, which consists of interconnected or interwoven parts (components) such that it becomes difficult to analyze it in respect of its reliability or a particular problem due to the constraints imposed by the existing techniques, algorithms, software (such as programming languages and operating networks), see **Figure** (2.10)

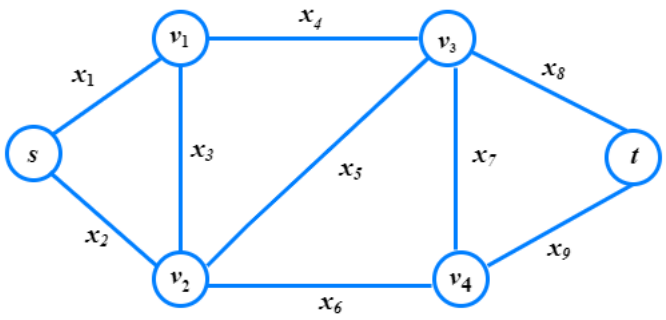



Figure 2.10: Network with cycle.

**Definition 2.4.5.** [67] [121] A path is a collection of components when they function, join the start node and the end node through other parts that are function , ensuring that the system is in a functioning state.

**Definition 2.4.6.** [40] [82] The minimal path set connects the source and sink nodes as long as it does not contain any cycles, the minimal path set cannot be reduced, as it has no redundant elements, and removing any of the edges from the path means that the source and sink nodes are no longer connected, as an example.

***Example*** 2.4.1. The minimal path in **Figure** (2.9) are

$MP_1 = \{e_{s,1}, e_{1,3}, e_{3,t}\} = \{x_1, x_3, x_7\}$,

$MP_2 = \{e_{s,2}, e_{2,4}, e_{4,t}\} = \{x_2, x_4, x_8\}$,

$MP_3 = \{e_{s,1}, e_{1,4}, e_{4,t}\} = \{x_1, x_5, x_8\}$,

$MP_4 = \{e_{s,2}, e_{2,3}, e_{3,t}\} = \{x_2, x_6, x_7\}$

and the minimal paths for network (2.10) are

$MP_1 = \{x_1, x_4, x_8\}, MP_2 = \{x_1, x_4, x_7, x_9\}, MP_3 = \{x_1, x_4, x_5, x_6, x_9\}, MP_4 = \{x_1, x_4, x_5, x_6, x_7, x_8\}, MP_5 = \{x_1, x_3, x_5, x_8\}, MP_6 = \{x_1, x_3, x_5, x_7, x_9\}, MP_7 = \{x_1, x_3, x_6, x_9\}, MP_8 = \{x_1, x_3, x_6, x_7, x_8\}, MP_9 = \{x_2, x_6, x_9\}, MP_{10} = \{x_2, x_6, x_7, x_8\}, MP_{11} = \{x_2, x_5, x_8\}, MP_{12} = \{x_2, x_5, , x_7, x_9\}, MP_{13} = \{x_2, x_3, x_4, x_8\}, MP_{14} = \{x_2, x_3, x_4, x_7, x_9\}.$

and cycles are

$c_1 = \{x_3, x_4, x_5\}, c_2 = \{x_3, x_4, x_7, x_6\}, c_3 = \{x_5, x_6, x_7\}.$

**Definition 2.4.7.** [20] A cut set is a set of system components which, when fail, cause failure of the entire system.

**Definition 2.4.8.** [12] [120] A cut set is called to be minimal if the set cannot be reduced without losing its status as a cut set.

***Example*** 2.4.2. The minimal cuts for **Figure** (2.9) are

$MC_1 = \{x_1, x_2\}, MC_2 = \{x_7, x_8\}, MC_3 = \{x_1, x_4, x_6\}, MC_4 = \{x_1, x_4, x_7\}, MC_5 = \{x_1, x_6, x_8\}, MC_6 = \{x_2, x_3, x_5\}, MC_7 = \{x_2, x_3, x_8\}, MC_8 = \{x_2, x_5, x_7\}, MC_9 = \{x_3, x_6, x_8\}, MC_{10} = \{x_4, x_5, x_7\}, MC_{11} = \{x_3, x_4, x_5, x_6\}.$

## 2.5. Reliability Function

**Definition 2.5.1.** [13] [120] Structure function we assume that there are only two possible states for the system and its component, working or failing. As a result, the state of each system component is a discrete random variable that can only take one of two potential values, which correspond to the states of operation and failure, respectively. Let

$x$ represent the state of component $i$ for $1 \leq i \leq n$

$$x_i = \begin{cases} 1 & \textit{if component } i \textit{ work} \\ 0 & \textit{if component } i \textit{ failed} \end{cases} \tag{2.10}$$

Then, vector $x = (x_1, x_2, ..., x_n)$ represents the states of all components and is called the component state vector. The components' states only determine the state of the system, which is also a binary random variable. In equation ( 2.11) describe the current state of the system

$$\phi = \begin{cases} 1 & \textit{if system work} \\ 0 & \textit{if system failed} \end{cases} \tag{2.11}$$

The system state is known if the states of every component are known. The component states determine the state of the system in a deterministic function. Thus, we often write

$$\phi = \phi(x) = \phi(x_1, x_2, ..., x_n)$$

$\phi(x)$ called structure function.

**Definition 2.5.2.** [138] System reliability $R$ is the probability that the system structure function equals 1

$$R = Pr(\phi(x)) = 1$$

Since $\phi(\cdot)$ is a binary random variable, the last formula can be written as

$$R = E(\phi(x))$$

**Definition 2.5.3.** [121] Series Systems. A system that is function if and only if all of its $k$ items are function is called a series system or a series structure

$$\phi(x_1.x_2.x_3.....x_k) = \prod_{i=1}^{k} x_i$$

and we observe that $\phi(x) = 1$ if and only if $f(x_i) = 1$ for all $i = 1, 2, .., k$, and the system reliability [155]

$$R(t) = \prod_{i=1}^{k} R_i(t)$$

The series system is illustrated by a reliability block diagram in **Figure** (2.11).


Figure 2.11: Series block diagram.

**Definition 2.5.4.** [110] Parallel Systems. A parallel structure is functioning if at least one of its $k$ components is function. The structure function is

$$\phi(x) = 1 - (1 - x_1)(1 - x_2)...(1 - x_k) = 1 - \prod_{i=1}^{k}(1 - x_i)$$

The reliability for a parallel system [125]

$$R(t) = 1 - \prod_{i=1}^{k}(1 - R_i(t))$$

a parallel structure of order $n$ is illustrated by the RBD in **Figure** (2.12)

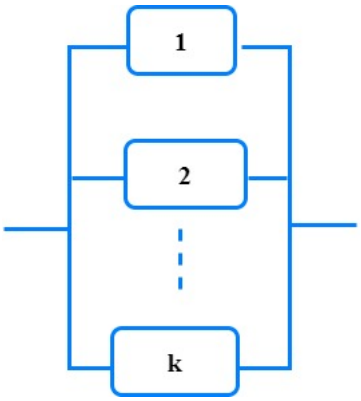



Figure 2.12: Parallel block diagram.

**Theorem 2.5.1.** *[54] Let $P_1, P_2, ..., P_m$ be minimal paths sets and $C_1, C_2, ..., C_k$ be*

*minimal cuts sets of the network, then we have*

$$\phi(x) = 1 - \prod_{j=1}^{m}(1 - \prod_{i \in P_j} x_i) \tag{2.12}$$

$$\phi(x) = \prod_{j=1}^{k}(1 - (\prod_{i \in c_j}(1 - x_i))) \tag{2.13}$$

**Definition 2.5.5.** [146] Parallel-Series Systems we refer to a system that has $m$ parallel subsystems, each of which has $k_i$ for $1 \leq i \leq m$ parts arranged in series, $1 \leq j \leq k_m$. **Figure** (2.13) shows a reliability block diagram for parallel series, representation minimal paths in parallel series we have structure function

$$\phi(x) = 1 - \prod_{i=1}^{m}(1 - \prod_{j=1}^{k_i} x_{ij}) \tag{2.14}$$

the reliability of a Parallel-Series is [86]

$$R(t) = 1 - \prod_{i=1}^{m}(1 - \prod_{j=1}^{k_i} r_{ij}(t)) \tag{2.15}$$

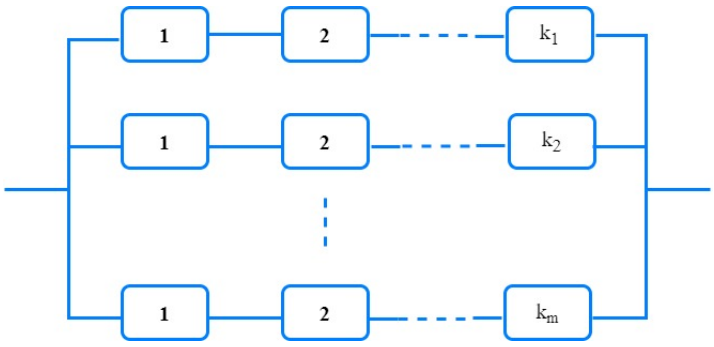



Figure 2.13: Parallel-Series block diagram.

For consider minimal paths in example (2.4.1) the system represented in **Figure** (2.14)

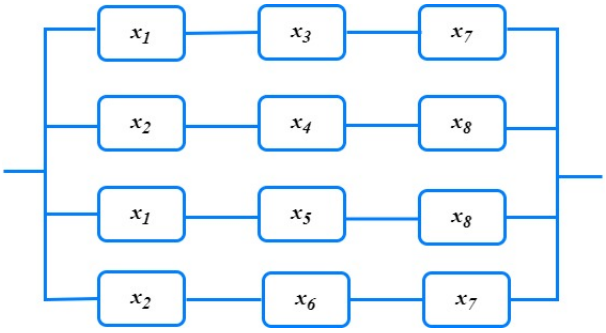



Figure 2.14: Parallel-Series block diagram for minimal paths.

**Definition 2.5.6.** Series-Parallel consists of $m$ modules that are connected in series, while module $i$ for $1 \leq i \leq m$ has $k_i$ components that are connected in parallel. A reliability block diagram for series parallel is shown in **Figure** (2.15), representing minimal cuts in series parallel, we have structure function [137].

$$\phi(x) = \prod_{i=1}^{m}(1 - (\prod_{j=1}^{k_i}(1 - x_{ij}))) \tag{2.16}$$

the reliability of a series- parallel is [139]

$$R(t) = \prod_{i=1}^{m}(1 - (\prod_{j=1}^{k_i}(1 - r_{ij}(t)))) \tag{2.17}$$

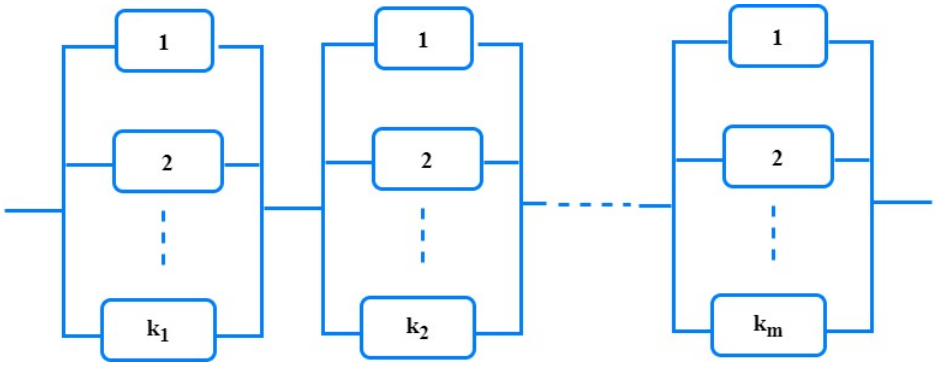



Figure 2.15: Block diagram of series-parallel.

For consider minimal cuts in example (2.4.2) the system represented in **Figure** (2.16)

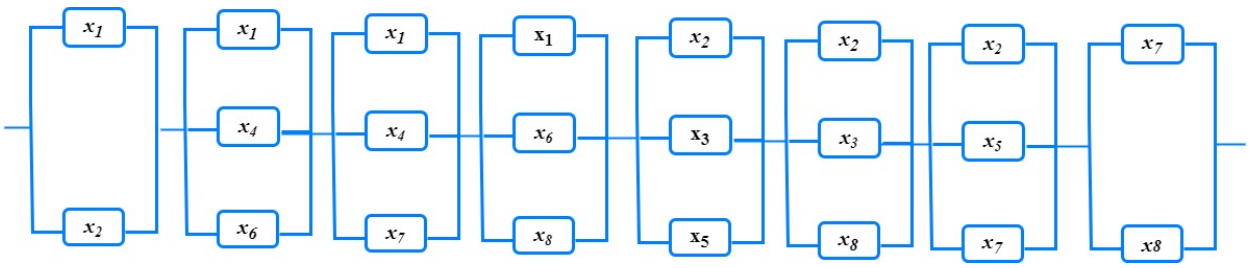



Figure 2.16: Block diagram of the minimal cuts.

**Definition 2.5.7.** [67] [86] [115] A component $x$ is in the system of its structural function $\phi$ irrelevant if its state does not affect the state of the system at all. The component $i$ is irrelevant if and only if $\phi(1_i, x) = \phi(0_i, x)$ for any component state vector $x$, otherwise $i$ is called relevant, where $\phi(1_i, x)$ represents a state vector where the state of the $i$th component equal 1, $\phi(0_i, x)$ represents a state vector where the state of the $i$th component equal 0. **Figure** (2.17) shows a system of where component $(x_2)$ is irrelevant.

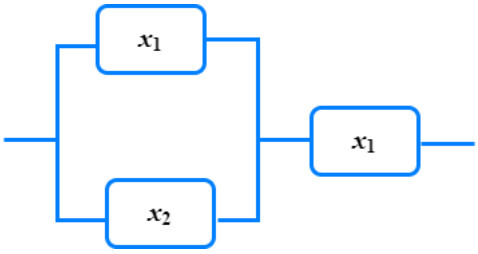



Figure 2.17: Irrelevant system.

The structure function for system (2.17) is

$$\phi(x_1, x_2) = x_1 + x_1x_2 - x_1x_2 = x_1$$

then $\phi(x_1, 1) = \phi(x_2, 0)$, hence component (2) is irrelevant in the system.

**Definition 2.5.8.** [18] A system of components is called to be coherent if all its components are relevant and the structure function is nondecreasing in each its arguments $\phi(0_i, x) \leq \phi(1_i, x)$ for all $x$.

**Theorem 2.5.2.** *[86] [122] Let $\phi(x)$ be the structure function of coherent system then $\phi(0) = 0$ and $\phi(1) = 1$ in other words*

- if all the components in a coherent system are functioning , then the system is functioning.
- if all the components in a coherent system are failed, then the system is failed state.

**Theorem 2.5.3.** *[18] [122] Let $\phi(x)$ be the structure function of coherent system of order $n$ then we have*

$$\prod_{i=1}^{n}(x_i) \leq \phi(x) \leq \coprod_{i=1}^{n}(x_i)$$

in other words the parallel structure is the best while the series system is the worst.

**Theorem 2.5.4.** *[111] [122] let $\phi$ be a coherent system of order $n$ then we have*

- $\phi(x \bigcup y) \geq \phi(x) \bigcup \phi(y)$
- $\phi(x \cdot y) \leq \phi(x) \cdot \phi(y)$

We obtain a better structure of a parallel system, by redundancy at the component level is preferable to redundancy at the system level. And connecting two systems, each with structure function in a series superior to replacing each element by a series connection of the two corresponding system elements.

Components are working correctly, and various systems which consist of combinations of series and parallel subsystems form a kind of network structure. All these are examples of coherent systems. *Coherent systems are a kind of system for which it is to measure the relation between system and component reliability* [37].

## 2.6. Formula of Optimization Problem

An optimization or mathematical programming problem is be defined in general mathematical terms as [97] [126] :

$$(P_1)\begin{cases} Optimize\ [\ min\ ]f(x) & \\ subject\ to & h_j(x)=0, j=1,2,...,l \\ & g_i(x)\leq 0, i=1,2,...,m \\ & x\in R^n \end{cases} \tag{2.18}$$

also if maximization problem can be represented as follows [134]

$$(P_2)\begin{cases} Optimize\ [\ max\ ]f(x) & \\ subject\ to & h_j(x)=0, j=1,2,...,l \\ & g_i(x)\geq 0, i=1,2,...,m \\ & x\in R^n \end{cases} \tag{2.19}$$

The maximization problem $maxf(x)$ can be cast in the standard form equation (2.18) by observing that $maxf(x) = -minf(x)$. Once the minimize $x^*$ is obtained, the maximum value of the original maximization problem is given by $-f(x^*)$ [132], as shown in **Figure** (2.18).

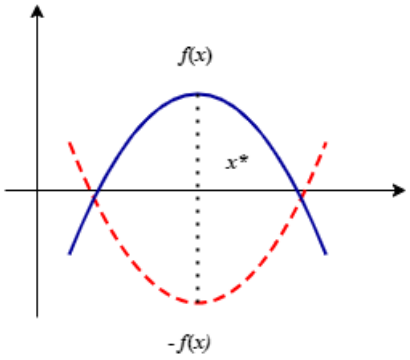



Figure 2.18: Maximization problem transformed to minimization problem.

Non-linear programming, mathematical programming, and numerical optimization are used to describe mathematical optimization model.

An optimization problem $f : R^n \rightarrow R$ purpose is to identify a collection of choice variables $x$ that maximizes or minimizes the objective function $f$ while maintaining the model's $h$

and $g$ constraints. **Figure** (2.19) describing. The optimization phase, which itself spans numerous stages, is a crucial component in many decision-making and design processes. The optimization process's goal is to assist in determining realistic and workable solutions for management design and decision-making processes [4].

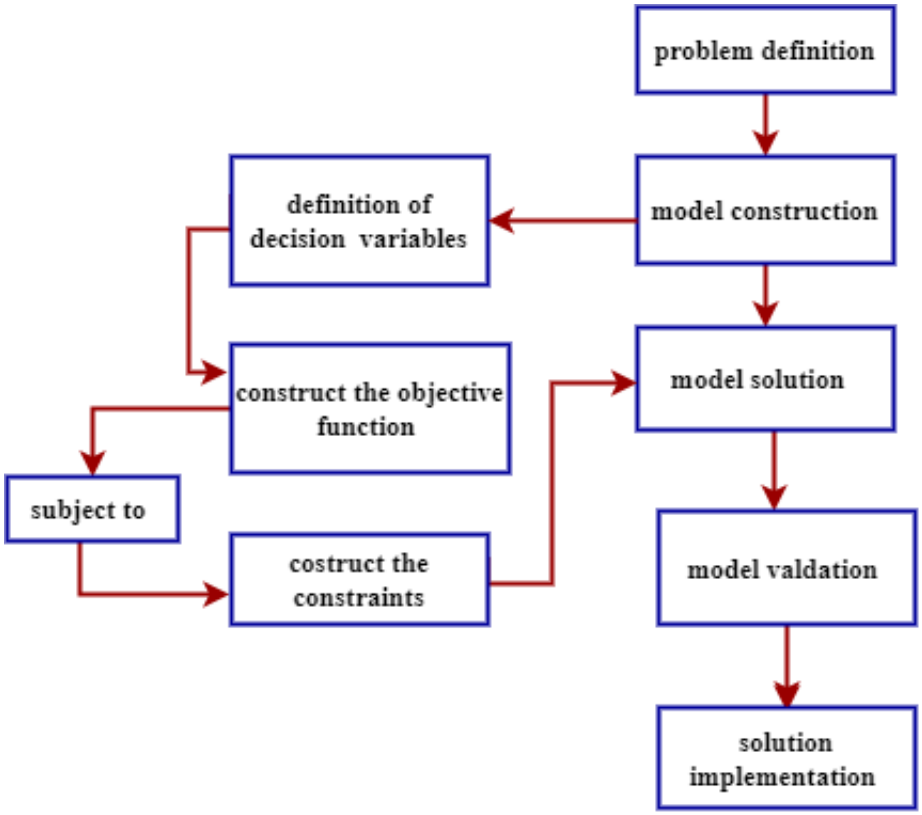



Figure 2.19: Construction of model for numerical optimization.

### 2.6.1 Concepts of Optimization Problem

**Definition 2.6.1. Objective Function.** [116] [154]. In a mathematical optimization issue, the objective function is a real-valued function whose value must be minimized or maximized across the set of possible options, for example see equations (2.18) and (2.19).

**Definition 2.6.2. Decision variables.** [65] In an optimization problem, decision variables are variables whose values may change throughout the feasible set of alternatives to improve or decrease the main objective function's value. The vector $x$

represents the vector of choice variables in the preceding problem equations (2.18) and (2.19).

**Definition 2.6.3. Constraint and unconstrained.** [116] The general form equations 2.18 and 2.19 problems may be characterized by the number of variables, their size, and the smoothness of their relationships with the objective function and constraints (linear, non-linear, convex), among other characteristics. Perhaps the most critical difference is between issues with and without changeable constraints.

**Definition 2.6.4. Feasible solution.** [148, 156] Points satisfies all the constraints, it is called a feasible solution, otherwise, it is infeasible. All the feasible points or vectors form the feasible regions in the search space, for example, consider the following optimization

$$\begin{aligned} min \quad & f(x,y) \\ \text{subject to}: \quad & x+y \leq 1 \\ & x \geq 0 \\ & y \geq 0 \end{aligned} \tag{2.20}$$

in Figure (2.20 ) shows that the feasible area is the collection of all possible points.

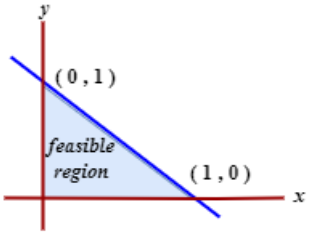



Figure 2.20: Feasible region.

**Definition 2.6.5. Linear programming problem.** [97] A linear program (LP) is a mathematical formulation of the optimization problems, an objective function is linear, and the constraints are made up of linear equality and inequality constraints. The issue takes on its typical shape.

$$\begin{aligned} &min && c^T y \\ &\text{subject to:} && Ay = b \quad ; y \geq 0 \end{aligned} \tag{2.21}$$

In this case, $y$ denotes an $n$-dimensional column vector, $c^T$ denotes a row vector with $n$ dimensions, $A$ represents an $m$-dimensional matrix, and $b$ denotes a column vector with $m$-dimensions, $y \geq 0$, shows that each of $y$ components is positive.

***Example*** 2.6.1.

$$\begin{aligned} &min && f(y_1, y_2) = -y_1 - y_2 \\ &\text{subject to :} && y_1 + 2y_2 \leq 3 \\ & && 2y_1 + y_2 \leq 3 \\ & && y_1, y_2 \geq 0 \end{aligned} \tag{2.22}$$

The graphical solution is shown in **Figure** (2.21), we have $f^* = -2$ with $y_1^* = y_2^* = 1$

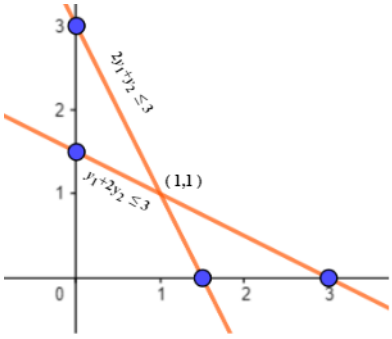



Figure 2.21: The graphical solution for example 2.6.1.

### 2.6.2 Non-Linear Programming

**Definition 2.6.6. Non-Linear programming.** [21] Problems involving non-linear programming (NLP) have a non-linear objective function or constraints, or both of these. No efficient approaches exist for tackling the non-linear programming issue in its entirety. When it comes to solving issues with ten variables and hundreds of variables, even the most straightforward situations can be difficult to solve. As a result, numerous techniques to solve the general non-linear programming issue have been developed, each

of which entails some degree of compromise.

$$\begin{aligned} max \qquad & z(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 2)^2 \\ \text{subject to :} \qquad & 3x_1^2 - x_2 - 3 \leq 0 \\ & x_1 + 2x_2 \leq 5 \\ & x_2 - 1 \leq 3 \\ & x_1 \geq 0 \end{aligned} \tag{2.23}$$

**Definition 2.6.7. Convex sets.** [88] [132] If a chord links any two points $p$ and $q$ in a set $S$ , then the set $S$ is convex. And is also

$$\lambda p + (1 - \lambda)q \in S, \quad for \quad 0 \leq \lambda \leq 1 \tag{2.24}$$

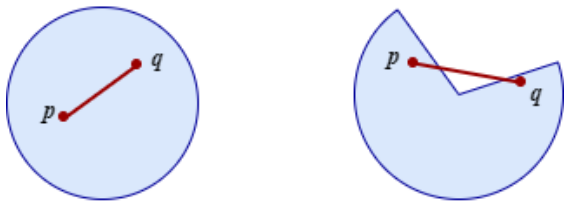



Figure 2.22: Convex and Non-Convex Set.

**Definition 2.6.8. Convex functions.** [21, 132] A function is said to be convex if the chord that connects two points on its curve is never lower than the curve, **Figure** (2.23)
Mathematically:
Let f: $R^n \to R$ and $S \in R^n$, for any $\chi_1, \chi_2 \in S$ then

$$f\left(\lambda\chi_1 + (1 - \lambda)\chi_2\right) \leq \lambda f\left(\chi_1\right) + (1 - \lambda) f\left(\chi_2\right), \quad 0 \leq \lambda \leq 1 \tag{2.25}$$

If the above inequality is true as a strict inequality for all $\chi_1 \neq \chi_2$

$$f\left(\lambda\chi_1 + (1 - \lambda)\chi_2\right) < \lambda f\left(\chi_1\right) + (1 - \lambda) f\left(\chi_2\right), \quad 0 \leq \lambda \leq 1 \tag{2.26}$$

then $f$ is called a strict convex function. When the function is reversed, it becomes concave:

$$f\left(\lambda\chi_1 + (1-\lambda)\chi_2\right) \geq \lambda f\left(\chi_1\right) + (1-\lambda) f\left(\chi_2\right), \quad 0 \leq \lambda \leq 1 \tag{2.27}$$

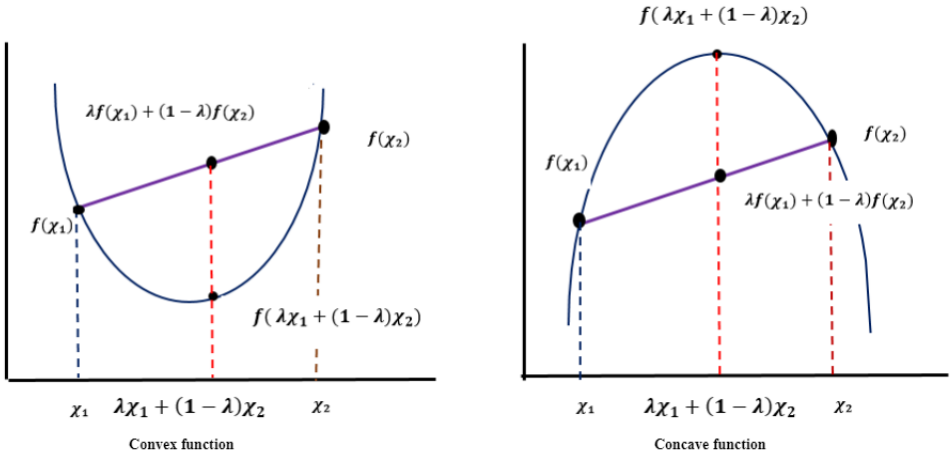



Figure 2.23: Convex and concave functions.

**Definition 2.6.9. Local.** [73] [104] A point $x^* \in R^n$ is a problem's local optimal if and only if exists $\delta > 0$ such that $f\left(x^*\right) \leq f(x)$ for every $x \in S \cap B\left(x^*, \delta\right)$. **Figure** (2.24) shown local minimum points.

**Definition 2.6.10. Global.** [73] [104] If $f\left(x^*\right) \leq f(x)$ for all $x \in S$, a point $x^* \in S$ be a global optimum of the problem 2.18 and 2.19. **Figure** (2.24) shown global minimum points.

In the above definitions, we replace $(\leq)$ with$(<)$ then we have strict local minimum and strict global minimum, respectively [73].

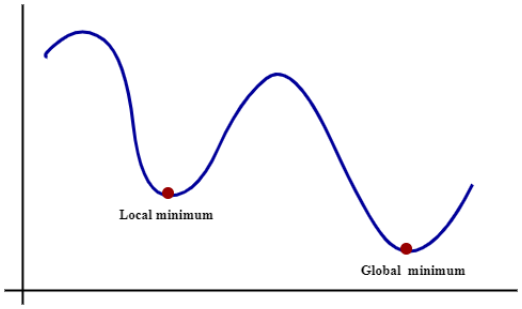



Figure 2.24: Local and global minimum.

**Theorem 2.6.1.** *[58] [135] Consider the convex set ($M \in R^n$) and the convex function f: $M \to R$.*

1. *If $x \in M$ is a local minimum of $f$ on $M$, then $x$ is a global minimum of $f$ on $M$.*
2. *If $f$ is strictly convex, then $x$ a unique global minimum.*

## 2.7. Optimality Conditions

**Definition 2.7.1.** [135] Let $M \in R^{n \times n}$ be symmetric matrix. $M$ is said to be positive definite if $x^T Mx > 0, \forall x \in R^n, x \neq 0$. $M$ is said to be positive semi definite if $x^T Mx \geq 0$, $\forall x \in R^n$. $M$ is said to be negative definite or negative semi definite if $-M$ is positive definite or positive semi definite. $M$ is said to be indefinite if it is neither positive semi definite nor negative semi definite.

### 2.7.1 Optimality Conditions for Unconstrained Optimization

**Theorem 2.7.1.** ***First-Order Necessary Condition (FONC)*** *[79] [142] An open set $S$ in which the local minimum is located may be defined as the set $f : R^n \to R$. $\bigtriangledown f(x)$ then equals zero.*

**Theorem 2.7.2.** ***Second-Order necessary Condition(SONC)*** *[30] [135]*
*Let $f : D \subset R^n \to R$ be twice continuously differentiable on an open set $D$. If $x^*$ is a local minimum of 2.7.1, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.*

**Theorem 2.7.3.** ***Second-Order sufficient Condition (SOSC)*** *[30] [135]*
*Let $f : R^n \to R$ be twice continuously differentiable. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, then $x^*$ is a strict local minimum.*

### 2.7.2 Optimality Conditions for Constraints

The constraint system can be converted to unrestricted using **(Lagrange Multiplier Rule)**

**Theorem 2.7.4.** ***Lagrange Theorem****(Lagrange Multiplier Rule) [36] If $x^*$ is a local minimum for a constraint problem, then there exists numbers $(\iota_1, \ldots, \iota_0)$ such that:*

$$\nabla f(x^*) - \sum_{1}^{\ell} \iota_i^* \nabla h_i(x^*) = 0^T \quad , i = 1 \ldots n$$

*And*

$$h_i(x^*) = 0, \quad i = 1, \ldots, \ell$$

**Theorem 2.7.5.** ***First-Order Necessary Condition*** *[36, 156] called condition is the Karush-Kuhn-Tucker (KKT) condition. Let $\left(f, g, h \in C^1\right)$, $x^*$ be a regular point and a local minimum for the problem of minimizing $f$ subject to $h(x) = 0, g(x) \leq 0$, and $\iota^* \in R^l, \xi^* \in R^m$ such that*

1. $\xi^* \geq 0$.
2. $\nabla f(x^{*T}) + \iota^* \nabla h(x^{*T}) + \xi^{*T} \nabla g(x^*) = 0$.
3. $\xi^{*T} g(x^*) = 0$.

*where $\iota$ the Lagrange multiplier vector, and $\xi$ the Karush-Kuhn-Tucker (KKT) multiplier vector*

**Theorem 2.7.6.** ***Second-Order Necessary Condition*** *[11, 36] let $x^*$ be local minimum of problem 2.18 and $\left(f, g, h \in C^2\right)$. Suppose $x^*$ is regular. Then, there exist $\iota^* \in R^l$ and $\xi^* \in R^m$ such that*

1. $\xi^* \geq 0, \nabla f(x^{*T}) + \iota^* \nabla h(x^{*T}) + \xi^{*T} \nabla g(x^*) = 0, \xi^{*T} g(x^*) = 0, and$
2. *For all $y \in \tau(x^*)$ we have $y^T L(x^*, \iota^*, \xi^*) y \geq 0$ and*

$$\tau(x^*) = \{y \in R^n : \nabla h(x^*) y = 0, \nabla g_j(x^*) y = 0, j \in J(x^*)\}$$

*where $J(x^*) = \{j : g_j(x^*) = 0\}$.*

**Theorem 2.7.7.** ***Second-order sufficient conditions:*** *[36, 135] Suppose* $\left(f, g, h \in C^2\right)$ *and there exist a feasible point* $x^* \in R^n$, *and* $\iota^* \in R^l, \xi^* \in R^m$ *such that*

1. $\xi^* \geq 0, \nabla f(x^{*T}) + \iota^* \nabla h(x^{*T}) + \xi^{*T} \nabla g(x^*) = 0, \xi^{*T} g(x^*) = 0,$ *and*

2. *For all* $y \in \tilde{\tau}(x^*, \xi^*)$ *we have* $y^T L(x^*, \iota^*, \xi^*) y \geq 0$ *and*

$$\tilde{\tau}(x^*, \xi^*) = \{y : \nabla h(x^*) y = 0, \nabla g_j(x^*) y = 0, j \in J(x^*, \xi^*)\}$$

*where* $J(x^*, \xi^*) = \{i : g_i(x^*) = 0, \xi^* \geq 0\}$.

*Then,* $x^*$ *is a strict local minimum of* $f$ *subject to* $h(x) = 0, g(x) \leq 0$.

## 2.8. Direct Methods

To determine when they have attained a local minimum or to direct them toward one, direct approaches do not rely on derivative information. The next search direction and the point at which they judge that they have converged are determined by other criteria. Only the objective function is used in direct procedures. These techniques are also referred to as black box, pattern search, zero-order, or derivative-free techniques, to either direct them toward a local minimum or to determine when they have reached one [79].

## 2.9. Nelder-Mead Simplex Method

The Nelder–Mead method (NM method) is technique a heuristic search a for unconstrained optimization without using derivatives, and it was first developed by J.A. Nelder and R. Mead in 1965 John Nelder and Roger Mead outlining a novel approach to unconstrained minimization [113]. The resulting optimization method has grown to become one of the most used and studied optimization algorithms worldwide. Nelder and Mead's approach was initially referred to as the " Simplex Method " ,as the approach relied on finding a minimum by employing function evaluations at a simplex's

vertices. Of fact, the term "Simplex Method" was already in use in the field of optimization, referring to Dantzig's method algorithm for solving linear programmes [148]. As a result, the technique was given the name Nelder-Mead (NM) method to avoid confusion Nelder-Mead approach that can converge to nonstationary positions. Although, it maintains a simplex of points at each iteration of the search and uses the function values at the simplex's vertices to direct the search. For $n$-dimensional issues, the simplex is a special sort of polynomial with $n+1$ vertices. A simplex in one dimension is a line, and in two dimensions it is a triangle see **Figure** (2.25) or a tetrahedron in three dimensions. The Nelder-Mead method uses a series of rules that dictate how the NM algorithm is updated based on evaluations of the objective function at its vertices. A flowchart outlines the procedure in **Figure** (2.26) [16] construct an initial $n$-simplex with $n+1$ vertices and to evaluate the objective function at the vertices. Then by ranking the objective values and re-ordering the vertices, we have an ordered set so that [72] [74]

$$f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$$

at $x_1, x_2, ..., x_{n+1}$, respectively. These are the steps that make up the NM algorithm

1. Calculate the centroid of the best $n$ points except for $f(x_{n+1})$ is denoted by

$$x_c = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq \text{ worst}}}^{n+1} x_i$$

2. Reflection: Compute $x_r = x_c + \rho\,(x_c - x_{n+1})$, Here, $\rho > 0$ and is typically set to 1.

   - If $f\,(x_r) < f\,(x_1)$ the algorithm continues with expansion.
   - If $f\,(x_1) \leq f\,(x_r) < f\,(x_n)$, replace he worst point $x_{n+1}$ with the reflected point $x_r$ and go to step 1.

3. Expansion it is calculated that the expanded point is

$$x_e = x_c + \mu\left(x_r - x_c\right)$$

Here, $\mu > \max(1, \rho)$ and is typically set to 2. similar to reflection, except the point that is reflected is sent further. And If $f\left(x_e\right) < f\left(x_r\right)$, replace the point $x_{n+1}$ by $x_e$ and go to step 1, otherwise, the In every other case, then accept $x_r$ and go to step 1.

4. Contraction the calculated contracted point is $f\left(x_n\right) \leq f\left(x_r\right)$ A contraction takes place. Two contractions are conceivable.

   - Outside: If $f\left(x_r\right) < f\left(x_{n+1}\right)$ contraction by the formulae.

$$x_{\text{con}} = x_c + \sigma\left(x_r - x_c\right), 0 \leq \sigma \leq 1$$

     If $f\left(x_{con}\right) < f\left(x_r\right)$ then by replacing the worst point $x_{n+1}$ with the contracted point $x_{con}$ and go to step 1, otherwise perform a shrink operation.

   - Inside: If $f\left(x_r\right) \geq f\left(x_{n+1}\right)$ calculate inside contraction

$$x_{\text{con}} = x_c - \sigma\left(x_c - x_{n+1}\right) \quad , \mathbf{0} \leq \sigma \leq 1$$

     and If $f\left(x_{\text{con}}\right) < f\left(x_{n+1}\right)$, accept then obtain a new simplex by replacing the worst point $x_{n+1}$, with the contracted point $x_{\text{con}}$ and go to step 1, otherwise go to calculated a shrink step.

5. Ashrink step calculate shrink by the formulae

$$v_i = x_1 + \delta\left(x_i - x_1\right), 0 < \delta < 1 \quad , \quad i = 2, \ldots, n+1$$

and go to step 1.

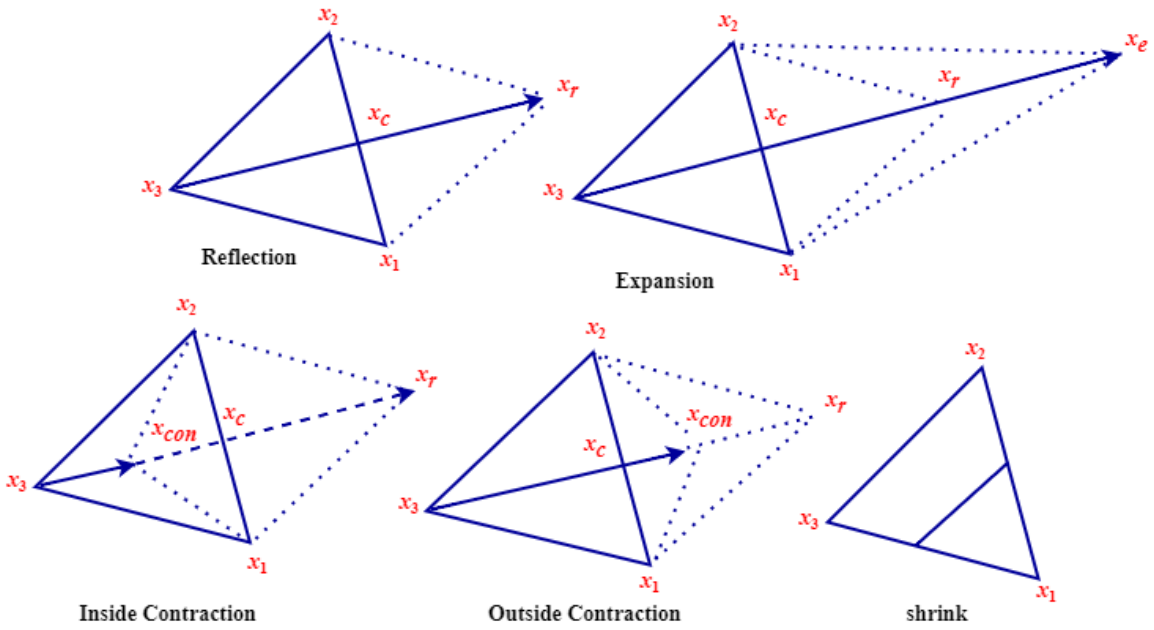



Figure 2.25: Nelder-Mead method operations visualized in two dimensions.

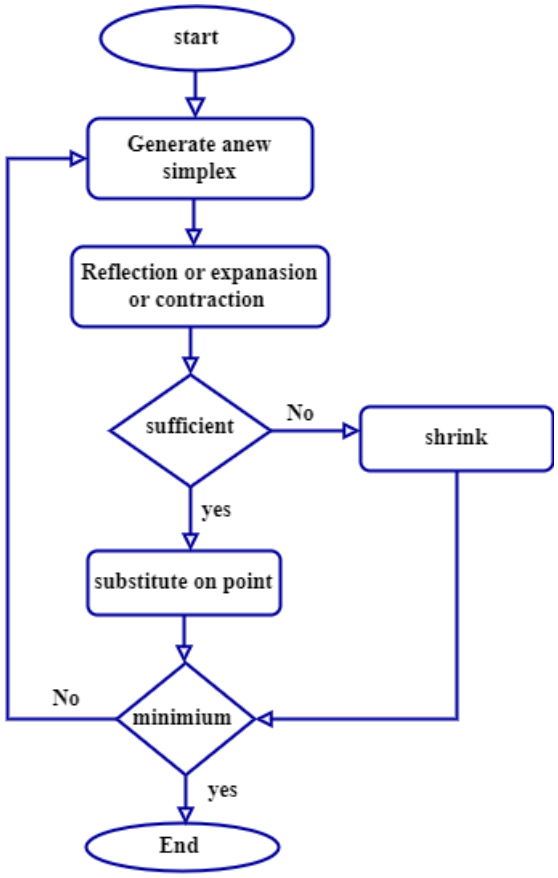



Figure 2.26: Flowchart of Nelder-Mead algorithm.

***Example*** 2.9.1. Illustrates the Nelder-Mead algorithm to find the minimum of

$$f(x, y) = (x - 10)^2 + (y - 10)^2$$

We calculate the first steps and illustrate them in **Figure** 2.27

- First iteration start with three vertices and $n_1 = (0, 0), n_2 = (2, 0), n_3 = (0, 6)$ the function values $f(x, y)$ at these points

  $$f(0, 0) = 200, f(2, 0) = 164, f(0, 6) = 116$$

  Since $f(0, 6) < f(2, 0) < f(0, 0)$, the vertices will to give the names

  $$x_l = (0, 6), x_g = (2, 0), x_h = (0, 0)$$

  The centroid of $x_l x_g$ and the reflected point is:

  $$x_c = \frac{x_l + x_g}{2} = (1, 3), x_r = 2x_c - x_h = (2, 6)$$

  The function value at $x_r$ ,$f(x_r) = f(2, 6) = 80 < f(x_g) < f(x_l)$ and the extended point $x_e$ will be calculated from:

  $$x_e = 2x_e - x_c = (3, 9)$$

  . The new triangle is $\triangle x_l x_g x_e$

- Second iteration since the function value at $x_e$ is $f(3, 9) = 50 < f(x_g) < f(x_l)$. The new centroid of $x_l$ $x_g$ and the reflected point $x_r$ are

  $$x_l = (3, 9), x_g = (0, 6), x_h = (2, 0)$$

The value of centroid and the reflected point is:

$$x_c = (1.5, 7.5), x_r = (1, 15)$$

$f(1, 15) = 106$ this is again less than the value at, $f(x_g) = f(0, 6) = 116$ but greater than $f(x_l) = f(3, 9) = 50$. Therefore, the extended point will not be calculated and the new triangle is $\triangle x_r x_g x_l$

- Third iteration because $f(3, 9) < f(1, 15) < f(0, 6)$, and

$$x_l = (3, 9), x_g = (1, 15), x_h = (0, 6)$$

The new centroid and the reflected point are $x_c = (2, 12)$ and $x_r = (4, 18)$ and the function takes at $x_r$ the value $f(x_r) = f(4, 18) = 100$. This is again less than the value at $x_g$ but it is not less than the value at $x_l$, thus the point $x_g$ will be a vertex of the next triangle $\triangle x_r x_g x_l$.

- Fourth iteration because $f(3, 9) < f(4, 18) < f(1, 15)$

$$x_l = (3, 9), x_g = (4, 18), x_h = (1, 15)$$

The value of centroid and the reflected point is

$$x_c = (3.5, 13.5), x_r = (6, 12)$$

Because $f(x_r) = f(6, 12) = 20 < f(x_l)$, we shall build the extended point $x_e$ $x_e = (8.5, 10.5)$ The function takes here the value $f(x_e) = f(8.5, 10.5) = 2.5$, thus the new triangle is $\triangle x_g x_l x_e$.

The calculations proceed until almost equal function values are found for all of the vertices. The procedure will still include a substantial number of steps before it is finished at the final stage $(10, 10)$.

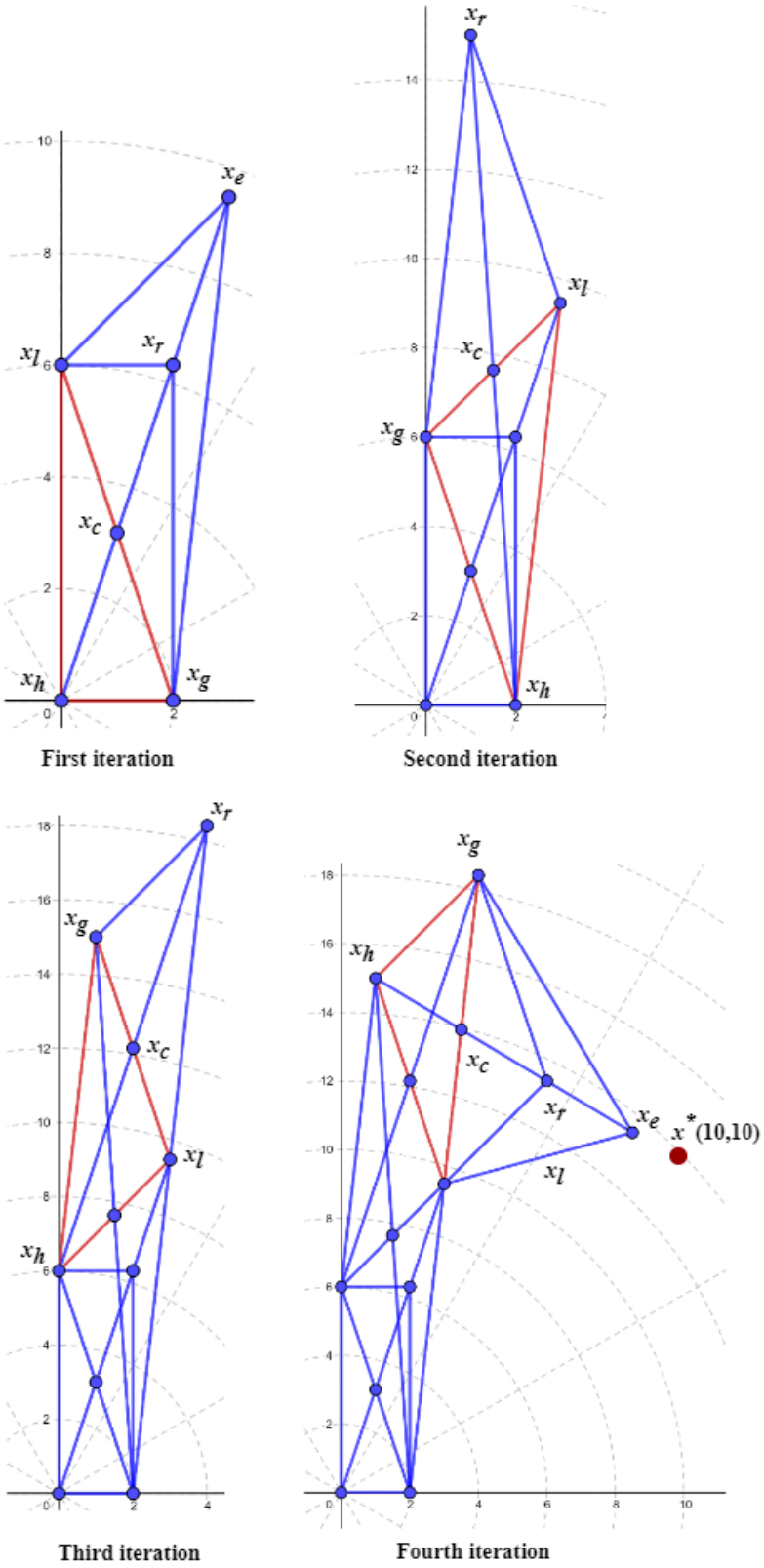



Figure 2.27: First iteration steps of the Nelder-Mead method to solve example 2.9.1.

## 2.10. Constrained to Unconstrained Problem

### 2.10.1 Penalty Function Method

The fundamental idea penalty methods is transform constrained optimization problems to an unconstrained optimization problems, the modified objective function with penalty terms is written as equation (2.28) [25] [132]

$$F(x) = f(x) + p(x, \kappa, \vartheta) \tag{2.28}$$

The penalty function $p$ is usually of the form

$$p(x, \kappa, \vartheta) = \sum_{j=1}^{l} \kappa_j h_j^2(x) + \sum_{i=1}^{m} \vartheta_i g_i(x)^2 \tag{2.29}$$

The parameters $\kappa_j$ and $\vartheta_i$ are called penalty parameters and the function

$$g_i(x) = max\{0, g_i(x)\} \tag{2.30}$$

the equation (2.29) the quadratic penalty. It is possible to create penalty functions that are precise in the sense that, for a given value of the penalty parameter, the solution to the penalty problem returns the exact solution to the original problem. These functions eliminate the need to work through an endless series of penalty issues in order to arrive at the right answer. However, the fact that a new difficulty introduced by these penalty functions is that they are non differentiable, and where the exact Penalty function is defined by [81] [127]

$$PF(x) = f(x) + \sum_{j}^{l} \kappa_j |h_j(x)| + \sum_{i}^{m} \vartheta_i g_i(x) \tag{2.31}$$

***Example*** 2.10.1. Consider the problem

$$\text{Minimize } (x_1 - 3)^2 + 2x_2^2$$

$$\text{subject to } x_1 + x_2 = 4$$

$$(x_1 - x_2)^2 \leq 9$$

After solving the prior problem with classical methods, we obtain $x^* = \left(\frac{7}{2}, \frac{1}{2}\right)$ with objective function value $\frac{3}{4}$, consider the following penalty function as a starting point for building theory around penalty functions

$$\psi(x_1, x_2) = (x_1 + x_2 - 4)^2 + \max\left\{0, (x_1 - x_2)^2 - 9\right\}^2$$

constraint $1 = |x_1 + x_2 - 4|$ constraint $2 = \max\left\{0, (x_1 - x_2)^2 - 9\right\}$. Therefore, our goal is to minimize

$$F(x_1, x_2) = (x_1 - 3)^2 + 2x_2^2 + \vartheta(x_1 + x_2 - 4)^2 + \vartheta \max\left\{0, (x_1 - x_2)^2 - 9\right\}^2$$

We determine the minimum of each iteration based on the current value. Every best solution serves as a jumping off point for the following iteration. When the violation is less than, the loop is terminated. As can be seen **Table** (2.1), we're getting close feasible region from.

Table 2.1: Penalty method.

| Iteration | $\mu$ | $x$ | Objective value |
|---|---|---|---|
| 1 | 10 | (3.46544586, 0.4648014) | 0.648721 |
| 2 | 100 | (3.49631232, 0.49624341) | 0.73884 |
| 3 | 1000 | (3.49962861, 0.49962167) | 0.74887 |
| 4 | 10000 | (3.49996271, 0.49996202) | 0.749887 |
| 5 | 100000 | (3.49999615, 0.49999608) | 0.74999 |

In **Table** (2.1) five iteration represents the value of $x_1$ and $x_2$ the objective value represents

the objective function in 0.00001 we get at (3.49999615, 0.49999608), which is quite near to the precise best solution.

### 2.10.2 Exterior Penalty Methods

For problems with inequality constraints, one straightforward idea is to convert a constrained optimization problem into a unconstrained one, which can be realized by applying a penalty for all violated constraints known as exterior penalty methods terms is written as equation 2.32 [10] [112]

$$PF(x) = f(x) + \vartheta_k \sum_{i=1}^{m} (g_i(x))^q \tag{2.32}$$

where $\vartheta_k$ is a positive penalty parameter, the exponent $q$ is a nonnegative constant, and the bracket function $(g_i(x))$is defined as follows

$$g_i(x) = max\{0, g_i(x)\} = \begin{cases} g(x) & if \quad g(x) > 0 \quad \text{constraint is violated} \\ 0 & if \quad g(x) \leq 0 \quad \text{constraint is satisfied} \end{cases}$$

***Example*** 2.10.2. Consider the problem

$$\begin{aligned} min \qquad & f(x_1) = 4x_1 + 3 \\ \text{subject to :} \qquad & -3 - x_1 \leq 0 \\ & x \geq 0 \end{aligned} \tag{2.33}$$

Let $\psi(x) = [\max\{g(x), 0\}]^2$ i.e, $\psi(x) = 0$, for $x \geq 3$ or $\psi(x) = (3-x)^2$, for $x < 3$. If $\psi(x) = 0$, then the optimal solution to minimize $F(x) = f(x) + \vartheta\psi(x)$ is at $x^* = -\infty$ and this is infeasible. Since this function is quadratic form, we can evaluate the minimizer by using first derivative, $PF'(x) = 4 - 2\vartheta(3-x) = 0$.
This implies that $x = 3 - \dfrac{2}{\vartheta}$ , which converges to $x^* = 3$ as $\vartheta \to \infty$. Therefore, the minimizer of the original problem is $x^* = 3$.

Table 2.2: Exterior penalty method.

| Iteration | $\mu$ | $x_k$ | Objective function |
|---|---|---|---|
| 1 | 10 | 2.7999999960 | 14.199999841369028 |
| 2 | 100 | 2.980 | 14.919999990439496 |
| 3 | 1000 | 2.997999990 | 14.991999972449538 |
| 4 | 10000 | 2.99979999 | 5.999599985208255 |
| 5 | 100000 | 2.99997999 | 14.99991997022185 |
| 6 | 1000000 | 2.999997990 | 14.999991970199893 |

In **Table** (2.2), we are got (2.999997990) and $f(2.999997990) = 14.999991970199893$ which is closed to the exact optimal solution $x^* = 3$.

## 2.11. Some Concepts of Multi-Objective Optimization Problems

When an optimization problem involves only one objective function, it is a single-objective optimization, the primary concept of multi-objective optimization is the multi-objective problem having several functions to be optimized (maximized or minimized). Multi-objective problems in which there is competition between objectives may have no single, unique optimal solution. Multi-objective optimization problems are also referred to as multicriteria or vector optimization problems. We can formulate a multi-objective optimization problem as an equation (2.34), find a decision variable that satisfies the given constraints, and optimizes a vector function whose components are objective functions. The general form of a multi-objective optimization problem can be mathematically stated as [41] [76]

$$(P_3)\begin{cases} min\ or\ max & f_m(x), m = 1, 2, ..., M \\ subject\ to & h_k(x) = 0, k = 1, 2, ..., K \\ & g_j(x) \geq 0, j = 1, 2, ..., J \\ & x_l \leq x \leq x_u \end{cases} \tag{2.34}$$

where $x$ is a vector of $n$ design variables given by $x = [x_1, x_2, ..., x_n]^T$. Finding a perfect multi-objective solution that simultaneously optimists each objective function is very impossible. It is appropriate to approach the solution of a multi-objective problem by investigating a set of solutions, each of which satisfies the objectives at a respectable level without being dominated by any other alternative [5] [19] optimum solution to a multi-objective problem's is Pareto optimality.

### 2.11.1 Pareto Optimality

A point, $x^* \in F(F$ is the feasible region) is Pareto optimal iff for every , $x \in F$, either $\forall_{i \in I}(f_i(x) = f_i(x^*))$, or there is at least one $i \in I$ such that $(f_i(x) > f_i(x^*))$ [81] [128].

### 2.11.2 Pareto Optimal Frontier

The Pareto optimal frontier $P^*$ is defined by the space in $R^n$ formed by all Pareto optimal solutions $P^* = \{x \in F | x$ is Pareto optimal$\}$. The Pareto optimal frontier is a set of optimal nondominated solutions, which may be infinite [49].

### 2.11.3 Pareto Front

The Pareto front $PF^*$ is defined by $PF^* = \{f(x) \in R^k | x \in P^*\}$. The Pareto front is the image set of the Pareto optimal frontier mapping into the objective space [33].
Combining all of the objective functions into a single objective function is the easiest way to solve a multi-objective optimization problem. Different objective functions can be combined into a single objective function using weighted sum method.

### 2.11.4 Weighted Sum Method

Let the following weighted sum of objective functions be given [96]

$$minf(x) = \sum_{i=1}^{k} w_i f_i(x)$$

where $k$ is the number of objective function, $w_i \geq 0$

$$\sum_{i=1}^{k} w_i = 1$$

This approach can be used since it is straightforward, simple to use, and computationally efficient for non-convex problems, on the other hand, as main disadvantages of the weighted sum method we can cite the difficulty to determine the appropriate weights, when we do not have enough information about the problem. As a result, the multi-objective optimization problem solution will depend on the coefficients used to combine the objective functions [5].

***Example*** 2.11.1. The Binh problem [35] Let us consider the following problem with two objective functions

$$minf_1(x_1, x_2) = x_1^2 + x_2^2$$

$$minf_2(x_1, x_2) = (x_1 + 5)^2 + (x_2 + 5)^2$$

$$\text{subject to } -5 \leq x_1, x_2 \leq 10$$

We begin by constructing a new objective function:

$$minF(x_1, x_2) = w_1(x_1^2 + x_2^2) + w_2((x_1 + 5)^2 + (x_2 + 5)^2) \tag{2.35}$$

with $w_1 + w_2 = 1, w_1, w_2 \geq 0$. So, we have

$$minF(x_1, x_2) = x_1^2 + x_2^2 + 10w_2(x_1 + x_2) + 50w_2 \tag{2.36}$$

$$\frac{\partial F(x_1, x_2)}{\partial x_1} = 2x_1 + 10w_2 \tag{2.37}$$

$$\frac{\partial F(x_1, x_2)}{\partial x_2} = 2x_2 + 10w_2 \tag{2.38}$$

Now, we look for the optimum of this objective function:

$$\frac{\partial^2 F(x_1, x_2)}{\partial x_1 x_2} = 2 \geq 0 \tag{2.39}$$

Equation (2.39) shows that a search for the points at which equations (2.37) and (2.38) are equal to zero will give us a minimum of this objective function.

$2x_1^* + 10w_2 = 0 \rightarrow x_1^* = -5w_2$

$2x_2^* + 10w_2 = 0 \rightarrow x_2^* = -5w_2$

We find the following objective functions:

$F(w_2) = 50w_2(1 - w_2)$

$f_1(w_2) = 50w_2^2$

$f_2(w_2) = 50(1 - w_2)^2$

We can compute some solutions of for some values of the $w_2$ weight. These results are gathered in **Table** (2.3), and results solutions is represented in **Figure** (2.28).

Table 2.3: Recapitulatory solutions for some values of the $w_2$.

| $w_2$ | $F(w_2)$ | $f_1(w_2)$ | $f_2(w_2)$ |
|---|---|---|---|
| 0 | 0 | 0 | 50 |
| 0.1 | 4.5 | 0.5 | 40.5 |
| 0.2 | 8 | 2 | 32 |
| 0.3 | 10.5 | 4.5 | 24.5 |
| 0.4 | 12 | 8 | 18 |
| 0.5 | 12.5 | 12.5 | 12.5 |
| 0.6 | 12 | 18 | 8 |
| 0.7 | 10.5 | 24.5 | 4.5 |
| 0.8 | 8 | 32 | 2 |
| 0.9 | 4.5 | 40.5 | 0.5 |
| 1 | 0 | 50 | 0 |

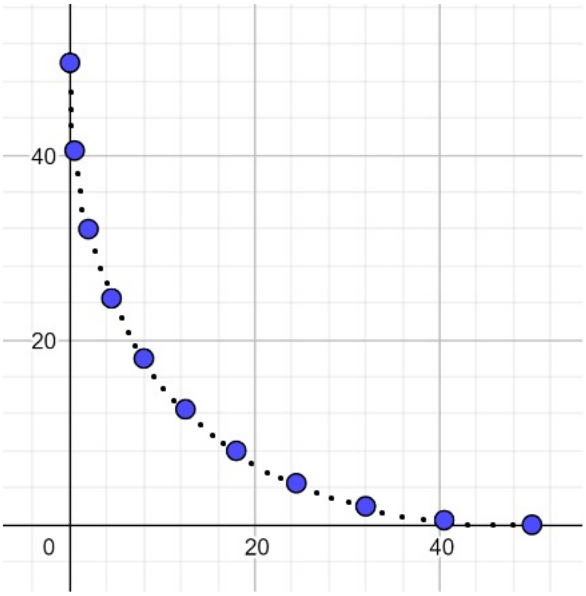



Figure 2.28: Curve solutions representation of the Binh problem.

## 2.12. Petri Net

The classical Petri net (PN) is a directed bipartite graph with two node types called places and transitions. The nodes are connected via directed arcs. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles. Places correspond to state variables of the system, while transitions correspond to actions that induce changes of states. A place may contain tokens that are represented by dots in the PN. The state of the PN is defined by its marking, which is represented by a vector $M_0 = \{I_1, I_2, ..., I_k\}$, where $I_k = M(p_k)$ is the number of tokens in place $p_k$. Here, $M(.)$ is a mapping function from a place to the number of tokens assigned to it [45].

**Definition 2.12.1.** [68] [90] A Petri net (PN) is formally defined by a quintuple $PN = (P, T, F, W, M_0)$ where

- $P = \{p_1, p_2, ..., p_n\}$ is a finite and non-empty set of places.
- $T = \{t_1, t_2, ..., t_m\}$is a finite and non-empty set of transitions.

- $F \subseteq (P \times T) \bigcup (T \times P)$ is the set of arcs which are between places and transitions.
- $W : F \to N$ is a weight function.
- $M_0 : F \to N$ called initial marking.

In **Figure** (2.29) a simple PN with all components. This PN has places $p_1, p_2, p_3, p_4, p_5$, and one transition $t_1, t_2, t_3, t_4$. $p_1$ has one token and $p_2, p_3, p_4, p_5$ has no token, that is $M(p_1) = 1$, $M(p_2) = M(p_3) = M(p_4) = M(p_5) = 0$.

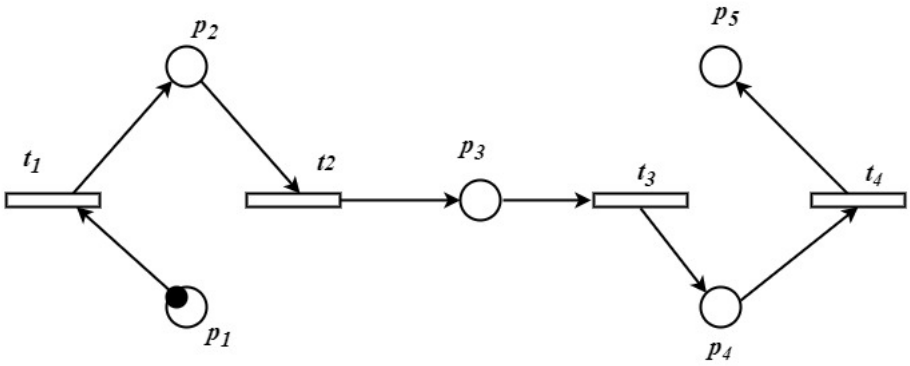



Figure 2.29: Petri net at its initial setting.

## 2.13. Rewriting Petri Nets as Directed Graphs

Petri nets are based on graphs and have some similarities to them. Transforming Petri nets into graphs opens up a whole set of new interesting possible implementations, can be obtained from Petri nets. There are different ways how to obtain graphs from Petri nets.

### 2.13.1 Places as Nodes and Transitions as Edges

The Petri net is turned into a graph in this instance. Places are replaced with nodes, and the transitions and their connecting arcs are replaced with a single edge. It must be made very clear that a transition for this type of conversion needs to have exactly both an output and an input arc. A suitable directed graph cannot be made if these prerequisites are not met. The only when locations have a single input do the translation of places into

nodes and connecting transitions to edges work successfully, points and exits with a single output can be converted with ease. For conversion to be possible, other more complicated Petri net classes must be simplified [109] [133]. The conversion techniques are illustrated with a few straightforward examples. A particular Petri net is used in **Figure** (2.30).

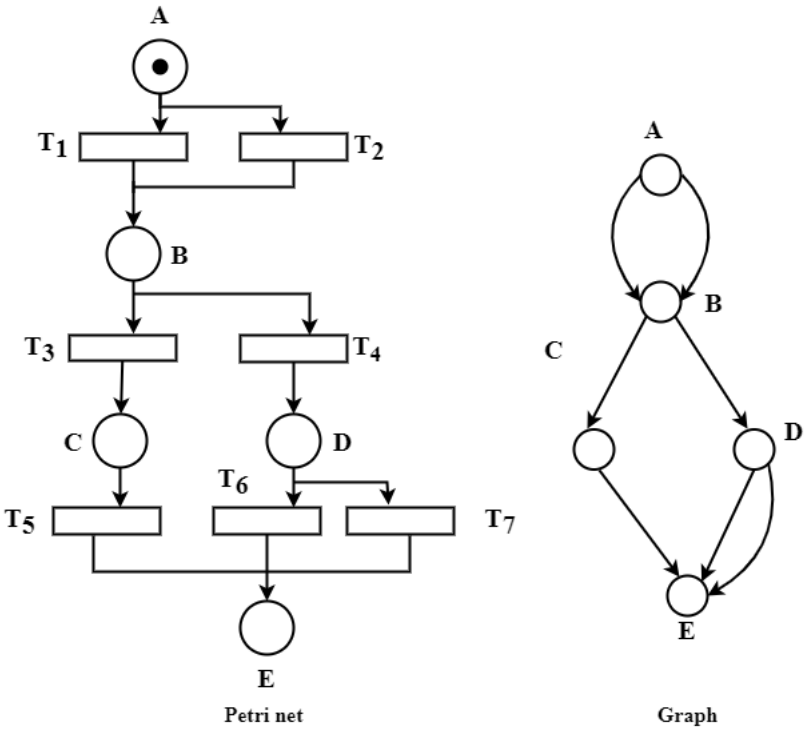



Figure 2.30: Petri net turned into a graph by places as nodes and transitions as edges.

### 2.13.2 Transitions as Nodes and Place as Edges

The Petri net is transformed into a graph and the linking input and output arcs are replaced with a single graph edge. The transitions are replaced with nodes. Only when locations have a single input point and a single output (exit) point do they effectively change into nodes and connect transitions to edges. It must be made extremely obvious for this type of conversion that each location in the Petri net must have exactly one input arc and one output arc. A suitable directed graph cannot be made if these prerequisites are not met [83] [133]. We provide straightforward example to understand conversion processes.A special Petri net is used in **Figure** (2.31).

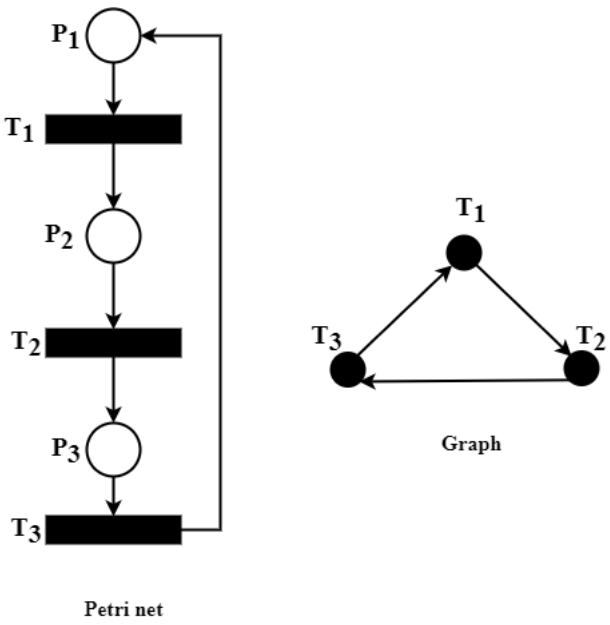



Figure 2.31: Petri net turned into a graph by transitions as nodes and places as edges.

## 2.14. Reactor Protection System(RPS)

An efficient nuclear power facility RPS is employed as model. Two subsystems. A and B, are present in the simplified system, and it is presumed that they are physically separate from one another. The structure is shown in the **Figure** (2.32). In this system, each subsystem uses two sets of independent sensors (SA and SB) used by each subsystem in this system are used to measure the corresponding measured values, which are then processed by the associated subsystem and compared to the set value. If the measured value is greater than the pre-set value, the bistable processor logic (BPL-A or BPL-B), in the subsystem generates a local trip signal. Each subsystem's signal processing procedure is the same. The two subsystems' local compliance logic (LCL-A and LCL-B) receives the generated local trip signals in turn . The received partial trip signal is voted on by each LCL in a two-out-of-two fashion. If two shutdown processors (RT-A1, RT-A2, RT-B1, RT-B2) attached to a specific LCL receive and process two partial trip signals from separate subsystems, the LCL will produce the associated electrical signals. A relay power-off signal is sent to the associated relay contacts by each shutdown logic processor

once it has converted the electrical signal (M1, M2, M3 and M4). M1 and M2 take two out of one vote, M3 and M4 take two out of one vote [151].

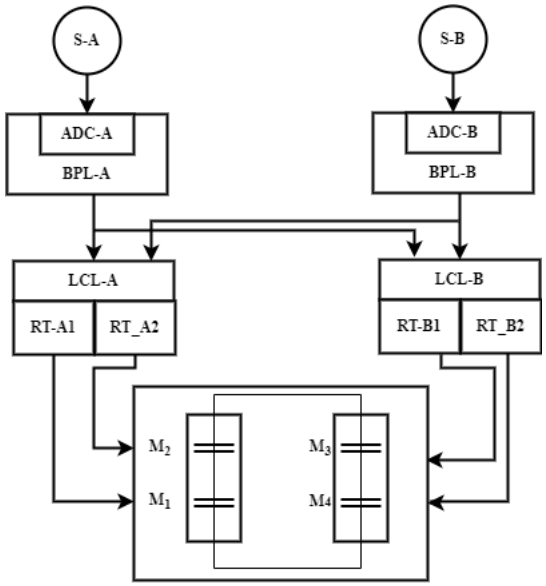



Figure 2.32: Structure of the general shutdown system's safety controls [151].

### 2.14.1 Modular Petri Net of RPS System

Modular PN according to function, splits the system into input, decision, and output modules. For systems that can be repaired, a state machine module is introduced to ascertain the module's condition and determine if data can be passed to the follow in module [151]. Model simplified as displayed in **Figure** (2.33).

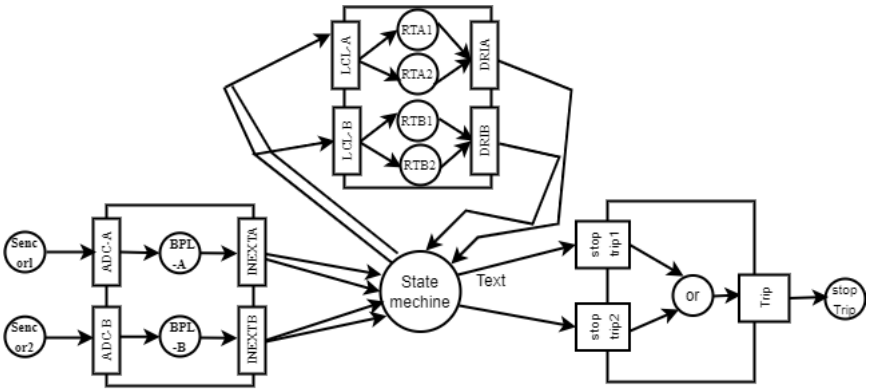



Figure 2.33: Modular Petri net model of shutdown system [151].

Each module is streamlined and integrated to prevent the explosion of state space, and the finished PN model is displayed in **Figure** (2.34).

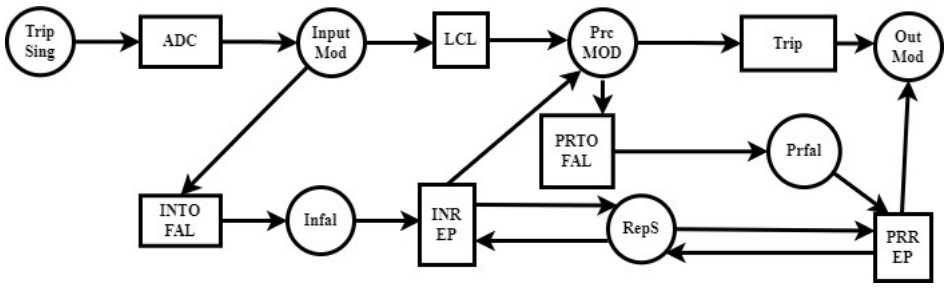



Figure 2.34: Simplified modular PN model of shutdown system [151].

## 2.15. Inverse Square Law

The inverse square law, also known as the basic law of radiometry, defines the relationship between the irradiance $E_e$ produced by a point light source, viewed at a distance such that its dimensions are negligible, and the same distance. The law states that the irradiance $E_e$ of the luminous flux through the element of area $dA$ is proportional to the intensity and radiant $I_e$ and varies inversely proportional to the square of the distance $r$ [102].

$$E_e = \frac{I_e}{r^2}$$

For example see section 4.2.1.

# CHAPTER 3

# SOME TECHNIQUES TO FIND RELIABILITY POLYNOMIAL OF A GIVEN NETWORK

## 3.1. Introduction

In this chapter, are discuss to extract the minimal path sets and the minimal cut sets, and we find the reliability polynomial of a network by using two techniques, which are sum of disjoint products technique and minimal cut technique.

## 3.2. Shutdown Network Case Study

Simplified modular Petri net of the shutdown system shown in subsection 2.14.1 and we turned **Figure** (2.34) into a network in this instance places are replaced with nodes, and the transitions and their connecting arcs are replaced with a single edge see in subsection 2.13. We get the network shown in **Figure** (3.1).

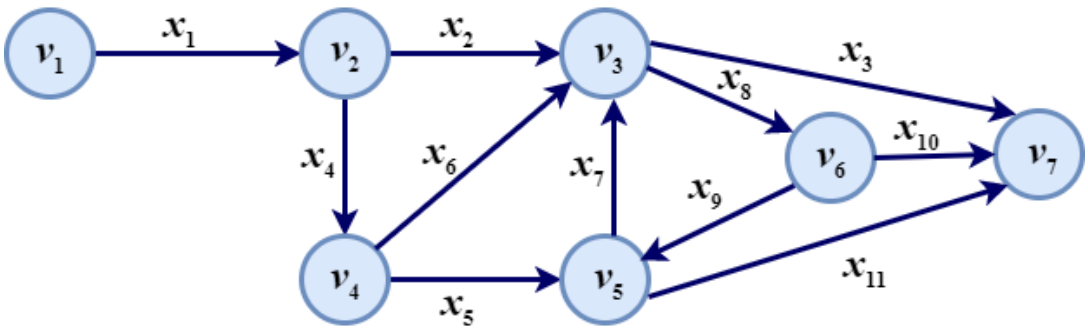



Figure 3.1: Network of simplified the shutdown system.

## 3.3. Using Some Techniques to Find Minimal Path Sets of the Shutdown Network

In this section some techniques are applied as follows:

### 3.3.1 Rai and Aggarwal's Algorithm

The algorithm can be broken down into four steps [105] [118]:

- Step 1: Create a connection matrix for graph of the network $[C]$.
- Step 2: Make all diagonal elements 1.
- Step 3: Remove the first columns and last rows and define the success determinant for the system $|\mathcal{D}|$.
- Step 4: Using the Boolean sum and product operations, expand the determinant $|\mathcal{D}|$.

The algorithm applies to the case study shutdown network in **Figure** (3.1). First we find the connection matrix equation 3.1 and make all diagonal elements 1.

$$C = \begin{bmatrix} 1 & x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & x_2 & x_4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & x_8 & x_3 \\ 0 & 0 & x_6 & 1 & x_5 & 0 & 0 \\ 0 & 0 & x_7 & 0 & 1 & 0 & x_{11} \\ 0 & 0 & 0 & 0 & x_9 & 1 & x_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

And remove the first columns and last rows and create success determinant.

$$\mathcal{D} = \begin{vmatrix} x_1 & 0 & 0 & 0 & 0 & 0 \\ 1 & x_2 & x_4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & x_8 & x_3 \\ 0 & x_6 & 1 & x_5 & 0 & 0 \\ 0 & x_7 & 0 & 1 & 0 & x_{11} \\ 0 & 0 & 0 & x_9 & 1 & x_{10} \end{vmatrix} \tag{3.2}$$

$$\begin{aligned} =& x_1x_2x_8(x_{10} - x_{11}x_9) - x_1x_2x_3 + x_1x_4x_5x_{11} - x_1x_4x_8x_6(x_{10} - x_9x_{11}) + \\ & x_1x_4x_8x_5x_7x_{10} + x_1x_4x_3x_6 - x_1x_4x_3x_5x_7 + x_1x_4x_8x_5x_7x_{10} + x_1x_4x_3x_6 \\ & - x_1x_4x_3x_5x_7 \end{aligned} \tag{3.3}$$

Expand equation (3.3) by the Boolean sum and product operations

$$\begin{aligned} S =& x_1x_2x_8x_{10} \vee x_1x_2x_8x_9x_{11} \vee x_1x_2x_3 \vee x_1x_4x_5x_{11} \vee x_1x_4x_8x_6x_{10} \\ & \vee x_1x_4x_8x_9x_{11} \vee x_1x_4x_8x_5x_7x_{10} \vee x_1x_4x_6x_3 \vee x_1x_4x_3x_5x_7 \end{aligned} \tag{3.4}$$

From equation (3.4) we get the minimal paths are

$$\{x_1, x_2, x_3\}, \{x_1, x_2, x_8, x_{10}\}, \{x_1, x_4, x_6, x_3\}, \{x_1, x_4, x_5, x_{11}\}, \{x_1, x_2, x_8, x_9, x_{11}\},$$
$$\{x_1, x_4, x_6, x_8, x_{10}\}, \{x_1, x_4, x_5, x_7, x_3\} \{x_1, x_4, x_6, x_8, x_9, x_{11}\} \text{ and } \{x_1, x_4, x_5, x_7, x_8, x_{10}\}$$

### 3.3.2 Matrix Multiplication

The process starts by iteratively multiplying the network graph's connection matrix by nodes, the most extensive path will be of size $(p-1)$. Besides, at each multiplication, one must apply the laws of Boolean algebra. The path sets of varied order from one node to another would be represented by the elements in the matrix of each order correspond to the given the specified node pair. Term collection with the number of minimal paths sets listed

by this approach would be reduced by the removal of redundant terms [20] [27] [78]. Apply this technique to find the minimal paths on the network highlighted in this dissertation in the **Figure** (3.1) using an equation 3.1, get

$$C^2 = \begin{bmatrix} 0 & 0 & x_1x_2 & x_1x_4 & 0 & 0 & 0 \\ 0 & 0 & x_4x_6 & 0 & x_4x_5 & x_2x_8 & x_2x_3 \\ 0 & 0 & 0 & 0 & x_8x_9 & 0 & x_8x_{10} \\ 0 & 0 & x_5x_7 & 0 & 0 & x_6x_8 & x_3x_6 + x_5x_{11} \\ 0 & 0 & 0 & 0 & 0 & x_7x_8 & x_7x_3 \\ 0 & 0 & x_7x_9 & 0 & 0 & 0 & x_9x_{11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C^3 = \begin{bmatrix} 0 & 0 & x_1x_4x_6 & 0 & x_1x_4x_5 & 0 & x_1x_2x_3 \\ 0 & 0 & x_4x_5x_7 & 0 & x_2x_8x_9 & x_4x_6x_8 & x_2x_8x_{10} + x_4x_6x_3 + x_4x_5x_{11} \\ 0 & 0 & x_7x_8x_9 & 0 & 0 & 0 & x_8x_9x_{11} \\ 0 & 0 & 0 & 0 & x_6x_8x_9 & x_5x_7x_8 & x_6x_8x_{10} + x_5x_7x_3 \\ 0 & 0 & 0 & 0 & x_7x_8x_9 & 0 & x_7x_8x_{10} \\ 0 & 0 & 0 & 0 & 0 & x_7x_8x_9 & x_9x_7x_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$C^4 = \begin{bmatrix} 0 & 0 & x_1x_4x_5x_7 & 0 & x_1x_2x_8x_9 & x_1x_4x_6x_8 & x_1x_2x_8x_{10} + x_1x_4x_6x_3 + x_1x_4x_5x_{11} \\ 0 & 0 & x_6x_7x_8x_9 & 0 & x_4x_6x_8x_9 & x_4x_5x_7x_8 & x_2x_8x_9x_{11} + x_4x_6x_8x_{10} + x_4x_5x_7x_3 \\ 0 & 0 & 0 & 0 & 0 & x_7x_8x_9 & x_7x_8x_9x_3 \\ 0 & 0 & x_6x_7x_8x_9 & 0 & x_5x_7x_8x_9 & 0 & x_6x_8x_9x_{11} + x_5x_7x_8x_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & x_7x_8x_9x_{11} \\ 0 & 0 & x_7x_8x_9 & 0 & x_7x_8x_9 & 0 & x_7x_8x_9x_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$C^5 = \begin{bmatrix} 0 & 0 & x_1x_2x_7x_8x_9 & 0 & x_1x_4x_6x_8x_9 & x_1x_4x_5x_7x_8 & y_1 \\ 0 & 0 & x_4x_6x_7x_8x_9 & 0 & x_4x_5x_7x_8x_9 & x_2x_4x_6x_8 + x_4x_5x_7x_8 & y_2 \\ 0 & 0 & 0 & 0 & 0 & x_7x_8x_9 & x_8x_9x_7x_3 \\ 0 & 0 & x_5x_7x_8x_9 & 0 & 0 & x_6x_7x_8x_9 & x_6x_7x_8x_9x_3 + x_5x_7x_3x_{11} \\ 0 & 0 & 0 & 0 & 0 & x_7x_8x_9 & x_7x_8x_9x_3 \\ 0 & 0 & x_7x_8x_9 & 0 & x_7x_8x_9 & 0 & x_7x_8x_9x_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where $y_1 = x_1x_2x_8x_9x_{11} + x_1x_4x_6x_8x_{10} + x_1x_4x_5x_7x_3$,

$y_2 = x_2x_7x_8x_9x_3 + x_4x_6x_8x_9x_{11} + x_4x_5x_7x_8x_{10}$

$$C^6 = \begin{bmatrix} 0 & 0 & x_1x_4x_6x_7x_7x_9 & 0 & x_1x_4x_5x_7x_8x_9 & x_1x_2x_4x_6x_8 + x_1x_4x_5x_7x_8 & z_1 \\ 0 & 0 & x_4x_5x_6x_7x_8x_9 & 0 & 0 & x_2x_7x_8x_9 + x_4x_6x_7x_8x_9 & z_2 \\ 0 & 0 & x_7x_8x_9 & 0 & x_7x_8x_9 & 0 & x_8x_9x_7x_3 \\ 0 & 0 & 0 & 0 & 0 & x_6x_7x_8x_9 + x_5x_7x_8x_9 & z_3 \\ 0 & 0 & 0 & 0 & 0 & x_7x_8x_9 & x_7x_8x_9x_3 \\ 0 & 0 & 0 & 0 & 0 & x_7x_8x_9 & x_7x_8x_9x_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where

$z_1 = x_1x_2x_3x_7x_8x_9 + x_1x_4x_6x_8x_9x_{11} + x_1x_2x_3x_7x_8x_9 + x_1x_4x_6x_8x_9x_{11} + x_1x_4x_5x_7x_8x_{10}$

$+ x_1x_4x_5x_7x_8x_{10}$.

$z_2 = x_2x_8x_9x_7x_3 + x_4x_6x_7x_8x_9x_3 + x_4x_5x_7x_9x_{11}$,and $z_3 = x_7x_8x_9x_3 + x_5x_7x_8x_9x_3$.

Minimal path sets between source node and sink node $(1, 7)$ are not find path from element $(1, 7)$ in $C^2$ and minimal paths in $C^3$ is $\{x_1, x_2, x_3\}$ and minimal paths in $C^4$ are $\{x_1, x_2, x_8, x_{10}\}$ and,$\{x_1, x_4, x_6, x_3\}$ and $\{x_1, x_4, x_5, x_{11}\}$ and minimal paths in $C^5$ are $\{x_1, x_2, x_8, x_9, x_{11}\}$ and $\{x_1, x_4, x_6, x_8, x_{10}\}$ and $\{x_1, x_4, x_5, x_7, x_3\}$ and minimal paths in $C^6$ are $\{x_1, x_4, x_6, x_8, x_9, x_{11}\}$ and $\{x_1, x_4, x_5, x_7, x_8, x_{10}\}$ but $\{x_1, x_2, x_3, x_7, x_8, x_9\}$ is not minimal path in $C^6$.

### 3.3.3 Path Set Enumeration

The algorithm technique for searching for all path sets between source and sink. Below is a list of all the many steps that make up path set enumeration [28] [56]:

1. Create the network's adjacency matrix first.

2. It begins with the source node by looking for nonzero elements in the adjacency

matrix in the row corresponding to the source node.

3. It builds a new vector with the source node number for every nonzero element found at a certain point in that row.

4. In this way, the number of new vectors produced is equal to the number of nonzero elements discovered throughout the search.

5. Using the same technique as for the row corresponding to the last element added in each of these vectors, select the row of the adjacency matrix corresponding to the most recently added entry.

6. This is done until the sink node is reached, at which point the path sets are retraced.

Technique applies to the case study shutdown network in **Figure** (3.1) by finding a adjacency matrix equation (3.5) and needed to impose the symbols. It should be kept in a matrix $[D]$,$[B]$ referred to as the matrix to source node's assigned row in the adjacency matrix is $[DD]$ contains $[D]$ with the non-zero positions of $[B]$ concatenated to each entry of $[D]$, $[BB]$ is $[B]$ temporary storage.

$$\begin{array}{c|ccccccc} \text{nodes} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \\ \hline v_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ v_4 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ v_5 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_6 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ v_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \tag{3.5}$$

The steps of the algorithm are shown in the **Table** (3.1)

Table 3.1: Steps to algorithm for the network of Figure 3.1

| Iteration no. | $[D]$ | $[B]$ | $[DD]$ | $[BB]$ |
|---|---|---|---|---|
| 1 | 1 | 0100000 | 12 | 0011000 |
| 2 | 12 | 0011000 | 123 | 0000011 |
| | | | 124 | 0010100 |
| 3 | 123 | 0000011 | 1236 | 0000101 |
| | 124 | 0010100 | **1237** | **0000010** |
| | | | 1243 | 0000011 |
| | | | 1245 | 0010001 |
| 4 | 1236 | 0000101 | 12365 | 0000001 |
| | 1243 | 0000011 | **12367** | **0000100** |
| | 1245 | 0010001 | 12436 | 0000101 |
| | | | **12437** | **0000010** |
| | | | 12453 | 0000011 |
| | | | **12457** | **0010000** |
| 5 | 12365 | 0000001 | **123657** | **0000000** |
| | 12436 | 0000101 | 124365 | 0000001 |
| | 12453 | 0000011 | **124367** | **0000100** |
| | | | 124536 | 0000001 |
| | | | **124537** | **0000010** |
| 6 | 124365 | 0000001 | **1243657** | **0000000** |
| | 124536 | 0000001 | **1245367** | **0000000** |

With source node 1, we begin append $[D]$ with these values and store it in $[DD]$ because $[D]$ has non-zero values at positions second. One iteration of the search is finished when we remove the rows of the adjacency matrix that correspond to the final element in each row of [DD], we mean by the second rows by setting elements the first and second then, $[D]$ is updated by $[DD]$, and $[B]$ is updated by $[BB]$, we perform two checks, the first is to determine whether any rows of $[DD]$ last 's element has a sink node, the second is to determine whether all entries in that rows of $[BB]$ are zero. We halt the procedure when

the iterative count is one less than the overall number of network nodes.

$$1 \to 2 \to 3 \to 7 = \{x_1, x_2, x_3\}$$

$$1 \to 2 \to 3 \to 6 \to 7 = \{x_1, x_2, x_8, x_{10}\}$$

$$1 \to 2 \to 4 \to 3 \to 7 = \{x_1, x_4, x_6, x_3\}$$

$$1 \to 2 \to 4 \to 5 \to 7 = \{x_1, x_4, x_5, x_{11}\}$$

$$1 \to 2 \to 3 \to 6 \to 5 \to 7 = \{x_1, x_2, x_8, x_9, x_{11}\}$$

$$1 \to 2 \to 4 \to 3 \to 6 \to 7 = \{x_1, x_4, x_6, x_8, x_{10}\}$$

$$1 \to 2 \to 4 \to 5 \to 3 \to 7 = \{x_1, x_4, x_5, x_7, x_3\}$$

$$1 \to 2 \to 4 \to 3 \to 6 \to 5 \to 7 = \{x_1, x_4, x_6, x_8, x_9, x_{11}\}$$

$$1 \to 2 \to 4 \to 5 \to 3 \to 6 \to 7 = \{x_1, x_4, x_5, x_7, x_8, x_{10}\}$$

### 3.3.4 Node-Child Matrix Algorithm

Algorithms are designed to determine the minimal paths in a network. The node-child matrix see in definition (2.3.4) is used in these algorithms. The first algorithm for acyclic network and second algorithm is the extended of first algorithm using no constraints (acyclic or not acyclic) [50], and these techniques employ search and to produce all of the minimal pathways, swap procedures, on the node-child matrix, search technique will go through all of use the node-child matrix to find the available nodes in the current path, to discover the next child node. If a destination node is reached at each level, the search will come to a conclusion, and the complete path will be saved in the path matrix as a row. Otherwise, use the branch node list. The current node is chosen from among the child node of the final node of a branch (other than the one that has already been processed). At the current node, the search procedure starts over first algorithm of acyclic network.

1. Select an origin node $i$ from $S$, set it as the current node, and then remove it from $S$.

2. To count the nonzero elements in the node-child matrix's $i_th$ row, say $k_i$. If $k_i = 0$, the node $i$ is a destination node, and the path is complete. On path matrix $P$, row l, save the whole path from origin to destination, add one to $l$, and go to step 3. Otherwise, if $k_i \leq 2$ ( if $k = l$, node $I$ is not a branch node), add $I$ to the list of branch nodes, make $B(i, l)$ the current node, $B(i, l) \rightarrow i$ and return to step 2.

3. If the list of branch nodes is empty, you've located all pathways that begin with the letter $I$ go to step 5 if not, proceed to step 4.

4. Process of swapping by taking the initial element $q$, make $B(q, j)$ the current node, list from the list of branch nodes as the current node. $(i \text{ , } B(q, j))$. Remove $q$ from the list of branch nodes, and set $j = 2$ if $B(q, j + 1) = 0$ or $j = p$, otherwise add one to $j$, go to the second stage.

5. Break out of the algorithm when in s, there are no more nodes, and all minimal path have been established. Add one to $j$ if it's not already there step 2.

Second algorithm for general network. The stack data structure is employed by the algorithm. It is designed to recognize directed cycles and not record paths that contain cycles, allowing the top element to be removed. This algorithm uses a search procedure to produce the path matrix. **Figure** (3.2) shows the algorithm's flowchart, the search procedure starts over algorithm for general network.

- **Case 1**: There is a child in this node, will choose a child of the current node and check to see if it is in the current path. We'll extend it and make the child the current node if it's not on the current path (find $(W, B(s, j)$ have a false value). The current node's child will be chosen again ( by adding one to $j$). If no additional offspring are found, we'll use a technique similar to the swap algorithm in the first algorithm (step 4, 5).

- **Case 2**: There are no children in this node. Using the definition of the target node, we were able to find a minimal path. Every time we arrive at a path's destination

node, we'll take one step backwards and inspect every other possible route to get there (with step of 2 and 3). Because the number of nodes is finite, the search process will reach the second instance after a finite number of steps. As a result, the algorithm's output will almost surely comprise all minimum path.

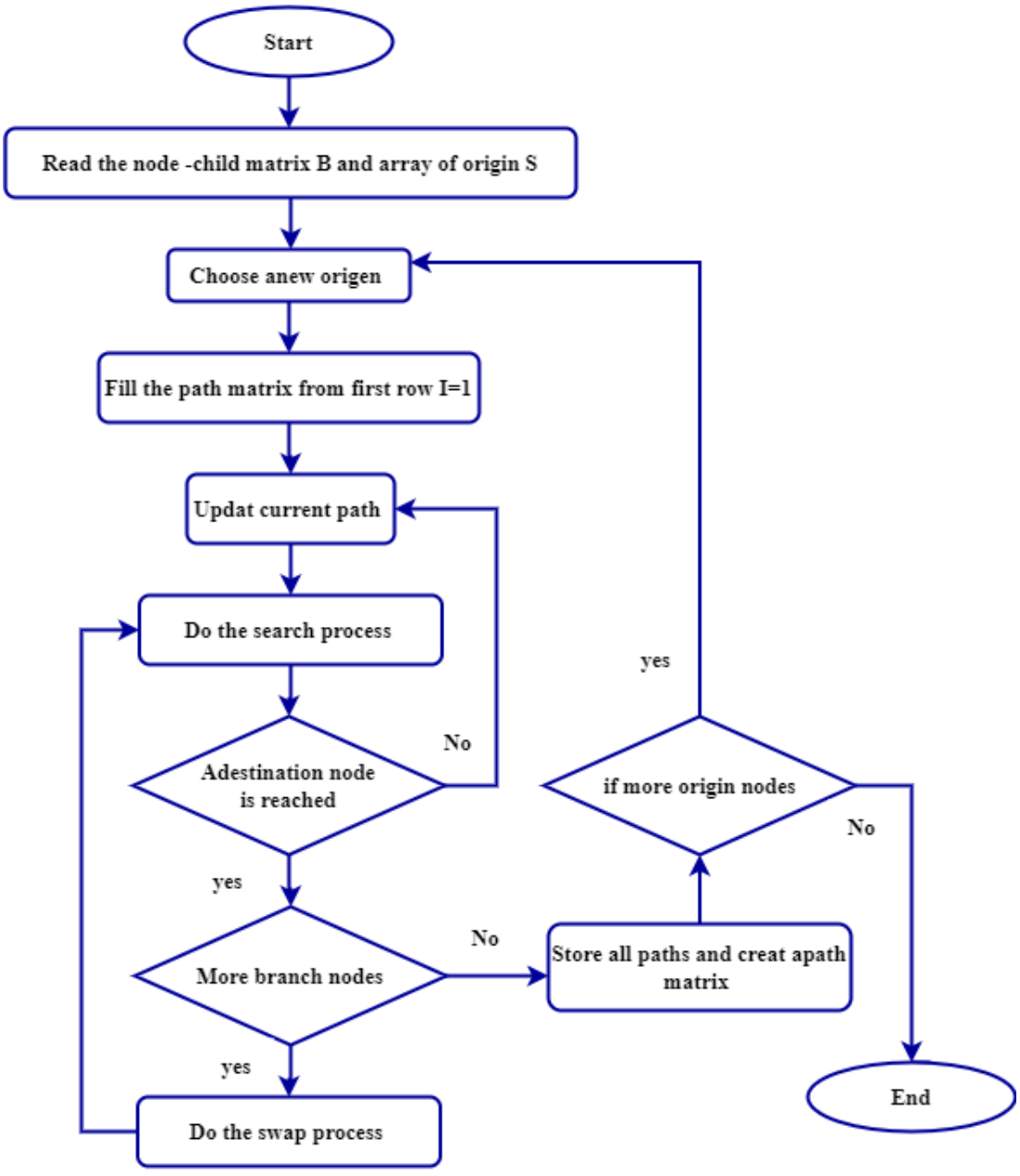



Figure 3.2: The node-child matrix algorithm flowchart

We use the node-child matrix algorithm applies to the case study shutdown network in **Figure** (3.1). First, we find an matrix

$$M = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{bmatrix} v_2 & 0 \\ v_3 & v_4 \\ v_6 & v_7 \\ v_3 & v_5 \\ v_3 & v_7 \\ v_5 & v_7 \\ 0 & 0 \end{bmatrix}. \tag{3.6}$$

We set $k = 1$ and origin node $S = v_1$ have stack $\mathcal{W} = \{v_1\}$ and **Table** (3.2). The rest of the application steps show the the node-child matrix algorithm for general network.

Table 3.2: Steps to node-child matrix algorithm for the network of **Figure** (3.1).

| $k$ | $j$ | $S$ | $M(s,j)$ | $\mathcal{W}$ | minimal path |
|---|---|---|---|---|---|
| 1 | 1 | $v_1$ | $v_2$ | $\{v_1, v_2\}$ | |
| | 1 | $v_2$ | $v_3$ | $\{v_1, v_2, v_3\}$ | |
| | 1 | $v_3$ | $v_6$ | $\{v_1, v_2, v_3, v_6\}$ | |
| | 1 | $v_6$ | $v_5$ | $\{v_1, v_2, v_3, v_6, v_5\}$ | |
| | 1 | $v_5$ | $v_3$ | $\{v_1, v_2, v_3, v_6, v_5\}$ | |
| | 2 | $v_5$ | $v_7$ | $\{v_1, v_2, v_3, v_6, v_5, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_3, v_6, v_5, v_7\}$ |
| 2 | 2 | $v_5$ remove | | $\{v_1, v_2, v_3, v_6\}$ | |
| | 2 | $v_6$ | $v_7$ | $\{v_1, v_2, v_3, v_6, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_3, v_6, v_7\}$ |
| 3 | 2 | $v_6$ remove | | $\{v_1, v_2, v_3\}$ | |
| | 2 | $v_3$ | $v_7$ | $\{v_1, v_2, v_3, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_3, v_7\}$ |

| | | | | | |
|---|---|---|---|---|---|
| 4 | 2 | $v_3$ remove | | $\{v_1, v_2\}$ | |
| | 2 | $v_2$ | $v_4$ | $\{v_1, v_2, v_4\}$ | |
| | 1 | $v_4$ | $v_3$ | $\{v_1, v_2, v_4, v_3\}$ | |
| | 1 | $v_3$ | $v_6$ | $\{v_1, v_2, v_4, v_3, v_6\}$ | |
| | 1 | $v_6$ | $v_5$ | $\{v_1, v_2, v_4, v_3, v_6, v_5\}$ | |
| | 1 | $v_5$ | $v_3$ | $\{v_1, v_2, v_4, v_3, v_6, v_5\}$ | |
| | 2 | $v_5$ | $v_7$ | $\{v_1, v_2, v_4, v_3, v_6, v_5, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_4, v_3, v_6, v_5, v_7\}$ |
| 5 | 2 | $v_5$ remove | | $\{v_1, v_2, v_4, v_3, v_6\}$ | |
| | 2 | $v_6$ | $v_7$ | $\{v_1, v_2, v_4, v_3, v_6, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_4, v_3, v_6, v_7\}$ |
| 6 | 2 | $v_6$ remove | | $\{v_1, v_2, v_4, v_3\}$ | |
| | 2 | $v_3$ | $v_7$ | $\{v_1, v_2, v_4, v_3, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_4, v_3, v_7\}$ |
| 7 | 2 | $v_3$ remove | | $\{v_1, v_2, v_4\}$ | |
| | 2 | $v_4$ | $v_5$ | $\{v_1, v_2, v_4, v_5\}$ | |
| | 1 | $v_5$ | $v_3$ | $\{v_1, v_2, v_4, v_5, v_3\}$ | |
| | 1 | $v_3$ | $v_6$ | $\{v_1, v_2, v_4, v_5, v_3, v_6\}$ | |
| | 1 | $v_6$ | $v_5$ | $\{v_1, v_2, v_4, v_5, v_3, v_6\}$ | |
| | 2 | $v_6$ | $v_7$ | $\{v_1, v_2, v_4, v_5, v_3, v_6, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_4, v_5, v_3, v_6, v_7\}$ |
| 8 | 2 | $v_6$ remove | | $\{v_1, v_2, v_4, v_5, v_3\}$ | |
| | 2 | $v_3$ | $v_7$ | $\{v_1, v_2, v_4, v_5, v_3, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_4, v_5, v_3, v_7\}$ |
| 9 | 2 | $v_3$ remove | | $\{v_1, v_2, v_4, v_5\}$ | |
| | 2 | $v_5$ | $v_7$ | $\{v_1, v_2, v_4, v_5, v_7\}$ | |
| | 1 | $v_7$ | 0 destination node | | $\{v_1, v_2, v_4, v_5, v_7\}$ |
| 10 | 2 | $v_5$ remove | | $\{v_1, v_2, v_4\}$ | |
| | 2 | $v_4$ remove | | $\{v_1, v_2\}$ | |
| | 2 | $v_2$ remove | | $\{v_1\}$ | |
| | 2 | $v_1$ | 0 | $\{v_1\}$ | stop |

In **Table** (3.2) we note when $k = 1$ and $S = v_5, j = 1, M(v_5, j) = v_3$, element $v_3$ have in the stack and we remove it and take element from the stack, we shall proceed, when

$s = v_7, j = 1, M(v_7, j) = 0$. Our destination node we get the first minimal path. Also in the set $k = 2, 3, 4, 5, 6, 7, 8, 9$, and $j = 2$, we remove $v_5, v_6, v_3, v_5, v_6, v_3, v_6, v_3$ respectively and take an element from the stack, when $M(v_7, j) = 0$ will get minimal path sets . Now we have $k = 10$ and remove $S = v_5, v_4, v_2$, and $j = 2$ we take an element from the stack and continue process we get $S = v_1, j = 2$ and $M(v_1, j) = 0$ is stop algorithm.

The path matrix is

$$P = \begin{bmatrix} v_1 & v_2 & v_3 & v_6 & v_5 & v_7 & 0 \\ v_1 & v_2 & v_3 & v_6 & v_7 & 0 & 0 \\ v_1 & v_2 & v_3 & v_7 & 0 & 0 & 0 \\ v_1 & v_2 & v_4 & v_3 & v_6 & v_5 & v_7 \\ v_1 & v_2 & v_4 & v_3 & v_6 & v_7 & 0 \\ v_1 & v_2 & v_4 & v_3 & 0 & v_7 & 0 \\ v_1 & v_2 & v_4 & v_5 & v_3 & v_6 & v_7 \\ v_1 & v_2 & v_4 & v_5 & v_3 & v_7 & 0 \\ v_1 & v_2 & v_4 & v_5 & v_7 & 0 & 0 \end{bmatrix}$$

We get minimal path sets

$$v_1 \to v_2 \to v_3 \to v_6 \to v_5 \to v_7 = \{x_1, x_2, x_8, x_9, x_{11}\}$$

$$v_1 \to v_2 \to v_3 \to v_6 \to v_7 = \{x_1, x_2, x_8, x_{10}\}$$

$$v_1 \to v_2 \to v_3 \to v_7 = \{x_1, x_2, x_3\}$$

$$v_1 \to v_2 \to v_4 \to v_3 \to v_6 \to v_5 \to v_7 = \{x_1, x_4, x_6, x_8, x_9, x_{11}\}$$

$$v_1 \to v_2 \to v_4 \to v_3 \to v_6 \to v_7 = \{x_1, x_4, x_6, x_8, x_{10}\}$$

$$v_1 \to v_2 \to v_4 \to v_3 \to v_7 = \{x_1, x_4, x_6, x_3\}$$

$$v_1 \to v_2 \to v_4 \to v_5 \to v_3 \to v_6 \to v_7 = \{x_1, x_4, x_5, x_7, x_8, x_{10}\}$$

$$v_1 \to v_2 \to v_4 \to v_5 \to v_3 \to v_7 = \{x_1, x_4, x_5, x_7, x_3\}$$

$$v_1 \to v_2 \to v_4 \to v_5 \to v_7 = \{x_1, x_4, x_5, x_{11}\}$$

### 3.3.5 Two Terminal Nodes Algorithm

This algorithm is based on a technique for counting all of a network's minimal path (has a loop or parallel linkages, whether it's orientated or not). This technique involves looking at networks from two different perspectives, one from the sink **t** and the other from the source **s**. Starting with the two terminal nodes, the algorithm gets started (source **s** and sink **t**). Since this technique allows us to continue by two edges at a time, at each iteration, looked for a link between the last node of the traversal that began with the source and the last node of the traversal that began with the sink [87]. Here are some of definitions that are needed.

**Definition 3.3.1.** push: Use this tool to add a new element to a collection.

**Definition 3.3.2.** pop: This tool returns and removes a collection element.

Algorithm steps

1. The supplied network source and sink nodes should be entered.
2. generate a $To - Treat$ that contains $\{[s, t, \{s\}, \{t\}]\}$ for the first time.
3. Using the tool pop, take an element from the $To - Treat$ collection and transform it into $\left[A_1, A_2, \bar{S}, \bar{T}\ \right]$ (despite the fact that the $To - Treat$ isn't empty). The set $\bar{S}$ connects $A_1$ with the sink node $t$, while the set $\bar{T}$ connects $A_2$ with the source node $s$.
4. $Adj - A_1 = \{a \in V \mid (A_1, a) \in E \text{ and } a \neq A_2,\ a \notin \bar{S}\ ,\ a \notin \bar{T}\}$.
5. $Adj - A_2 = \{a \in V \mid (a, A_2) \in E \text{ and } a \neq A_1\ ,\ a \notin \bar{S},\ a \notin \bar{T}\}$.

6. We take the next element from the set $To - Treat$ if one of the two sets $Adj - A_1$ or $Adj - A_2$ is empty. Otherwise , we create all two-order combinations of $Adj - A_1$ and $Adj - A_2$ with $\{(n, m) \mid n \in Adj - A_1 \text{ and } m \in Adj - A_2\}$.

7. By concatenating the sets $\bar{S}, n, m$, and $\bar{T}$, the algorithm discovered a new path, if there is a link between the two nodes $(n, m)$ or $n = m$. We next verify if $n \neq m$, and if they aren't equal, we use the function push to add the new element $[n, m, \bar{S} \cup \{n\}, \{m\} \cup \bar{T}]$ to $To - Treat$.

This algorithm applies to the case study shutdown network in **Figure** (3.1), and let $v_1 = s, v_7 = t$, we find initialization of the variables $To - Treat = \{[s, t, \{s\}, \{t\}]\}$.

- **Iteration 1:**

  To - Treat is not empty

  $$\left[A_1, A_2, \bar{S}, \bar{T}\right] \leftarrow \text{pop( To - Treat)}$$
  $$A_1 = s, A_2 = t, \bar{S} = \{s\}, \bar{T} = \{t\}, Adj - A_1 = \{2\}, Adj - A_2 = \{3, 5, 6\}$$

  All potential combinations have been considered $[(2, 3), (2, 5), (2, 6)]$, for $(2, 3)$ there's a link between node 2 and node 3. We are given a new path $MP_1 = \{s, 2, 3, t\}$ and push $[2, 3, \{s, 2\}, \{3, t\}]$, $(2, 5)$ there is no connection between the two nodes then push $[2, 5, \{s, 2\}, \{5, t\}]$ and $(2, 6)$ there is no connection between the two nodes then push $[2, 6, \{s, 2\}\{6, t\}]$.

- **Iteration 2:**

  $A_1 = 2, A_2 = 3, \bar{S} = \{s, 2\}, \bar{T} = \{3, t\}, Adj - A_1 = \{4\}, Adj - A_2 = \{4, 5\}$. All potential combinations have been considered $[(4, 4), (4, 5)]$,$(4, 4)$ since $4 = 4$, we get new path $MP_2 = \{s, 2, 4, 3, t\}$, $(4, 5)$ there is a link between nodes 4 and 5. We are given a new path $MP_3 = \{s, 2, 4, 5, 3, t\}$ and push $[4, 5, \{s, 2, 4\}, \{5, 3, t\}]$.

- **Iteration 3**:

  $A_1 = 2, A_2 = 5, \bar{S} = \{s, 2\}, \bar{T} = \{5, t\}, Adj - A_1 = \{3, 4\}, Adj - A_2 = \{4, 6\}$. All

potential combinations have been considered $[(3,4)(3,6),(4,4),(4,6)]$, $(3,4)$ isn't link between node 3 and node 4 then push $[3,4,\{s,2,3\},\{4,5,t\}]$, $(3,6)$ there is link between node 3 and node 6, we are given a new path. $MP_4 = \{s,2,3,6,5,t\}$ then push $[3,6,\{s,2,3\},\{6,5,t\}]$,$(4,4)$ since $4=4$ get new path $MP_5 = \{s,2,4,5,t\}$, and $(4,6)$ there is no connection between the two nodes then push $[4,6,\{s,2,4\},\{6,5,t\}]$.

- **Iteration 4:**

  $A_1 = 2, A_2 = 6, \bar{S} = \{s,2\}, \bar{T} = \{6,t\}, Adj - A_1 = \{3,4\}, Adj - A_2 = \{3\}$. All potential combinations have been considered $[(3,3),(4,3)]$, $(3,3)$ since $3=3$, we get new path $MP_6 = \{s,2,3,6,t\}$,$(4,3)$, and $(4,3)$ there is link between node 4 and node 3. We are given a new path $MP_7 = \{s,2,4,3,6,t\}$ then push $[4,3,\{s,2,4\},\{3,6,t\}]$.

- **Iteration 5:**

  $A_1 = 4, A_2 = 5, \bar{S} = \{s,2,4\}, \bar{T} = \{5,3,t\}, Adj - A_1 = \emptyset, Adj - A_2 = \{6\}$, go to the next iteration.

- **Iteration 6:**

  $A_1 = 3, A_2 = 4, \bar{S} = \{s,2,3\}, \bar{T} = \{4,5,t\}, Adj - A_1 = \{6\}, Adj - A_2 = \emptyset$, go to the next iteration.

- **Iteration 7:**

  $A_1 = 3, A_2 = 6, \bar{S} = \{s,2,3\}, \bar{T} = \{6,5,t\}, Adj - A_1 = \emptyset, Adj - A_2 = \emptyset$, go to the next iteration.

- **Iteration 8:**

  $[4,6,\{s,2,4\}\{6,5,t\}]$

  $A_1 = 4, A_2 = 6, \bar{S} = \{s,2,4\}, \bar{T} = \{6,5,t\}, Adj - A_1 = \{3\}, Adj - A_2 = \{3\}$. All potential combinations have been considered $[(3,3)], (3,3)$ since $3=3$, we get new path $MP_8 = \{s,2,4,3,6,5,t\}$.

- **Iteration 9:**

  $A_1 = 4, A_2 = 3, \bar{S} = \{s,2,4\}, \bar{T} = \{3,6,t\}, Adj - A_1 = \{5\}, Adj - A_2 = \{5\}$. All

potential combinations have been considered $[(5,5)]$, $(5,5)$ since $5=5$, we get new path $MP_9 = \{s, 2, 4, 5, 3, 6, t\}$.

- **Iteration 10:**

  To -Treat is $\emptyset$.

The minimal path are

$MP_1 = \{s, 2, 3, t\} = \{x_1, x_2, x_3\}$

$MP_2 = \{s, 2, 4, 3, t\} = \{x_1, x_4, x_6, x_3\}$

$MP_3 = \{s, 2, 4, 5, 3, t\} = \{x_1, x_4, x_5, x_7, x_3\}$

$MP_4 = \{s, 2, 3, 6, 5, t\} = \{x_1, x_4, x_8, x_9, x_{11}\}$

$MP_5 = \{s, 2, 4, 5, t\} = \{x_1, x_4, x_5, x_{11}\}$

$MP_6 = \{s, 2, 3, 6, t\} = \{x_1, x_2, x_8, x_{10}\}$

$MP_7 = \{s, 2, 4, 3, 6, t\} = \{x_1, x_4, x_6, x_8, x_{10}\}$

$MP_8 = \{s, 2, 4, 3, 6, 5, t\} = \{x_1, x_4, x_6, x_8, x_9, x_{11}\}$

$MP_9 = \{s, 2, 4, 5, 3, 6, t\} = \{x_1, x_4, x_5, x_7, x_8, x_{10}\}$

In this section, the techniques for enumerating minimal path sets can broadly be divided into two categories:

1. By doing the use of and exploitation of the data in the matrix representation of a network.

2. Graph traversal or exhaustive search technique using an appropriate data structure holding the network information.

The first technique is Rai and Aggarwal's algorithm, the second is matrix multiplication, and the third is the path set enumeration algorithm. These techniques are based on an adjacency matrix. Three approaches examine precisely computing the minimal path of a reliable network system. If a network with many nodes and the matrix size will give all the path was big lies the difficulty of repeating the matrix multiplication technique. This

way, the first technique is easier to find the minimal path. As the expansion of the matrix takes place, pointers can be used third technique to seek a simple and fast technique and can find paths by vertices instead of edges. While the fourth technique, which is the node-child matrix algorithm, uses a particular matrix to know which vertices are associated with each other, and paths can also be found in vertices instead of edges. Finally, the fifth technique, which is the two terminal nodes algorithm, is a unique technique that takes advantage of the network structure to find paths from both terminal nodes. This algorithm can be applied to networks containing indirect or directed.

## 3.4. Using Some Techniques to Find Minimal Cut Sets of the Shutdown Network

In this section, applied some techniques to find as follows:

### 3.4.1 The Nodes and Edges Algorithm

Algorithm for finding all the minimal cuts (MC) in different types of networks such as directed or undirected networks, cyclic or acyclic networks, etc. first determine all the MCV sets. Then, the algorithms transform each the minimal cuts vertices MCV to an MC by determining all the existing arcs between the two associating vertices sets of the MCV [52], needed some concepts. let $G(M, E(M))$ be the sub network of $G(V, E)$ including only the set of nodes $\mathcal{M}$, and $v \in M$, the edges of the set.

$$E(v, \bar{M}) = \left\{ e_{vm} \in E \mid m \in \overline{M} \right\}$$

$$E(M - \{v\}, v) = \{ e_{nv} \in E \mid n \in \mathcal{M} \}$$

Algorithm steps

- Step 0 : let $i = k = 0, S = \{s\}, T = V - \{s\}, A = MC_0 = Q_0 = E(s, T), U_0 =$

$\{t\}, B = \varphi$ and $p = \{MC_0\}$.

- Step 1: If there is node $n \in T - \{B \cup U_0\}$, adjacent to $S$ then go to step 2 else go to step 4.
- Step 2: If $G(T - \{n\}, E(T - \{n\}))$ is a connected network then let $B = \varphi$ and go to step 3, else let $B = B \cup \{n\}$ and go to step 1.
- Step 3: Let $i = i + 1, k = k + 1$, $T = T - \{n\}, A = MC_i = Q_k = A \cup E(n,\ T) - E(S, n), S = \{s\} \cup \{n\}, U_i = U_{i-1}, p = p \cup \{MC_k\}$ and go to step 1.
- Step 4: If $i = 1$ then stop, else remove the last node $n$ in S, let $i = i - 1, A = Q_i, U_i = U_i \cup \{n\}, T = T \cup \{n\}$, and go to step 1.

**Figure** (3.3) show the Algorithm flowchart.

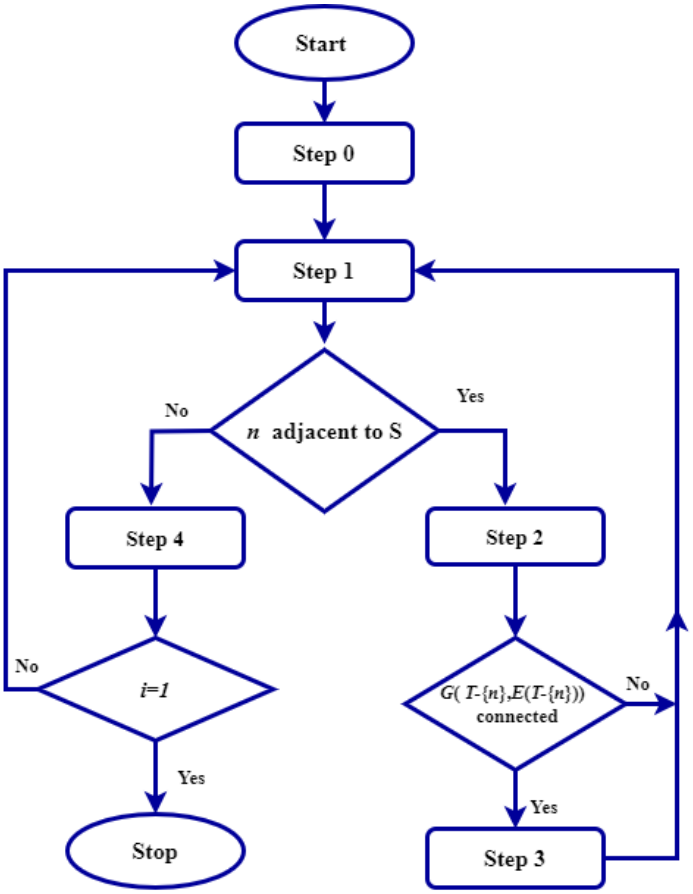



Figure 3.3: Flowchart of the nodes and edges algorithm.

Applying the algorithm to the case study shown in **Figure** (3.1)

Let $v_1 = s, v_7 = t$

**Step 0:** let $i = k = 0, S = \{s\}, T = \{2,3,4,5,6\}, A = MC_0 = Q_0 = \{e_{12}\}, U_0 = \{t\}, B = \varphi$ and $p = \{MC_0\}$.

**Step 1 :** Since $T - \{B \cup U_0\} = \{2,3,4,5,6\}, n = 2$ adjacent to $S$ then go to step 2.

**Step 2:** Step 2: If $G(T - \{2\}, E(T - \{2\})) = \{3,4,5,6,t\}$ is a connected network then $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i + 1 = 1, k = k + 1 = 1, , T = \{3,4,5,6,t\}, A = MC_1 = Q_1 = \{e_{23}, e_{24}\}, S = \{s,3\}, U_1 = \{t\}, p = \{MC_0, MC_1\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_1\} = \{3,4,5,6\}, n = 3$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{3\}, E(T - \{3\})) = \{4,5,6,t\}$ is a connected network then $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i + 1 = 2, k = k + 1 = 2, T = \{4,5,6,t\}, A = MC_2 = Q_2 = \{e_{24}, e_{36}, e_{3t}\}, S = \{s,2,3\}, U_2 = \{t\}, p = \{MC_0, MC_1, MC_2\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_2\} = \{4,5,6\}, n = 4$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{4\}, E(T - \{4\})) = \{5,6,t\}$ is a connected network then $B = \varphi$ and go to step 3.

**Step 3:** Let$i = i+1 = 3, k = k+1 = 3,\ T = \{5,6,t\}, A = mc_3 = Q_3 = \{e_{45}, e_{36}, e_{3t}\}$,$S = \{s,2,3,4\}, U_3 = \{t\}, p = \{MC_0, MC_1, MC_2, MC_3\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_3\} = \{5,6\}, n = 5$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{5\}, E(T - \{5\})) = \{6,t\}$ is a connected network then let $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i+1 = 4, k = k+1 = 4,\ T = \{6, t\}, A = MC_4 = Q_4 = \{e_{5t}, e_{3t}, e_{36}\}, S = \{s, 2, 3, 4, 5, 6\}, U_4 = \{t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4\}$ and go to step 1.

**Step 1:** In $T - \{B \cup U_4\} = \{5, 6\}, n = 6$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{6\}, E(T - \{6\})) = \{t\}$ is a connected network then $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i + 1 = 5, k = k + 1 = 5,\ T = \{t\}, A = MC_5 = Q_5 = \{e_{5t}, e_{3t}, e_{6t}\}, S = \{s, 2, 3, 4, 5, 6\}, U_5 = \{t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4, MC_5\}$ and go to Step 1.

**Step 1:** Since $T - \{B \cup U_5\} = \varphi$, go to step 4.

**Step 4:** If $i = 5 > 0$ then remove the last node $n = 6$ in $S, S = \{s, 2, 3, 4, 5\}$, let $i = i - 1 = 4, A = Q_4 = \{e_{5t}, e_{36}, e_{3t}\}, U_4 = U_4 \cup \{6\} = \{6, t\}, T = \{6, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_4\} = \varphi$, go to step 4.

**Step 4:** If $i = 4 > 0$ then $S - \{5\} = \{s, 2, 3, 4\}$, let $i = i - 1 = 3, A = Q_3 = \{e_{45}, e_{36}, e_{3t}\}, U_3 = U_3 \cup \{5\} = \{5, t\}, T = \{5, 6, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_3\} = \{6\}, n = 6$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{6\}, E(T - \{6\})) = \{5, t\}$ is a connected network then let $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i+1 = 4, k = k+1 = 6, T = \{5, t\}, A = MC_6 = Q_6 = \{e_{45}, e_{3t}, e_{6t}, e_{65}\}$, $S = \{s, 2, 3, 4, 6\}, U_4 = \{5, t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4, MC_5, MC_6\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_4\} = \varphi$, go to step 4.

**Step 4:** If $i = 4 > 0$ then $S - \{6\} = \{s, 2, 3, 4\}$, let $i = i - 1 = 3, A = Q_3 = \{e_{45}, e_{36}, e_{3t}\}, U_3 = U_3 \cup \{6\} = \{6, 5, t\}, T = \{6, 5, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_3\} = \varphi$, go to step 4.

**Step 4:** If $i = 3 > 0$ then $S - \{4\} = \{s, 2, 3\}$, let $i = i - 1 = 2, A = Q_2 = \{e_{24}, e_{36}, e_{3t}\}, U_2 = U_2 \cup \{4\} = \{4, t\}, T = \{4, 6, 5, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_2\} = \{6, 5\}, n = 6$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{6\}, E(T - \{6\})) = \{4, 5, t\}$ is a connected network then let $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i + 1 = 3, k = k + 1 = 7, T = \{4, 5, t\}, A = MC_7 = Q_7 = \{e_{24}, e_{3t}, e_{6t}, e_{65}\}, S = \{s, 2, 3, 6\}, U_3 = \{4, t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4, MC_5, MC_6, MC_7\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_3\} = \{5\}, n = 5$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{5\}, E(T - \{5\})) = \{4, t\}$ isn't a connected network then let $B = \{5\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_3\} = \varphi$, go to step 4.

**Step 4:** If $i = 3 > 0$ then $S - \{6\} = \{s, 2, 3\}$, let $i = i - 1 = 2, A = Q_2 = \{e_{24}, e_{36}, e_{3t}\}, U_2 = U_2 \cup \{6\} = \{4, 6, t\}, T = \{6, 4, 5, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_2\} = \varphi$, then go to step 2.

**Step 2:** If $i = 2 > 0$ then $S - \{3\} = \{s, 2\}$, let $i = i - 1 = 1, A = Q_1 = \{e_{23}, e_{24}\}, U_1 = U_1 \cup \{3\} = \{3, t\}, T = \{3, 6, 4, 5, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_1\} = \{6, 4\}, n = 4$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{4\}, E(T - \{4\})) = \{3, 6, 5, t\}$ is a connected network then $B = \varphi$ and go to step 3.

**Step 3:** Let $i = i + 1 = 2, k = k + 1 = 8,\ T = \{3, 6, 5, t\}, A = MC_8 = Q_8 = \{e_{23}, e_{45}, e_{43}\}, S = \{s, 2, 4\}, U_2 = \{3, t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4, MC_5, MC_6, MC_7, MC_8\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_2\} = \{6, 5\}, n = 5$ adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{5\}, E(T - \{5\})) = \{6, t\}$ is a connected network then let $B = \{5\}$ and go to step 3.

**Step 3:** Let $i = i + 1 = 3, k = k + 1 = 9, T = \{3, 6, t\}, A = MC_9 = Q_9 = \{e_{23}, e_{43}, e_{53}, e_{5t}\}, S = \{s, 2, 4, 5\}, U_3 = \{3, t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4, MC_5, MC_6, MC_7, MC_8, MC_9\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_3\} = \{6\}$, adjacent to $S$ then go to step 2.

**Step 2:** If $G(T - \{6\}, E(T - \{6\})) = \{3, t\}$ is a connected network then $B =$ and go to step 3.

**Step 3:** Let $i = i + 1 = 4, k = k + 1 = 10, T = \{3, t\}, A = MC_{10} = Q_{10} = \{e_{23}, e_{43}, e_{53}, e_{5t}\},\ MC_{10} = MC_9, S = \{s, 2, 4, 5, 6\}, U_4 = \{3, t\}, p = \{MC_0, MC_1, MC_2, MC_3, MC_4, MC_5, MC_6, MC_7, MC_8, MC_9\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_4\} = \varphi$ go to step 4.

**Step 4:** If $i = 4 > 0$ then $S - \{6\} = \{s, 2, 4, 5\}$, let $i = i - 1 = 3, A = Q_3 = \{e_{24}, e_{36}, e_{3t}\}$, $U_3 = U_3 \cup \{6\} = \{6, 3, t\}, T = \{6, 3, t\}$ and go to step 1. Since $T - \{B \cup U_3\} = \varphi$, go to step 4.

**Step 4:** If $i = 3 > 0$ then $S - \{5\} = \{s, 2, 4\}$, let $i = i - 1 = 2, A = Q_2 = \{e_{24}, e_{36}, e_{3t}\}, U_2 = U_2 \cup \{5\} = \{5, 3, t\}, T = \{5, 6, 3, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_2\} = \{6\}, n = 6$ is not adjacent to $S$ go to step 4.

**Step 4:** If $i = 2 > 0$ then $S - \{4\} = \{s, 2\}$, let $i = i - 1 = 1, A = Q_1 = \{e_{23}, e_{24}\}$, $U_1 = U_1 \cup \{4\} = \{4, 3, t\}, T = \{4, 5, 6, 3, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_1\} = \{5, 6\}, n = 6$ is no adjacent to $S$ and go to step 4.

**Step 4:** If $i = 1 > 0$ then $S - \{2\} = \{s\}$, let $i = i - 1 = 0$, $A = Q_0 = \{e_{12}\}, U_0 = U_0 \cup \{2\} = \{2, t\}, T = \{2, 4, 5, 6, 3, t\}, U_1 = U_0 \cup \{2\} = \{4, 3, t\}, T = \{4, 5, 6, 3, t\}$ and go to step 1.

**Step 1:** Since $T - \{B \cup U_0\} = \{4, 5, 6, 3\}$, there is no node adjacent to $S$ and transfer is made to step 4.

**Step 4:** If $i = 0$ stop.

The minimal cuts are

$MC_0 = \{e_{s2}\} = \{x_1\}$

$MC_1 = \{e_{23}, e_{24}\} = \{x_2, x_4\}$

$MC_2 = \{e_{24}, e_{36}, e_{3t}\} = \{x_4, x_8, x_3\}$

$MC_3 = \{e_{45}, e_{36}, e_{3t}\} = \{x_5, x_8, x_3\}$

$MC_4 = \{e_{5t}, e_{3t}, e_{36}\} = \{x_{11}, x_3, x_8\}$

$MC_5 = \{e_{5t}, e_{3t}, e_{6t}\} = \{x_{11}, x_3, x_{10}\}$

$MC_6 = \{e_{45}, e_{3t}, e_{6t}, e_{65}\} = \{x_5, x_3, x_{10}, x_9\}$

$MC_7 = \{e_{24}, e_{3t}, e_{6t}, e_{65}\} = \{x_4, x_3, x_{10}, x_9\}$

$MC_8 = \{e_{23}, e_{45}, e_{43}\} = \{x_2, x_5, x_6\}$

$MC_9 = \{e_{23}, e_{43}, e_{53}, e_{5t}\} = \{x_2, x_6, x_7, x_{11}\}$.

### 3.4.2 Shal Algorithm

The steps of the algorithm are explained as follows [8] [27]:

1. Create the network graph's connection matrix.

2. Collect all of the symbols in the first row that are source minimal cuts, as well as all of the symbols in the last column that are destination minimal cuts.

3. Form a collection 'S' of all column combinations of order 1 to $(n-3)$, $(n-4$ provides the trivial cases), and columns 2 to $(n-1)$. Delete a combination by observing:

(a) If the combination consists of only those column having zeros in the first row.

(b) If the combination is made up of rows with nonzero entries in the last column.

4. Take one combination and collect all of the symbols in these rows, including the first, with the exception of those in the combination's columns. Another cut set will be created using this combination.

5. Repeat steps 4 and 5 for all other combinations.

6. Delete those cut sets which contain all the symbols of some other cutest.

Applying the algorithm to the case study shown in **Figure** ( 3.1) , so first create the connection matrix.

$$C = \begin{bmatrix} 0 & x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2 & x_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_8 & x_3 \\ 0 & 0 & x_6 & 0 & x_5 & 0 & 0 \\ 0 & 0 & x_7 & 0 & 0 & 0 & x_{11} \\ 0 & 0 & 0 & 0 & x_9 & 0 & x_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- From the matrix $C$ above, collecting terms from first row $\{x_1\}$, and last column, $\{x_3, x_{11}, x_{10}\}$, produces two minimal cut sets.

- Here $n = 7$, order of combinations to be formed $= (7 - 3 = 4)$, i.e. orders of combination are $1, 2, 3, 4$ using only column numbers 2 to 6.
  $S = \{\{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{2,3\}, \{2,4\}, \{2,6\}, \{3,4\}, \{3,5\}, \{3,6\}, \{4,5\}\{4,6\}$
  $\{4,6\}, \{2,3,4\}, \{2,3,5\}, \{2,3,6\}, \{2,4,5\}, \{2,4,6\}, \{2,5,6\}, \{3,4,5\}, \{3,4,6\}, \{3,5,6\},$
  $\{4,5,6\}, \{2,3,4,5\}, \{2,3,4,6\}, \{2,3,5,6\}, \{2,4,5,6\}, \{2,4,5,6\}, \{3,4,5,6\}\}$.
  By step 2 cancel.

$\{3\},\{4\},\{5\},\{6\},\{3,4\},\{3,5\},\{3,6\},\{4,5\}\{4,6\},\{4,6\},\{3,4,5\},\{3,4,6\},\{3,5,6\}$
$\{4,5,6\},\{3,4,5,6\}$.

As for the rest of the cases, they are shown in the **Table** (3.3). Because of step 6, will delete the sets in the rows $4,5,7,10,11$.

Table 3.3: Steps Shal algorithm for the network of **Figure** (3.1.)

| n | Combination | Minimal cut |
|---|---|---|
| 1 | $\{2\}$ | $\{x_2,x_4\}$ |
| 2 | $\{2,3\}$ | $\{x_4,x_3,x_8\}$ |
| 3 | $\{2,4\}$ | $\{x_2,x_6,x_5\}$ |
| 4 | $\{2,5\}$ | $\{x_2,x_4,x_6\}$ delete |
| 5 | $\{2,6\}$ | $\{x_2,x_4,x_9,x_{10}\}$ delete |
| 6 | $\{2,3,4\}$ | $\{x_5,x_3,x_8\}$ |
| 7 | $\{2,3,5\}$ | $\{x_4,x_3,x_8,x_{11}\}$ delete |
| 8 | $\{2,3,6\}$ | $\{x_4,x_3,x_9,x_{10}\}$ |
| 9 | $\{2,4,5\}$ | $\{x_2,x_6,x_7,x_{11}\}$ |
| 10 | $\{2,4,6\}$ | $\{x_2,x_6,x_5,x_9,x_{10}\}$ delete |
| 11 | $\{2,5,6\}$ | $\{x_2,x_4,x_7,x_6,x_{11}\}$ delete |
| 12 | $\{2,3,4,5\}$ | $\{x_8,x_3,x_{11}\}$ |
| 13 | $\{2,3,4,6\}$ | $\{x_5,x_3,x_9,x_{10}\}$ |
| 14 | $\{2,3,5,6\}$ | $\{x_4,x_3,x_9,x_{11}\}$ delete |
| 15 | $\{2,4,5,6\}$ | $\{x_2,x_6,x_7,x_{10},x_{11}\}$delete |

### 3.4.3 Roaa Technique

A proposed technique depends on determining the $MC$ set based on the set of minimal path vertices $MPV$, found the $MC$ set for a direct network using the vertices for minimal paths sets, the technique using the relationship between the set $MPV$ and its complement. Let $\omega_i$ be vertices for minimal path where $\omega_i \subseteq V, \overline{\omega_i} = V - \omega_i, i = 1,\dots n$, where $n$ number of minimal paths. The steps that make up the technique are as follows:

- Step 1: Delete the sink vertex $\boldsymbol{t}$ from each minimal paths sets $(p)$. $\omega = p - \{t\}$.

- Step 2: Find complement for $\omega$ and $\{s\}, \{t\}$.
- Step 3: Find MC sets.

$$\mathrm{MC}(\omega) = \{e_{um} \in X \mid u \in \omega, m \in \bar{\omega}\}$$

and

$$\mathrm{MC}(t) = \{e_{ut} \in X \mid u \in \bar{\omega}\}$$

- Step 4: If the network contains a cycle, delete the vertices that make up the cycle from the path and go back to step 2.

Applying the algorithm to the case study shown in **Figure** (3.1 ) get $MPV$ sets from

$$\{s, v_2, v_3, v_6, v_5, t\}, \{s, v_2, v_3, v_6, t\}, \{v_1, v_2, v_3, t\}, \{s, v_2, v_4, v_3, v_6, v_5, t\}$$
$$\{s, v_2, v_4, v_3, v_6, t\}, \{s, v_2, v_4, v_3, t\}, \{s, v_2, v_4, v_5, v_3, v_6, t\}, \{s, v_2, v_4, v_5, t\}$$
$$\{s, v_2, v_4, v_5, v_3, t\}$$

and note that the network contains a cycle between the nodes $\{v_3, v_6, v_5\}$, there are three paths that contain by step 4 and delete the nodes from paths, details of the results are shown in the **Table** (3.5). The rest of the paths are shown in the **Table** (3.4). The delete sink vertex from all minimal paths, and find complements for the path nodes, then find the $MC$ sets of edges by taking node from the minimal paths set and vertex from the complements set, provided that the direction is from the path set to the complements set, except for the sink node by a vertex from the set of complements to sink node.

Table 3.4: Steps to find $MC$ set for $MPV$ sets that do not contain cycle.

| $w = p - \{t\}$ and $\{s\}, \{t\}$ | $\bar{w}$ | $MC$ |
|---|---|---|
| $\{s\}$ | $\{v_2, v_3, v_4, v_5, v_6, t\}$ | $\{e_{s2}\}$ |
| $\{t\}$ | $\{s, v_2, v_3, v_4, v_5, v_6\}$ | $\{e_{3t}, e_{5t}, e_{6t}\}$ |
| $\{s, v_2, v_3, v_6\}$ | $\{v_4, v_5, t\}$ | $\{e_{24}, e_{65}, e_{3t}, e_{6t}\}$ |
| $\{s, v_2, v_3\}$ | $\{v_4, v_5, v_6, t\}$ | $\{e_{24}, e_{36}, e_{3t}\}$ |
| $\{s, v_2, v_4, v_3, v_6\}$ | $\{v_5, t\}$ | $\{e_{45}, e_{56}, e_{3t}, e_{6t}\}$ |
| $\{s, v_2, v_4, v_3\}$ | $\{v_5, v_6, t\}$ | $\{e_{45}, e_{36}, e_{3t}\}$ |
| $\{s, v_2, v_4, v_5, v_3\}$ | $\{v_6, t\}$ | $\{e_{36}, e_{5t}, e_{3t}\}$ |
| $\{s, v_2, v_4, v_5, \}$ | $\{v_3, v_6, t\}$ | $\{e_{23}, e_{43}, e_{53}, e_{5t}\}$ |

complements set, except for the sink vertex by a vertex from the set of complements to sink node.

Table 3.5: Steps to find $MC$ sets for $MPV$ sets that contain cycle.

| path | $w = p - \{t\}$ with delete cycle | $\bar{w}$ | $MC$ |
|---|---|---|---|
| $\{s, v_2, v_3, v_6, v_5, t\}$ | $\{s, v_2\}$ | $\{v_3, v_6, v_5, v_4, t\}$ | $\{e_{23}, e_{24}\}$ |
| $\{s, v_2, v_4, v_5, v_3, v_6, t\}$ | $\{s, v_2, v_4\}$ | $\{v_3, v_6, v_5, t\}$ | $\{e_{23}, e_{45}, e_{43}\}$ |
| $\{s, v_2, v_4, v_3, v_6, v_5, t\}$ | $\{s, v_2, v_4\}$ | $\{v_3, v_6, v_5, t\}$ | $\{e_{23}, e_{45}, e_{43}\}$ |

The three paths containing the points of the cycle. Although the path passes through each of these points once, deleted the points, delete the sink node, and find a $MC$ of edges, the same way used it for other paths, which are indicated in the **Table** (3.5). In this section the techniques for enumerating minimal path sets can broadly be divided into two categories:

1. Direct techniques that take advantage of the network graph's structure, as well as its representation through an appropriate data structure and subsequent manipulation.

2. Path sets based techniques with the use of set theoretic laws and concepts.

Introduced, the first technique was nodes and edges algorithm for finding all the minimal cuts by determine all minimal cuts vertices sets and transform to minimal cuts edges, the second technique is Shall algorithm is a technology based on network connection matrix and find that the minimum cut edges are modeled for the enumeration plots of the network have a source and a terminal node. Simply operate on an enumeration of all the combinations and uses only those that lead to minimal cuts. This algorithm can be applied to networks that contain indirect or directed. Finally, proposed technique is the Roaa technique, depends technique is minimal path vertices to find a minimum cut of edges for directed graphs with a cycle. It is a simplified technique and does not contain computational complications at the time of calculation less than other techniques.

## 3.5. Using Some Techniques to Find Reliability Polynomial of The Shutdown Network

Network reliability evaluation is computationally laborious. The techniques discussed in this section are being applied to system of the shutdown network (3.1). reviewed some techniques.

### 3.5.1 Sum of Disjoint Products Technique

A sum of disjoint simple products created by Abraham. The technique based on Sum of disjoint product (SDP) have been used effectively and efficiently and reducing the complexity of generated structural functions to provide reliability evaluation, the SDP technique is to take each minimal path sets and make it disjoint with each preceding a path sets using Boolean algebra [93] [105], the minimal path set $MP = \{A_1, A_2, ..., A_n\}$ as show

$$\cup_{i=1}^{n} A_i = A_1 \cup (\bar{A}_1 A_2) \cup \cdots \cup (\bar{A}_1 \bar{A}_2 \ldots \bar{A}_{n-1} A_n) \tag{3.7}$$

Equation (3.7) is called the disjointed process. Compute the equation reliability expression

is used to evaluate the probability.

$$R_{sys} = Pr(A_1) + Pr(\bar{A}_1 A_2) + \cdots + Pr(\bar{A}_1 \bar{A}_2 \ldots \bar{A}_{n-1} A_n) \tag{3.8}$$

To compute for reliability system of shutdown network in **Figure** (3.1), using The minimal path sets are $MP_1 = \{x_1, x_2, x_3\}$, $MP_2 = \{x_1, x_2, x_8, x_{10}\}$, $MP_3 = \{x_1, x_4, x_6, x_3\}$, $MP_4 = \{x_1, x_4, x_5, x_{11}\}$, $MP_5 = \{x_1, x_2, x_8, x_9, x_{11}\}$, $MP_6 = \{x_1, x_2, x_8, x_9, x_{11}\}$, $MP_7 = \{x_1, x_4, x_6, x_8, x_{10}\}$, $MP_8 = \{x_1, x_4, x_5, x_7, x_3\}$, $MP_9 = \{x_1, x_4, x_5, x_7, x_8, x_{10}\}$ then the system reliability

$$\begin{aligned} R_{sys} = {} & Pr[x_1x_2x_3] + Pr[(1 - x_1x_2x_3)(x_1x_2x_8x_{10})] + Pr[(1 - x_1x_2x_3)(1 - x_1x_2x_8 \\ & x_{10})(x_1x_4x_6x_3)] + Pr[(1 - x_1x_2x_3)(1 - x_1x_2x_8x_{10})(1 - x_1x_4x_6x_3)(x_1x_4x_5x_{11}) \\ & ] + Pr[(1 - x_1x_2x_3)(1 - x_1x_2x_8x_{10})(1 - x_1x_4x_6x_3)(1 - x_1x_4x_5x_{11})(x_1x_2x_8x_9 \\ & x_{11})] + Pr[(1 - x_1x_2x_3)(1 - x_1x_2x_8x_{10})(1 - x_1x_4x_6x_3)(1 - x_1x_4x_5x_{11})(1 - x_1x_2 \\ & x_8x_9x_{11})(x_1x_2x_8x_9x_{11})] + Pr[(1 - x_1x_2x_3)(1 - x_1x_2x_8x_{10})(1 - x_1x_4x_6x_3)(1 - x_1 \\ & x_4x_5x_{11})(1 - x_1x_2x_8x_9x_{11})(1 - x_1x_2x_8x_9x_{11})(x_1x_4x_6x_8x_{10})] + Pr[(1 - x_1x_2x_3)(1 \\ & - x_1x_2x_8x_{10})(1 - x_1x_4x_6x_3)(x_1x_4x_5x_{11})(1 - x_1x_2x_8x_9x_{11})(1 - x_1x_2x_8x_9x_{11})(1- \\ & x_1x_4x_6x_8x_{10})(x_1x_4x_5x_7x_3)] + Pr[(1 - x_1x_2x_3)(1 - x_1x_2x_8x_{10})(1 - x_1x_4x_6x_3)(1- \\ & x_1x_4x_5x_{11})(1 - x_1x_2x_8x_9x_{11})(1 - x_1x_2x_8x_9x_{11})(1 - x_1x_4x_6x_8x_{10})(1 - x_1x_4x_5x_7x_3) \\ & (x_1x_4x_5x_7x_8x_{10})] \end{aligned}$$

Simplifying the calculations using Boolean algebra are get.

$$\begin{aligned} R_{sys} = {} & R_1R_2R_3 + R_1R_3R_4R_6 + R_1R_2R_8R_{10} + R_1R_4R_5R_{11} - R_1R_2R_3R_4R_6 + R_1R_3R_4 \\ & R_5R_7 - R_1R_2R_3R_8R_{10} + R_1R_4R_6R_8R_{10} + R_1R_2R_8R_9R_{11} - R_1R_2R_3R_4R_5R_7- \\ & R_1R_2R_3R_4R_5R_{11} - R_1R_3R_4R_5R_6R_7 - R_1R_3R_4R_5R_6R_{11} - R_1R_2R_4R_6R_8R_{10}- \\ & R_1R_3R_4R_5R_7R_{11} - R_1R_3R_4R_6R_8R_{10} - R_1R_2R_3R_8R_9R_{11} + R_1R_4R_5R_7R_8R_{10}+ \end{aligned}$$

$$R_1R_4R_6R_8R_9R_{11} - R_1R_2R_8R_9R_{10}R_{11} + R_1R_2R_3R_4R_5R_6R_7 + R_1R_2R_3R_4R_5R_6$$
$$R_{11} + R_1R_2R_3R_4R_5R_7R_{11} + R_1R_2R_3R_4R_6R_8R_{10} - R_1R_2R_4R_5R_7R_8R_{10} + R_1R_3$$
$$R_4R_5R_6R_7R_{11} - R_1R_3R_4R_5R_7R_8R_{10} - R_1R_2R_4R_5R_8R_9R_{11} - R_1R_2R_4R_5R_8R_{10}$$
$$R_{11} - R_1R_2R_4R_6R_8R_9R_{11} - R_1R_4R_5R_6R_7R_8R_{10} - R_1R_3R_4R_6R_8R_9R_{11} + R_1R_2$$
$$R_3R_8R_9R_{10}R_{11} - R_1R_4R_5R_6R_8R_9R_{11} - R_1R_4R_5R_6R_8R_{10}R_{11} - R_1R_4R_5R_7R_8R_{10}$$
$$R_{11} - R_1R_4R_6R_8R_9R_{10}R_{11} - R_1R_2R_3R_4R_5R_6R_7R_{11} + R_1R_2R_3R_4R_5R_7R_8R_{10} +$$
$$R_1R_2R_3R_4R_5R_8R_9R_{11} + R_1R_2R_4R_5R_6R_7R_8R_{10} + R_1R_2R_3R_4R_5R_8R_{10}R_{11} + R_1$$
$$R_2R_3R_4R_6R_8R_9R_{11} + R_1R_3R_4R_5R_6R_7R_8R_{10} + R_1R_2R_4R_5R_6R_8R_9R_{11} + R_1R_2$$
$$R_4R_5R_6R_8R_{10}R_{11} + R_1R_3R_4R_5R_6R_8R_9R_{11} + R_1R_2R_4R_5R_7R_8R_{10}R_{11} + R_1R_3R_4$$
$$R_5R_6R_8R_{10}R_{11} + R_1R_3R_4R_5R_7R_8R_{10}R_{11} + R_1R_2R_4R_5R_8R_9R_{10}R_{11} + R_1R_2R_4R_6$$
$$R_8R_9R_{10}R_{11} + R_1R_3R_4R_6R_8R_9R_{10}R_{11} + R_1R_4R_5R_6R_7R_8R_{10}R_{11} + R_1R_4R_5R_6R_8$$
$$R_9R_{10}R_{11} - R_1R_2R_3R_4R_5R_6R_7R_8R_{10} - R_1R_2R_3R_4R_5R_6R_8R_9R_{11} - R_1R_2R_3R_4R_5$$
$$R_6R_8R_{10}R_{11} - R_1R_2R_3R_4R_5R_7R_8R_{10}R_{11} - R_1R_2R_3R_4R_5R_8R_9R_{10}R_{11} - R_1R_2R_3R_4$$
$$R_6R_8R_9R_{10}R_{11} - R_1R_2R_4R_5R_6R_7R_8R_{10}R_{11} - R_1R_3R_4R_5R_6R_7R_8R_{10}R_{11} - R_1R_2R_4$$
$$R_5R_6R_8R_9R_{10}R_{11} - R_1R_3R_4R_5R_6R_8R_9R_{10} + R_{11} + R_1R_2R_3R_4R_5R_6R_7R_8R_{10}R_{11} +$$
$$R_1R_2R_3R_4R_5R_6R_8R_9R_{10}R_{11} \tag{3.9}$$

By definition 2.5.8 the equation (3.9) is a coherent network.

### 3.5.2 Minimal Cuts Technique

The basic ides is to find polynomial reliability, used the minimal cut sets and convert into the serial-parallel reliability block diagram [67] [92], and calculate the system structure by using theorem (2.5.1).

Applying this technique to the case study shutdown network in **Figure** (3.1), get

minimal cut sets from section 3.4 are

$$\{x_1\}, \{x_2, x_4\}, \{x_4, x_8, x_3\}, \{x_5, x_8, x_3\}, \{x_5, x_3, x_{10}, x_9\}, \{x_4, x_3, x_{10}, x_9\},$$
$$\{x_2, x_6, x_7, x_{11}\}, \{x_{11}, x_3, x_{10}\}, \{x_{11}, x_3, x_8\}, \{x_2, x_5, x_6\}$$

. and representation by block diagram **Figure** (3.4). Compute the system reliability.

$$\begin{aligned}\phi(x) = [x_1][(1-(1-x_2)(1-x_4)[1-(1-x_3)(1-x_4)(1-x_8)][1-(1-x_3)(1-x_5)(1-\\ x_8)][1-(1-x_2)(1-x_6)(1-x_7)(1-x_{11})][1-(1-x_3)(1-x_4)(1-x_9)(1-x_{10})][1\\ -(1-x_3)(1-x_5)(1-x_9)(1-x_{10})][1-(1-x_3)][1-(1-x_8)(1-x_{11})][1-(1-\\ x_3)(1-x_{10})1-(1-x_{11})][1-(1-x_2)(1-x_5)(1-x_6)]\end{aligned} \tag{3.10}$$

The result that has obtained is the same result in equation (3.9)

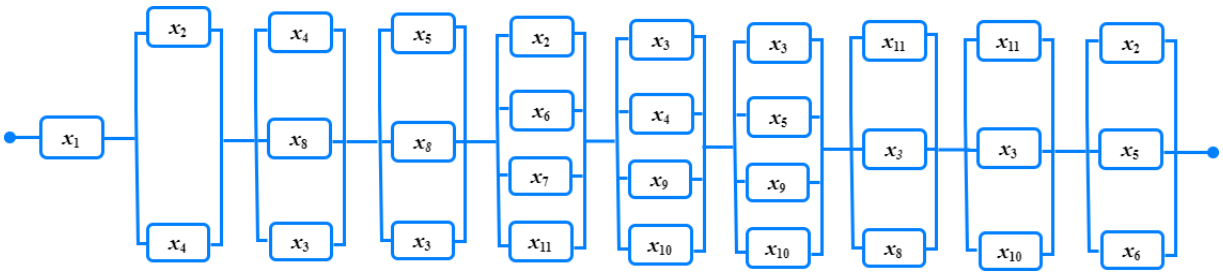



Figure 3.4: Series-parallel reliability block diagram of shutdown network.

Two different techniques are studied for calculating the reliability of the shutdown network polynomial, are introduced the first technique is the SDP simple products technique depends on the laws of probability, and the reliability was calculated based on the set of minimal paths and the second technique is a minimal cut technique, using minimal cut sets to get the network polynomial was the same using both techniques, and this indicates that our results were correct and that all techniques can be relied upon to calculate the reliability of shutdown network.

# CHAPTER 4

# META-HEURISTICS ALGORITHMS

## 4.1. Introduction

Many meta-heuristics and evolutionary algorithms are optimization techniques frequently inspired by natural systems, used to calculate good approximations to solutions optimization problems that are difficult or impossible to solve with other optimization techniques like linear programming, nonlinear programming, integer programming, and dynamic programming. Evolutionary and meta-heuristics algorithms are widely applicable, problem-independent techniques that successfully address various challenging, real-world engineering issues. To solve complicated engineering optimization problems, meta-heuristics and evolutionary algorithms have gained popularity [22]. Because of the importance of these algorithms, highlight them by presenting some of them, as well as trying to improve them by conducting a hybridization process, which will be explained through this chapter. In this chapter, two hybrid algorithms are introduced to the Honey Badger Algorithm with the Nelder-Mead method and a hybrid algorithm combining Dwarf Mongoose optimization algorithm and the Nelder- Mead method.

## 4.2. The Honey Badger Algorithm

The honey badger algorithm (HBA) imitates the foraging style for the honey badger (HB). Either the HB searches for food sources by sniffing and digging or by tracking the honey guide bird. The first circumstance is known as the digging mode, whereas the second is known as the honey mode. It locates the prey by using its sniffing abilities. Once there, it explores the area around the prey to determine the optimum location for digging and catching it. It locates the prey by using its sniffing abilities. Once there, it explores the area around the prey to determine the optimum location for digging and catching it. In the final option, the beehive is immediately located by the honey badger using the honey guide bird as a guide [64] [98].

### 4.2.1 The ABA Mathematical Model

This subsections provide an explanation of the HBA's mathematical models [64] [98]. Since it incorporates both the exploration and exploitation stages. Theoretically, the HBA is a global optimization approach. The population of potential solutions ($X$) in the HBA is represented, mathematically as:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1D} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nD} \end{bmatrix}$$

Every badger in the population is represented by its position $x_i = [x_i^1, x_i^2, ..., x_i^D]$, where $D$ signifies the number of variables and $i = 1, 2, 3, ..., n$, where $n$ represents population size.in detail as follows

- Step 1: Initialization phase. Equation (4.1) can be used to initialize the respective positions of honey badgers with $n$ population.

$$x_i = lb_i + r_1(ub_i - lb_i), r_1 \in [0, 1] \tag{4.1}$$

  where $x_i$ is the $i-th$ honey badger position and corresponds to a candidate solution, while $ub_i$, $lb_i$, denote the upper and lower bounds of the search space, respectively.

- Step 2: Defining Intensity ($I$). Prey concentration strength and the distance between it and the $i-th$ honey badger are the two key factors that affect intensity ($I$), $I_i$ is the prey's scent intensity, when the scent is weak, the prey moves slowly, and vice versa. It depicts the prey's scent strength. It is provided by Inverse Square Law (ISL) see in section 2.15 in as shown by equation (4.2) in

**Figure** (4.1).

$$I_i = r_2 \times \frac{\mathtt{St}}{(4\pi d_i^2)}, r_2 \in [0,1] \tag{4.2}$$

$$\mathtt{St} = (x_i - x_{(i+1)})^2 \tag{4.3}$$

$$d_i = x_{prey} - x_i \tag{4.4}$$

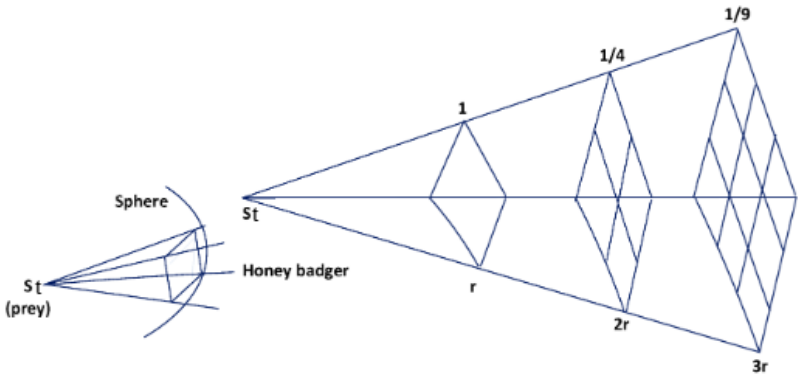



Figure 4.1: ISL. $I$ is smell intensity, $\mathtt{St}$ is prey position, and $r_2 \in [0,1]$.

where $\mathtt{St}$ stands for the strength of the source or concentration, the distance $d_i$ is what separates the $i$th badger from its victim.

- Step 3: Update density factor. The density factor ($\alpha$) controls time-varying randomization to ensure smooth transition from exploration to exploitation. Update decreasing factor ($\alpha$) that decreases with iterations to decrease randomization with time, use equation (4.5 )

$$\alpha = C \times exp(\frac{-t}{t_{max}}) \tag{4.5}$$

where ($C$) is a constant number more than 1 (the default value is 2), and denotes $t_{max}$ the maximum number of iterations.

- Step 4: Escaping from local optimum. The HBA uses the current step and the following two in order to leave the local solution area. The HBA optimization

technique employs a flag ($F$) that modifies the search direction to provide agents more opportunities to precisely scan the search area.

- Step 5: Update the agents' positions. The HBA position update process ($x_{new}$), as previously mentioned, is divided into two stages, the " digging phase" and the " honey phase". What follows is a more detailed explanation:

    1. Digging phase: In digging phase, a honey badger performs action similar to Cardioid shape as shown in **Figure** (4.2). The Cardioid motion can be simulated by equation (4.6 )

$$x_{new} = x_{prey} + F \times \beta \times I \times x_{prey} + F \times r_3 \times \alpha \times d_i \times |cos(2\pi r_4)[1 - cos(2\pi r_5)]| \quad (4.6)$$

    where $x_{prey}$ represents the global optimum, or the best position of the prey as of yet. $\beta \geq 1$ indicates the honey badger's capacity to find food (default value is 6). $d_i$ is distance between prey and the $i$th honey, $r_3, r_4$ ,and $r_5$ are three different generated random numbers inside the range $[0, 1]$. The search direction is controlled by the flag $F$, which is defined by ( 4.7 )

$$F = \begin{cases} 1 & , if \ r_6 \leq 0.5 \\ -1 & otherwise \end{cases} \quad (4.7)$$

    where $r_6 \in [0, 1]$.

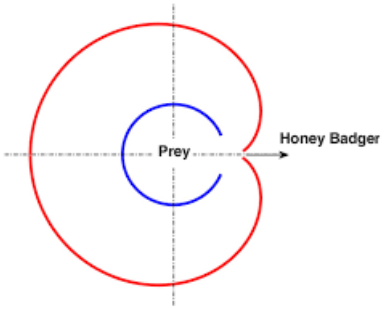



Figure 4.2: Digging phase: the red outline indicates the strength of the smell, while the blue circular line indicates the position of the prey.

2. Honey phase. Equation (4.8) describes how the honey badger is guided to the hive by the guide bird.

$$x_{new} = x_{prey} + F \times r_7 \times \alpha \times d_i \tag{4.8}$$

where $r_7 \in [0, 1]$, the prey position is represented by $x_{prey}$, the badger's most recent location is represented by $x_{new}$, $\alpha$ and $F$ are determined using equations (4.5) and (4.7), respectively. From equation (4.8), it can be observed that a honey badger performs search close to prey location $x_{prey}$ found so far, based on distance information $d_i$. At this stage, time-varying search behavior $\alpha$ has an impact on the search. A honey badger may also find $F$ disturbance.

The flowchart of the HBA manner shown in **Figure** (4.3) and in Appendix (6.2) the algorithm's pseudo code is described.

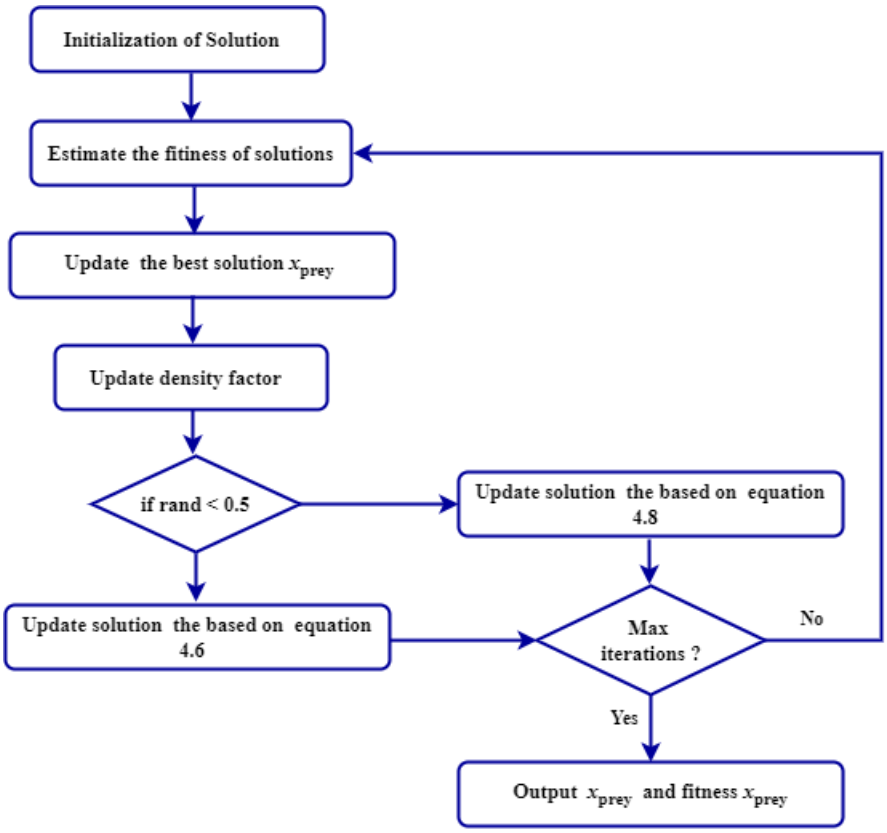



Figure 4.3: Flowchart of the HBA.

## 4.3. The Proposed Hybrid Honey Badger Nelder Mead Algorithm

The suggested honey badger Nelder Mead method (HBNMA) follows the same steps the conventional the Honey Badger Algorithm (HBA) then apply solution obtained from HBA in the Nelder-Mead method see in section 2.9 for same of iteration, to improve the best result from the previous step of the HBA .

## 4.4. Benchmark Functions

A wide variety of experiments are carried out to evaluate the performance and effectiveness of the suggested technique. Used to test the HBNMA on 23 test functions are traditional benchmark functions used in many different types of research, characteristics of test functions are modality, basins, valleys, separability and dimensionality [47] [108]. Seven unimodal test functions, six multimodal functions with high dimensions, and ten multimodal functions with fixed dimensions all of them have been used. **Table** (4.1) shows the list of the unimodal functions ($f_1$–$f_7$), these issues have a single minimum in the search space. The major goal of solving these problems is to examine how well optimization techniques converge to the global optima. **Table** (4.2) shows the high-dimensional multi modal functions ($f_8$–$f_{13}$) which have several local optimal regions in the search space. The major goal of optimizing these kinds of issues is to assess an optimization algorithm's capacity to cross non-optimal regions and so determine the primary optimal region. These two sets of benchmarks are employed in 10, 30 and 50 dimensions, indicates **Table** (4.3) to the last class is fixed dimensional multimodal functions ($f_{14}$–$f_{23}$) because it is crucial to identify both the optimal region and the domain of convergence to the optimal solution simultaneously in this type of problem. This makes them appropriate for examining the global and local search capabilities of optimization algorithms. The effectiveness of optimization algorithms in

striking the ideal balance between local search and global search is assessed by this kind of functions. **Figure** (4.4) shows a two variables representation of some mathematical functions.

Table 4.1: Unimodal test functions.

| functions | Dim | Rang | $f_{min}$ |
|---|---|---|---|
| $f_1 = \sum_{i=1}^{n} x_i$ | 10,30,50 | [-10,10] | 0 |
| $f_2 = \sum_{i=1}^{m} \lvert x_i \rvert + \prod_{i=1}^{m} \lvert x_i \rvert$ | 10,30,50, | [-10,10] | 0 |
| $f_3 = \sum_{i=1}^{m} (\sum_{j=1}^{i} x_j)^2$ | 10,30,50 | [-100,100] | 0 |
| $f_4 = max\{\lvert x_i \rvert, 1 \leq i \leq m\}$ | 10,30,50 | [-12,12] | |
| $f_5 = \sum_{i=1}^{m-1} [100(x_i + 1 - x_i^2)^2 + (x_i - 1)^2]$ | 10,30,50 | [-30,30] | 0 |
| $f_6 = \sum_{i=1}^{m} ([x_i + 0.5])^2$ | 10,30,50 | [-100,100] | 0 |
| $f_7 = \sum_{i=1}^{m} x_i^4 + random(0,1)$ | 10,30,50 | [-1,1] | 0 |

Table 4.2: Multimodal test functions.

| functions | Dim | Rang | $f_{min}$ |
|---|---|---|---|
| $f_8 = \sum_{i=1}^{m} -x_i sin(\sqrt{\lvert x_i \rvert})$ | 10,30,50 | [-100,100] | -1.2569E+04 |
| $f_9 = \sum_{i=1}^{m} [x_i^2 - 10\cos(2\pi x_i) + 10]$ | 10,30,50 | $[-5.2, 5.2]$ | 0 |

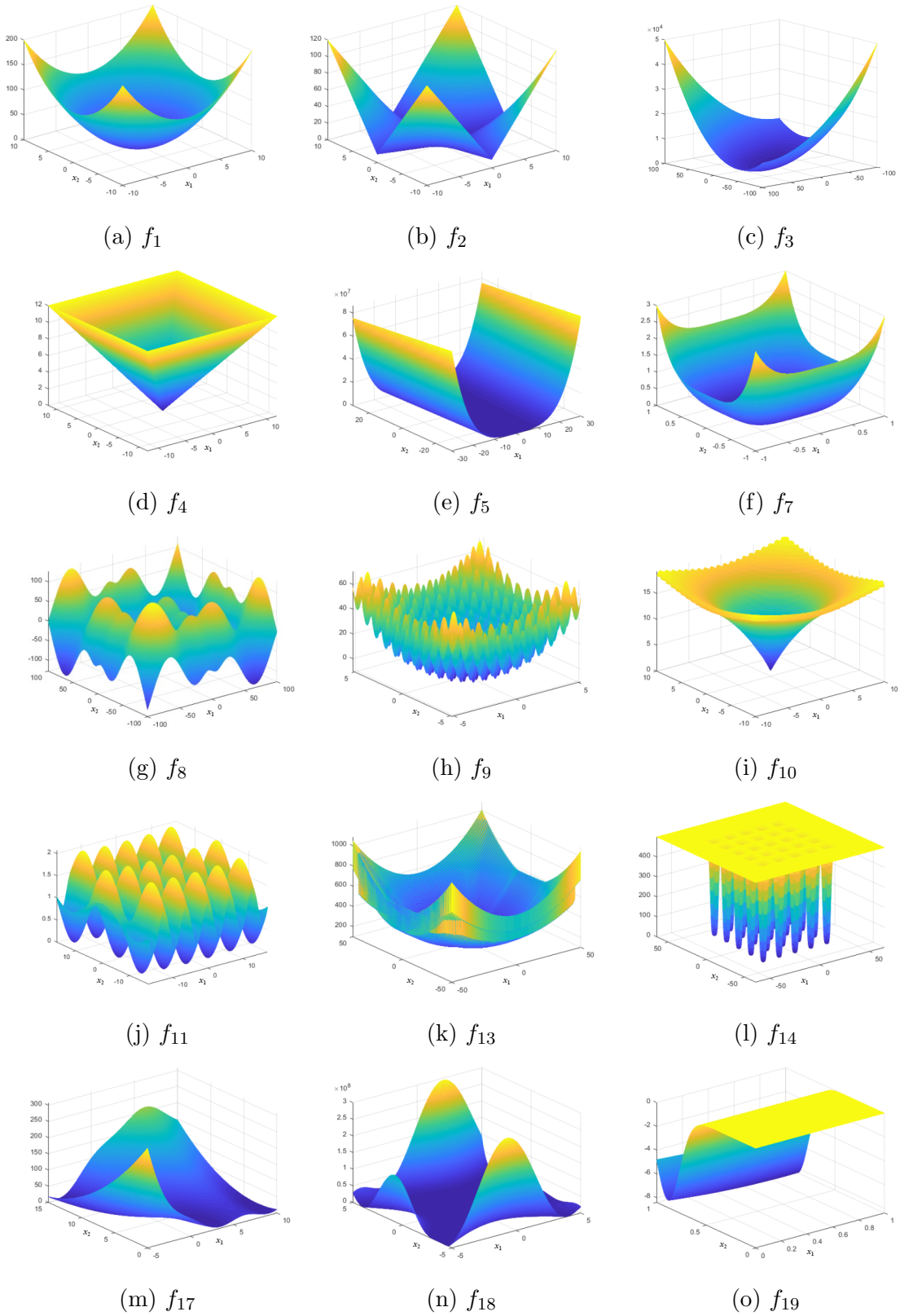
| functions | Dim | Rang | $f_{min}$ |
|---|---|---|---|
| $f_{10} = -20\exp(-0.2\sqrt{\frac{1}{m}\sum_{i=1}^{m} x_i^2}) - \exp(\frac{1}{m}\sum_{i=1}^{m}\cos(2\pi x_i)) + 20 + e$ | 10,30,50 | [-10,10] | 0 |
| $f_{11} = \frac{1}{4000}\sum_{i=1}^{m} x_i^2 - \prod_{i=1}^{m}\cos(\frac{x_i}{\sqrt{i}}) + 1$ | 10,30,50 | [-17,17] | 0 |
| $f_{12} = \frac{\pi}{m}\{10\sin(\pi y_1) + \sum_{i=1}^{m}(y_i-1)^2 + [1+10\sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i,10,100,4)$ <br> $u(x_i,a,i,,n) = \begin{cases} k(x_i-a)^n, x_i > -a \\ 0, -a < x_i < a \\ k(-x_i-a)^n, x_i < -a \end{cases}$ | 10,30,50 | [-13,13] | 0 |
| $f_{13} = 0.1\{\ \sin^2(3\pi x_1) + [1+ sin^2(3\pi x_1 + 1 + (x_n-1]^2[1+ sin^2(2\pi x_m)\} + \sum_{i=1}^{m} u(x_i,5,100,4)$ | 10,30,50 | [-50,50] | 0 |

Table 4.3: Fixed dimensional multimodal test functions.

| functions | Dim | Rang | $f_{min}$ |
|---|---|---|---|
| $f_{14} = (\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$ <br> $a = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & .... & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & ... & 32 & 32 & 32 \end{bmatrix}$ | 2 | [- 65.5,65.5] | 0.98 |

| functions | Dim | Rang | $f_{min}$ |
|---|---|---|---|
| $f_{15} = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$ | | | |
| $a = [0.1957, 0.1947, 0.1735, 0.16, 0.0844,$ $0.0627, 0.0456, 0.0342, 0.0323, 0.0235,$ $0.0246]$, $b = [0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16]$ | 4 | [-5,5] | 0.00030 |
| $f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{5.1}{4\pi}x_1^6 + x_1x_2$ $- 4x_2^4 + 4x_2^4$ | 2 | [-5,5] | -1.0316 |
| $f_{17} = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + x_1 - 6)^2$ $+ 10(1 - \frac{1}{8\pi})\cos x_1 + 10$ | 2 | $[-3, 0] \times [9, 12]$ | 0.398 |
| $f_{18} = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1$ $+ 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\times$ $[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 +$ $12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ | 2 | [- 2,2] | 3 |
| $f_{19} = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$ | | | |
| $c = (1, 1.2, 1.3, 3.2)$, | | | |
| $a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 01 & 10 & 35 \end{bmatrix}$, | 3 | [ 0,1] | -3.86 |
| $p = 10^{-4}\begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$ | | | |

| functions | Dim | Rang | $f_{min}$ |
|---|---|---|---|
| $f_{20} = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$<br>$c = (1, 1.2, 3, 3.2)$,<br>$a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$,<br>$p = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$ | 6 | [ 0,0.8] | -3.32 |
| $f_{21} = -\sum_{i=1}^{7} [(x - a_i)(x - a_i)^T + 6c_i]^{-1}$<br>$c = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]$<br>$a = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}$ | 4 | [ 0,9] | -10.1532 |
| $f_{22} = -\sum_{i=1}^{5} [(x - a_i)(x - a_i)^T + 6c_i]^{-1}$<br>where $a, c$ same value in $f_{21}$ | 4 | [ 0,9] | -10.4029 |
| $f_{23} = -\sum_{i=1}^{10} [(x - a_{1i})(x - a_{2i})^T + 6c_i]^{-1}$<br>$c = (1, 2, 2, 4, 4, 6, 3, 7, 5, 5)$,<br>$a = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix}$ | 4 | [ 0,9] | -10.5364 |



(a) $f_1$ (b) $f_2$ (c) $f_3$

(d) $f_4$ (e) $f_5$ (f) $f_7$

(g) $f_8$ (h) $f_9$ (i) $f_{10}$

(j) $f_{11}$ (k) $f_{13}$ (l) $f_{14}$

(m) $f_{17}$ (n) $f_{18}$ (o) $f_{19}$

Figure 4.4: Some of test function in two variables.

## 4.5. Results Experiment for HBA and HBNMA

Presentation of experimental findings utilizing statistical analytic methods, the average (Avg), of the generated solutions' and standard deviations (Std) equations (4.9) and (4.10) have been employed in the calculation of these two criteria.

$$Avg = \frac{1}{N}\sum_{i=1}^{N} BQS_i \tag{4.9}$$

$$Std = \sqrt{\frac{1}{N}(\sum_{i=1}^{N} BQS_i - avg)^2} \tag{4.10}$$

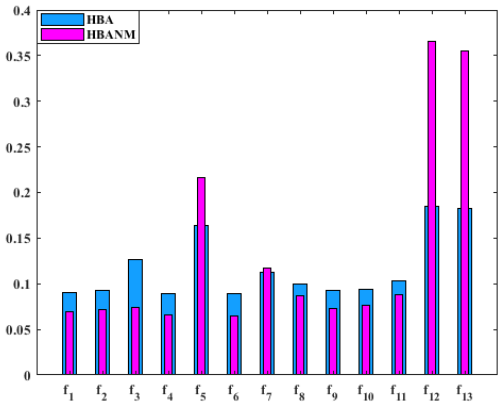
where $N$ is the number of independent implementations and $BQS_i$ is the candidate solution in the $i-th$ execution. Compared run times of the HBA with HBNMA . The maximum number of iterations for is 500, and the best results are shown in bold throughout all tables for all algorithms. The experimental results are in **Table** (4.4).

Table 4.4: Comparison Statistical results of HBNMA and HBA on unimodal and multimodal functions in 10 dimension.

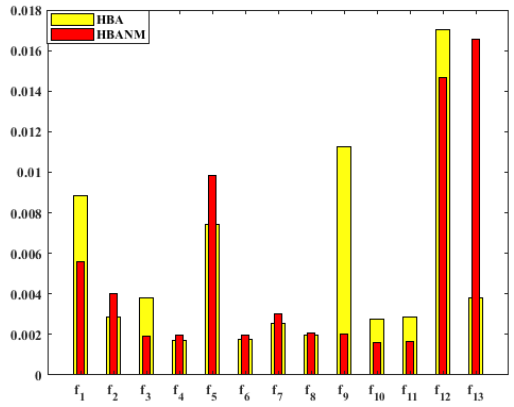
| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_1$ | HBA | 2.989E-164 | **0** | 9.045E-02 | 8.867E-03 | 0 |
| | HBNMA | **3.495E-203** | **0** | 6.934E-02 | 5.603E-03 | 0 |
| $f_2$ | HBA | 1.021E-86 | 5.236E-86 | 9.275E-02 | 2.850E-03 | 0 |
| | HBNMA | **1.063E-106** | **1.143E-105** | 7.161E-02 | 4.015E-03 | 0 |
| $f_3$ | HBA | 1.417E-127 | 2.279E-126 | 1.260E-01 | 3.826E-03 | 0 |
| | HBNMA | **0** | **0** | 7.356E-02 | 1.905E-03 | 0 |
| $f_4$ | HBA | 8.523E-74 | 7.328E-73 | 9.205E-02 | 5.063E-03 | 0 |
| | HBNMA | **4.125E-93** | **7.640E-92** | 6.597E-02 | 1.993E-03 | 0 |
| $f_5$ | HBA | **0** | **0** | 1.643E-01 | 7.447E-03 | 0 |
| | HBNMA | **0** | **0** | 2.162E-01 | 9.842E-03 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $f_6$ | HBA | 3.189E-15 | 2.692E-14 | 8.977E-02 | 1.780E-03 | 0 |
| | HBNMA | **3.567E-19** | **6.043E-18** | 6.507E-02 | 1.966E-03 | 0 |
| $f_7$ | HBA | 3.144E-04 | 2.640E-04 | 1.121E-01 | 2.560E-03 | 0 |
| | HBNMA | **2.298E-04** | **2.042E-04** | 1.173E-01 | 3.025E-03 | 0 |
| $f_8$ | HBA | **-5.405E+02** | 9.170E+01 | 9.921E-02 | 1.956E-03 | -627.12 |
| | HBNMA | **-5.405E+02** | **7.563E+01** | 8.720E-02 | 2.066E-03 | -636.350 |
| $f_9$ | HBA | **0** | **0** | 9.253E-02 | 1.124E-02 | 0 |
| | HBNMA | **0** | **0** | 7.242E-027 | 2.013E-03 | 0 |
| $f_{10}$ | HBA | **8.882E-16** | **0** | 9.353E-02 | 2.742E-03 | 0 |
| | HBNMA | **8.882E-16** | **0** | 7.637E-02 | 1.602E-03 | 0 |
| $f_{11}$ | HBA | **0** | **0** | 1.030E-01 | 2.853E-03 | 0 |
| | HBNMA | **0** | **0** | 8.757E-02 | 1.658E-03 | 0 |
| $f_{12}$ | HBA | 3.596E-15 | 5.739E-14 | 1.850E-01 | 1.705E-02 | 0 |
| | HBNMA | **4.712E-31** | **2.367E-45** | 3.652E-01 | 1.469E-02 | 0 |
| $f_{13}$ | HBA | 1.473E-02 | 3.457E-02 | 1.830E-01 | 3.815E-03 | 0 |
| | HBNMA | **7.740E-05** | **1.223E-03** | 3.556E-01 | 1.657E-02 | 0 |

Show function $\{f_1 - f_{13}\}$ in 10 dimension, the average values of the functions $f_5, f_9 - f_{11}$ are the same for both algorithms. It has been found that the proposed HBNMA performs better than the HBA for the rest functions. And the value of the standard deviation was equal for the functions $f_1, f_5, f_9, f_{10}, f_{11}$ in both algorithms, while the HABNM improved results in the rest of the functions. For the execution time show in **Figure** (4.5) of finding the execution time for average, note that the implementation time was less for the proposed algorithm for the functions $f_1 - f_4, f_6, f_9 - f_{11}$, the implementation time for the standard deviation was less for the proposed algorithm in functions $f_1, f_3, f_9 - f_{12}$. The value of the objective functions was equal to the optimal value except the value $f_8$ results of the proposed algorithm function that was closer to the optimal value from HBA .




(a) Time implement Avg ($f_1$–$f_{13}$)




(b) Time implement Std($f_1$–$f_{13}$)

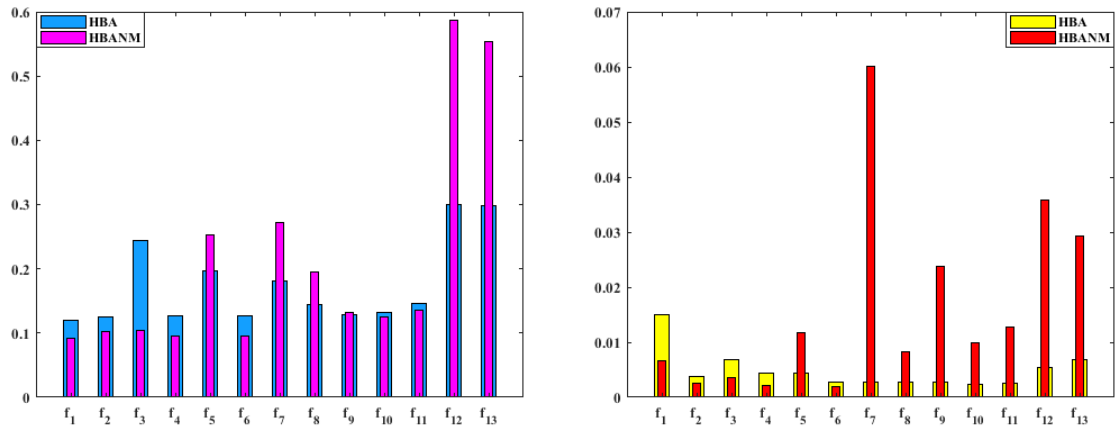
Figure 4.5: Comparison time implement Avg and Std of $f_1$–$f_{13}$ for HBNMA and HBA in 10 dimension.

Table 4.5: Comparison Statistical results of HBNMA and HBA on unimodal and multimodal functions in 30 dimension.

| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_1$ | HBA | 6.435E-137 | 5.742E-136 | 1.202E-01 | 1.510E-02 | 0 |
| | HBNMA | **6.067E-161** | **1.357E-159** | 9.140E-02 | 6.618E-03 | 0 |
| $f_2$ | HBA | 2.218E-71 | 4.194E-70 | 1.256E-01 | 3.775E-03 | 0 |
| | HBNMA | **9.192E-87** | **1.191E-85** | 1.027E-01 | 2.505E-03 | 0 |
| $f_3$ | HBA | 2.601E-94 | 5.360E-93 | 2.444E-01 | 6.876E-03 | 0 |
| | HBNMA | **0** | **0** | 1.036E-01 | 3.647E-03 | 0 |
| $f_4$ | HBA | 1.552E-57 | 9.837E-57 | 1.272E-01 | 4.373E-03 | 0 |
| | HBNMA | **7.671E-72** | **1.005E-70** | 9.536E-02 | 2.204E-03 | 0 |
| $f_5$ | HBA | **0** | **0** | 1.971E-01 | 4.380E-03 | 0 |
| | HBNMA | **0** | **0** | 2.533E-01 | 1.175E-02 | 0 |
| $f_6$ | HBA | 2.527E-02 | 7.666E-02 | 1.261E-01 | 2.787E-03 | 0 |
| | HBNMA | **5.478E-04** | **1.118E-02** | 9.532E-02 | 1.912E-03 | 0 |
| $f_7$ | HBA | 3.951E-04 | 2.877E-04 | 1.814E-01 | 2.749E-03 | 0 |
| | HBNMA | **2.582E-04** | **2.441E-04** | 2.719E-01 | 6.012E-02 | 0 |
| $f_8$ | HBA | -1.377E+03 | 3.455E+02 | 1.436E-01 | 2.785E-03 | -1824 |
| | HBNMA | **-1.530E+03** | **3.099E+02** | 1.958E-01 | 8.361E-03 | -1825 |
| $f_9$ | HBA | **0** | **0** | 1.289E-01 | 2.662E-03 | 0 |
| | HBNMA | **0** | **0** | 1.314E-01 | 2.381E-02 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $f_{10}$ | HBA | **8.882E-16** | **0** | 1.323E-01 | 2.289E-03 | 0 |
| | HBNMA | **8.882E-16** | **0** | 1.252E-01 | 9.833E-03 | 0 |
| $f_{11}$ | HBA | **0** | **0** | 1.460E-01 | 2.479E-03 | 0 |
| | HBNMA | **0** | **0** | 1.364E-01 | 1.272E-02 | 0 |
| $f_{12}$ | HBA | 5.393E-04 | 4.844E-03 | 3.001E-01 | 5.414E-03 | 0 |
| | HBNMA | **4.712E-31** | **2.367E-45** | 5.871E-01 | 3.587E-02 | 0 |
| $f_{13}$ | HBA | 4.181E-01 | 3.107E-01 | 2.983E-01 | 6.910E-03 | 0 |
| | HBNMA | **3.870E-05** | **8.654E-04** | 5.540E-01 | 2.934E-02 | 0 |

The results are in **Figure** (4.5) show functions $\{f_1 - f_{13}\}$ in 30 dimension. The average values and the standard deviation value were equal in the functions $f_5, f_9, f_{10}, f_{11}$ are the same for both algorithms. It was found that the results of the HBNMA improved the results of the HBA for the remainder functions. For the execution time show in **Figure** (4.6) the execution time of finding the average was less for the proposed algorithm with respect to the functions $f_1 - f_4, f_6, f_{10}, f_{11}$, while the execution time of the algorithm at the standard deviation was better in the functions $f_1 - f_4, f_6$ than the algorithm. The value of the objective functions was equal to the optimal value, except that the function $f_8$, a result of the proposed algorithm, was closer to the optimal value from the result of the HBA .




(a) Time implement Avg ($f_1$–$f_{13}$) (b) Time implement Std ($f_1$–$f_{13}$)

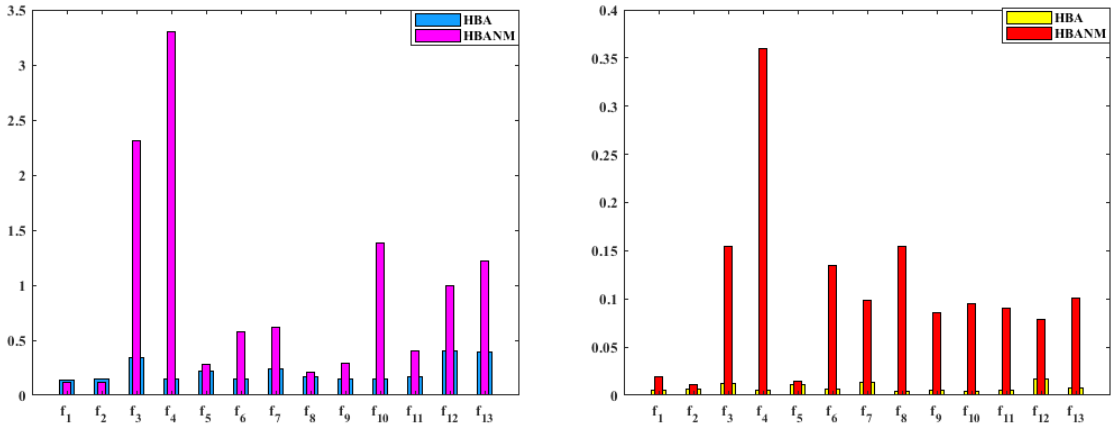
Figure 4.6: Comparison time implement Avg and Std of $f_1$–$f_{13}$ for HBNMA and HBA in 30 dimension.

Table 4.6: Comparison Statistical results of HBNMA and HBA on unimodal and multimodal functions in 50 dimension.

| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_1$ | HBA | 3.783E-129 | 4.597E-128 | 1.420E-01 | 5.331E-03 | 0 |
| | HBNMA | **1.603E-153** | **3.449E-152** | 1.190E-01 | 1.943E-02 | 0 |
| $f_2$ | HBA | 1.864E-68 | 8.733E-68 | 1.476E-01 | 6.374E-03 | 0 |
| | HBNMA | **1.567E-81** | **1.713E-80** | 1.224E-01 | 1.063E-02 | 0 |
| $f_3$ | HBA | 2.755E-84 | 5.720E-83 | 3.420E-01 | 1.196E-02 | 0 |
| | HBNMA | **0** | **0** | 2.307E+00 | 1.551E-01 | 0 |
| $f_4$ | HBA | 6.071E-50 | 8.316E-49 | 1.436E-01 | 5.265E-03 | 0 |
| | HBNMA | **2.375E-65** | **3.326E-64** | 3.302E+00 | 3.598E-01 | 0 |
| $f_5$ | HBA | **0** | **0** | 2.195E-01 | 1.166E-02 | 0 |
| | HBNMA | **0** | **0** | 2.786E-01 | 1.427E-02 | 0 |
| $f_6$ | HBA | 9.996E-01 | 4.295E-01 | 1.475E-01 | 6.116E-03 | 0 |
| | HBNMA | **2.646E-01** | **2.511E-01** | 5.814E-01 | 1.345E-01 | 0 |
| $f_7$ | HBA | 4.117E-04 | 3.029E-04 | 2.436E-01 | 1.288E-02 | 0 |
| | HBNMA | **2.580E-04** | **2.290E-04** | 6.161E-01 | 9.880E-02 | 0 |
| $f_8$ | HBA | -2.139E+03 | 5.523E+02 | 1.665E-01 | 3.903E-03 | -2978.54 |
| | HBNMA | **-2.276E+03** | **5.320E+02** | 2.067E-01 | 1.547E-01 | -3006.18 |
| $f_9$ | HBA | **0** | **0** | 1.479E-01 | 4.820E-03 | 0 |
| | HBNMA | **0** | **0** | 2.938E-01 | 8.568E-02 | 0 |
| $f_{10}$ | HBA | **8.882E-16** | **0** | 1.519E-01 | 4.238E-03 | 0 |
| | HBNMA | **8.882E-16** | **0** | 1.387E+00 | 9.514E-02 | 0 |
| $f_{11}$ | HBA | **0** | **0** | 1.636E-01 | 4.878E-03 | 0 |
| | HBNMA | **0** | **0** | 4.055E-01 | 9.076E-02 | 0 |
| $f_{12}$ | HBA | 1.589E-02 | 9.449E-03 | 4.071E-01 | 1.644E-02 | 0 |
| | HBNMA | **4.712E-31** | **2.367E-45** | 9.989E-01 | 7.897E-02 | 0 |
| $f_{13}$ | HBA | 2.456E+00 | 6.533E-01 | 3.956E-01 | 7.746E-03 | 0 |
| | HBNMA | **1.161E-04** | **1.496E-03** | 1.215E+00 | 1.006E-01 | 0 |

The results are in **Table** (4.6) show functions $\{f_1 - f_{13}\}$ in 50 dimension. The average values and the standard deviation value were equal in the functions $f_5, f_9, f_{10}, f_{11}$ are the same for both algorithms. It was found that the results of the HBNMA a improved the

results of the HBA for the remainder functions. For the execution time show in **Figure** (4.7). The execution time of finding the average was less for the proposed algorithm for the functions $f_1 - f_2$. While the implementation time the standard deviation was less for an HBA in all functions. The value of the objective functions was equal to the optimal value, except that the function $f_8$, a result of the value of the objective function of a better HBA from the result of the proposed algorithm.



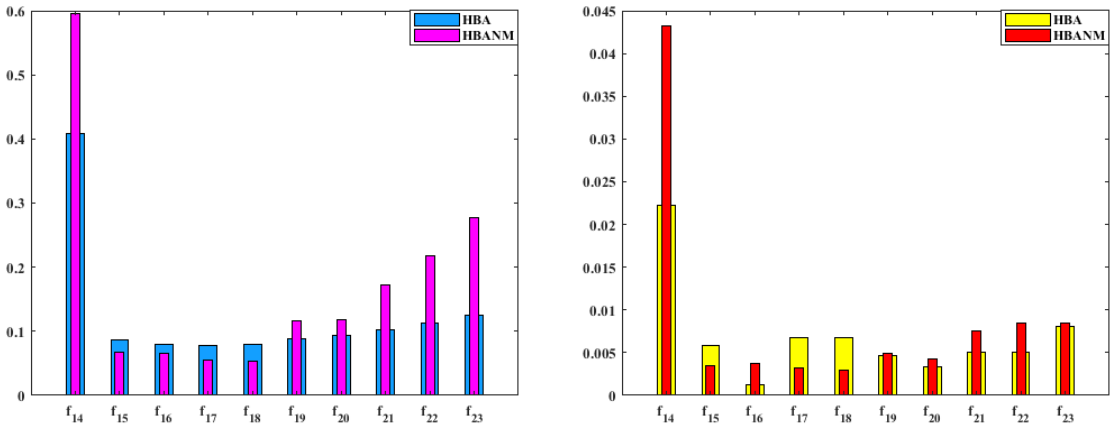

(a) Time implement Avg ($f_1$–$f_{13}$) (b) Time implement Std ($f_1$–$f_{13}$)

Figure 4.7: Comparison time implement Avg and Std of $f_1$–$f_{13}$ for HBNMA and HBA in 50 dimension.

Table 4.7: Comparison Statistical results of HBNMA and HBA fixed dimensional multimodal functions.

| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_{14}$ | HBA | 1.4766 | 1.5712 | 0.4086 | 0.0223 | 0.998 |
| | HBNMA | **1.4570** | **1.5266** | 0.5951 | 0.0432 | 0.998 |
| $f_{15}$ | HBA | 0.0064 | 0.0105 | 0.0865 | 0.0058 | 0.00031 |
| | HBNMA | **0.0052** | **0.0089** | 0.0668 | 0.0035 | 0.00031 |
| $f_{16}$ | HBA | **-1.0316** | **0** | 0.0795 | 0.0013 | -1.0316 |
| | HBNMA | **-1.0316** | **0** | 0.0663 | 0.0037 | -1.0316 |
| $f_{17}$ | HBA | 0.5269 | 0.2662 | 0.0784 | 0.0067 | 3 |
| | HBNMA | **0.5253** | **0.2728** | 0.0558 | 0.0032 | 3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $f_{18}$ | HBA | 4.7820 | 10.3144 | 0.0798 | 0.0067 | 0.398 |
| | HBNMA | **4.4040** | **9.8675** | 0.0532 | 0.0030 | 0.398 |
| $f_{19}$ | HBA | -3.8509 | 0.0908 | 0.0889 | 0.0047 | -3.86 |
| | HBNMA | **-3.8579** | **0.0598** | 0.1160 | 0.0049 | -3.86 |
| $f_{20}$ | HBA | **-3.2443** | 0.1531 | 0.0941 | 0.0034 | -3.32 |
| | HBNMA | -3.2416 | **0.1453** | 0.1185 | 0.0043 | -3.32 |
| $f_{21}$ | HBA | -9.4521 | 2.2130 | 0.1029 | 0.0051 | -10.1532 |
| | HBNMA | **-9.6717** | **1.8431** | 0.1721 | 0.0075 | -10.1532 |
| $f_{22}$ | HBA | **-9.0191** | **2.9604** | 0.1127 | 0.0051 | -10.4029 |
| | HBNMA | -8.7483 | 3.1619 | 0.2178 | 0.0084 | -10.4029 |
| $f_{23}$ | HBA | -8.4546 | 3.4329 | 0.1259 | 0.0081 | -10.5364 |
| | HBNMA | **-8.6414** | **3.3076** | 0.2769 | 0.0084 | -10.5364 |




(a) Time implement Avg ($f_{14}$–$f_{23}$) (b) Time implement Std ($f_{14}$–$f_{23}$)

Figure 4.8: Comparison time implement Avg and Std of functions $f_{14}$–$f_{23}$ for HBNMA and HBA.

The results are in **Table** (4.7) show functions $\{f_{14} - f_{23}\}$ in Fixed dimensional multimodal functions. The average values and the standard deviation value were equal in the functions $f_{16}$ are the same for both algorithms. The HBNMA performs better than the HBA in functions most functions $f_{14} - f_{19}, f_{21}, f_{23}$ except $f_{20}, f_{22}$. Show in **Figure** (4.8) for the execution time of finding the average was less for the proposed algorithm for the functions $f_{15} - f_{18}$. While the implementation time, the standard

deviation for the functions $f_{15}, f_{17}, f_{18}$ was less for an HBA. The value of the objective functions was equal to the optimal value was the same for both algorithms.

## 4.6. Dwarf Mongoose Optimization Algorithm

Dwarf Mongoose Optimization Algorithm (DMOA) is a population-based meta heuristic algorithm that drew its inspiration from the foraging and social behavior of dwarf mongooses. Each dwarf mongoose hunts for food on his own, searching for a sleeping mound the dwarf mongooses do not build a nest for their young, they move them from one sleeping mound to another and do not return to the previously foraged site, because of the semi nomadic nature of these animals. To address optimization issues, the programmer uses a mathematical model of this animal's way of existence. All population-based optimization algorithms commence with random initialization. After that, because of the intensification and diversification rules, every solution gathers around the global best optimal [7] [101].

## 4.7. The DMOA Mathematical Model

The DMOA starts its solution by initializing the mongoose's candidate population. This population is generated stochastically between a particular problem's lower and upper bounds.

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1d} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & X_{n3} & \dots & X_{nd} \end{bmatrix} \tag{4.11}$$

where $X$ is the set of current candidate populations, which are generated randomly using equation (4.12 ), $n$ indicates the population size, and $d$ is the dimension of the problem.

$$X_{i,j} = l_j + rand \times (u_j - l_j) \tag{4.12}$$

where $rand \in [0, 1]$, $u_j$ and $l_j$ are the search domain's boundaries. The Alpha Group, Scouts, and Babysitters are the three groups that make up the DMOA. Each group has a unique strategy for capturing the food, and the information about these groups is as follows:

### 4.7.1 Alpha Group

Once the population has been started, the fitness ($fit_i$) of each solution is calculated. The probability value for each population fitness is calculated using equation (4.13), and the alpha female ($G_\alpha$ ) is selected based on this probability.

$$G_\alpha = \frac{fit_i}{\sum_{i=1}^{n} fit_i} \tag{4.13}$$

The number of mongooses in the alpha group is represented by the $n - bs$, where $bs$ denotes the number of nannies and peep denotes the vocalization made by the dominant female who keeps the family on track. To create a candidate food position, the DMOA uses the expression in equation (4.14).

$$X_{i+1} = X_i + phi \times peep \tag{4.14}$$

where $phi$ is a uniformly distributed random number $[-1, 1]$. Each mongoose sleeps in the first sleeping mound, which is set to $\phi$, the sleeping mound is as given in equation(4.15).

$$sm_i = \frac{fit_{i+1} - fit_i}{max(|fit_{i+1}, fit_i|)} \tag{4.15}$$

Equation (4.16) provides the average number of the discovered sleeping mounds.

$$\phi = \frac{\sum_{i=1}^{n} fit_i}{n} \tag{4.16}$$

The algorithm moves on to the scouting phase, when the next food source or sleeping mound is taken into account, after the babysitting exchange requirement is satisfied.

### 4.7.2 Scout Group

Scouts look for the next sleeping mound to ensure exploration because mongooses are known to avoid old sleeping mounds. According to scouting and foraging are carried out simultaneously in our model. His movement is based on a successful or futile search for a brand-new sleeping mound. In other words, the mongooses' migration depends on how well they do overall. that the family will discover a fresh sleeping mound if they forage far enough. Equation simulates the scout mongoose (4.17) is a representation of the scout mongoose.

$$X_{i+1} = \begin{cases} X_i - cf \times phi \times rand \times [X_i - \vec{M}] & ,\mathrm{if}\phi_{i+1} > \phi_i \\ X_i + cf \times phi \times rand \times [X_i - \vec{M}] & ,otherwise \end{cases} \tag{4.17}$$

where the value of $rand$ is a random number between $[0, 1]$, the value of $(cf)$ is determined by equation (4.18) denotes the parameter that controls the collective-volitive movement of the mongoose group and is decreased linearly along with the iterations, and the value of $M$ is the vector that determines the movement of the mongoose to the new sleeping mound is determined by equation (4.19).

$$cf = (1 - \frac{iter}{max_{iter}})^{2\times\frac{iter}{max_{iter}}} \tag{4.18}$$

$$\vec{M} = \frac{\sum_{i=1}^{n} X_i \times sm_i}{X_i} \tag{4.19}$$

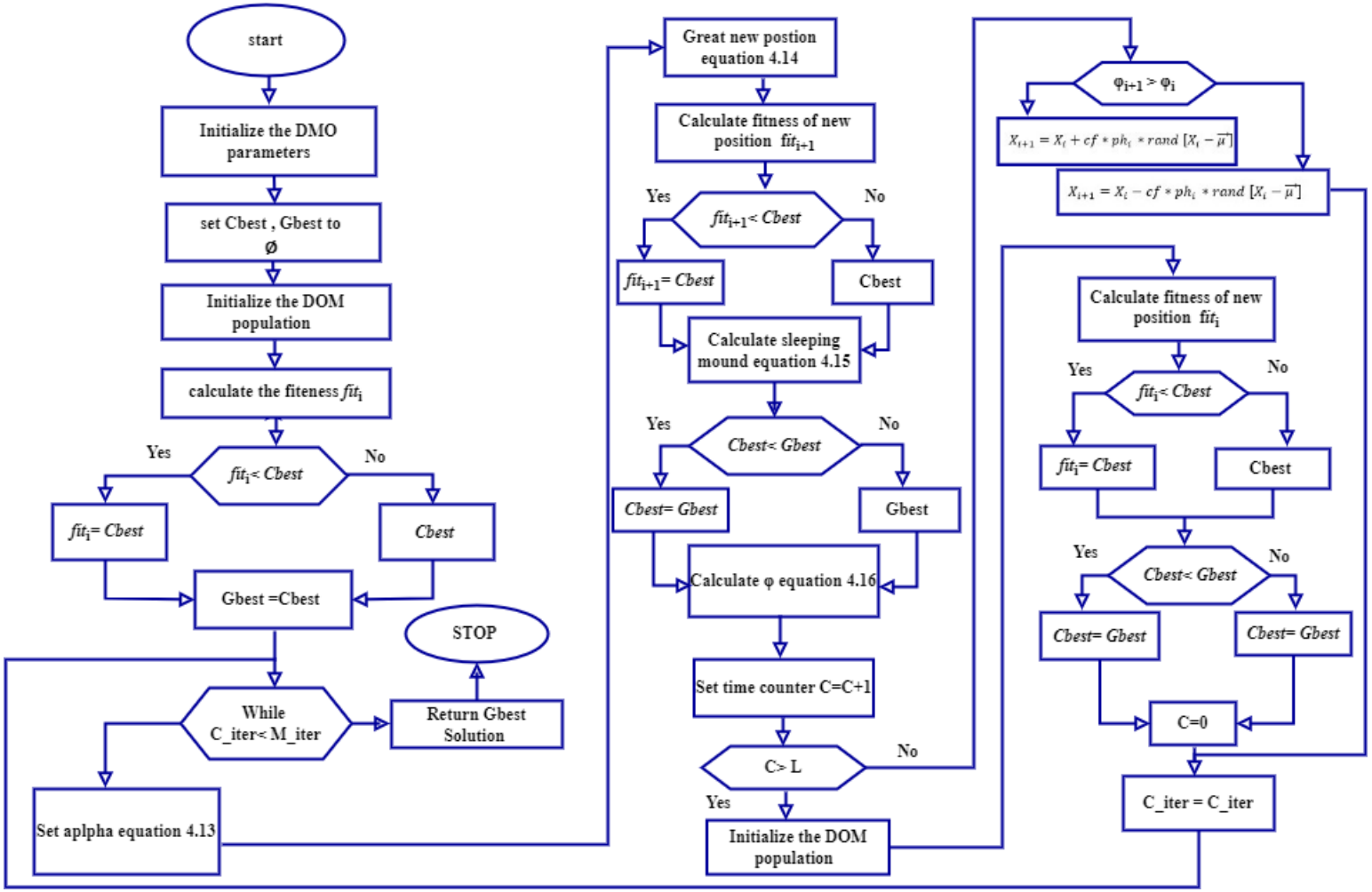



Figure 4.9: Flowchart of the DMOA.

### 4.7.3 Babysitters Group

Babysitters are usually inferior group members that remain with the young and are rotated regularly to allow the alpha female to lead the remainder of the group on daily foraging. Babysitters influence the algorithm by reducing the total population size based on the percentage set, their number is proportional to the size of the population. The babysitter exchange parameter resets the scouting and food source information previously held by the family members replacing them. The simple steps within DMOA manner are presented in the flowchart shown in the **Figure** (4.9) see in Appendix (6.2) contains the pseudo code for the algorithm.

## 4.8. The Proposed Hybrid Dwarf Mongoose Optimization Nelder Mead Algorithm

The proposed Dwarf Mongoose Optimization Nelder Mead algorithm is DMONMA follows the same steps the traditional Dwarf Mongoose Optimization Algorithm . Then to improve the best result from the previous step of the DMOA, the solution obtained from the Dwarf Mongoose Optimization Algorithm is applied to the Nelder Mead method see in section (2.9) for the same of iteration.

## 4.9. Results Experiment for DMOA and DMONMA

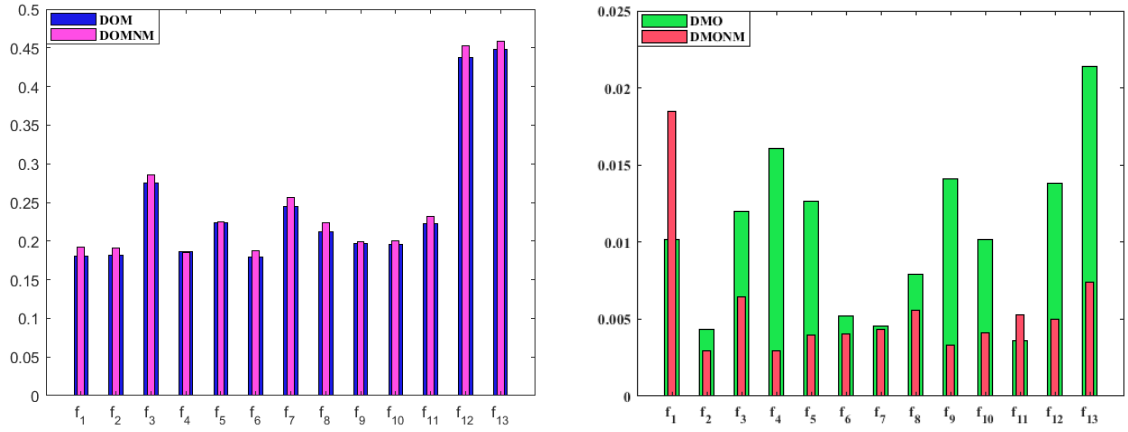
In this section. It was mentioned section (4.4) a set of different benchmark functions, Standard deviation (Std), and the average (Avg), execution time and the best solution of objective functions have been used to evaluate the performance of the DMONMA is compared with DMOA and runs in 500 iterations.

Table 4.8: Comparison Statistical results of DMOA and DMONMA algorithms on unimodal and multimodal functions in 10 dimension.

| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_1$ | DMOA | **1.867E-11** | **2.457E-11** | 1.808E-01 | 1.014E-02 | 0 |
| | DMONMA | 2.439E-11 | 5.679E-11 | 1.920E-01 | 1.846E-02 | 0 |
| $f_2$ | DMOA | **6.304E-07** | **4.244E-07** | 1.816E-01 | 4.331E-03 | 0 |
| | DMONMA | 1.704E-05 | 2.807E-05 | 1.911E-01 | 2.985E-03 | 0 |
| $f_3$ | DMOA | 3.040E+01 | 2.266E+01 | 2.754E-01 | 1.202E-02 | 2.64 |
| | DMONMA | **1.875E-02** | **2.481E-02** | 2.859E-01 | 6.443E-03 | 0 |
| $f_4$ | DMOA | 4.412E-02 | 2.040E-02 | 1.868E-01 | 1.609E-02 | 0 |
| | DMONMA | **4.624E-04** | **7.684E-04** | 1.852E-01 | 2.947E-03 | 0 |
| $f_5$ | DMOA | 1.709E+01 | 2.024E+01 | 2.239E-01 | 1.263E-02 | 0.75 |
| | DMONMA | **6.701E+00** | **9.968E+00** | 2.249E-01 | 3.991E-03 | 0 |
| $f_6$ | DMOA | **1.823E-09** | **2.591E-09** | 1.796E-01 | 5.189E-03 | 0 |
| | DMONMA | 2.550E-09 | 7.883E-09 | 1.878E-01 | 4.070E-03 | 0 |
| $f_7$ | DMOA | 2.224E-02 | 8.666E-03 | 2.452E-01 | 4.590E-03 | 0 |
| | DMONMA | **3.438E-03** | **1.867E-03** | 2.562E-01 | 4.346E-03 | 0 |
| $f_8$ | DMOA | **-5.892E+02** | 3.502E+01 | 2.122E-01 | 7.897E-03 | -636.03 |
| | DMONMA | -5.607E+02 | **2.058E+01** | 2.232E-01 | 5.588E-03 | -624.947 |
| $f_9$ | DMOA | 2.469E+01 | **6.406E+00** | 1.969E-01 | 1.411E-02 | 6.45 |
| | DMONMA | **1.227E+01** | 7.517E+00 | 1.988E-01 | 3.325E-03 | 0 |
| $f_{10}$ | DMOA | 4.572E-06 | **2.487E-06** | 1.955E-01 | 1.016E-02 | 0 |
| | DMONMA | **3.659E-06** | 4.867E-06 | 2.004E-01 | 4.087E-03 | 0 |
| $f_{11}$ | DMOA | 3.129E-01 | 1.013E-01 | 2.230E-01 | 3.584E-03 | 0 |
| | DMONMA | **5.057E-02** | **8.803E-02** | 2.319E-01 | 5.303E-03 | 0 |
| $f_{12}$ | DMOA | **2.668E-10** | **9.512E-10** | 4.375E-01 | 1.378E-02 | 0 |
| | DMONMA | 9.978E-03 | 4.969E-02 | 4.530E-01 | 5.004E-03 | 0 |
| $f_{13}$ | DMOA | **8.014E-08** | **2.580E-07** | 4.482E-01 | 2.136E-02 | 0 |
| | DMONMA | 1.078E-03 | 2.965E-03 | 4.583E-01 | 7.418E-03 | 0 |

**Table** (4.8) show function $\{f_1 - f_{13}\}$ in 10 dimension,results in the proposed DMONMA better for seven the average values of the functions $f_3 - f_5, f_7, f_9 - f_{11}$ from unimodal and multimodal functions, while improving performance the DMONMA the results of

the standard deviation of the functions $f_3 - f_5, f_7, f_8, f_{11}$, in **Figure** (4.10) show the implementation time of finding the average was less for the proposed algorithm in $f_4, f_5$ implementation time of the standard deviation of the DMONMA was less in the functions $f_2 - f_{10}, f_{12}, f_{13}$ . The objective function value in the dimension 10 was equal to the value of the best solution to the functions $f_1, f_2, f_4, f_6, f_7, f_{10} - f_{13}$ in two algorithms, while in the functions $f_3, f_5, f_9$, got values equal to the optimal value in the DMONMA, while in the objective function value of $f_8$ was closer to the optimal in the DMOA.




(a) Time implement Avg ($f_1$–$f_{13}$) (b) Time implement Std ($f_1$–$f_{13}$)

Figure 4.10: Comparison time implement Avg and Std of test functions for DMONMA and DMOA in 10 dimension.

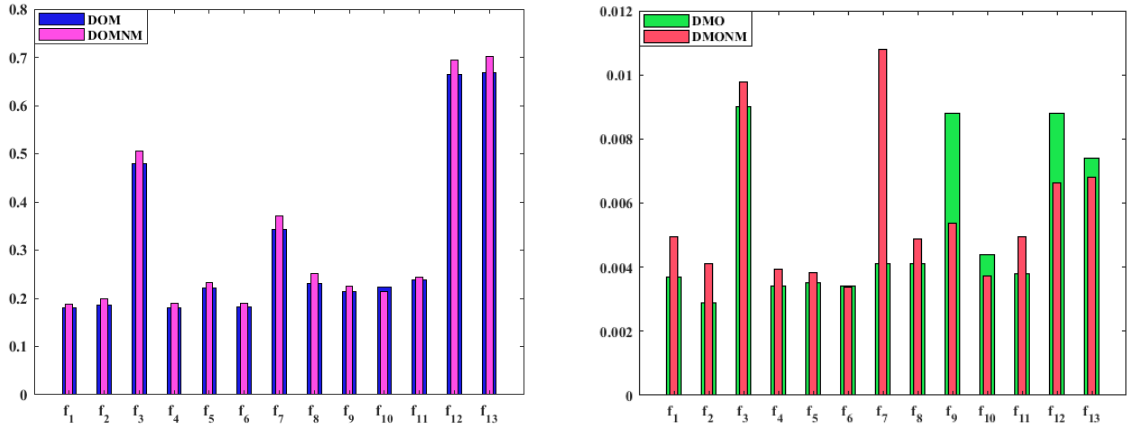
Table 4.9: Comparison Statistical results of DMOA and DMONMA on unimodal and multimodal functions in 30 dimension.

| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_1$ | DMOA | 2.3949 | 0.7485 | 0.1806 | 0.0037 | 0.8 |
| | DMONMA | **8.128E-07** | **5.851E-07** | 1.882E-01 | 4.959E-03 | 0 |
| $f_2$ | DMOA | 67.9443 | 28.4976 | 0.1868 | 0.0029 | 21.3 |
| | DMONMA | **1.022E-01** | **1.528E+00** | 1.985E-01 | 4.124E-03 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $f_3$ | DMOA | 38403.84 | 5559.65 | 0.4800 | 0.0090 | 21697.2 |
| | DMONMA | **1.272E+02** | **6.615E+01** | 5.050E-01 | 9.763E-03 | 28.9 |
| $f_4$ | DMOA | 7.9646 | 0.6810 | 0.1802 | 0.0034 | 5.7 |
| | DMONMA | **7.802E-01** | **2.930E-01** | 1.893E-01 | 3.926E-03 | 0 |
| $f_5$ | DMOA | 674870.96 | 347123.49 | 0.2211 | 0.0035 | 96266.8 |
| | DMONMA | **4.800E+01** | **3.473E+01** | 2.326E-01 | 3.827E-03 | 17.5 |
| $f_6$ | DMOA | 239.7622 | 80.9341 | 0.1825 | 0.0034 | 86.9 |
| | DMONMA | **8.209E-05** | **5.445E-05** | 1.888E-01 | 3.388E-03 | 0 |
| $f_7$ | DMOA | 0.3622 | 0.1006 | 0.3435 | 0.0041 | 0 |
| | DMONMA | **1.101E-02** | **4.061E-03** | 3.703E-01 | 1.081E-0 | 0 |
| $f_8$ | DMOA | **-1108.6010** | **95.5077** | 0.2310 | 0.0041 | -1632.1 |
| | DMONMA | -1.084E+03 | 6.207E+01 | 2.516E-01 | 4.879E-03 | -1325.3 |
| $f_9$ | DMOA | 246.3873 | 17.2681 | 0.2142 | 0.0088 | 177.0 |
| | DMONMA | **1.137E+02** | **5.428E+01** | 2.258E-01 | 5.373E-03 | 5.5 |
| $f_{10}$ | DMOA | 2.9794 | 0.2953 | 0.2236 | 0.0044 | 2.1 |
| | DMONMA | **6.009E-04** | **2.261E-04** | 2.145E-01 | 3.711E-03 | 0 |
| $f_{11}$ | DMOA | 0.6401 | 0.1183 | 0.2391 | 0.0038 | 0.3 |
| | DMONMA | **2.015E-02** | **8.341E-02** | 2.437E-01 | 4.963E-03 | 0 |
| $f_{12}$ | DMOA | 14.1601 | 4.0196 | 0.6642 | 0.0088 | 4.5 |
| | DMONMA | **9.438E-03** | **3.431E-02** | 6.948E-01 | 6.644E-03 | 0 |
| $f_{13}$ | DMOA | 2636754.69 | 2004365.94 | 0.6690 | 0.0074 | 2232895.8 |
| | DMONMA | **3.350E-03** | **7.695E-03** | 7.024E-01 | 6.804E-03 | 0 |

**Table** (4.8) shows function $\{f_1 - f_{13}\}$ in 30 dimension. The average values and the standard deviation value of the proposed DMONMA. It was better in all functions except $f_8$ compared to the results DMOA, in **Figure** (4.11) show the implementation time of finding the average was less for the proposed algorithm in $f_4, f_9, f_{10}$, implementation time of the standard deviation of the DMONMA was less in the functions time was less in the functions $f_6, f_9, f_{10}, f_{12}, f_{13}$. As for the value of the function in dimension 30, got values equal to the values of the objective functions $f_1, f_2, f_4, f_6, f_7, f_{10}, f_{11}, f_{12}, f_{13}$ of the DMONMA, while the values of the functions $f_3, f_5, f_8, f_9$ were closer to the optimal value of the DMONMA. In the function $f_5$, the

value was equal to the optimal value of both algorithms.



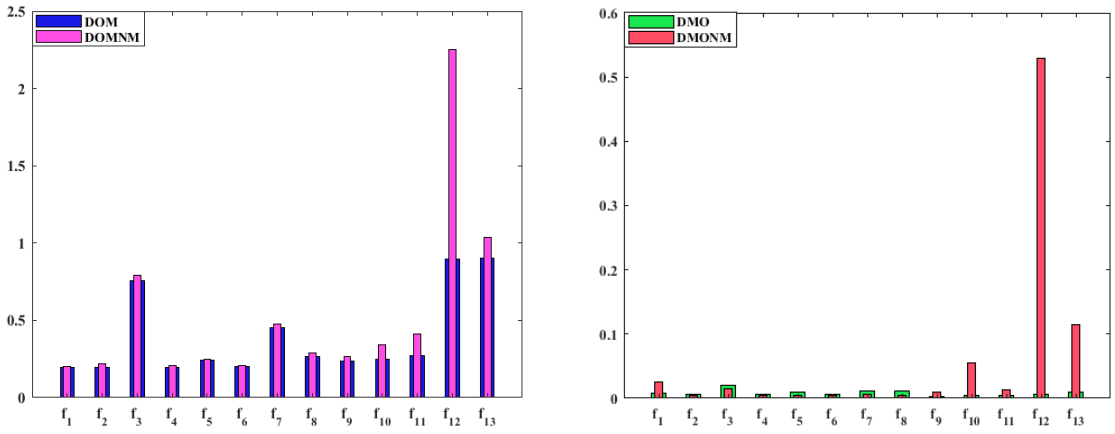

(a) Time implement Avg ($f_1$–$f_{13}$) (b) Time implement Std ($f_1$–$f_{13}$)

Figure 4.11: Comparison time implement Avg and Std of test functions for DMONMA and DMOA in 30 dimension.

Table 4.10: Comparison Statistical results of DMOA and DOMNM on Unimodal and Multimodal functions in 50 dimension.

| Fun | algorithm | Average | Std | Average Time | Std Time | Fun value |
|---|---|---|---|---|---|---|
| $f_1$ | DMOA | 120.1960 | 20.3088 | 0.1946 | 0.0084 | 64.1 |
| | DMONMA | **0.0008** | **0.0004** | 0.2019 | 0.0259 | 0 |
| $f_2$ | DMOA | 6.126E+16 | 1.093E+18 | 0.1975 | 0.0064 | 368.6 |
| | DMONMA | **1.4227** | **3.8658** | 0.2165 | 0.0043 | 0 |
| $f_3$ | DMOA | 119893.04 | 12093.74 | 0.7546 | 0.0209 | 73820.1 |
| | DMONMA | **1097.9611** | **388.7210** | 0.7903 | 0.0147 | 339.5 |
| $f_4$ | DMOA | 10.8098 | 0.3810 | 0.1981 | 0.0063 | 9.1 |
| | DMONMA | **1.5816** | **0.2828** | 0.2056 | 0.0036 | 0.9 |
| $f_5$ | DMOA | 48337923.86 | 14100514.76 | 0.2444 | 0.0096 | 12893706.8 |
| | DMONMA | **159.3110** | **75.3948** | 0.2507 | 0.0042 | 49 |
| $f_6$ | DMOA | 12058.52 | 2101.885 | 0.2000 | 0.0061 | 6622.1 |
| | DMONMA | **0.0828** | **0.0429** | 0.2077 | 0.0041 | 0 |
| $f_7$ | DMOA | 10.8875 | 2.9096 | 0.4541 | 0.0110 | 4.6 |
| | DMONMA | **0.0232** | **0.0077** | 0.4786 | 0.0056 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $f_8$ | DMOA | **-1430.4048** | **114.085** | 0.2637 | 0.01084 | -1814.4 |
| | DMONMA | -1422.9765 | 78.9202 | 0.2899 | 0.0042 | -1738.5 |
| $f_9$ | DMOA | 532.9105 | 25.0751 | 0.2387 | 0.0033 | 433.4 |
| | DMONMA | **235.9255** | **106.4477** | 0.2662 | 0.0088 | 15.2 |
| $f_{10}$ | DMOA | 7.2372 | 0.3823 | 0.2508 | 0 | 0.6 |
| | DMONMA | **0.0258** | **0.1296** | 0.3396 | 0.0557 | 0 |
| $f_{11}$ | DMOA | 1.0915 | 0.0157 | 0.2710 | 0.0052 | 1 |
| | DMONMA | **0.0100** | **0.0702** | 0.4110 | 0.0129 | 0 |
| $f_{12}$ | DMOA | 1140.5349 | 1279.7424 | 0.8972 | 0.0060 | 22.1 |
| | DMONMA | **0.0092** | **0.0248** | 2.2506 | 0.5284 | 0 |
| $f_{13}$ | DMOA | 218165112 | 67690258.3 | 0.90384 | 0.00961 | 61666108.7 |
| | DMONMA | **1.8046** | **4.7521** | 1.0343 | 0.1139 | 0 |




(a) Time implement Avg ($f_1$–$f_{13}$) (b) Time implement Std ($f_1$–$f_{13}$)

Figure 4.12: Comparison time implement Avg and Std of test functions for DMONMA and DMOA in 50 dimension.

**Table** (4.8) shows show function $\{f_1 - f_{13}\}$ in 50 dimension. The results of the proposed DMONMA , compared to the results DMOA algorithm. It was better in all functions except $f_8$ the execution time of the DMONMA was less in the functions $f_2 - f_8$. In **Figure** (4.12) show the implementation time for the average execution time of the DOMA was less than the DMONMA for all functions, the implementation time of the standard deviation of the DMONMA was less in the functions $f_2 - f_8$. The values of the functions

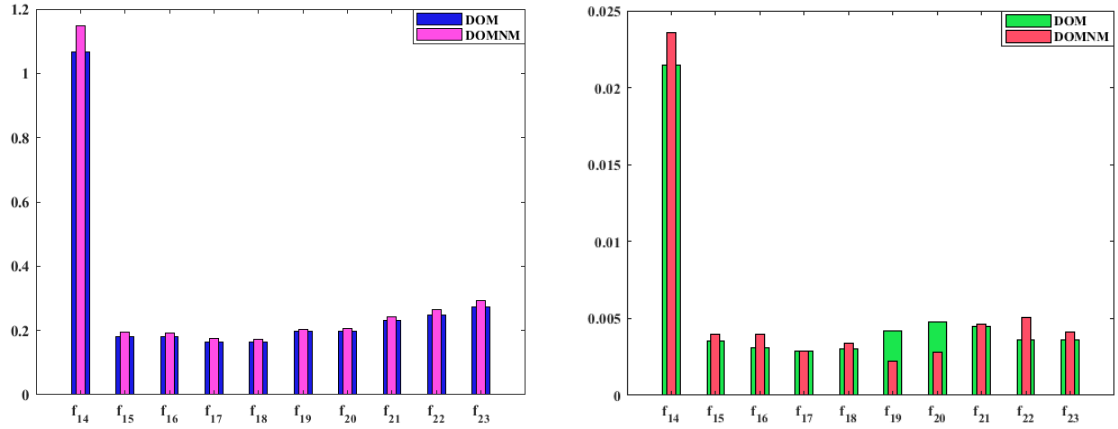
$f_1, f_2, f_6, f_7, f_{10} - f_{13}$ of the DMONMA, while the values of the functions $f_3, f_4, f_5, f_8, f_9$ were closer to the optimal value of the DMONMA .

Table 4.11: Comparison Statistical results of DMOA and DMONNMA fixed dimensional multimodal functions.

| Fun | algorithm | Average | Std | Average Time | Std Time | fun value |
|---|---|---|---|---|---|---|
| $f_{14}$ | DMOA | **0.9980** | **0.0001** | 1.0661 | 0.0215 | 0.998 |
| | DMONMA | 1.1841 | 0.3304 | 1.1481 | 0.0236 | 0.998 |
| $f_{15}$ | DMOA | **0.0012** | **0.0004** | 0.1819 | 0.0035 | 0.0008 |
| | DMONMA | 0.0032 | 0.0023 | 0.1947 | 0.0040 | 0.0007 |
| $f_{16}$ | DMOA | **-1.0316** | **0** | 0.1806 | 0.0031 | -1.0316 |
| | DMONMA | **-1.0316** | **0** | 0.1915 | 0.0040 | -1.0316 |
| $f_{17}$ | DMOA | 0.4707 | 0.0347 | 0.1649 | 0.0029 | 0.398 |
| | DMONMA | **0.4360** | **0.0312** | 0.1753 | 0.0029 | 0.398 |
| $f_{18}$ | DMOA | **3.0000** | **0** | 0.1630 | 0.0030 | 3 |
| | DMONMA | 3.1416 | 0.2031 | 0.1726 | 0.0034 | 3 |
| $f_{19}$ | DMOA | -3.8627 | **0.0005** | 0.1966 | 0.0042 | -3.86 |
| | DMONMA | **-3.8628** | 0.0007 | 0.2045 | 0.0022 | -3.86 |
| $f_{20}$ | DMOA | -3.2703 | 0.0930 | 0.1984 | 0.0048 | -3.32 |
| | DMONMA | **-3.3073** | **0.0591** | 0.2067 | 0.0028 | -3.32 |
| $f_{21}$ | DMOA | **-6.3301** | **1.8023** | 0.2301 | 0.0045 | -10.1513 |
| | DMONMA | -5.2053 | 3.1308 | 0.2422 | 0.0046 | -10.1532 |
| $f_{22}$ | DMOA | -6.0355 | **2.0582** | 0.2492 | 0.0036 | -10.3985 |
| | DMONMA | **-6.3383** | 3.5938 | 0.2647 | 0.0051 | -10.4029 |
| $f_{23}$ | DMOA | -7.2862 | **2.4546** | 0.2738 | 0.0036 | -10.5357 |
| | DMONMA | **-8.9193** | 3.0373 | 0.2927 | 0.0041 | -10.5364 |

**Table** (4.11) shows in fixed dimensional multimodal, the proposed DMONMA algorithm was able to improve five functions for average values $f_{17}, f_{19} - f_{23}$ compared to the results of the DMOA algorithm and the value was equal in $f_{16}$ both algorithms,and the standard deviation value of the proposed DMONMA as able to improve five functions $f_{17}, f_{20}$. In **Figure** (4.13) show the implementation time of finding the average was less for the DMOA algorithm in all functions, the implementation time of the standard deviation of

the DMONMA was less in the functions time was less in the functions $f_{17}, f_{19}, f_{20}$, the objective value, got the exact value of the objective function of the fixed-dimensional multimodal functions in the two algorithms for the functions $f_{14}, f_{16} - f_{20}$. In contrast, got the same values of the functions $f_{21}, f_{22}, f_{23}$ for the proposed DMONMA, and the value of the function $f_{16}$ were closer to the DMONMA.




(a) Time implement Avg ($f_{14}$–$f_{23}$) (b) Time implement Std ($f_{14}$–$f_{23}$)

Figure 4.13: Comparison time implement Avg and Std of fixed dimensional multimodal functions for DMONMA and DMOA.

## 4.10. Comparison Statistical Results and Analysis for All Algorithms

Compared are HBA, HBNMA, DMOA and DMONMA.

1. The **Tables** (4.4, 4.8) and the **Figure** (4.14) show that the results of the average in the dimension equal 10 to the algorithms were better HBNMA results in functions $f_1, f_4, f_6, f_7, f_{12}$. While the function $f_8, f_{13}$ was better in the DMOA. The values were equal for the functions $f_5, f_9 - f_{11}$ in the two algorithms are HBA and HBNMA.

2. The results of the standard deviation were better in the functions $f_1, f_3, f_4, f_6, f_7, f_{12}$ with the HBNMA. In comparison, the results of the functions $f_8, f_{13}$ were better

about the DMONMA and DMOA , respectively, and the values of equality were at the functions $f_5, f_9 - f_{11}$ in the HBNMA and HBA.

3. The implementation time of the Avg it was the HBNMA that had the least execution time in the functions $f_1 - f_4, f_6, f_8 - f_{11}$, while HBA had the least execution time in the functions $f_5, f_7, f_{12}, f_{13}$.

4. The execution time of the standard deviation was less in the functions $f_1, f_3, f_9 - f_{12}$ of the HBNMA and in the HBA it was less execution of the functions $f_2, f_4, f_6, f_7, f_8, f_{13}$. In comparison, the time of execution of the function $f_5$ was less when the DMONMA.

5. Show the **Tables** (4.5, 4.9) and the **Figure** (4.15) in 30 dimensions the HBNMA was able to improve the results of the functions $f_1 - f_4, f_6, f_8, f_{12}, f_{13}$ while it was equal to the HBA algorithm in the functions $f_5, f_7, f_9 - f_{11}$. The results of the standard deviation were better in the functions $f_1, f_3, f_4, f_7, f_{12}, f_{13}$ with HBNMA. In contrast, the DMONMA and DMOA improved the functions $f_6, f_8$ respectively. In contrast, the improvement of the HBNMA and HBA was equal in the functions $f_5, f_9 - f_{11}$.

6. The execution time, the Avg and the standard deviation were less when the HBNMA in the functions $f_2, f_3, f_4, f_6$. In contrast, in the HBA when the functions $f_7, f_8 - f_{13}$, and the time of executing the functions $f_1, f_5$ was less than the DMOA.

7. Show the **Tables** (4.6, 4.10) and the **Figure** (4.16) in 50 dimension the HBNMA was able to improve the results of the functions $f_1 - f_4, f_6, f_8, f_{12}, f_{13}$. At the same time, it was equal to the HBA in the functions $f_5, f_7, f_9, f_{10}, f_{11}$.

8. The results were the standard deviation, where the HBNMA improved the values of each of the functions $f_1, f_3, f_4, f_6, f_7, f_{12}, f_{13}$. In contrast, the DMONMA improved the function $f_8$ and the values of the HBNMA, and HBA were equal in the functions $f_5, f_9 - f_{11}$. The results were where the HBNMA improved the values of each of the

functions $f_1, f_3, f_4, f_6, f_7, f_{12}, f_{13}$. In contrast, the DMONMA improved the function $f_8$ and the values of the HBNMA, and HBA were equal in the functions $f_5, f_9, f_{10}, f_{11}$.

9. The execution time of the Avg was the least HBNMA in the functions $f_1, f_2$ and the execution time of the functions $f_3 - f_{13}$ in the HBA. As for the execution time, the standard deviation was less when the DMOA in the functions $f_9, f_{10}, f_{12}$, while in the DMONMA when the functions $f_2, f_4 - f_6$, and the time of executing the functions $f_1, f_3, f_{11}, f_{13}$ was less than the HBA.

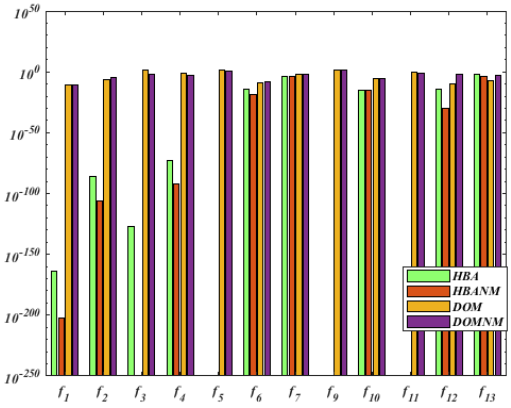
10. Show the **Tables** (4.7, 4.11) and the **Figure** (4.17). In fixed dimensional multimodal functions that the results of the Avg were better DMONMA results in functions $f_{17}, f_{19}, f_{20}, f_{23}$. While the algorithm improved the functions and the best values of the functions $f_{21}, f_{22}$ in the HABNM and HAB, respectively.

11. The value of the standard deviation was equal for the function $f_{16}$ in all algorithms, and the results of the two functions $f_{17}, f_{20}$ were better in the DMONMA while the DMOA $f_{14}, f_{15}, f_{18}, f_{19} - f_{23}$ functions better results in the DMOA.

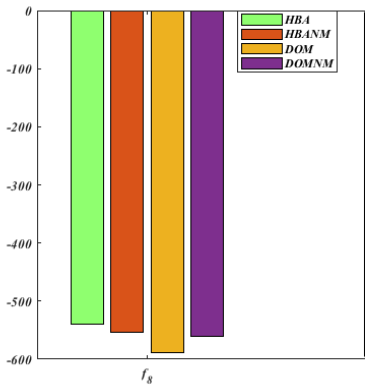
12. Show the implementation time of the Avg the HBNMA performs better functions $f_{15} - f_{18}$ while the implementation time the average the HBA was less in functions $f_{14}, f_{19} - f_{23}$, the implementation time for the standard deviation the execution time was equal in function $f_{15}$ for the two HBNMA and DMOA, and functions $f_{14} - f_{23}$ that were used in the comparison the HBNMA performs better than the HBA in functions most functions $f_{14} - f_{19}, f_{21}, f_{23}$ except $f_{20}, f_{22}$, and the values of the function $f_{16}$ are equal in both algorithms.

13. The implementation time the average was less for the proposed algorithm in the functions $f_{15} - f_{18}$, the time to implement the standard deviation on fixed dimensional multimodal for the proposed algorithm in the functions $f_{16}, f_{20}$. The execution time was better for the HBA, while it was better for the DMOA than for the functions $f_{14} - f_{18}, f_{22}, f_{23}$.

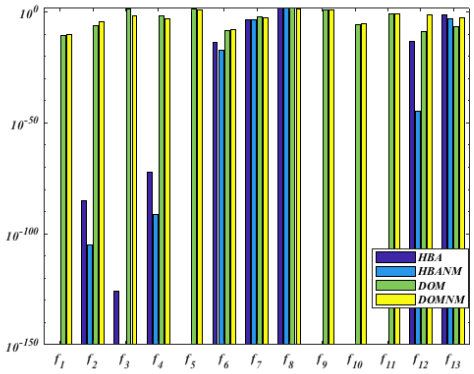
14. The value of the objective functions was equal to the optimal value in all dimensions (10,30,50) except the value $f_8$ results of the proposed algorithm function in 10,30 dimensions that was closer to the optimal value from the HBA. In contrast, in the 50 dimensions, the value of the objective function of a better HBA was equal, and as for the value $f_{12}$ of the function, it was similar in both algorithms. Close to the optimal solution. The exact value of the objective function of the fixed dimensional multimodal functions in the two algorithms for the functions $f_{14}, f_{16} - f_{20}$. In contrast, got the same values of the functions $f_{21} - f_{23}$ for the proposed DMONMA, and the value of the function $f_{16}$ was closer to the DMONMA.



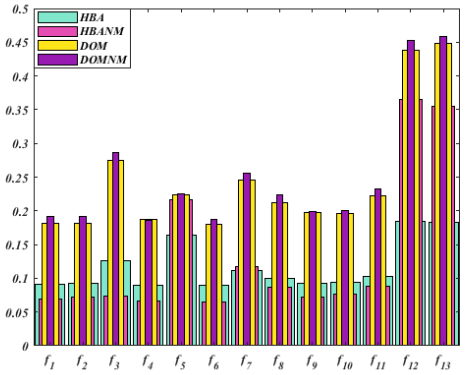

(a) Avg of unimodal and multimodal functions



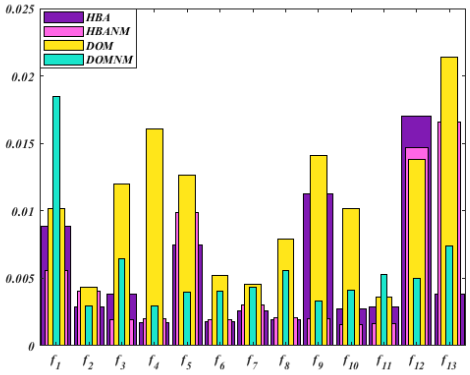

(b) Avg for $f_8$




(c) Std for functions $f_1 - f_{13}$

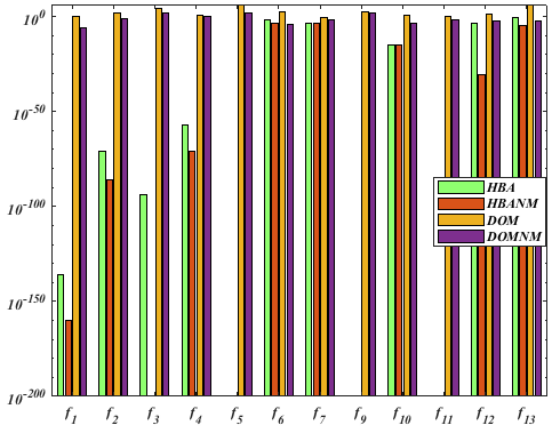



(d) Time implement Avg for functions $f_1 - f_{13}$

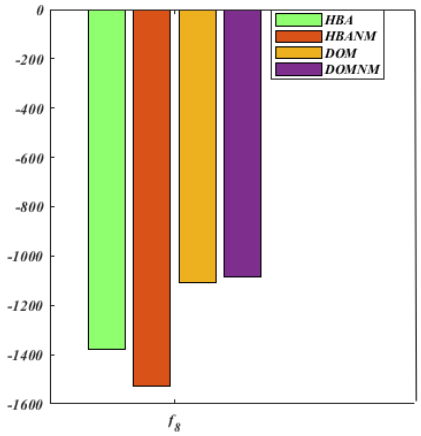



(e) Time implement Std for functions $f_1 - f_{13}$

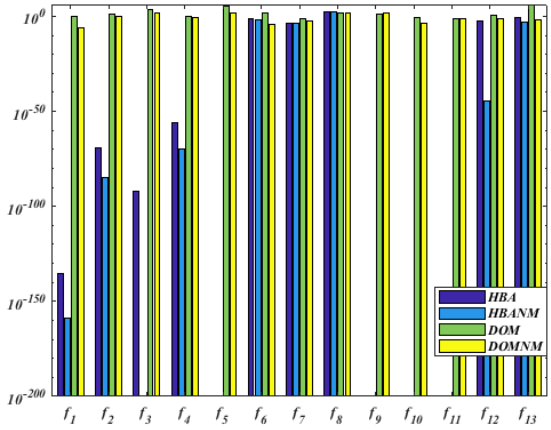
Figure 4.14: Comparison HBA, HBNMA,DMOA and DMONMA in 10 dimensions.



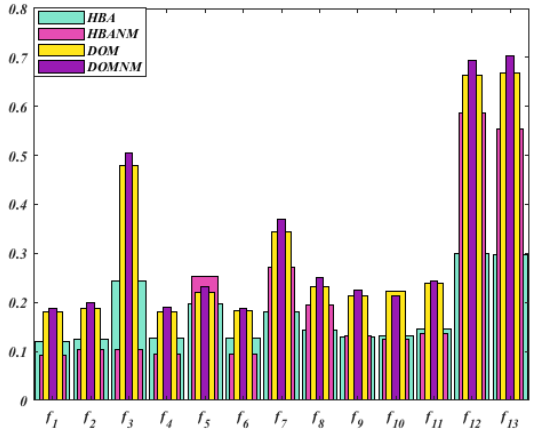

(a) Avg of unimodal and multimodal functions



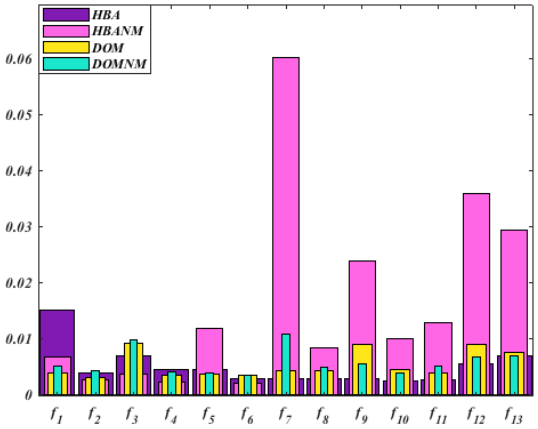

(b) Avg for $f_8$




(c) Std for functions $f_1 - f_{13}$

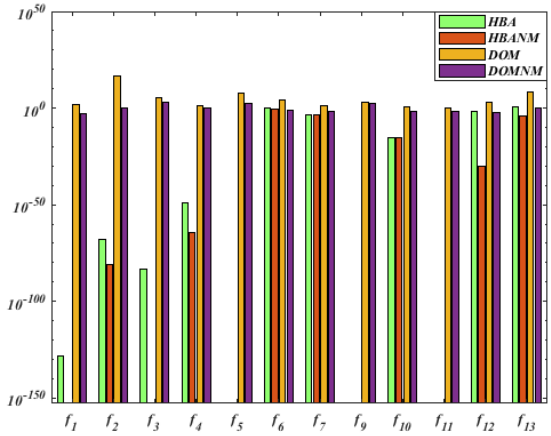



(d) Time implement Avg for functions $f_1 - f_{13}$

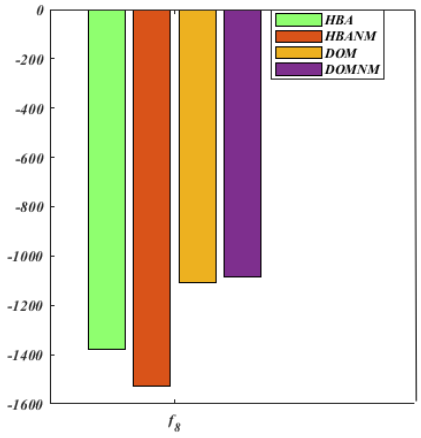



(e) Time implement Std for functions $f_1 - f_{13}$

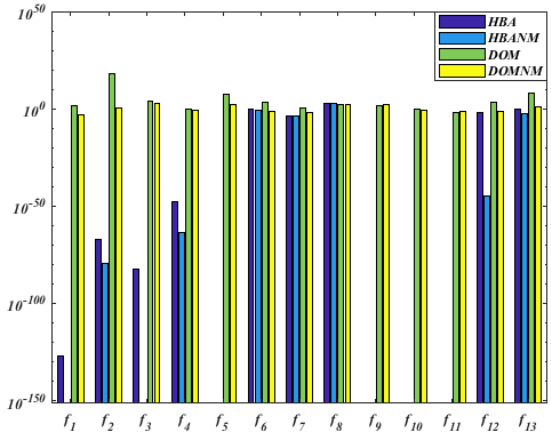
Figure 4.15: Comparison HBA, HBNMA,DMOA and DMONMA in 30 dimensions.



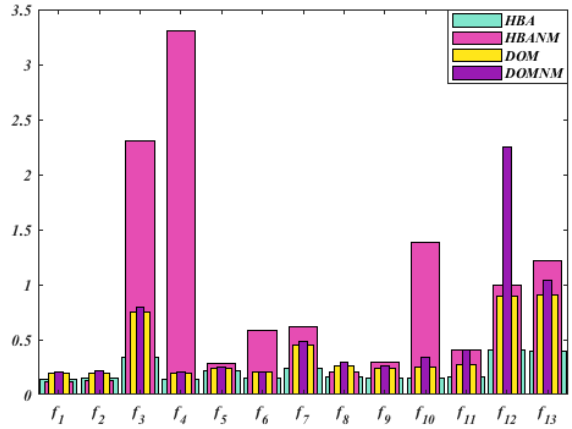

(a) Avg of unimodal and multimodal functions



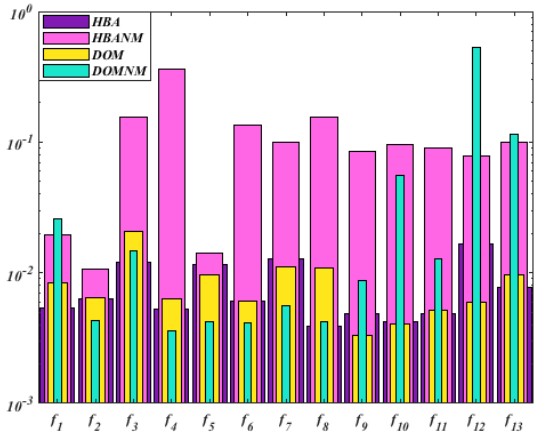

(b) Avg for $f_8$




(c) Std for functions $f_1 - f_{13}$

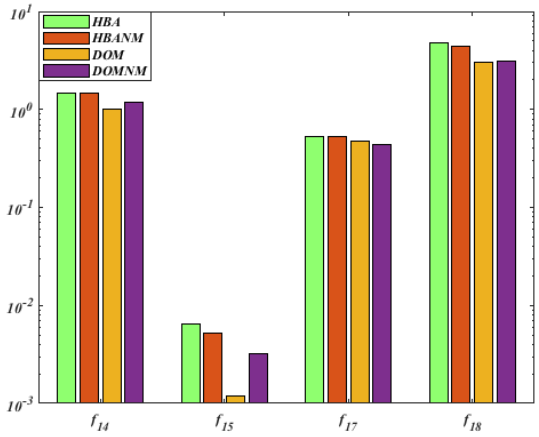



(d) Time implement Avg for functions $f_1 - f_{13}$

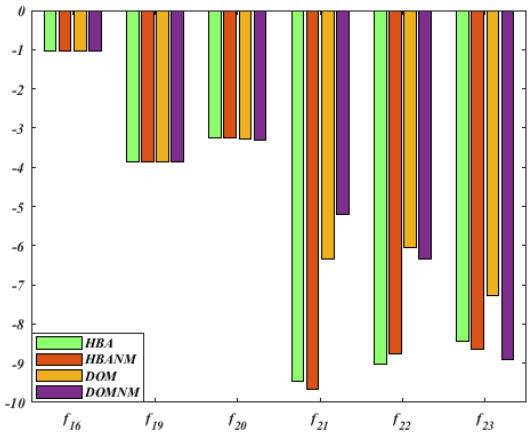



(e) Time implement Std for functions $f_1 - f_{13}$

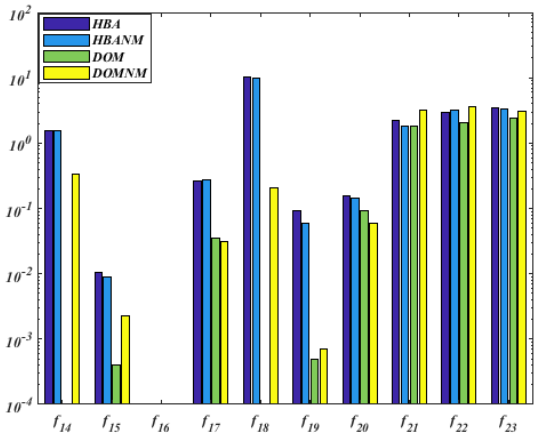
Figure 4.16: Comparison HBA, HBNMA, DMOA and DMONMA in 50 dimensions.



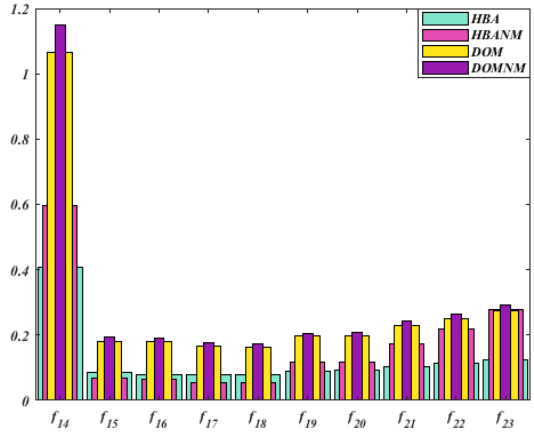

(a) Avg of functions $f_{14}, f_{15}, f_{17}, f_{18}$



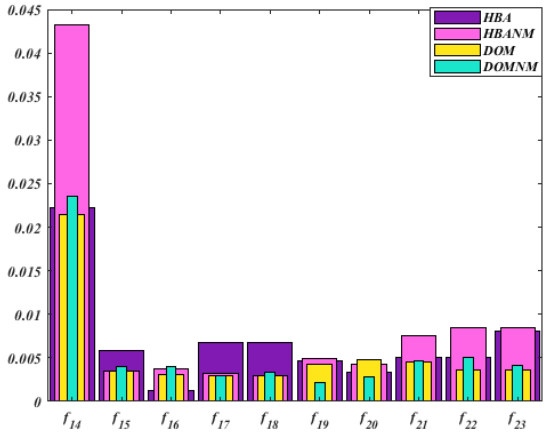

(b) Avg of functions $f_{16}, f_{19}, f_{23}$




(c) Std for functions $f_{14} - f_{23}$




(d) Time implement Avg for functions $f_{14} - f_{23}$




(e) Time implement Std for functions $f_{14} - f_{23}$

Figure 4.17: Comparison Avg of functions for multimodal functions with fixed dimensions.

CHAPTER 5

# PENALTY METHOD BASED OF META-HEURISTICS ALGORITHMS FOR SOLVING RELIABILITY OPTIMIZATION PROBLEMS

## 5.1. Introduction

To provide an algorithm that reaches the best solution values for solving constrained nonlinear optimization problems by combining the penalty method, as illustrated in section (2.10), with algorithms as explained in previous chapter HBA, HBNMA, DMOA, and DMONMA achieving better solutions for solving multi-objectives function problems which dealing with both objectives of design is a tremendous challenge for the decision maker to reliability system create a model. It is usually preferable to simultaneously maximize system dependability and reduce cost usage when creating a reliability system. Even though a single solution that is optimal for all objectives may not exist, it is preferable to use a multi-objective approach to system design when the limitations on cost usage are variable or need to be accurately and adequately determined. The challenge for the designer in these circumstances is to optimize all objectives concurrently [85], and used the weighted sum method mentioned in subsection 2.11.4 to solve this problem and find pareto optimal solutions.

## 5.2. Multi-Objective System Reliability Optimization

The mathematical model for multi-objective system reliability allocation by using objective function is written as follows [103]:

$$\begin{aligned} & max \quad R_s(R_i) \\ & min \quad C_s(R_i) \\ & \text{subject to :} \qquad R_{min} \leq R_s \leq R_{max} \\ & \qquad\qquad a \leq R_i \leq b \ , for \quad i = 1, ..., n, \quad a, b \in [0.5, 1] \end{aligned} \tag{5.1}$$

where $R_s$ is the overall system reliability, $C_s$ is the system cost, $R_{min}$ is its minimum allowable value $R_{max}$ is its maximum allowable value, $R_i$ is the $i$-th component reliability. In order to resolve the multi-objective system reliability optimization problem, the weighted sum method see in subsection 2.11.4 has been used. The single-objective function is created by converting the objective functions of equation (5.1) and $maxR_s = -minR_s$ as follows:

$$minf(R_i) = \omega_1 C_s - \omega_2 R_s + \psi(R_i) \tag{5.2}$$

where $\omega_1$ and $\omega_2$ is the weight vector, such that $\omega_1 + \omega_2 = 1$, where $\psi(R_i)$ is the penalty function (see in section (2.10)) is computed as follows

$$\psi(R_i) = \vartheta_1 max\{0, R_{max} - R_s\} + \vartheta_2 max\{0, R_s - R_{min}\} \tag{5.3}$$

where $\vartheta_1, \vartheta_2$ are the penalty factor present three the mathematical model to case studies for system configuration. Using the algorithms mentioned in the previous chapter are HBA, HBNMA, DMOA, and DMONMA, calculating them in two ways, using the penalty with method and another without using the penalty method, used number of iterations equal 500. To find out the best reliability values of a complex network the results were rounded to four decimal places, and compare results values of components reliability $R_i$, reliability system $R_s$, total cost $C_s$ and compared run times in seconds, in study cases take $\omega_1 = \omega_2 = 0.5$ (to avoid being biased towards the objective function in favour of another objective function).

***Note***. *The nature of reliability function and cost function are not two contradictory functions. That is, increasing the value of one leads to visiting the other and decreasing the value of one of them leads to reducing the value of the other. That is, if the reliability increases, the cost increases because the cost calculation depends on the value of reliability.*

### 5.2.1 Mathematical model I

$$\begin{aligned} &max \quad R_s(R_i) \\ &min \quad C_s(R_i) = \sum_{i=1}^{11} a_i(tan(\frac{\pi}{2})R_i)^{\kappa_i} \\ &\text{subject to} : 0.95 \leq R_S \leq 0.9999 \\ &\qquad 0.7 \leq R_i \leq 0.9999, \quad for \quad i = 1, ..., 11 \end{aligned} \tag{5.4}$$

maximize the reliability network $R_s$ represented in equation (3.9), and $a_i = 0.0001$, $\kappa_i = 2$ for all $i$.

#### 5.2.1.1 Computational Results of the HBA

To find out the best value of reliability of system by using the HBA and penalty method with HBA is (P-HBA). Illustrate the **Table** (5.1) and the **Figure** (5.1) shows that the HBA results for values between $0.7015 \leq R_i \leq 0.9635$ and P-HBA results for values between $0.8357 \leq R_i \leq 0.9933$, the best value of reliability system in HBA was $R_s = 0.9500$ and $R_s = 0.9924$ in P-HBA , total cost in HBA was $C_s = 0.0586$, in P-HBA was $C_s = 1.1775$, and the HBA execution time was 0.2643 and P-HBA execution time was 0.5806.

Table 5.1: Comparison for results values of P-HBA and HBA for model I.

| Components No | HBA | | P-HBA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9635 | 0.0304 | 0.9933 | 0.9028 |
| $R_2$ | 0.9228 | 0.0067 | 0.9777 | 0.0814 |
| $R_3$ | 0.9068 | 0.0046 | 0.9668 | 0.0367 |
| $R_4$ | 0.9165 | 0.0057 | 0.9766 | 0.0739 |

| | | | | |
|---|---|---|---|---|
| $R_5$ | 0.8707 | 0.0024 | 0.9527 | 0.0180 |
| $R_6$ | 0.8364 | 0.0014 | 0.9385 | 0.0106 |
| $R_7$ | 0.7015 | 3.8971e-04 | 0.8357 | 0.0014 |
| $R_8$ | 0.8736 | 0.0025 | 0.9573 | 0.0222 |
| $R_9$ | 0.7053 | 4.0147e-04 | 0.8642 | 0.0021 |
| $R_{10}$ | 0.8484 | 0.0017 | 0.9458 | 0.0137 |
| $R_{11}$ | 0.8724 | 0.0024 | 0.9473 | 0.0145 |
| $R_s$ | 0.9500 | | 0.9924 | |
| $C_s$ | 0.0586 | | 1.1775 | |
| time-run | 0.2643 | | 0.5806 | |

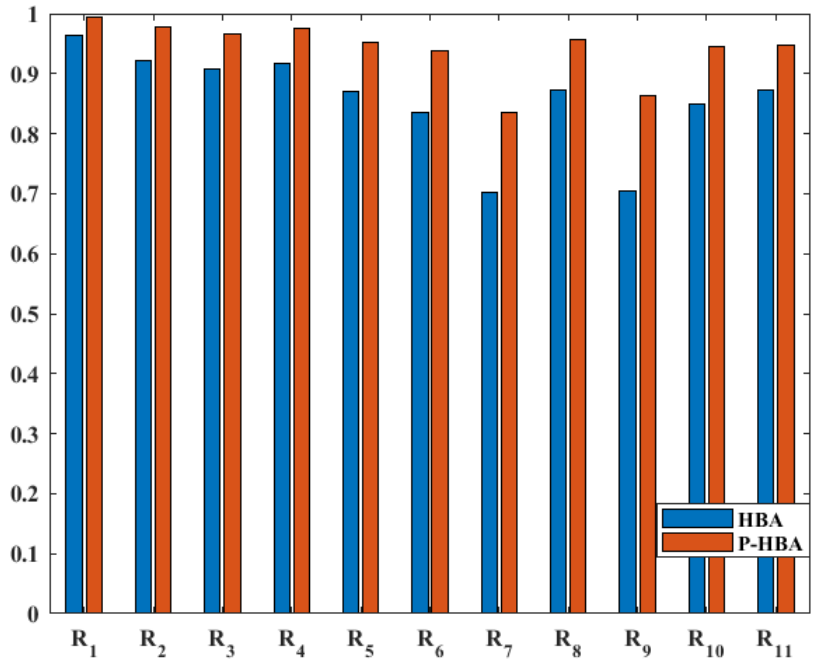



Figure 5.1: Comparison results values $R_i$ of P-HBA and HBA for model I.

#### 5.2.1.2 Computational Results of the HBNMA

Employing the HBNMA algorithm and the Penalty method with HBNMA P-HBNMA, in **Table** (5.2) illustrate and the **Figure** (5.2) shows the HBNMA results for values between 07844 $\leq R_i \leq$ 0.9668 and P-HBNMA results for values between 0.8277 $\leq R_i \leq$ 0.9933. The best reliability system value for the HBNMA was $R_s = 0.9547$ and $R_s = 0.9925$ for the P-HBNMA . Total cost for the HBA was $C_s = 0.0685$ while for the P-HBNMA it was $C_s = 1.1788$, HBNMA execution time was 0.1479 while for the P-HBNMA execution time it was 0.1934.

Table 5.2: Comparison for results values of HBNMA and P-HBNMA for model I.

| Components No | HBNMA | | P-HBNMA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9668 | 0.0367 | 0.9933 | 0.9028 |
| $R_2$ | 0.9239 | 0.0069 | 0.9775 | 0.0800 |
| $R_3$ | 0.8913 | 0.0034 | 0.9659 | 0.0348 |
| $R_4$ | 0.9226 | 0.0067 | 0.9767 | 0.0746 |
| $R_5$ | 0.8923 | 0.0034 | 0.9535 | 0.0187 |
| $R_6$ | 0.8307 | 0.0013 | 0.9365 | 0.0100 |
| $R_7$ | 0.7920 | 8.7083e-04 | 0.8277 | 0.0013 |
| $R_8$ | 0.8891 | 0.0032 | 0.9564 | 0.0213 |
| $R_9$ | 0.7844 | 8.0600e-04 | 0.8634 | 0.0021 |
| $R_{10}$ | 0.8404 | 0.0015 | 0.9442 | 0.0129 |
| $R_{11}$ | 0.8954 | 0.0036 | 0.9555 | 0.0204 |
| $R_s$ | 0.9547 | | 0.9925 | |
| $C_s$ | 0.0685 | | 1.1788 | |
| time-run | 0.1479 | | 0.1934 | |

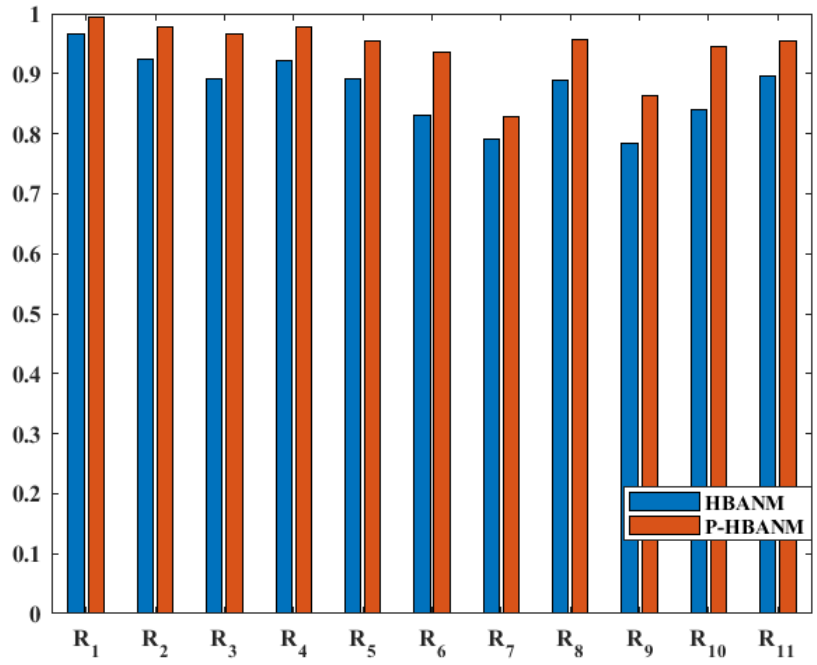



Figure 5.2: Comparison results values $R_i$ of P-HBNMA and HBNMA for model I.

#### 5.2.1.3 Computational Results of the DMOA

In order to identify which system reliability by using the DMOA and the Penalty method with DMOA is P-DMOA. Results were represented in the **Table** (5.3) and the **Figure** (5.3) shows the DMOA results for values between 07105 $\leq R_i \leq$ 0.9639 and P-DMOA results for values between 0.7477 $\leq R_i \leq$ 0.9933, the best reliability system value for the DMOA was $R_s = 0.9513$ and $R_s = 0.9921$ for the P-DMOA m. Total cost for the DMOA was $C_s = 0.0626$ while for the P-DMOA, it was $C_s = 1.1354$ and DMOA execution time was 31.5609 while for the P-DMOA execution time it was 83.2784.

Table 5.3: Comparison for results values of P-DMOA and DMOA for model I.

| Components No | DMOA | | P-DMOA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9639 | 0.0310 | 0.9933 | 0.9028 |
| $R_2$ | 0.9236 | 0.0069 | 0.9718 | 0.0509 |
| $R_3$ | 0.9215 | 0.0065 | 0.9610 | 0.0266 |
| $R_4$ | 0.9222 | 0.0066 | 0.9776 | 0.0807 |
| $R_5$ | 0.8615 | 0.0020 | 0.9659 | 0.0348 |
| $R_6$ | 0.8094 | 0.0010 | 0.8963 | 0.0037 |
| $R_7$ | 0.8710 | 0.0024 | 0.8324 | 0.0014 |
| $R_8$ | 0.8696 | 0.0023 | 0.9450 | 0.0133 |
| $R_9$ | 0.7105 | 4.1833e-04 | 0.7477 | 5.7109e-04 |
| $R_{10}$ | 0.8251 | 0.0013 | 0.9158 | 0.0057 |
| $R_{11}$ | 0.8629 | 0.0021 | 0.9481 | 0.0150 |
| $R_s$ | 0.9510 | | 0.9921 | |
| $C_s$ | 0.0626 | | 1.1354 | |
| time-run | 31.5609 | | 83.2784 | |

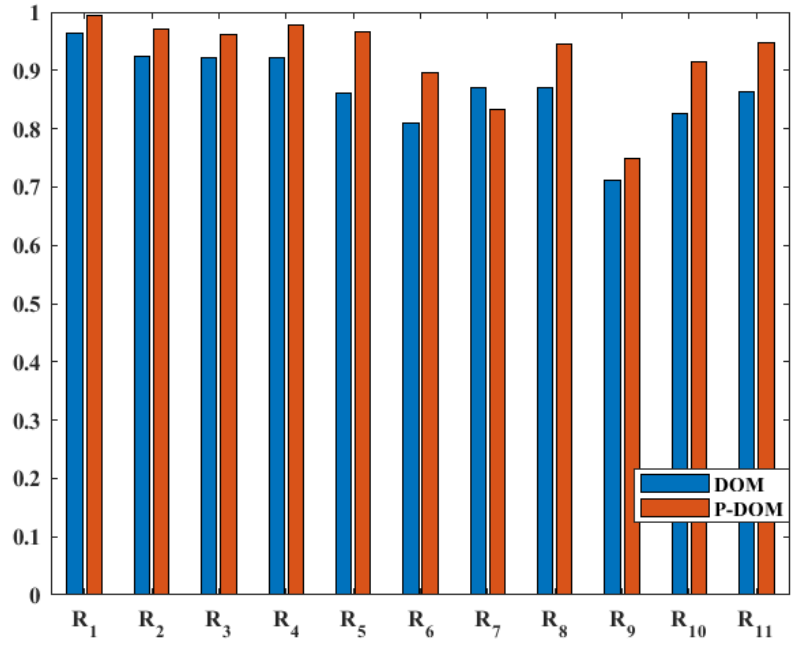



Figure 5.3: Comparison results values $R_i$ of P-DMOA and DMOA for model I.

#### 5.2.1.4 Computational Results of the DMONMA

The using DMONMA algorithm without penalty method and the Penalty method with DMONMA is P-DMONMA. Results were represented in the **Table** (5.4) and the **Figure** (5.4) were shown the DMONMA results for values between 07362 $\leq R_i \leq$ 0.9649 and P-DMONMA results for values between 0.8009 $\leq R_i \leq$ 0.9933. The best reliability system value for the DMONMA was $R_s = 0.9513$ and $R_s = 0.9924$ for the P-DMONMA. Total cost for the DMONMA was $C_s = 0.0624$ while for the P-DMONMA it was $C_s = 1.1659$ and DMONMA execution time was 3.8252 while for the P-DMONMA execution time it was 0.2412.

Table 5.4: Comparison for results values of P-DMONMA and DMONMA for model I.

| Components No | DMONMA | | P-DMONMA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9649 | 0.0328 | 0.9933 | 0.9028 |
| $R_2$ | 0.9103 | 0.0050 | 0.9794 | 0.0954 |
| $R_3$ | 0.9090 | 0.0048 | 0.9650 | 0.0330 |
| $R_4$ | 0.9175 | 0.0059 | 0.9742 | 0.0608 |
| $R_5$ | 0.8924 | 0.0034 | 0.9388 | 0.0108 |
| $R_6$ | 0.8428 | 0.0016 | 0.9349 | 0.0095 |
| $R_7$ | 0.7881 | 8.3669e-04 | 0.8241 | 0.0012 |
| $R_8$ | 0.9005 | 0.0040 | 0.9594 | 0.0245 |
| $R_9$ | 0.7362 | 5.1690e-04 | 0.8009 | 9.5639e-04 |
| $R_{10}$ | 0.8407 | 0.0015 | 0.9448 | 0.0132 |
| $R_{11}$ | 0.8596 | 0.0020 | 0.9456 | 0.0136 |
| $R_s$ | 0.9513 | | 0.9924 | |
| $C_s$ | 0.0624 | | 1.1659 | |
| time-run | 3.8252 | | 0.2412 | |

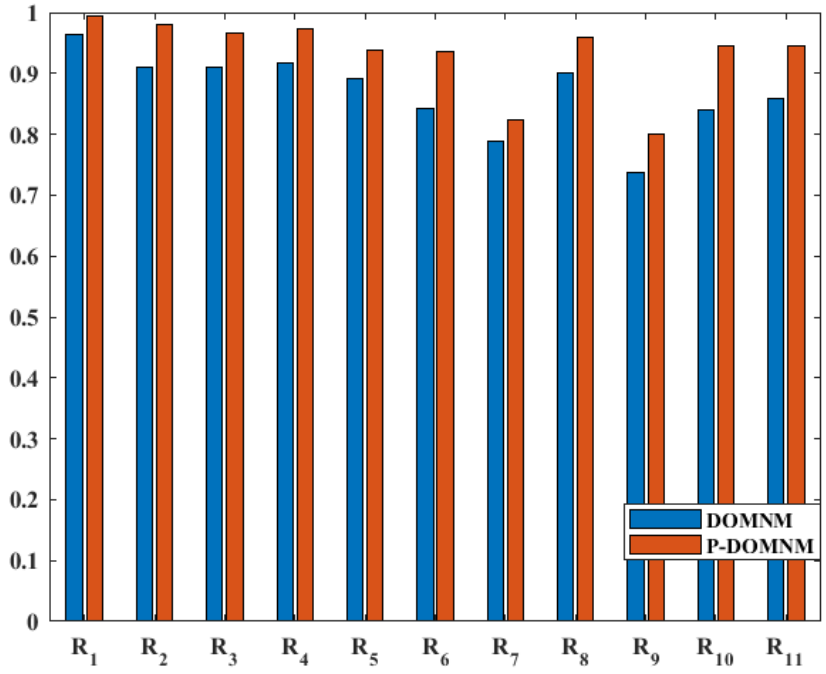



Figure 5.4: Comparison results values $R_i$ of P-DMONMA and DMONMA for model I.

### 5.2.2 Mathematical model II

$$\begin{aligned} &max \quad R_s(R_i) \\ &min \quad C_s(R_i) = \sum_{i=1}^{11} a_i \exp(\frac{b}{1-R_i}) \\ &\text{subject to} : 0.95 \leq R_s \leq 0.9999 \\ &\qquad\qquad 0.7 \leq R_i \leq 0.9999, \quad for \quad i = 1, ..., 11 \end{aligned} \tag{5.5}$$

where $R_s$ represented in equation (3.9), and $a_i = 0.01$ and $b = 0.03$ for all $i$.

#### 5.2.2.1 Computational Results of the HBA

The HBA and the Penalty method with HBA is (P-HBA), illustrate the **Table** (5.5) and the **Figure** (5.5) shows HBA results for values between $0.7000 \leq R_i \leq 0.9708$ and P-HBA results for values between $0.7000 \leq R_i \leq 0.9924$, the best value of reliability system in HBA was $R_s = 0.9586$ and $R_s = 0.9918$ in P-HBA, total cost in HBA was $C_s = 0.1561$ in P-HBA algorithm was $C_s = 0.8063$, and HBA execution time was 0.2196 and P-HBA execution time was 1.0234.

Table 5.5: Comparison for results values of P-HBA and HBA for model II

| Components No | HBA | | P-HBA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9708 | 0.0279 | 0.9924 | 0.5180 |
| $R_2$ | 0.9321 | 0.0156 | 0.9823 | 0.0545 |
| $R_3$ | 0.9176 | 0.0144 | 0.9806 | 0.0469 |
| $R_4$ | 0.9225 | 0.0147 | 0.9824 | 0.0550 |
| $R_5$ | 0.8699 | 0.0126 | 0.9700 | 0.0272 |
| $R_6$ | 0.8105 | 0.0117 | 0.9454 | 0.0173 |
| $R_7$ | 0.7000 | 0.0111 | 0.7000 | 0.0111 |
| $R_8$ | 0.8666 | 0.0125 | 0.9559 | 0.0197 |
| $R_9$ | 0.7000 | 0.0111 | 0.7000 | 0.0111 |
| $R_{10}$ | 0.8297 | 0.0119 | 0.7000 | 0.0111 |
| $R_{11}$ | 0.8705 | 0.0126 | 0.9758 | 0.0345 |
| $R_s$ | 0.9586 | | 0.9918 | |
| $C_s$ | 0.1561 | | 0.8063 | |
| time-run | 0.2196 | | 1.0234 | |

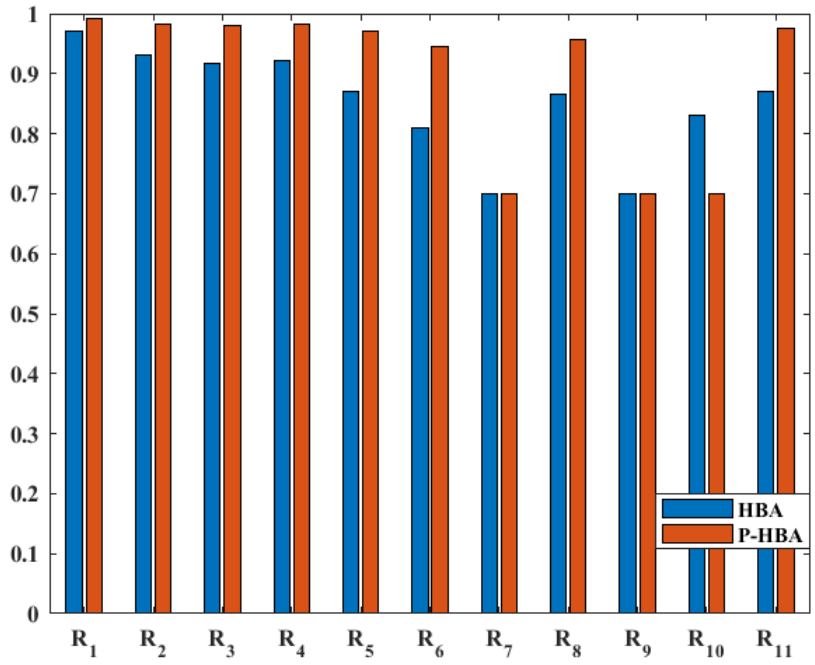



Figure 5.5: Comparison results values $R_i$ of P-HBA and HBA for model II

#### 5.2.2.2 Computational Results of the HBNMA

Using the HBNMA and the Penalty method with HBNMA is P-HBNMA, the **Table** (5.6) and the **Figure** (5.6) shows the results for the HBNMA for values between $07666 \leq R_i \leq 0.9731$ and the results for the P-HBNMA for values between $0.7919 \leq R_i \leq 0.9924$ . The best reliability system value for the HBNMA was $R_s = 0.9616$ while the P-HBNMA was $R_s = 0.9919$ the HBNMA total cost was $C_s = 0.1605$ and total cost for was P-HBNMA algorithm was $C_s = 0.7893$, and its execution time was 0.2830 while the P-HBNMA was 0.2025.

Table 5.6: Comparison for results values of HBNMA and P-HBNMA for model II.

| Components No | HBNMA | | P-HBNMA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9731 | 0.0305 | 0.9924 | 0.5180 |
| $R_2$ | 0.9363 | 0.0160 | 0.9825 | 0.0555 |

| | | | | |
|---|---|---|---|---|
| $R_3$ | 0.9021 | 0.0136 | 0.9733 | 0.0308 |
| $R_4$ | 0.9199 | 0.0145 | 0.9821 | 0.0534 |
| $R_5$ | 0.8883 | 0.0131 | 0.9636 | 0.0228 |
| $R_6$ | 0.8448 | 0.0121 | 0.9483 | 0.0179 |
| $R_7$ | 0.7917 | 0.0115 | 0.7919 | 0.0116 |
| $R_8$ | 0.8722 | 0.0126 | 0.9656 | 0.0239 |
| $R_9$ | 0.7666 | 0.0114 | 0.8485 | 0.0122 |
| $R_{10}$ | 0.8391 | 0.0120 | 0.9554 | 0.0196 |
| $R_{11}$ | 0.8850 | 0.0130 | 0.9652 | 0.0237 |
| $R_s$ | 0.9616 | | 0.9919 | |
| $C_s$ | 0.1605 | | 0.7893 | |
| time-run | 0.2830 | | 0.2025 | |

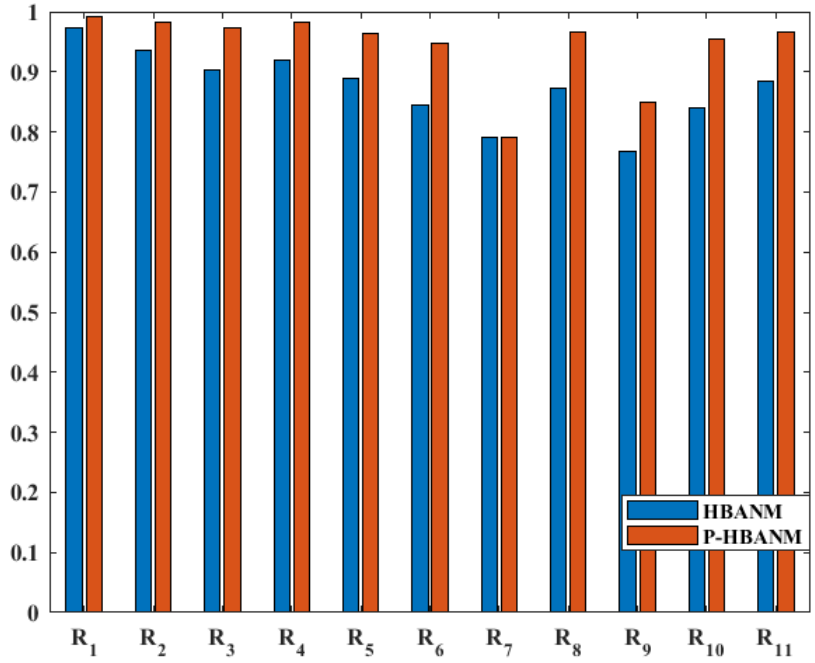



Figure 5.6: Comparison results values $R_i$ of HBNMA and P-HBNMA for model II.

#### 5.2.2.3 Computational Results of the DMOA

The DMOA and the Penalty method with DMOA is P-DMOA. Results were displayed in the **Table** (5.7) and the Figure (5.7) shows the P-DMOA results for values between $0.7333 \leq R_i \leq 0.9923$ and the DMOA results for values between $0.7333 \leq R_i \leq 0.9702$. The best reliability system value for the P-DMOA was $R_s = 0.9917$, and the best reliability system value for the DMOA was $R_s = 0.9585$. The total cost for the DMOA was $C_s = 0.1563$, but the total cost for the P-DMOA was $C_s = 0.7372$. The execution time for the DMOA was 13.2938, whilst the execution time for the P-DMOA was 42.9952.

Table 5.7: Comparison for results values of P-DMOA and DMOA for model II.

| Components No | DMOA | | P-DMOA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9702 | 0.0274 | 0.9923 | 0.4921 |
| $R_2$ | 0.9331 | 0.0157 | 0.9817 | 0.0515 |
| $R_3$ | 0.9188 | 0.0145 | 0.9669 | 0.0248 |
| $R_4$ | 0.9198 | 0.0145 | 0.9799 | 0.0445 |
| $R_5$ | 0.8865 | 0.0130 | 0.9607 | 0.0215 |
| $R_6$ | 0.8102 | 0.0117 | 0.9312 | 0.0155 |
| $R_7$ | 0.7333 | 0.0112 | 0.7333 | 0.0112 |
| $R_8$ | 0.8588 | 0.0124 | 0.9615 | 0.0218 |
| $R_9$ | 0.7498 | 0.0113 | 0.8105 | 0.0117 |
| $R_{10}$ | 0.8204 | 0.0118 | 0.9484 | 0.0179 |
| $R_{11}$ | 0.8811 | 0.0129 | 0.9670 | 0.0248 |
| $R_s$ | 0.9585 | | 0.9917 | |
| $C_s$ | 0.1563 | | 0.7372 | |
| time-run | 13.2938 | | 42.9952 | |

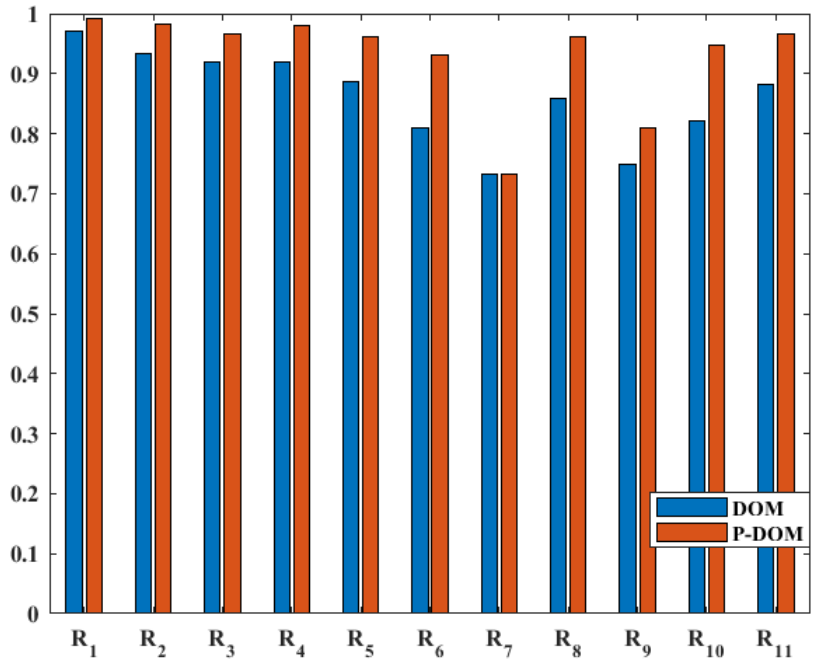



Figure 5.7: Comparison results values $R_i$ of P-DMOA and DMOA for model I

#### 5.2.2.4 Computational Results of the DMONMA

The using without the penalty method and the P-DMONMA with the penalty method. The results were represented in the **Table** (5.8) and the **Figure** (5.8) shows the DMONMA algorithm results were shown for values between $0.7440 \leq R_i \leq 0.9713$ and the P-DMONMA results were shown for values between $0.7931 \leq R_i \leq 0.9924$. The best reliability system value for the DMONMA was $R_s = 0.9609$ and the best reliability system value for the P-DMONMA was $R_s = 0.9919$ and the total cost for the DMONMA was $C_s = 0.1591$, where as the total cost for the P-DMONMA was $C_s = 0.7871$, and the execution time was 1.3424 as opposed to 0.2382 for the P-DMONMA.

Table 5.8: Comparison for results values of P-DMONMA and DMONMA for model II.

| Components No | DMONMA | | P-DMONMA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9713 | 0.0284 | 0.9924 | 0.5180 |
| $R_2$ | 0.9393 | 0.0164 | 0.9824 | 0.0550 |
| $R_3$ | 0.9159 | 0.0143 | 0.9730 | 0.0304 |
| $R_4$ | 0.9236 | 0.0148 | 0.9819 | 0.0525 |
| $R_5$ | 0.8774 | 0.0128 | 0.9632 | 0.0226 |
| $R_6$ | 0.8283 | 0.0119 | 0.9486 | 0.0179 |
| $R_7$ | 0.7649 | 0.0114 | 0.7931 | 0.0116 |
| $R_8$ | 0.8910 | 0.0132 | 0.9650 | 0.0236 |
| $R_9$ | 0.7440 | 0.0112 | 0.8190 | 0.0118 |
| $R_{10}$ | 0.8580 | 0.0124 | 0.9576 | 0.0203 |
| $R_{11}$ | 0.8599 | 0.0124 | 0.9649 | 0.0235 |
| $R_s$ | 0.9609 | | 0.9919 | |
| $C_s$ | 0.1591 | | 0.7871 | |
| time-run | 1.3424 | | 0.2382 | |

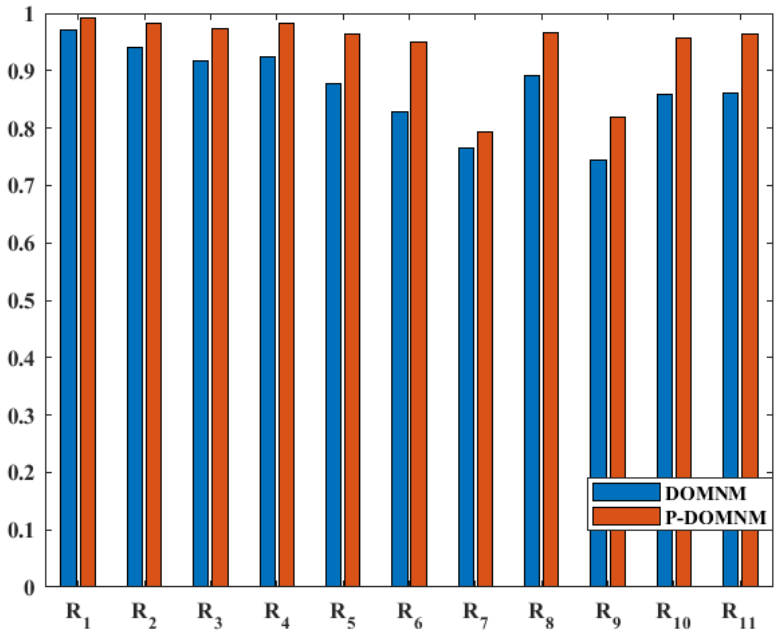



Figure 5.8: Comparison results value $R_i$ of P-DMONMA and DMONMA for model II.

### 5.2.3 Mathematical model III

$$
\begin{aligned}
&max \quad R_s(R_i) \\
&min \quad C_s(R_i) = \sum_{i=1}^{11} a_i \ln(\frac{1}{1-R_i}) \\
&\text{subject to} : 0.95 \leq R_s \leq 0.9999 \\
&\qquad\qquad 0.7 \leq R_i \leq 0.9999, \quad for \quad i = 1, ..., 11
\end{aligned}
\tag{5.6}
$$

where $R_s$ represented in equation (3.9), and $a_i = 0.2$ for all $i$.

#### 5.2.3.1 Computational Results of the HBA

Using HBA and P-HBA, illustrate the **Table** (5.9) and the **Figure** (5.9) shows the HBA results for values between $0.7000 \leq R_i \leq 0.9831$ and P-HBA results for values between $0.7000 \leq R_i \leq 0.9993$. The best reliability system value in the HBA was $R_s = 0.9500$ for the P-HBA was $R_s = 0.9978$. Total cost for the HAB was $C_s = 4.1215$ while for the P-HBA it was $C_s = 6.0226$ and HBA execution time was 0.3380 while for the P-HBA execution time it was 0.4691.

Table 5.9: Comparison for results value of P-HBA and HBA for model III.

| Components No | HBA | | P-HBA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9831 | 0.8161 | 0.9993 | 1.4529 |
| $R_2$ | 0.9545 | 0.8161 | 0.9980 | 1.4529 |
| $R_3$ | 0.9391 | 0.5597 | 0.9974 | 1.1904 |
| $R_4$ | 0.7036 | 0.2432 | 0.7000 | 0.2408 |
| $R_5$ | 0.7000 | 0.2408 | 0.7000 | 0.2408 |
| $R_6$ | 0.7000 | 0.2408 | 0.7000 | 0.2408 |
| $R_7$ | 0.7005 | 0.2411 | 0.7000 | 0.2408 |
| $R_8$ | 0.7002 | 0.2409 | 0.7000 | 0.2408 |

| | | | | |
|---|---|---|---|---|
| $R_9$ | 0.7001 | 0.2409 | 0.7000 | 0.2408 |
| $R_{10}$ | 0.7000 | 0.2408 | 0.7000 | 0.2408 |
| $R_{11}$ | 0.7005 | 0.2411 | 0.7000 | 0.2408 |
| $R_s$ | 0.9500 | | 0.9978 | |
| $C_s$ | 4.1215 | | 6.0226 | |
| time-run | 0.3380 | | 0.4691 | |

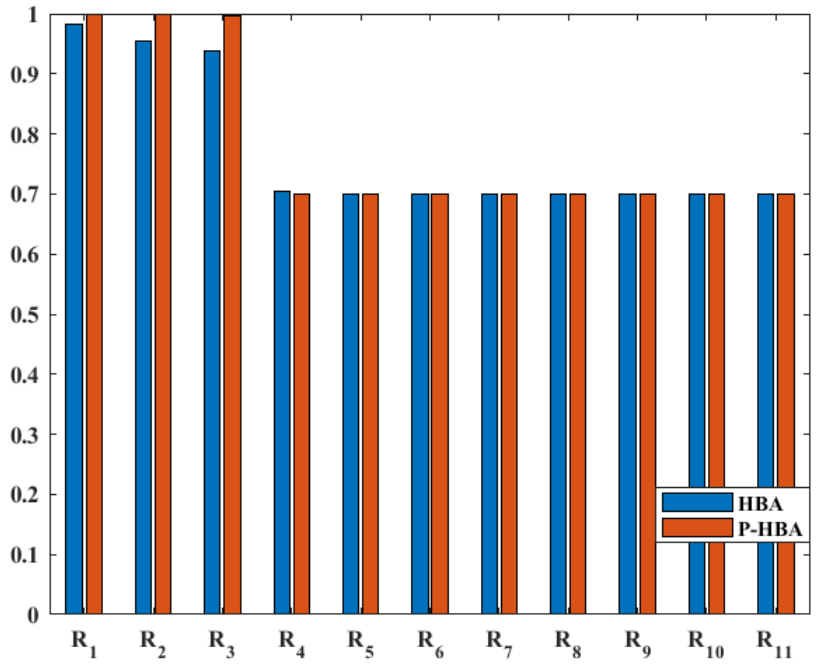



Figure 5.9: Comparison results value $R_i$ of P-HBA and HBA for model III.

#### 5.2.3.2 Computational Results of the HBNMA

Employing the HBNMA and the P-HBNMA. Results were displayed in the **Table** (5.10) and the **Figure** (5.10) shows the HBNMA results for values between $0.7083 \leq R_i \leq 0.9917$ and P-HBNMA results for values between $0.7000 \leq R_i \leq 0.9999$. The best reliability system value for the HBNMA was $R_s = 0.9587$ and $R_s = 0.9998$ for the P-HBNMA . Total cost for the HBNMA was $C_s = 4.5704$ while for the P-HBNMA, it was $C_s = 13.8577$ and HBNMA execution time was 0.2930 while for the P-HBNMA execution time it was 0.2225.

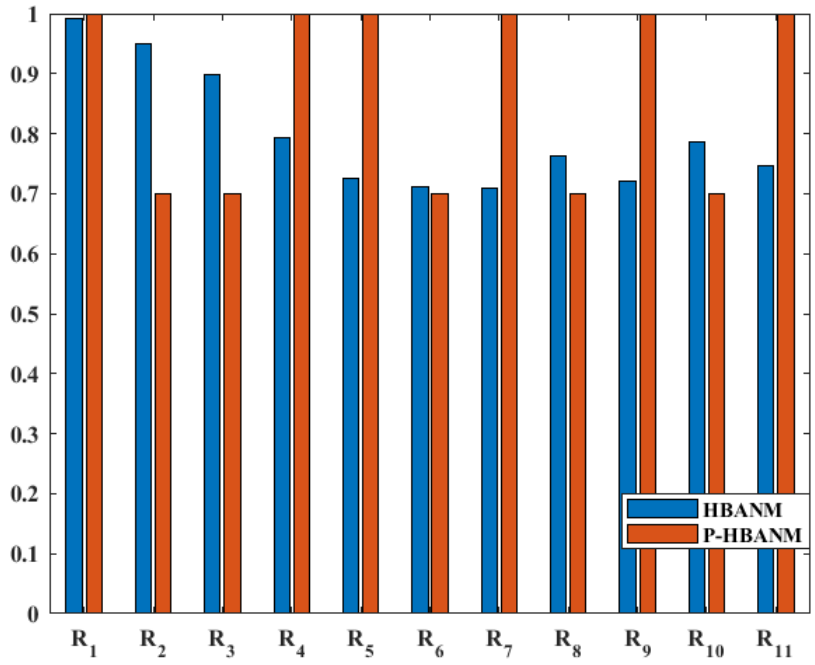



Figure 5.10: Comparison results value $R_i$ of P-HBNMA and HBNMA for model III.

Table 5.10: Comparison for results values of of P-HBNMA and HBNMA for model III.

| Components No | HBNMA | | P-HBNMA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9917 | 0.9583 | 0.9999 | 1.8421 |
| $R_2$ | 0.9487 | 0.9583 | 0.7000 | 1.8421 |
| $R_3$ | 0.8991 | 0.4587 | 0.7000 | 0.2408 |
| $R_4$ | 0.7927 | 0.3147 | 0.9999 | 1.8421 |
| $R_5$ | 0.7244 | 0.2578 | 0.9999 | 1.8421 |
| $R_6$ | 0.7118 | 0.2488 | 0.7000 | 0.2408 |
| $R_7$ | 0.7083 | 0.2464 | 0.9999 | 1.8421 |
| $R_8$ | 0.7628 | 0.2878 | 0.7000 | 0.2408 |
| $R_9$ | 0.7217 | 0.2558 | 0.9999 | 1.8421 |

| | | | | |
|---|---|---|---|---|
| $R_{10}$ | 0.7862 | 0.3085 | 0.7000 | 0.2408 |
| $R_{11}$ | 0.7475 | 0.2753 | 0.9999 | 1.8421 |
| $R_s$ | 0.9587 | | 0.9998 | |
| $C_s$ | 4.5704 | | 13.8577 | |
| time-run | 0.2930 | | 0.2225 | |

#### 5.2.3.3 Computational Results of the DMOA

Using the DMOA and P-DMOA. The **Table** (5.11) and the **Figure** (5.11) shows the P-DMOA results for values between $0.7033 \leq R_i \leq 0.9994$ and the DMOA results for values between $0.7011 \leq R_i \leq 0.9860$ and the P-DMOA, the optimal reliability system value was $R_s = 0.9974$, while for the DMOA, it was $R_s = 0.9511$. The P-DMOA overall cost was $C_s = 6.1574$ while the DMOA total cost was $C_s = 4.4383$. The P-DMOA execution time was 66.1896 while the DMOA execution time was 23.1391.

Table 5.11: Comparison for results value of P-DMOA and DMOA for model III.

| Components No | DMOA | | P-DMOA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9860 | 0.8537 | 0.9994 | 1.4837 |
| $R_2$ | 0.9197 | 0.8537 | 0.9961 | 1.4837 |
| $R_3$ | 0.9276 | 0.5251 | 0.9960 | 1.1043 |
| $R_4$ | 0.8138 | 0.3362 | 0.8151 | 0.3376 |
| $R_5$ | 0.7356 | 0.2661 | 0.7108 | 0.2481 |
| $R_6$ | 0.7262 | 0.2591 | 0.7067 | 0.2453 |
| $R_7$ | 0.7286 | 0.2608 | 0.7297 | 0.2616 |
| $R_8$ | 0.7478 | 0.2755 | 0.7093 | 0.2471 |
| $R_9$ | 0.7416 | 0.2706 | 0.7033 | 0.2430 |

| | | | | |
|---|---|---|---|---|
| $R_{10}$ | 0.7722 | 0.2959 | 0.7140 | 0.2504 |
| $R_{11}$ | 0.7011 | 0.2415 | 0.7171 | 0.2525 |
| $R_s$ | 0.9511 | | 0.9974 | |
| $C_s$ | 4.4383 | | 6.1574 | |
| time-run | 23.1391 | | 66.1896 | |

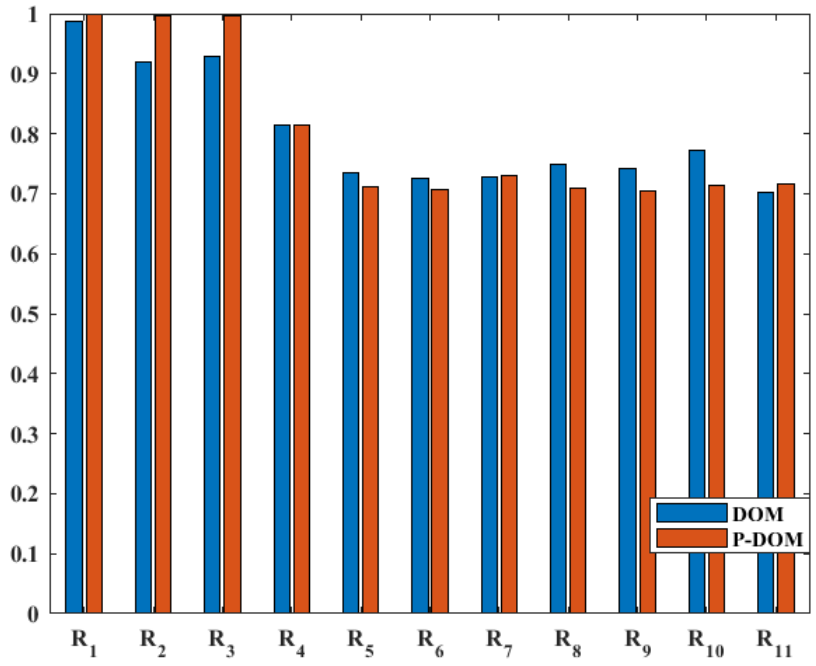



Figure 5.11: Comparison results value $R_i$ of of P-DMOA and DMOA for model III.

#### 5.2.3.4 Computational Results of the DMONMA

Both the P-DMONMA and the DMONMA were employed. The findings were shown in the **Table** (5.12) and the **Figure** (5.12) shows the DMONMA results were shown for values between $0.7055 \leq R_i \leq 0.9819$ and the P-DMONMA results were shown for values between $0.7000 \leq R_i \leq 0.9999$. The best reliability system value for the DMONMA was $R_s = 0.9560$, while the best reliability system value for the P-DMONMA was $R_s = 0.9984$. The total cost for the DMONMA was $C_s = 4.4738$, while the total cost for the P-DMONMA was $C_s = 6.8895$, and the DMONMA execution time was 14.7126 as opposed

to 0.2544 for P-DMONMA.

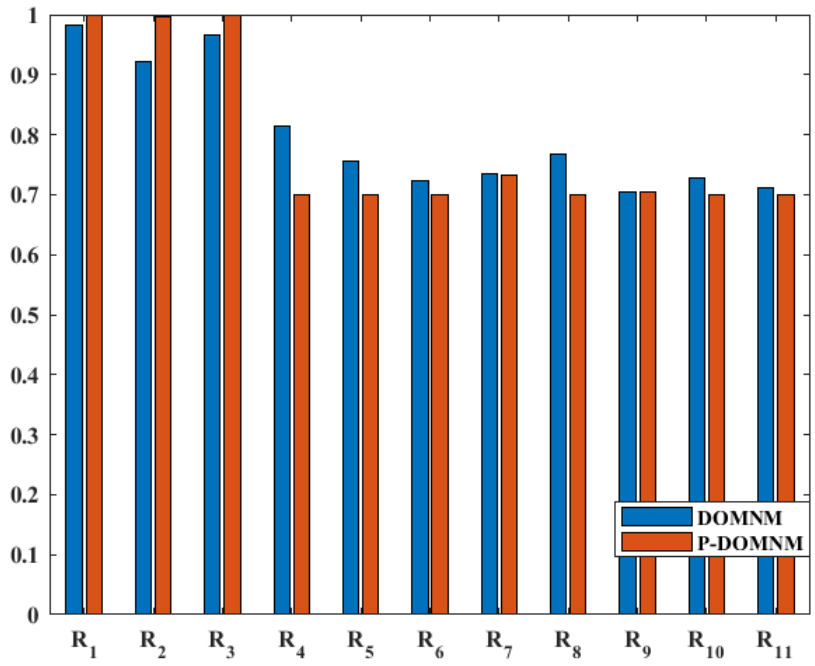



Figure 5.12: Comparison results value $R_i$ of P-DMONMA and DMONMA for model III.

Table 5.12: Comparison for results value of P-DMONMA and DMONMA for model III.

| Components No | DMONMA | | P-DMONMA | |
|---|---|---|---|---|
| | $R_i$ | $C_i$ | $R_i$ | $C_i$ |
| $R_1$ | 0.9819 | 0.8024 | 0.9999 | 1.8421 |
| $R_2$ | 0.9223 | 0.8024 | 0.9974 | 1.8421 |
| $R_3$ | 0.9668 | 0.6810 | 0.9981 | 1.2532 |
| $R_4$ | 0.8152 | 0.3377 | 0.7000 | 0.2408 |
| $R_5$ | 0.7552 | 0.2815 | 0.7000 | 0.2408 |
| $R_6$ | 0.7238 | 0.2573 | 0.7000 | 0.2408 |
| $R_7$ | 0.7345 | 0.2652 | 0.7323 | 0.2636 |
| $R_8$ | 0.7680 | 0.2922 | 0.7000 | 0.2408 |
| $R_9$ | 0.7055 | 0.2445 | 0.7046 | 0.2439 |

| | | | | |
|---|---|---|---|---|
| $R_{10}$ | 0.7284 | 0.2607 | 0.7000 | 0.2408 |
| $R_{11}$ | 0.7119 | 0.2489 | 0.7000 | 0.2408 |
| $R_s$ | 0.9560 | | 0.9984 | |
| $C_s$ | 4.4738 | | 6.8895 | |
| time-run | 14.7126 | | 0.2544 | |

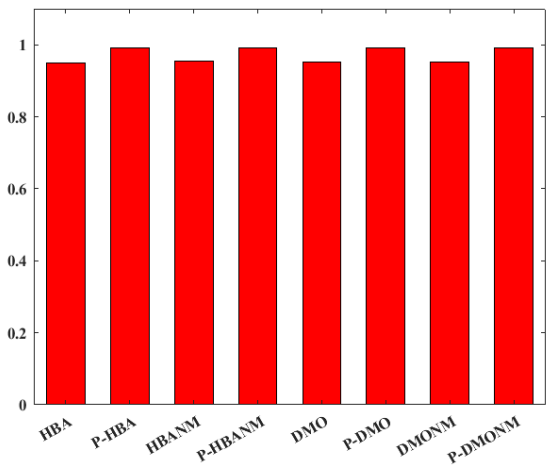
## 5.3. Comparing Results of Algorithms

This section discusses comparing the values that got from the algorithms, Compared each model separately, and then compare the three models to find out the best algorithm that gives the best value to the reliability of the system and the cost value and the best and least execution time for the implementation algorithms, as well as to know the best model that can be used to improve the study the shutdown network in **Figure** (3.1).

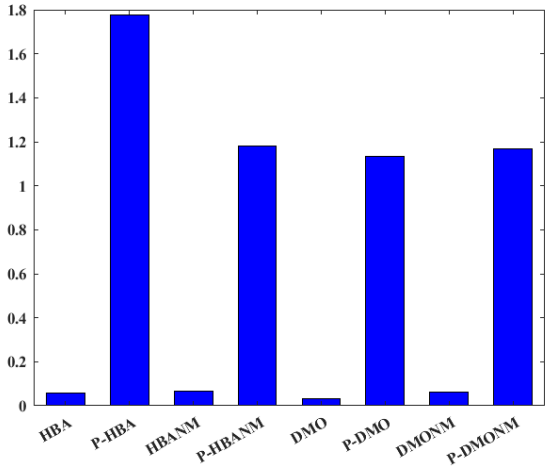
### 5.3.1 Comparing Results of Algorithms for Model I

Notice from **Table** (5.13) and **Figure** (5.13a) that the value $R_s$ of the HBA has been improved by both algorithms are HBNMA and P-HBA, and the results of the two algorithms are HBNMA and P-HBA have been improved by the P-HBNMA is better, as well as for the DMOA whose results have been improved by DMONMA and P-DMOA, and the results of the two algorithms are DMONMA and P-DMOA have been improved by P-DMONMA, and have obtained the best four results using are P-HBA, P-HBNMA, P-DMOA and P-DMONMA the best results for the best reliability of system was P-HBNMA. For comparison, the implementation time, show the **Figure** (5.13c), note that the time of the HBNMA is less than the time of the HBA, as well as the time of implementation of the P-HBNMA, is less than the time of implementation P-HBA, and the execution time of the DMONMA and P-DMONMA is less than the execution time of the DMOA and P-DMOA respectively. As for the time of implementation of the

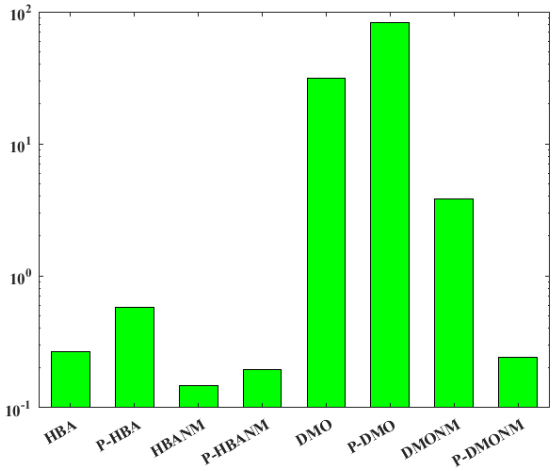
algorithms are P-HBA, P-HBNMA, P-DMOA and P-DMONMA that got the best results the best reliability of system, the highest time of the P-DMOA and the lowest time implementation was P-HBNMA. Comparing the value of the total cost was the lowest cost value corresponding to the best reliability value of the system of the algorithms are P-HBA, P-HBNMA, P-DMOA and P-DMONMA, the highest value is the comparison of the total cost value show **Figure** (5.13b), the highest value was P-HBNMA and the lowest value was when the P-DMOA.




(a) Results $R_s$




(b) Results $C_s$




(c) Results time of implementation

Figure 5.13: Comparison results for algorithms values of model I.

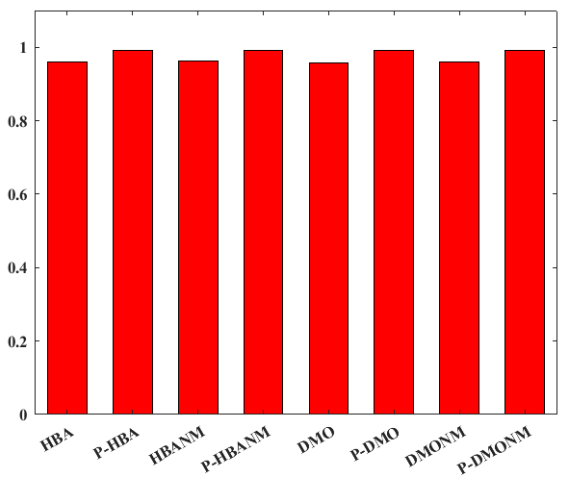
Table 5.13: Comparison results $R_s$ and $C_s$ for algorithms values of for model I.

| Algorithms | $R_s$ | $C_s$ | Time-run |
|---|---|---|---|
| HBA | 0.9500 | 0.0586 | 0.2643 |
| P-HBA | 0.9924 | 1.1775 | 0.5806 |
| HBNMA | 0.9547 | 0.0685 | 0.1479 |
| P-HBNMA | 0.9925 | 1.1788 | 0.1934 |
| DMOA | 0.9510 | 0.0626 | 31.5609 |
| P-DMOA | 0.9921 | 1.1354 | 83.2784 |
| DMONMA | 0.9513 | 0.0624 | 3.8252 |
| P-DMONMA | 0.9924 | 1.1659 | 0.2412 |

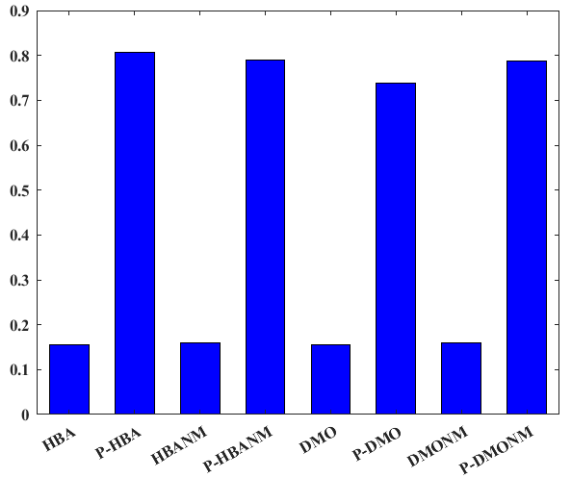
### 5.3.2 Comparing Results of Algorithms for Model II

The **Table** (5.14) and the **Figure** (5.14a) shows for comparison, the best results of system reliability $R_s$ that the HBA value has been enhanced by both the HBNMA and P-HBA, and that the P-HBNMA has improved the results of the two algorithms are HBNMA and P-HBA. Similarly, the DMOA effects have been enhanced by the DMONMA and P-DMOA, and the P-DMONMA has enhanced the results of the DMONMA and P-DMOA, yields the best outcomes in terms of system reliability in this model, obtained the four best results were the P-HBA, P-HBNMA, P-DMOA and P-DMONMA, either the best value was equal for the two algorithms were P-HBNMA and P-DMONMA. For comparison, the implementation time is shown in **Figure** (5.14c). It should be noted that the HBNMA executes faster than the HBA, the P-HBNMA executes faster than the P-HBA, and the DMONMA and P-DMONMA execute faster than the DMOA and P-DMOA algorithms, respectively. These algorithms are P-HBA, P-HBNMA, P-DMOA, and P-DMONMA were the ones that gave us the best results and the most reliable system. The P-DMOA took the longest time to implement, where as P-HBNMA took the shortest time. Compare the best values obtained the reliability value of the system of the algorithms are P-HBA, P-HBNMA, P-DMOA and P-DMONMA to find the value of the total cost show the **Figure** (5.14b)

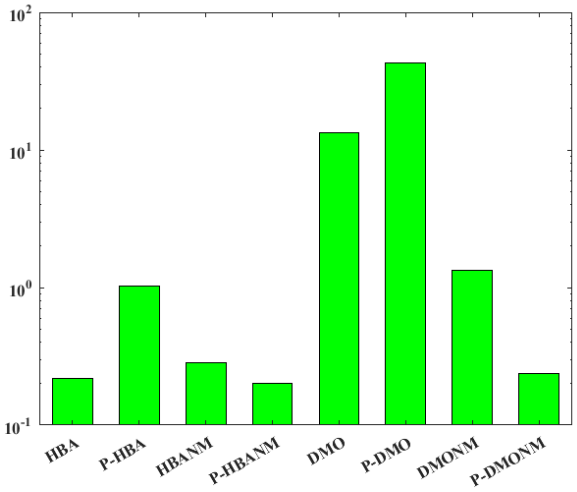
was the lowest cost value P-DMOA and the highest value is P-HBA.




(a) Results $R_s$




(b) Results $C_s$




(c) Results time of implementation

Figure 5.14: Comparison results for algorithms values of model II.

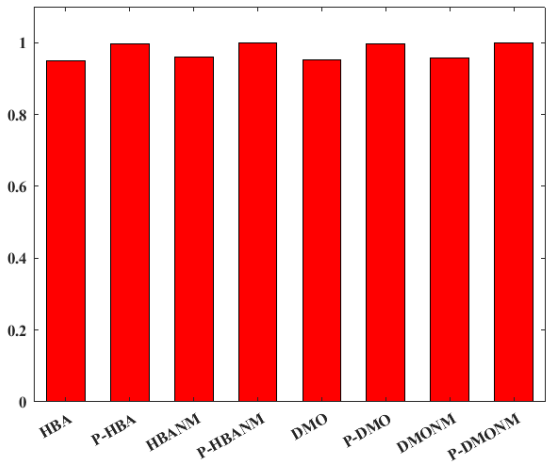
Table 5.14: Comparison for results algorithms value of for model II.

| Algorithms | $R_s$ | $C_s$ | Time-run |
|---|---|---|---|
| HBA | 0.9586 | 0.1561 | 0.2196 |
| P-HBA | 0.9918 | 0.8063 | 1.0234 |
| HBNMA | 0.9616 | 0.1605 | 0.2830 |
| P-HBNMA | 0.9919 | 0.7893 | 0.2025 |
| DMOA | 0.9585 | 0.1563 | 13.2938 |
| P-DMOA | 0.9917 | 0.7372 | 42.9952 |
| DMONMA | 0.9609 | 0.1591 | 1.3424 |
| P-DMONMA | 0.9919 | 0.7871 | 0.2382 |

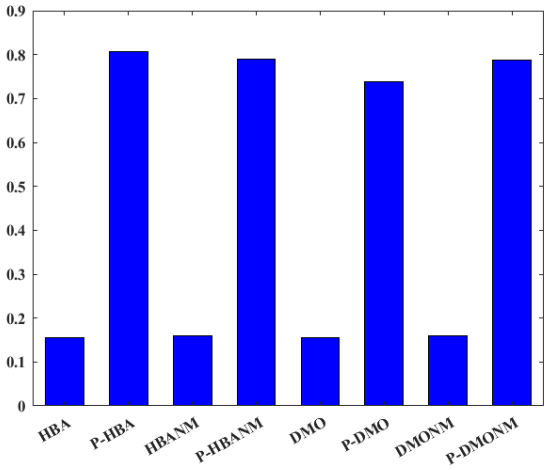
### 5.3.3 Comparing Results of Algorithms for Model III

Illustrate the **Table** 5.15 and the **Figure** (5.15a), the HBNMA and P-HBA have both improved the value $R_s$ of the HBA , and the P-HBNMA has improved the results of the two algorithms are HBNMA and P-HBA the best. Similarly, the DMOA results have been enhanced by the DMONMA and P-DMOA , and the P-DMONMA has enhanced the results of the two algorithms are DMONMA and P-DMOA, also in this model, obtained the four best results for the algorithms are P-HBA, P-HBNMA, P-DMOA and P-DMONMA, and the best value was the system reliability of the algorithm was P-HBNMA. **Figure** (5.15c) provides a comparison of the implementation time. Notably, the HBNMA, P-HBNMA, DMONMA and P-DMONMA runs faster than the HBA, P-HBA , DMOA, P-DMOA, respectively. It was the execution time of the algorithms that gave us the best results P-HBA, P-HBNMA, P-DMOA, and P-DMONMA the most accurate results and the most reliable system, and it was the best execution time for P-HBNMA while the P-DMOA took the longest time to implement. To determine the value of the overall cost show in the **Figure** (5.15b). Compared the best values could find for the reliability value of the system for algorithms are P-HBA, P-HBNMA, P-DMOA, and P-DMONMA. The P-HBA had the lowest cost, and the P-HBA strategy

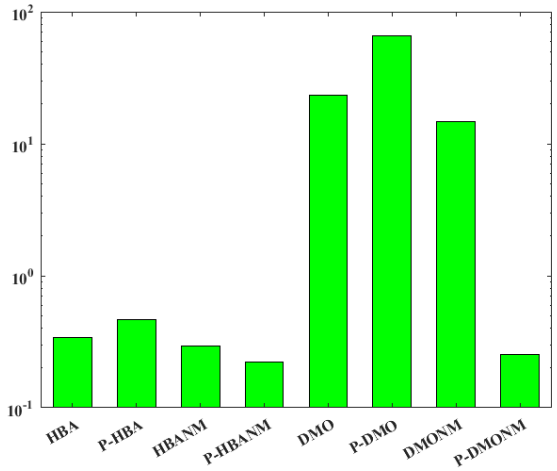
had the highest P-HBNMA.




(a) Results $R_s$




(b) Results $C_s$




(c) Results time of implementation

Figure 5.15: Comparison results for algorithms values of model III.

Table 5.15: Comparison for results algorithms value of for model III.

| Algorithms | $R_s$ | $C_s$ | Time-run |
|---|---|---|---|
| HBA | 0.9500 | 4.1215 | 0.3380 |
| P-HBA | 0.9978 | 6.0226 | 0.4691 |
| HBNMA | 0.9587 | 4.5704 | 0.2930 |
| P-HBNMA | 0.9998 | 13.8577 | 0.2225 |
| DMOA | 0.9511 | 4.4383 | 23.1391 |
| P-DMOA | 0.9974 | 6.1574 | 66.1896 |
| DMONMA | 0.9560 | 4.4738 | 14.7126 |
| P-DMONMA | 0.9984 | 6.8895 | 0.2544 |

## 5.4. Comparing Results of Algorithms for All Models

Illustrate the **Table** (5.16), and the **Figure** (5.16), the best results of system reliability $R_s$ the algorithms are P-HBA, P-HBNMA, P-DMOA and P-DMONMA for three models. It was the best value of the P-HBNMA for the model III, show in the **Figure** (5.18), the best time to implement these algorithms for all models was for the P-HBNMA for the model I. The **Figure** ( 5.17) shows that the lowest cost value of the three models was for the P-DOM of the model.

Table 5.16: Comparison for results value of algorithms for models

| algorithms | values of $R_s$ | | | values of $C_s$ | | | values of time-run | | |
|---|---|---|---|---|---|---|---|---|---|
| | model I | model II | model III | model I | model II | model III | model I | model II | model III |
| HBA | 0.9500 | 0.9586 | 0.9500 | 0.0586 | 0.1561 | 4.1215 | 0.2643 | 0.2196 | 0.3380 |
| P-HBA | 0.9924 | 0.9918 | 0.9978 | 1.1775 | 0.8063 | 6.0226 | 0.5806 | 1.0234 | 0.4691 |
| HBNMA | 0.9547 | 0.9616 | 0.9587 | 0.0685 | 0.1605 | 4.5704 | 0.1479 | 0.2830 | 0.2930 |
| P-HBNMA | 0.9925 | 0.9919 | 0.9998 | 1.1788 | 0.7893 | 13.8577 | 0.1934 | 0.2025 | 0.2225 |
| DMOA | 0.9510 | 0.9585 | 0.9511 | 0.0626 | 0.1563 | 4.4383 | 31.5609 | 13.2938 | 23.1391 |
| P-DOM | 0.9921 | 0.9917 | 0.9974 | 1.1354 | 0.7372 | 6.1574 | 83.2784 | 42.9952 | 66.1896 |
| DMONMA | 0.9513 | 0.9609 | 0.9560 | 0.0624 | 0.1591 | 4.4738 | 3.8252 | 1.3424 | 14.7126 |
| P-DMONMA | 0.9924 | 0.9919 | 0.9984 | 1.1659 | 0.7871 | 6.8895 | 0.2412 | 0.2382 | 0.2544 |

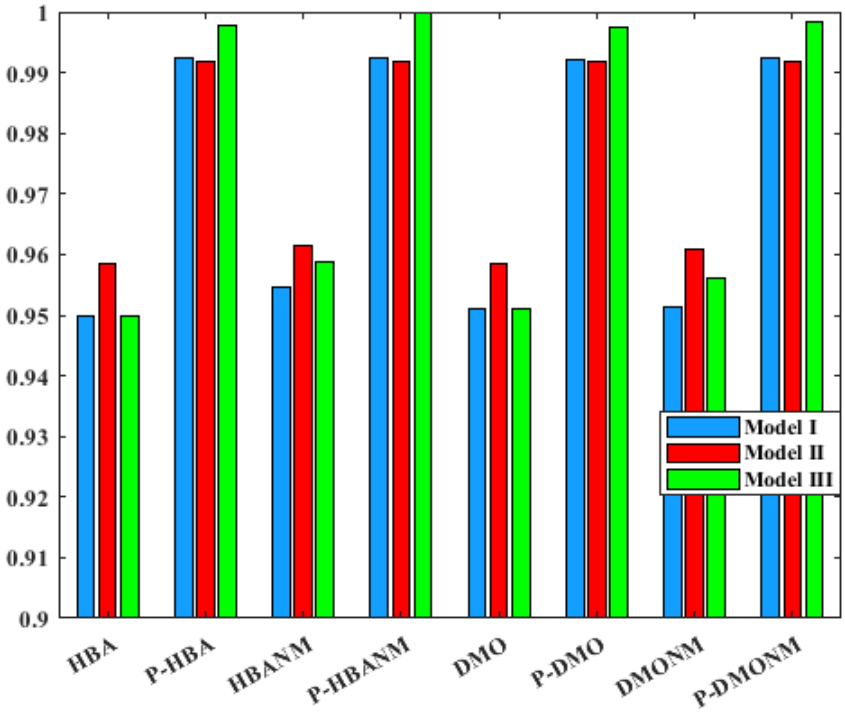



Figure 5.16: Comparison results $R_s$ for algorithms values for all models.

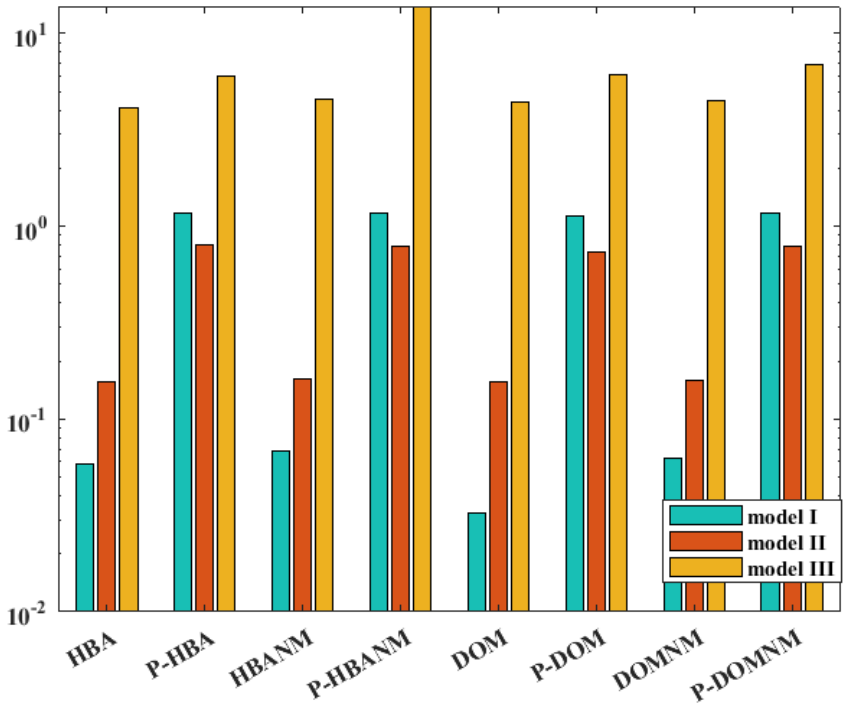



Figure 5.17: Comparison results $C_s$ for algorithms values for all models

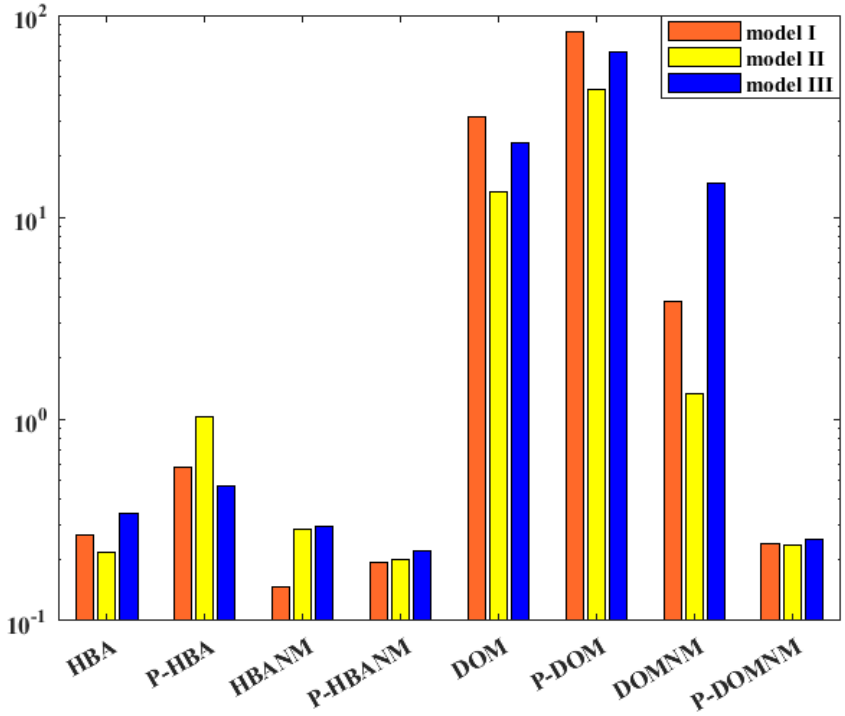



Figure 5.18: Comparison results time of implementation for algorithms for all models

# CHAPTER 6

# CONCLUSIONS AND FUTURE WORKS

## 6.1. Conclusions

The dissertation includes several works. Through study and research, had reached new ideas and visions that will present according as follows

1. Create a networks from a shutdown system Petri net. It a directed network containing cycle,

2. Analysis of a system to find minimal path sets, the using three techniques were Rai and Aggarwal algorithm, matrix multiplication, and the path set enumeration algorithm, depend on the connection matrix. The challenge of repeatedly using the matrix multiplication technique arises when a network has a large number of nodes and the matrix size will give all the paths is large. This makes finding the minimal path using the first technique simpler. a quick and easy technique that finds paths by vertices rather than edges as the matrix expands can be employed as a third technique. While the node-child matrix technique is the fourth approach, finding a minimal path in vertices rather than edges by using a specific matrix to identify which vertices are connected. The two terminal nodes algorithm is the fifth technique, which is a special technique that makes use of the network structure to locate paths from both terminal nodes. This technique can be used in networks directed or indirect.

3. The techniques of find minimal cut sets. The first technique involves using the nodes and edges algorithm to locate all minimal cuts, which depends on how to identify each minimal cut vertices to transform for a minimal cut of edges, second technique Shall algorithm is based on the network graph connection matrix, and it discovers that the minimal cut edges are modeled for the graph enumeration plots that contain a source and a terminal node. Use just the combinations that result in the fewest minimal cut sets after listing all possible combinations networks indirect or directed can use this technique. Finally, suggested technique is the Roaa technique , which

relies on finding the lowest cut of edges for directed networks with a cycle using minimal path vertices. It is a less complicated procedure than other techniques without any computational difficulties.

4. Calculation the reliability of the shutdown network polynomial, by two different techniques were studied for doing so. The first technique is the sum of disjoint simple products technique, which relies on the laws of probability was calculated using a set of minimal paths, and the reliability was calculated using that set. The second technique is a minimal cut technique. Using minimal cut sets to get the network polynomial was the same using both techniques, indicating that our results were accurate.

5. Studied two new optimization methods , the hybrid algorithms are HBNMA from combining the meta-heuristics algorithm, the honey badger algorithm (HBA), with the Nelder Mead method (NM), hybrid algorithm was DMONMA and from the Dwarf Mongoose Optimization Algorithm (DMOA) with the Nelder Mead method. To know the performance of hybrid algorithms, used the experimental results and statistical significance were average, standard deviations and execution time, experiments were conducted on several well-known benchmark functions with different dimensions, was the performance of the HBNMA better than the HBA for most functions and also the DMONMA improved the results of the DMOA in most functions with the test in various dimensions, but the execution time was better in executing some of the functions. Found that the preference in most of the functions of the HBNMA when comparing with HBA,DMOA, and DMONMA.

6. The HBA, HBNMA, DMOA, and DMONMA are applied to solve the problem of multi-objective nonlinear problems for shutdown network. For the reliability optimization, used the weighted sum method to transform the multi-objective problems are reduced to a single-objective problems. The constraint handling is done via a penalty function, three mathematical models are used to calculate

reliability and cost. The results were compared using a penalty function is used and when it is not used. oted that the proposed two algorithms were HBNMA and DMONMA improved the network reliability results, but the results were better using the penalty method with the algorithm showed improved results were P-HBA, P-HBNMA, P-DMOA, and P-DMONMA compared with those using the algorithm without penalty method, the best results of system reliability maximizing and the best time to implement these algorithms for all models was the third model, and first model respectively for the P-HBNMA, and the best model, less cost of the third model for the P-DMOA.

## 6.2. Future Works

1. Propose to study convert Petri net of the shutdown system in another method to network see subsection 2.13.2.

2. Propose to study hybrid combining Powell method with the Honey Badger algorithm or Dwarf Mongoose optimization algorithm.

3. Propose to study augmented lagrangians based for the Honey Badger algorithm or Dwarf Mongoose optimization algorithm.

# REFERENCES


[1] Fouad Hamza Abd Alsharify and Zahir Abdul Haddi Hassan. Bat and grey wolf algorithms to optimize complex network reliability. In *AIP Conference Proceedings*, volume 2591, page 050010. AIP Publishing LLC, 2023.

[2] Ghazi Abdullah and Zahir Abdul Haddi Hassan. Using of particle swarm optimization (pso) to addressed reliability allocation of complex network. In *Journal of Physics: Conference Series*, volume 1664, page 012125. IOP Publishing, 2020.

[3] Saad Abbas Abed, Hatem Kareem Sulaiman, and Zahir Abdul Haddi Hassan. Reliability allocation and optimization for (ross) of a spacecraft by using genetic algorithm. In *Journal of Physics: Conference Series*, volume 1294, page 032034. IOP Publishing, 2019.

[4] Yasser MR Aboelmagd. Linear programming applications in construction sites. *Alexandria Engineering Journal*, 57(4):4177–4187, 2018.

[5] Ajith Abraham and Lakhmi Jain. *Evolutionary multiobjective optimization*. Springer, London, second edition, 2005.

[6] KK Aggarwal and Shashwati Guha. Reliability allocation in a general system with non-identical components—a practical approach. *Microelectronics Reliability*, 33(8):1089–1093, 1993.

[7] Jeffrey O Agushaka, Absalom E Ezugwu, and Laith Abualigah. Dwarf mongoose optimization algorithm. *Computer methods in applied mechanics and engineering*, 391:114570, 2022.

[8] S Hasanuddin Ahmad. Simple enumeration of minimal cutsets of acyclic directed graph. *IEEE transactions on reliability*, 37(5):484–487, 1988.

[9] Ahmed F Ali and Mohamed A Tawhid. A hybrid cuckoo search algorithm with nelder mead method for solving global optimization problems. *SpringerPlus*, 5(1):1–22, 2016.

[10] Niclas Andréasson, Anton Evgrafov, and Michael Patriksson. *An introduction to continuous optimization: foundations and fundamental algorithms*. Courier Dover Publications, Mineola, New York, first edition, 2020.

[11] Jasbir Arora. *Introduction to optimum design*. Elsevier, America, second edition, 2004.

[12] Terje Aven. *Reliability and risk analysis*. Springer Science & Business Media, London and New York, firstdasgupta2013evolutionary edition, 2012.

[13] Terje Aven and Uwe Jensen. *Stochastic models in reliability*. Springer, New York, first edition, 1999.

[14] Michael O Ball, Charles J Colbourn, and J Scott Provan. Network reliability. *Handbooks in operations research and management science*, 7, 1995.

[15] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, Verlag London, second edition, 2008.

[16] Reza Barati. Parameter estimation of nonlinear muskingum models using nelder-mead simplex algorithm. *Journal of Hydrologic Engineering*, 16(11):946–954, 2011.

[17] Russell R Barton and John S Ivey Jr. Modifications of the nelder-mead simplex method for stochastic simulation response optimization. Technical report, Institute of Electrical and Electronics Engineers (IEEE), 1991.

[18] Frank Beichelt and Peter Tittmann. *Reliability and maintenance: networks and systems*. Crc Press, New York, first edition, 2012.

[19] MC Bhuvaneswari. *Application of evolutionary algorithms for multi-objective optimization in VLSI and embedded systems*. Springer, 2014.

[20] Roy Billinton and Ronald Norman Allan. *Reliability evaluation of engineering systems*, volume 792. Springer, New York, second edition, 1992.

[21] Vandenberghe Lieven Boyd, Stephen. *Convex optimization*. Cambridge university press, New York, first edition, 2004.

[22] Omid Bozorg-Haddad, Mohammad Solgi, and Hugo A Loáiciga. *Meta-heuristic and evolutionary algorithms for engineering optimization*. John Wiley & Sons, USA, first edition, 2017.

[23] Jason Brownlee. *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, Australia, first edition, 2011.

[24] P Giuggioli Busacca, Marzio Marseguerra, and Enrico Zio. Multiobjective optimization by genetic algorithms: application to safety systems. *Reliability Engineering & System Safety*, 72(1):59–74, 2001.

[25] Kuang-Hua Chang. *Design theory and methods using CAD/CAE: The computer aided engineering design series*. Academic Press, 2014.

[26] Kuei Hu Chang, Cheng Shih Liaw, Thing Yuan Chang, and Yung-Chia Chang. Fmea-based dematel apportionment approach. *Chung Cheng Ling Hsueh Pao/Journal of Chung Cheng Institute of Technology*, 42(1):41–58, 2013.

[27] Sanjay Kumar Chaturvedi. *Network reliability: measures and evaluation*. John Wiley & Sons, 2016.

[28] SK Chaturvedi and KB Misra. An efficient multi-variable inversion algorithm for reliability evaluation of complex systems using path sets. *International Journal of Reliability, Quality and Safety Engineering*, 9(03):237–259, 2002.

[29] Long Chen, Yingying Xu, Fangyi Xu, Qian Hu, and Zhenzhou Tang. Balancing the trade-off between cost and reliability for wireless sensor networks: a multi-objective optimized deployment method. *Applied Intelligence*, pages 1–26, 2022.

[30] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 75. John Wiley & Sons, 2013.

[31] Bastien Chopard and Marco Tomassini. *An introduction to metaheuristics for optimization*. Springer, Switzerland,, first edition, 2018.

[32] Lewis M Clement. Reliability of military electronic equipment. *Journal of the British Institution of Radio Engineers*, 16(9):488–495, 1956.

[33] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[34] David W Coit and Alice E Smith. Penalty guided genetic search for reliability design optimization. *Computers & industrial engineering*, 30(4):895–904, 1996.

[35] Yann Collette and Patrick Siarry. *Multiobjective optimization: principles and case studies*. Springer Science & Business Media, New York, second edition, 2004.

[36] Richard Cottle, Mukund N Thapa, et al. *Linear and nonlinear optimization*, volume 253. Springer, New York, first edition, 2017.

[37] MJ Crowder, AC Kimber, RL Smith, and TJ Sweeting. *Statistical Analysis of Reliability Data*. Chapman and Hall, London, first edition, 1991.

[38] Filippo De Carlo. Reliability and maintainability in operations management. *Operations Management*, 1:32, 2013.

[39] Fabio De Felice and Antonella Petrillo. Human factors and reliability engineering for safety and security in critical infrastructures. *Decision Making, Theory, and Practice, Springer, Cham*, 2018.

[40] Gilberto Francisco Martha De Souza. *Thermal power plant performance analysis*. Springer, London, 2012th edition, 2012.

[41] Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, London, first edition, 2011.

[42] Darren L Deeter and Alice E Smith. Heuristic optimization of network design considering all-terminal reliability. In *Annual reliability and maintainability symposium*, pages 194–199. IEEE, 1997.

[43] Berna Dengiz, Fulya Altiparmak, and Alice E Smith. Local search genetic algorithm for optimal design of reliable networks. *IEEE transactions on Evolutionary Computation*, 1(3):179–188, 1997.

[44] Narsingh Deo. *Graph theory with applications to engineering and computer science*. Courier Dover Publications, New Jersey, first edition, 2017.

[45] Michel Diaz. *Petri nets: fundamental models, verification and applications*. John Wiley & Sons, British, first edition, 2013.

[46] Reinhard Diestel. *Graph Theory*. Springer, Germany, fifth edition, 2010.

[47] Jason G Digalakis and Konstantinos G Margaritis. On benchmarking functions for genetic algorithms. *International journal of computer mathematics*, 77(4):481–506, 2001.

[48] JF Dopazo and HM Merrill. Optimal generator maintenance scheduling using integer programming. *IEEE Transactions on Power Apparatus and Systems*, 94(5):1537–1545, 1975.

[49] Ke-Lin Du, MNS Swamy, et al. Search and optimization by metaheuristics. *Techniques and Algorithms Inspired by Nature*, 2016.

[50] H Salehi Fathabadi, M Soltanifar, A Ebrahimnejad, and SH Nasseri. Determining all minimal paths of a network. *Australian Journal of Basic and Applied Sciences*, 3(4):3771–3777, 2009.

[51] Peter J Fleming, Robin C Purshouse, and Robert J Lygoe. Many-objective optimization: An engineering design perspective. In *International conference on evolutionary multi-criterion optimization*, pages 14–32. Springer, 2005.

[52] Majid Forghani-elahabad and Nezam Mahdavi-Amiri. An algorithm to search for all minimal cuts in a flow network. *Advances in Systems Science and Applications*, 20(4):1–10, 2020.

[53] Ilya Gertsbakh and Yoseph Shpungin. *Network reliability and resilience*. Springer Science & Business Media, New York, 2011th edition, 2011.

[54] Ilya Gertsbakh and Yoseph Shpungin. *Network reliability: a lecture course*. Springer, Singapore, first edition, 2020.

[55] Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical methods and optimization in finance*. Academic Press, United Kingdom, second edition, 2019.

[56] Neeraj Kumar Goyal and S Rajkumar. *Interconnection Network Reliability Evaluation: Multistage Layouts*. John Wiley & Sons, USA, first edition, 2020.

[57] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, USA, second edition, 2003.

[58] Osman Güler. *Foundations of optimization*, volume 258. Springer Science & Business Media, London, first edition, 2010.

[59] Ranjan Kumar Gupta, Asoke Kumar Bhunia, and D Roy. A ga based penalty function technique for solving constrained redundancy allocation problem of series system with interval valued reliability of components. *Journal of Computational and Applied Mathematics*, 232(2):275–284, 2009.

[60] Michael S Hamada, Harry F Martz, C Shane Reese, and Alyson G Wilson. *Bayesian reliability*, volume 15. Springer, USA, first edition, 2008.

[61] Military Handbook. Electronic reliability design handbook. In *MIL-HDBK-338, DoD*. USA, first edition, 1988.

[62] FRANK HARARY. *Graph theory*. CRC Press Taylor and Francis Group, New York, first edition, 2018.

[63] John M Harris, Jeffry L Hirst, and Michael J Mossinghoff. *Combinatorics and graph theory*. Springer Science & Business Media, 2008.

[64] Fatma A Hashim, Essam H Houssein, Kashif Hussain, Mai S Mabrouk, and Walid Al-Atabany. Honey badger algorithm: New metaheuristic algorithm for solving optimization problems. *Mathematics and Computers in Simulation*, 192:84–110, 2022.

[65] Eligius MT Hendrix, G Boglárka, et al. *Introduction to nonlinear and global optimization*, volume 37. Springer, London, 2010.

[66] Khalid Housni. An efficient algorithm for enumerating all minimal paths of a graph. *International Journal of Advanced Computer Science and Applications*, 10(1), 2019.

[67] Arnljot Hoyland and Marvin Rausand. *System reliability theory: models and statistical methods*. John Wiley & Sons, Canada., third edition, 2021.

[68] Branislav Hrúz and MengChu Zhou. *Modeling and control of discrete-event dynamic systems: With petri nets and other tools*, volume 59. Springer, London, 2007.

[69] G Hu, L Hu, J Song, P Li, X Chen, and H Li. Advances in neural networks-isnn 2010, 2010.

[70] Ching-Lai Hwang, Frank A Tillman, and Way Kuo. Reliability optimization by generalized lagrangian-function and reduced-gradient methods. *IEEE Transactions on reliability*, 28(4):316–319, 1979.

[71] Tongdan Jin. *Reliability engineering and services*. John Wiley & Sons, 2019.

[72] Yaochu Jin, Handing Wang, and Chaoli Sun. *Data-driven evolutionary optimization*. Springer, Switzerland, first edition, 2021.

[73] Hubertus Th Jongen, Klaus Meer, and Eberhard Triesch. *Optimization theory*. Springer Science & Business Media, Boston, first edition, 2007.

[74] Nocedal Jorge and J Wright Stephen. *Numerical optimization*. Spinger, USA, second edition, 2006.

[75] Dieter Jungnickel and D Jungnickel. *Graphs, networks and algorithms*, volume 3. Springer, Berlin, fourth edition, 2005.

[76] Peter Dueholm Justesen. Multi-objective optimization using evolutionary algorithms. *University of Aarhus, Department of Computer Science, Denmark*, 33, 2009.

[77] Man Cheol Kim. Reliability block diagram with general gates and its application to system reliability analysis. *Annals of Nuclear Energy*, 38(11):2456–2461, 2011.

[78] Young H Kim, Kenneth E Case, and PM Ghare. A method for computing complex system reliability. *IEEE Transactions on Reliability*, 21(4):215–219, 1972.

[79] Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, USA, 2019.

[80] Khee-meng Koh, Fengming Dong, and Eng Guan Tay. *Introduction to graph theory: H3 mathematics*. World Scientific Publishing Company, USA, 2007.

[81] Slawomir Koziel and Xin-She Yang. *Computational optimization, methods and algorithms*, volume 356. Springer, Berlin, 2011.

[82] Hiromitsu Kumamoto. *Satisfying safety goals by probabilistic risk assessment*. Springer Science & Business Media, London, 2007th edition, 2007.

[83] Sunita Kumawat. A graph theoretic approach: Petri net. *International Journal of Mathematical Sciences and Applications*, 1(3):1637–1641, 2011.

[84] Prajna Kunche and KVVS Reddy. *Metaheuristic applications to speech enhancement*. Springer, Switzerland, first edition, 2016.

[85] Way Kuo and Rajendra Prasad. System reliability optimization: an overview. *Mathematical Reliability: An Expository Perspective*, pages 31–54, 2004.

[86] Way Kuo and Ming J Zuo. *Optimal reliability modeling: principles and applications*. John Wiley & Sons, USA, first edition, 2003.

[87] Yasser Lamalem and Khalid Housni. New and efficient method to find all minimal paths. In *2020 3rd International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–4. IEEE, 2020.

[88] Steven R Lay. *Convex sets and their applications*. Courier Corporation, 2007.

[89] Massimo Lazzaroni. *Reliability engineering: basic concepts and applications in ICT*. Springer Science & Business Media, Berlin, first edition, 2011.

[90] Lei Lei, Chuang Lin, and Zhangdui Zhong. *Stochastic Petri Nets for Wireless Networks*. Springer, London, first edition, 2019.

[91] Gregory Levitin et al. *The universal generating function in reliability analysis and optimization*, volume 6. Springer, London, 2005th edition, 2005.

[92] Hongbin Li and Qing Zhao. A cut/tie set method for reliability evaluation of control systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1048–1053. IEEE, 2005.

[93] Jie Li and Wei Liu. *Lifeline Engineering Systems*. Springer, Canada, first edition, 2021.

[94] Wei Li, Qiang Zeng, Ling Shen, and Ze Bin Zhang. A multi-objective optimization method for system reliability allocation. In *Advanced Materials Research*, volume 860, pages 2766–2773. Trans Tech Publ, 2014.

[95] Cheng-Shih Liaw, Yung-Chia Chang, Kuei-Hu Chang, and Thing-Yuan Chang. Me-owa based dematel reliability apportionment method. *Expert Systems with Applications*, 38(8):9713–9723, 2011.

[96] Fran Sérgio Lobato and Valder Steffen Jr. *Multi-objective optimization problems: concepts and self-adaptive parameters with mathematical and engineering applications*. Springer, Switzerland, first edition, 2017.

[97] David G Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer,, London, third edition, 2016.

[98] Jie Ma, Sen Yu, and Wei Cheng. Composite fault diagnosis of rolling bearing based on chaotic honey badger algorithm optimizing vmd and elm. *Machines*, 10(6):469, 2022.

[99] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[100] Nikos E Mastorakis. Genetic algorithms and nelder-mead method for the solution of boundary value problems with the collocation method. *WSEAS Transactions on Information Science and Applications*, 2(11):2016–2020, 2005.

[101] Khizer Mehmood, Naveed Ishtiaq Chaudhary, Zeshan Aslam Khan, Khalid Mehmood Cheema, Muhammad Asif Zahoor Raja, Ahmad H Milyani, and Abdullah Ahmed Azhari. Dwarf mongoose optimization metaheuristics for autoregressive exogenous model identification. *Mathematics*, 10(20):3821, 2022.

[102] Manuel Melgosa. Standard colorimetry: Definitions, algorithms and software claudio oleari chichester: John wiley; 2016 isbn: 9781118894446; 509 pp. 95.00paperback; 76.99, 2017.

[103] Mohamed Arezki Mellal and Enrico Zio. An adaptive particle swarm optimization method for multi-objective system reliability optimization. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 233(6):990–1001, 2019.

[104] Lucija Mihalj. *Penalty methods in constrained optimization*. PhD thesis, University of Zagreb. Faculty of Science. Department of Mathematics, 2019.

[105] Krishna B Misra. *Reliability analysis and prediction: A methodology oriented treatment*. Elsevier, New York, first edition, 2012.

[106] Mohammad Modarres. *What every engineer should know about reliability and risk analysis*. CRC Press, New York, first edition, 2018.

[107] Nuha Sami Mohsin, Rafah Shihab Alhamdani, and Buthainah Fahran Abd Al-Dulaimi. An integrated grey wolf optimizer with nelder-mead method for workflow

scheduling problem. In *2020 Emerging Technology in Computing, Communication and Electronics (ETCCE)*, pages 1–6. IEEE, 2020.

[108] Marcin Molga and Czesław Smutnicki. Test functions for optimization needs. *Test functions for optimization needs*, 101:48, 2005.

[109] Michael K. Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on computers*, 31(09):913–917, 1982.

[110] Dodderi Narshima Prabhakar Murthy, Marvin Rausand, and Trond Østerås. *Product reliability: specification and performance*. Springer, London, 2008th edition, 2008.

[111] Bent Natvig. *Multistate systems reliability theory with applications*. John Wiley & Sons, United Kingdom, first edition, 2010.

[112] Sukanta Nayak. *Fundamentals of Optimization Techniques with Algorithms*. Academic Press, United States, first edition, 2020.

[113] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[114] Andrew N O'Connor. *Probability distributions used in reliability engineering*. RiAC, USA, first edition, 2011.

[115] K Rajmohan Padiyar and Anil M Kulkarni. *Dynamics and control of electric transmission and microgrids*. Wiley-IEEE Press, USA., first edition, 2019.

[116] Konstantinos E Parsopoulos and Michael N Vrahatis. *Particle swarm optimization and intelligence: advances and applications: advances and applications*. IGI global, New York, first edition, 2010.

[117] Md Saidur Rahman et al. *Basic graph theory*, volume 9. Springer, 2017.

[118] Suresh Rai and KK Aggarwal. An efficient method for reliability evaluation of a general network. *IEEE Transactions on Reliability*, 27(3):206–211, 1978.

[119] Abhishek Rajan and T Malakar. Optimal reactive power dispatch using hybrid nelder–mead simplex based firefly algorithm. *International Journal of Electrical Power & Energy Systems*, 66:9–24, 2015.

[120] Marvin Rausand. *Risk assessment: theory, methods, and applications*, volume 115. John Wiley & Sons, Canada, first edition, 2013.

[121] Marvin Rausand. *Reliability of safety-critical systems: theory and applications*. John Wiley & Sons, Canada, first edition, 2014.

[122] Marvin Rausand and Arnljot Hoyland. *System reliability theory: models, statistical methods, and applications*. John Wiley & Sons, Canada, second edition, 2003.

[123] Mohamed-Larbi Rebaiaia and Daoud Ait-Kadi. A new technique for generating minimal cut sets in nontrivial network. *AASRI Procedia*, 5:67–76, 2013.

[124] Gintaras V Reklaitis, A Ravindran, and Kenneth M Ragsdell. *Engineering optimization: Methods and applications*. Wiley New York, 1983.

[125] Sheldon M Ross. *Introduction to probability models*. Academic press, USA, tenth edition, 2014.

[126] Andrzej Ruszczynski. *Nonlinear optimization*. Princeton university press, USA, first edition, 2011.

[127] Ruhul Amin Sarker and Charles S Newton. *Optimization modelling: a practical approach*. CRC press, USA, first edition, 2007.

[128] Yoshikazu Sawaragi, HIROTAKA NAKAYAMA, and TETSUZO TANINO. *Theory of multiobjective optimization*. Elsevier, London, first edition, 1985.

[129] Yuanlong Shen. A new simple algorithm for enumerating all minimal paths and cuts of a graph. *Microelectronics Reliability*, 35(6):973–976, 1995.

[130] Oliver Sinnen. *Task scheduling for parallel systems.* John Wiley & Sons, 2007.

[131] Keyue Smedley. Reliability analysis for leb ring magnet power system in ssc. *IEEE Transactions on Nuclear Science*, 39(4):1170–1174, 1992.

[132] Jan A Snyman and Daniel N Wilke. Practical mathematical optimization: Basic optimization theory and gradient-based algorithms, o$\lambda$. 133, 2018.

[133] A Spiteri Staines. Rewriting petri nets as directed graphs. *International Journal of Computers*, 5(2):289–297, 2011.

[134] Ivan Stanimirović. *Advances in optimization and linear programming.* Apple Academic Press, Canada, first edition, 2022.

[135] Wenyu Sun and Ya-Xiang Yuan. *Optimization theory and methods: nonlinear programming*, volume 1. Springer Science & Business Media, 2006.

[136] El-Ghazali Talbi et al. *Hybrid metaheuristics*, volume 166. Springer, London, 2013.

[137] Roberta Terruggia. Reliability analysis of probabilistic networks. *XXII Ciclo Gennaio*, 2010.

[138] Michael Tortorella. *Reliability theory: With applications to preventive maintenance.* Taylor & Francis, NewYork, second edition, 2001.

[139] Michael Tortorella. *Reliability, maintainability, and supportability: best practices for systems engineers.* John Wiley & Sons, Canada, first edition, 2015.

[140] C Vasudev. *Graph theory with applications.* New Age International, New Delhi, first edition, 2006.

[141] VI Voloshin. *Introduction to Graph and Hypergraph Theory. Nova Science Publishers, Inc.* 2009.

[142] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[143] Jianzhong Xu and Fu Yan. Hybrid nelder–mead algorithm and dragonfly algorithm for function optimization and the training of a multilayer perceptron. *Arabian Journal for Science and Engineering*, 44(4):3473–3487, 2019.

[144] Xiang Xu, Xinbo Chen, Zhe Liu, Junhao Yang, Yanan Xu, Yong Zhang, and Yunkai Gao. Multi-objective reliability-based design optimization for the reducer housing of electric vehicles. *Engineering Optimization*, pages 1–17, 2021.

[145] Zia Yamayee, Kathleen Sidenblad, and Miki Yoshimura. A computationally efficient optimal maintenance scheduling method. *IEEE Transactions on Power Apparatus and Systems*, (2):330–338, 1983.

[146] Guang Yang. *Life cycle reliability engineering*. John Wiley & Sons, Canada., first edition, 2007.

[147] Xin-She Yang. *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, Canada, second edition, 2010.

[148] Xin-She Yang. *Optimization techniques and applications with examples*. John Wiley & Sons, 2018.

[149] Ali Riza Yildiz. A novel hybrid whale–nelder–mead algorithm for optimization of design and manufacturing problems. *The International Journal of Advanced Manufacturing Technology*, 105(12):5091–5104, 2019.

[150] Erwie Zahara and Yi-Tung Kao. Hybrid nelder–mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36(2):3880–3886, 2009.

[151] Dongliang Zhang, Kaiwen Zhang, Liufeng Wang, and Qinqin Hong. Reliability modeling and analysis of reactor protect system based on petri net. In *Journal of Physics: Conference Series*, volume 1754, page 012059. IOP Publishing, 2021.

[152] Jian-Hua Zhao, Zhaoheng Liu, and My-Thien Dao. Reliability optimization using multiobjective ant colony system approaches. *Reliability Engineering & System Safety*, 92(1):109–120, 2007.

[153] Kaifeng Zheng, Chao Wang, Weizheng An, and Likun Ge. A hybrid method combining tab search and nelder-mead algorithms for global continuous optimization problems. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 17–24. IEEE, 2018.

[154] Minghui Zhu and Sonia Martinez. On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Automatic Control*, 57(1):151–164, 2011.

[155] Enrico Zio. *An introduction to the basics of reliability and risk analysis*, volume 13. World scientific, London, first edition, 2007.

[156] Peter Zörnig. Nonlinear programming. De Gruyter, Berlin/Boston, first edition, 2014.

# APPENDIX

ALGORITHM 1: Pseudo code of the HBA

Step 1:Set the parameters initialized $t_{max}$, $\beta$, $C$
Step 2:Set the initial number of options $N$.
Step 3:Use an objective function to assess each honey badger position's fitness, and assign to $f_i$, $i \in [1, 2, ..., N]$.
Step 4:Assign fitness to $f_{prey}$ and save the optimal position for $x_{prey}$ .
Step 5: Repeat
Step 6: Utilizing equation ( 4.5 ), update the decreasing factor
Step 7:Equation ( 4.2 )should be used to calculate the intensity $I_i$.
Step 8: for $i : 1toN$ do
Step 9: if $r < 0.5$ then
Step 10:Equation ( 4.6 ) is used to update the location $x_{new}$.
Step11: Else
Step12:By utilizing equation( 4.8 ), modify position $x_{new}$
Step13: end if Assess the new position, then give it to $f_{new}$
Step14:if $f_{new} \leq f_i$ then
Step15:Set $x_i = x_{new}$ and $f_i = f_{new}$
Step16:end if $f_{new} \leq f_{prey}$ then
Step17: Set $x_{prey} = x_{new}$ and $f_{prey} = f_{new}$
Step18: end if
Step19: end for the iteration $(t)$ criterion has been satisfied.
Step20:Return the $x_{prey}$

ALGORITHM : Pseudo code of the DMOA.

Input: Set the algorithm's requirements and solutions.
Initialize algorithmic parameters settings and the solution
For iter= $1 : max_iter$
Determine the Mongoose Fitness Function.
Establish a timer $(C)$.
Using Equation(4.13 ),determine the alpha value.
using Equation(4.14 ), locate a potential food position

Estimate new fitness $X_{i+1}$

Calculate the average value for the sleeping mound as determined by Equation (4.15 ).

Equation (4.16 ) can be used to calculate the sleeping mound's average.

Equation ( 4.19 )be utilized to determine the movement vector.

Based on Equation, simulate the next location of the scout mongoose (4.17 ).

end for

$T = T + 1$

end while

Output: Return the best solution $(X)$

```
function [Xprey, Food_Score,CNVG] = HBA(objfunc,
dim,lb,ub,tmax,N)
beta      = ;    % the ability of HB to get the food Eq.(4)
C      = ;    %constant in Eq. (3)
vec_flag=[1,-1];
%initialization
X=initialization(N,dim,ub,lb);
%Evaluation
fitness = fun_calcobjfunc(objfunc, X);
[GYbest, gbest] = min(fitness);
Xprey = X(gbest,:);
for t = 1:tmax
   alpha=C*exp(-t/tmax);   %density factor in Eq. (3)
   I=Intensity(N,Xprey,X);  %intensity in Eq. (2)
   for i=1:N
      r =rand();
      F=vec_flag(floor(2*rand()+1));
      for j=1:1:dim
         di=((Xprey(j)-X(i,j)));
         if r<.5
            r3=rand;              r4=rand;              r5=rand;

            Xnew(i,j)=Xprey(j)  +F*beta*I(i)*
Xprey(j)+F*r3*alpha*(di)*abs(cos(2*pi*r4)*(1-cos(2*pi*r5)));
         else
            r7=rand;
            Xnew(i,j)=Xprey(j)+F*r7*alpha*di;
         end
      end

FU=Xnew(i,:)>ub;FL=Xnew(i,:)<lb;Xnew(i,:)=(Xnew
(i,:).*(~(FU+FL)))+ub.*FU+lb.*FL;

      tempFitness = fun_calcobjfunc(objfunc,
Xnew(i,:));
      if tempFitness<fitness(i)
         fitness(i)=tempFitness;
         X(i,:)= Xnew(i,:);
      end
   end
```

```
FU=X>ub;FL=X<lb;X=(X.*(~(FU+FL)))+ub.*FU+lb.*FL;
   [Ybest,index] = min(fitness);
   CNVG(t)=min(Ybest);
   if Ybest<GYbest
      GYbest=Ybest;
      Xprey = X(index,:);
   end
end
Food_Score = GYbest;
end
function Y = fun_calcobjfunc(func, X)
N = size(X,1);
for i = 1:N
   Y(i) = func(X(i,:));
end
end

function I=Intensity(N,Xprey,X)
for i=1:N-1
   di(i) =( norm((X(i,:)-Xprey+eps))).^2;
   S(i)=( norm((X(i,:)-X(i+1,:)+eps))).^2;
end
di(N)=( norm((X(N,:)-Xprey+eps))).^2;
S(N)=( norm((X(N,:)-X(1,:)+eps))).^2;
for i=1:N
   r2=rand;
   I(i)=r2*S(i)/(4*pi*di(i));
end
end
function [X]=initialization(N,dim,up,down)
if size(up,2)==1
   X=rand(N,dim).*(up-down)+down;
end
if size(up,2)>1
   for i=1:dim
      high=up(i);low=down(i);
      X(:,i)=rand(N,1).*(high-low)+low;
   end
```

```
VarMin=; %VarMax=
 MaxIt=1000;
 nPop=;
nBabysitter=;
nAlphaGroup=nPop-nBabysittergroup
nScout=nAlphaGroup;
empty_mongoose.Position=[];
empty_mongoose.Cost=[];
pop=repmat(empty_mongoose,nAlphaGroup,1);
BestSol.Cost=inf;
tau=inf;
Iter=1;
sm=inf(nAlphaGroup,1);
for i=1:nAlphaGroup
   pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
   pop(i).Cost=F_obj(pop(i).Position);
   if pop(i).Cost<=BestSol.Cost
      BestSol=pop(i);
end
C=zeros(nAlphaGroup,1);
CF=(1-Iter/MaxIt)^(2*Iter/MaxIt);
BestCost=zeros(MaxIt,1);
for it=1:MaxIt
F=zeros(nAlphaGroup,1);
    MeanCost = mean([pop.Cost]);
   for i=1:nAlphaGroup
F(i) = exp(-pop(i).Cost/MeanCost);
   end
      P=F/sum(F);
for m=1:nAlphaGroup
i=RouletteWheelSelection(P);
K=[1:i-1  i+1:nAlphaGroup];
      k=K(randi([1  numel(K)]));
phi=(peep/2)*unifrnd(-1,+1,VarSize);
newpop.Position=pop(i).Position+phi.*(pop(i).Position-pop(k).Position);
if newpop.Cost<=pop(i).Cost
pop(i)=newpop;
else
C(i)=C(i)+1;
 end
```

```
  for i=1:nScout
K=[1:i-1  i+1:nAlphaGroup];
      k=K(randi([1  numel(K)]));
phi=(peep/2)*unifrnd(-1,+1,VarSize);
newpop.Position=pop(i).Position+phi.*(pop(i).Position-pop(k).Position);
newpop.Cost=F_obj(newpop.Position);

sm(i)=(newpop.Cost-pop(i).Cost)/max(newpop.Cost,pop(i).Cost);
if newpop.Cost<=pop(i).Cost
          pop(i)=newpop;
else
C(i)=C(i)+1;
 end
  end
for i=1:nBabysitter
      if C(i)>=L
pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
pop(i).Cost=F_obj(pop(i).Position);
C(i)=0;
end
end for i=1:nAlphaGroup
if pop(i).Cost<=BestSol.Cost
BestSol=pop(i);
End
newtau=mean(sm);
  for i=1:nScout
      M=(pop(i).Position.*sm(i))/pop(i).Position;
      if newtau>tau
        newpop.Position=pop(i).Position-CF*phi*rand.*(pop(i).Position-M);
      else
        newpop.Position=pop(i).Position+CF*phi*rand.*(pop(i).Position-M);
      end
      tau=newtau;
  end
for i=1:nAlphaGroup
      if pop(i).Cost<=BestSol.Cost
         BestSol=pop(i);
      end
   end
   BestCost(it)=BestSol.Cost;
   BEF=BestSol.Cost;
   BEP=BestSol.Position;
```

Nelder-Mead Method

```
 rho = ; psi =; sigma =; gamma=;
[cost,m] = sort(cost);
fv1=X(m,:);
X(:,:)=fv1;
xbar = sum(X(1:nAlphaGroup-1,:),  1)/(nAlphaGroup-1);
xr = xbar + rho*(xbar-X(end,:));
xr = correct(xr, lb, ub);
if (cost(1)<Obj_func(xr) && cost(end-1)>Obj_func(xr))
X(end,:)=xr;
end
if (cost(1)>Obj_func(xr))
xe = xbar +gamma*(xr-xbar);
xe = correct(xe, lb, ub);
 else
xcc = (1-psi)*xbar + psi*X(end,:);
xcc = correct(xcc, lb, ub);
if Obj_func(xcc)<cost(end)
X(end,:)=xcc;
else
for k=2:nAlphaGroup
X(k,:)= X(k,:)+sigma*(X(k,:)-X(1,:));
X(k,:) = correct(X(k,:), lb, ub);
end
else
xcc = (1-psi)*xbar + psi*X(end,:);
xcc = correct(xcc, lb, ub);
if Obj_func(xcc)<cost(end)
X(end,:)=xcc;
End
```

# الملخص


تقدم هذه الاطروحة تقنيات جديدة تهدف الى زيادة موثوقية شـــبكة الإطفاء الموجودة داخل مفاعل نووي باســـتعمال خوارزميات هجينة مكونة من الخوارزميات الحدســـية وفوق الحدسـية ، وقد تم إنشـاء النظام عن طريق تحويل شـبكة بيتري لنظام اطفاء التشـغيل الى شـبكة معقدة.

وللعثور على الحد الأدنى من مجموعات المســـارات قدمنا خمس تقنيات هي خوارزمية Rai و Aggarwal ، ضـرب المصـفوفة ، تعداد مجموعة المسـار، خوارزمية مصـفوفة العقدة التابعة، وخوارزمية العقدتين الطرفيتين. كما تم اســـتخدام ثلاث تقنيات هي خوارزمية العقد والحواف، خوارزمية Shal وتقنية جديدة تدعى تقنية رؤى لإيجاد الحد الأدنى من مجموعات القطع. لحساب متعدد الحدود لشبكة الاطفاء ، سنستخدم تقنية مجموع المنتجات المنفصلة وتقنية الحد الأدنى من القطع.

ولدراسة تحسين وزيادة موثوقية نظام الاطفاء، نعمل اولا" على تصميم نموذجيين رياضـيين للتحسين من خلال الجمع بين الخوارزميات فوق الحدسية مع طريقة نيلدر – ميد ، حيث تم تهجين خوارزمية غرير العسل (HBA) مع طريقة نيلدر – ميد (NM)، حيث كانت نتائج التهجين هي الخوارزمية الهجينة HBNMA، ومن تهجين خوارزمية النمس القزم (DMOA) مع طريقة نيلدر – ميد حصلنا على الخوارزمية الهجينة DMONMA.

وللتحقق من متانة وفعالية الخوارزميات الهجينة، تم اســتخدام ثلاث وعشــرين دالة اختبارية، وتم التحقق من نتائج هذه الدوال باســتخدام التقييم الإحصــائي للمتوســط والانحراف المعياري وزمن تنفيذ الخوارزميات. وجدنا أن الخوارزميات الهجينة حســـنت معظم الوظائف مقارنة بالخوارزميات الأصلية.

الخطوة اللاحقة لإجراء التحســـين وزيادة الموثوقية للنظام المعطى ســـتتم عن طريق تهجين طريقة الجزاء مع خوارزميات P-HBA، P-HBNMA، P-DMOA، وP-DMONMA

وسنقدم مقارنة بين نتائج الخوارزميات التي تستخدم تقنية التهجين مع نتائج الخوارزميات التي لا تستخدم هذه التقنية. وللحصول على حلول بيتري لحل مشاكل الامثلية غير الخطية متعددة الأهداف لشبكة إيقاف التشغيل، تم استخدام طريقة مجموع الوزن لتحويل مسألة دالة متعددة الأهداف إلى دالة هدف واحدة مع دراسة ثلاثة نماذج رياضية لتصميم نموذج رياضي للامثلية متعددة الأهداف لزيادة قيمة الموثوقية وتقليل التكلفة ، وقد أظهرت نتائج الخوارزميات الهجينة لـ P-HBA، P-HBNMA، P-DMOA وP-DMONMA ، انه يمكن العثور على أفضل قيمة للموثوقية وأقل تكلفة مقارنة بالخوارزميات الأخرى. نستنتج أن الدمج بين HBA و DMOA مع طريقة Penalty و طريقة نيلدر – ميد يعطي أفضل الحلول بالإضافة إلى تقليل زمن التنفيذ مقارنة بزمن تنفيذ الخوارزميات الأخرى.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية التربية للعلوم الصرفة
قسم الرياضيات

# تحسين موثوقية نظام الاطفاء باستخدام خوارزميات فوق الحدسية المهجنة

اطروحة

مقدمة الى مجلس كلية التربية للعلوم الصرفة / جامعة بابل وهي

جزء من متطلبات نيل درجة الدكتوراه فلسفة في التربية /

الرياضيات

من قبل الطالبة

**رؤى عزيز فاضل محسن**

بإشراف

**أ. د. زاهر عبد الهادي حسن**

**1445** هـ **2023** م