Republic of Iraq

Ministry of Higher Education & Scientific Research

University of Babylon

College of Education for Pure Sciences

Department of Mathematics

Solving Fuzzy Reliability Optimization Problems by Using Fuzzy Non Linear Programming

A Dissertation

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

Babylon University in Partial Fulfillment of the Requirements for

Doctor of Philosophy in Education / Mathematics.

By

Ahmed Abdulhussein Jabber Hamoud

Supervised by

prof.Dr. Audi Sabri Abd ALRazzaq

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

يَرْفَعْ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ

صدق الله العظيم

سورة المجادلة: من الآية ١١

The Supervisor Certificate

I certify that this dissertation entitled " " was prepared by the student " Ahmed Abdulhussein Jabber Hamoud " under my supervision at the University of Babylon, College of Education for Pure Sciences as a partial fulfillment of the requirement the Doctor of Philosophy in Education /Mathematics.

Signature: Name: prof.Dr. Audi Sabri Abd ALRazzaq

Scientific Grade: Professor

Date: / / 2023

According to the available recommendation, I forward this dissertation for debate by the examining committee.

Signature:

Name: Dr.

Scientific Grade:

Address:

Date: / / 2023

Scientific Supervisor's Certification

This is to certify that I have read this dissertation entitled " Solving Fuzzy Reliability Optimization Problems by Using Fuzzy Non Linear Programming " and I found it is qualified for debate.

Signature:

Name:

Title:

Date: / / 2023

Linguistic Supervisor's Certification

This is to certify that I have read this dissertation entitled " Solving Fuzzy Reliability Optimization Problems by Using Fuzzy Non Linear Programming" and I found it is qualified for debate.

Signature:

Name:

Title:

Date: / / 2023

Examining Committee Certificate

We certify that we have read this dissertation entitled " Solving Fuzzy Reliability Optimization Problems by Using Fuzzy Non Linear Programming " as an examination committee, examined the student " Ahmed Abdulhussein Jabber Hamoud " in its contents and that in our opinion it meets the standard of a dissertation for the Doctor of Philosophy in Education/ Mathematics.

Signature:
Name:
Title:
Date : / / 2023
Chairman

Signature:
Name :
Title:
Date : / / 2023
Member

Signature:
Name:
Title:
Date : / / 2023
Member

Signature:
Name:
Title:
Date : / / 2023
Member

Signature:
Name :
Title:
Date : / / 2023
Member / Advisor

Approved by the Dean of College

Signature:
Name:
Title:
Address: Dean of the College of Education for Pure Sciences
Date : / / 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 1: **Notations Used in the Dissertation**

$f(x)$	Objective Function
$Minf(x)$	The Minimum of Objective Function
$Maxf(x)$	The Maximum of Objective Function
$g(x)$	The Inequality Constraint
$h(x)$	The Equality Constraint
C	The Convex Set
$\mathbb{R}$	Real Numbers
$\mathbb{R}^n$	Dimensional Space
$\nabla f(x)$	The Gradient of f
$\nabla^2 f(x)$	The Hessian of f
x^*	The Minimizer (Local, Global)
x^{k+1}	The Next Iterate
x^k	The Current Iterate
α_k	The Step Length
$NLPP$	Non Linear Programming Problems
KKT	Karush-Kuhn-Tucker Conditions
DFP	Method of Davidon, Fletcher, and Powell
$BFGS$	Method of Broyden, Fletcher, Golfarb, and Shanno
$\mathcal{C}$	The Cone
$\mathcal{L}$	The Lagrangian Function
$\mathcal{P}(x)$	The Penalty Function
$B(x)$	The Interior Penalty Function
$E(x)$	The Exterior Penalty Function
$ALAG$	The Augmented Lagrangian Penalty Function
GP	General Problem

Table 2: **Notations Used in the Dissertation**

$NRBDO$	Non-linear Probabilistic Reliability
$RBDO$	Reliability Based Design Optimization
VS	Vague
$MOLP$	Multi-Objective Linear problem
MCS	Minimal Cut Set
FFT	Fuzzy Fault treat
$RBDO$	Reliability Based Design Optimization

Abstract

Both the Adam method and gradient method are optimization algorithms used to solve constraint optimization problems. However, there are some key differences between these two methods. The gradient method is a first-order optimization algorithm that updates the parameters by taking small steps in the direction of the negative gradient of the objective function. The main disadvantage of the gradient method is that it can get stuck in local minima, and it may take a long time to converge to the global minimum. On the other hand, the Adam method is a second-order optimization algorithm that combines the advantages of the gradient method with a momentum-based approach. Adam uses adaptive learning rates, which allows it to converge faster than the gradient method, and it is less likely to get stuck in local minima. In terms of convergence rate, Adam is generally faster than the gradient method, especially for problems with high-dimensional or noisy data. However, Adam can be sensitive to the choice of hyperparameters, and it may converge to suboptimal solutions if the hyperparameters are not tuned properly. In summary, while both methods are useful for solving constraint optimization problems, the Adam method may be more effective for problems with high-dimensional or noisy data, while the gradient method may be more suitable for simpler problems. The choice of method ultimately depends on the specific problem and the available computational resources. Engineering faces a critical challenge in optimizing system reliability, which aims to maximize the potential of the system to function properly throughout its lifespan. However, in real-world situations, the precise reliability objective may be uncertain and unclear. As a result, a novel approach is presented in this dissertation to address the optimization of the fuzzy objective function for system reliability using a hybrid algorithm. This algorithm combines two meta-heuristic optimization techniques, particle swarm optimization and simulated annealing , to leverage their respective advantages.

ACKNOWLEDGEMENTS

I might want to express my special thanks to many people who so lavishly take part to this study. My great respect to everyone who helped me scientifically and sentimentally I every time praise be to Allah who gave me the power to conduct this work.

First of all, my keen supervisor merit a special mention,prof.Dr. Audi Sabri Abd ALRazzaq. My Ph.D. has been an amazing experience, and I thank prof.Dr. Audi Sabri Abd ALRazzaq whole- heatedly, not only for his great academic support but also for giving me the Chance to learn so many wonderful life skills. He was always there when I needed him. He taught me how to be a good researcher and optimistic, even in the dark. He made every thinks voyage.

I owe special thanks to my family who supported me and helped me throughout my life and during this study. It is very hard to express how much support I have . Finally, thanks go to all of my friends for almost every support and chance they have provided me to finish my work successfully. I dedicate this work to all of those important people in my world.

Devotion

For you, father, mother, wife, my children and brothers

CHAPTER 1

INTRODUCTION

As the scope of optimization continues to expand through advancements in science the problems being tackled are becoming increasingly larger and more complex [74]. Modeling is the process of identifying the objective factors and requirements for a given problem, and optimization is a crucial numerical technique used to determine the optimal values of these factors that will yield the maximum benefit for a given mathematical function [91]. Optimization algorithms are a powerful and efficient approach to solving numerical problems, typically with the aid of a computer. The optimization process involves finding the best solution from a set of solutions to a particular problem. Linear programming refers to a system with linear constraints and a linear objective function, while Non linear programming involves at least one Non-linear constraint or objective function [60]. If both the constraint and objective functions are convex, it is referred to as convex programming. In mathematics, the goal is to find the minimum or maximum value of the objective function with i variables, where i equals 1 to n. Mathematical optimization is a field of applied mathematics that is widely applicable across various domains. Optimizing the reliability of a system with a fuzzy objective function requires a multi-objective optimization approach. In such an approach, multiple objectives are considered simultaneously,The objective of finding a

set of solutions that optimize all objectives simultaneously can be tackled using a fuzzy set theory-based approach. This approach involves defining the objective function using fuzzy sets and fuzzy logic [98]. Fuzzy sets are characterized by membership functions that are not binary but instead reflect varying degrees of membership ranging from 0 to 1. Fuzzy logic provides a mathematical framework for handling uncertainty and imprecision [86]. To solve the problem using a fuzzy set theory-based approach, we first need to define the objectives and constraints of the problem in terms of fuzzy sets. For example, the reliability of the system could be defined as a fuzzy set with membership functions that represent the degree of reliability of the system. Similarly, the constraints of the problem, such as cost or time, could be defined as fuzzy sets [43]. Once the objectives and constraints are defined in terms of fuzzy sets, we can use fuzzy optimization algorithms to find the optimal solution(s). Fuzzy optimization algorithms use fuzzy set theory and fuzzy logic to evaluate and compare solutions based on their degree of satisfaction of the objectives and constraints [82]. One popular fuzzy optimization algorithm is the fuzzy-genetic algorithm, which is a hybrid of the genetic algorithm and fuzzy set theory. The fuzzy-genetic algorithm uses a population-based approach to search for optimal solutions, and evaluates the fitness of solutions based on their degree of satisfaction of the fuzzy objectives and constraints [85]. In summary, solving a fuzzy objective function of the system reliability optimization requires a multi-objective optimization approach that incorporates fuzzy set theory and fuzzy logic [22]. Fuzzy optimization algorithms, such as the fuzzy-genetic algorithm, can be used to find optimal solutions that simultaneously optimize all objectives and constraints [46]. System reliability optimization is the process of improving the reliability of a system to meet certain performance and safety requirements [15]. A reliable system is one that can operate for a given period without failure or downtime. The optimization process involves identifying the weakest links in the system and addressing them through design improvements or maintenance.

Related Work

The earlier works that are itemized in the structure of this paper are isolated into five fundamental classifications:

1. **Mathematical Model (Linear and Non linear Programming)**
2. **Optimization Technique**
3. **Reliability Systems**
4. **Fuzzy logic**
5. **Optimization Reliability**
6. **Fuzzy optimization**
7. **Fuzzy Reliability**
8. **Optimization Algorithms**

1.0.1 Mathematical Model (Linear and Non linear Programming)

The mathematical theory of linear programming is highly adaptable and has been effective in representing practical problems. Nonetheless, several fascinating optimization problems are nonlinear, necessitating a diverse blend of linear algebra, multivariate calculus, numerical analysis, and computing methods [60]. Important areas of research in nonlinear optimization include devising computational algorithms (such as interior point techniques for linear programming), analyzing and interpreting the geometry of convex sets and functions, and exploring structured problems such as quadratic programming. Non linear optimization is critical in comprehending mathematical analysis and is extensively utilized in various domains, including engineering design, inventory management, regression analysis, geophysical exploration,

and economics [91]. Linear programming has a rich history of development, with significant advancements made by numerous mathematicians over the years. In 1762, Lagrange tackled optimization problems that had simple equality constraints. In 1820, Gauss solved linear systems of equations using the now-famous Gaussian elimination method. In 1866, Wilhelm Jordan refined the method to find least squared errors, which is now known as the Gauss-Jordan method. The emergence of digital computers in 1945 opened up new possibilities for solving optimization problems [74]. In 1947, George Dantzig invented the Simplex Method, which is a widely used method for solving linear programming problems. In 1968, Fiacco and McCormick introduced the Interior Point Method for solving linear programming problems. In 1984, Karmarkar applied the Interior Point Method to solve Linear Programs and introduced his innovative analysis, which greatly improved the efficiency of the method [81]. Quadratic programming is a mathematical model used in optimization that involves solving problems where the objective function and constraints are quadratic. In contrast, linear programming deals with problems where the objective function and constraints are linear. Nonlinear programming (NLP) is a subfield of mathematical optimization that involves finding the extreme values of an objective function over a set of real variables while satisfying a system of equality and inequality constraints [7]. NLP is used to solve optimization problems where some or all of the constraints or the objective function are nonlinear. Nonlinear programming is a crucial area of study in mathematics, as many real-world optimization problems are inherently nonlinear in nature. Mathematical modeling involves using mathematical concepts and language to describe a system. It is widely used in various fields, including natural sciences such as physics, biology, and chemistry, engineering disciplines like computer science and electrical engineering, and social sciences like economics, psychology, and political science [48]. Operations research also utilizes mathematical models to solve problems in business or military operations. Mathematical models are even used in music, linguistics, and philosophy, such as in analytic philosophy. Nonlinear programming (NLP) addresses the problem of optimizing

an objective function within the constraints of equality and inequality. If all functions are linear, it is referred to as a linear program; otherwise, it is a nonlinear program [39]. The development of highly efficient and robust linear programming algorithms and software, along with the advent of high-speed computers, has contributed to the importance of LP in solving problems in various fields. However, the non linearity of the objective functions and constraints in some practical situations cannot be fully explained or predicted through LP alone. Consequently, significant efforts have been made in recent decades to efficiently address such nonlinear problems. The field of nonlinear modeling of real-world problems has seen rapid progress in the past century. Due to the uncertainty that exists in all aspects of nature and human life, fuzzy systems are often used to view these models [69].

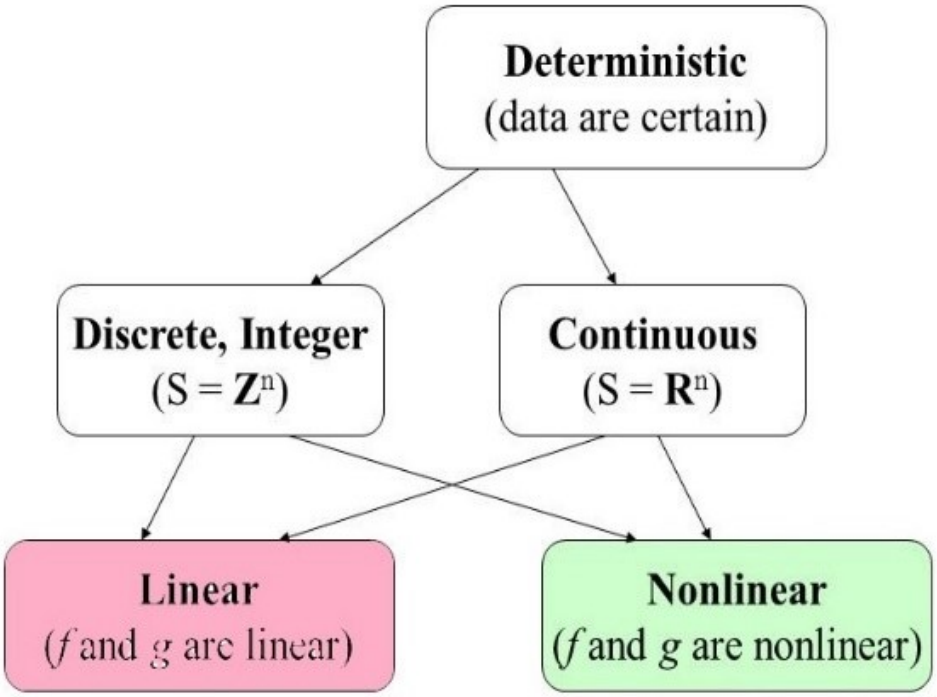

Figure 1.1: Tree linear and non linear

1.0.2 Optimization Technique

Optimization models aim to mathematically express the best solution to a problem, whether it involves maximizing profit, minimizing loss, increasing efficiency, reducing risk, or achieving any other goal [77]. These models are ubiquitous and arise in almost every area of application, such as designing a bridge to minimize weight or maximize strength, or selecting a flight plan for an aircraft to minimize time or fuel use. Optimization models have been used for centuries, and their importance has only grown in recent times as businesses become larger and more complex, and engineering designs become more ambitious [75]. In many cases, decision-making without the aid of optimization models is no longer possible or economically feasible. For instance, in a large multinational corporation, a minor improvement in operations could lead to a significant increase in profit, but analyzing all divisions of the corporation to achieve this improvement would be an enormous task [64]. Similarly, designing a new computer chip with millions of transistors would be nearly impossible without the use of such models. Optimization models have even been used to explain natural laws, such as Fermat's derivation of the law of refraction for light. In 1947, the U.S. Air Force initiated intensive work that led to the development of the linear programming model [80]. This model was chosen because it was relatively simple mathematically, yet provided a general and practical framework for representing interdependent activities that share scarce resources. In the linear programming model, the system to be optimized is viewed as comprising various activities that require inputs (such as labor and raw materials) and produce outputs (such as finished goods and services) proportional to the level of activity. Activity levels are represented by non-negative numbers. The innovative aspect of this approach is the expression of the goal of the decision process in terms of minimizing or maximizing a linear objective function, such as maximizing possible sorties for the Air Force or maximizing profits in industry. Before 1947, practical planning was characterized by a set of rules and priorities imposed authoritatively [69].Previously, general objectives were not explicitly stated,

likely due to the difficulty of performing calculations to minimize an objective function while adhering to constraints. However, in 1947, the simplex method was introduced, and it proved to be an efficient solution to practical problems, as described in the section on the method. Interest in linear programming increased rapidly, and by 1951, its application had spread to industry. Today, it is challenging to find an industry that does not use some form of mathematical programming, although the extent and nature of its use vary widely, even within the same industry [80].

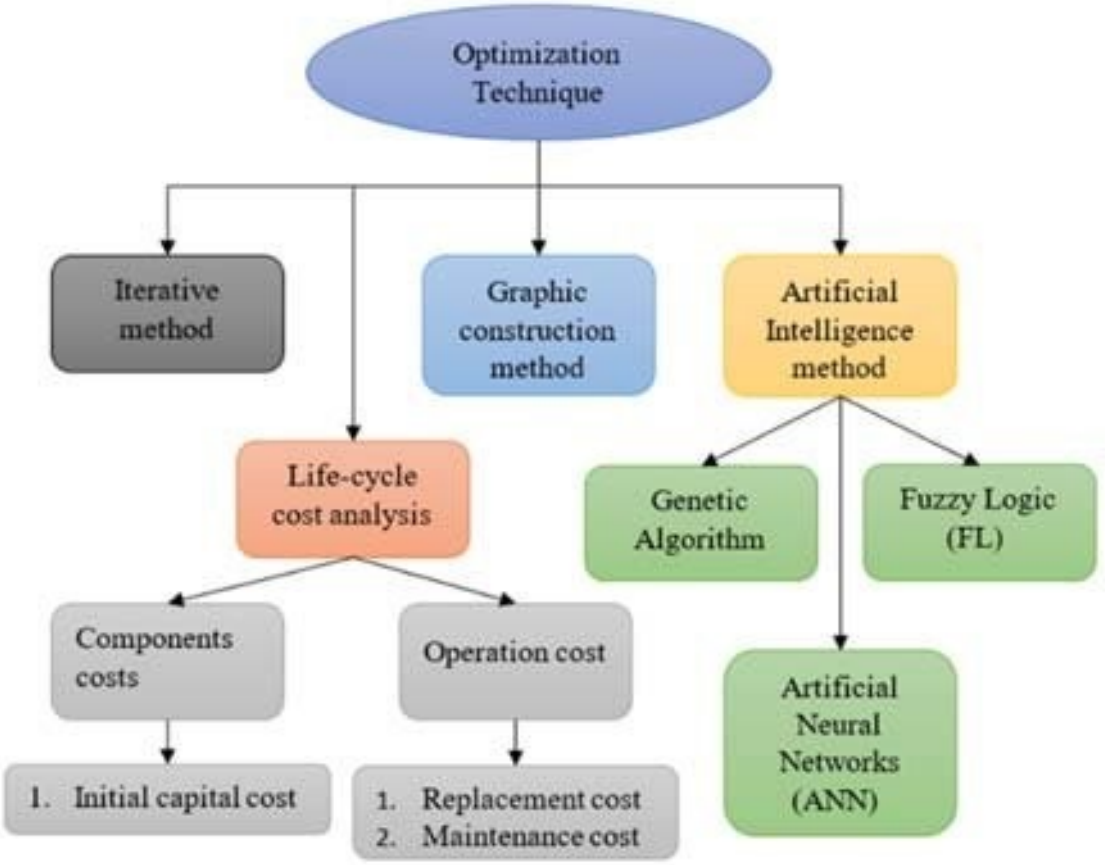

Figure 1.2: Logic Fuzzy tree

1.0.3 Reliability Systems

Reliability is the likelihood of an object completing its function over a specified amount of time under the operational parameters encountered. In 1961, H. A. Watson of Bell Telephone Laboratories devised the fault tree (FT) in response to a contract from the

United States Air Force to study the minuteman missile launch control system [78] . Sandler G. 1963 studied system reliability engineering and reliability estimation for series and parallel systems that did not require maintenance [78]. The fault tree analysis is a graphical representation of the logic involved with the evolution of a specific system failure (T) to fundamental failures (primary events). Fault trees are currently used in a variety of sectors to assess qualitative and quantitative risk. Using Boolean algebra, the FTA is used to transform the logical link between the top event and the fundamental events into mathematical equations known as minimal cut sets (MCS). A MCS is defined as "a cut set with the restriction that the absence of any one basic event from this set results in the absence of the top event" [8]. By evaluating the multiple minimal cut sets, a qualitative FTA may be utilized to assist understand how the top event happened. The top event probability (T) is obtained by assigning values to the probabilities of the underlying event using Boolean algebra. Dr. Lotfi Zadeh was assigned to look for and construct a fuzzy logic theory in 1965. The prospective method described by Zadeh, or fuzzy set theory, has played an important role in many various domains of science and technology, notably in system reliability and safety analyses [71]. Following that, the Japanese applied this reasoning to their manufacturing and industries, and it evolved to encompass the most technological areas. In)1983(K.T. Atanassov introduced intuitionistic fuzzy sets concepts. In (1990([13] Singer D. demonstrated how to use a fuzzy set in fault tree and reliability analysis. In (1993) [6] Gau, et al. introduced the notion of vague sets. In)2008(Jing-Shing Yao, et al. using triangular fuzzy numbers, the fuzzy system reliability Analysis was used [58]. In (2010) Wang Limin using fault tree analysis studied the reasons of an oil tank fire and explosion. In (2018) [8] Jiang, Ge, et alA novel approach to fuzzy dynamic fault tree analysis was introduced, utilizing the weakest n-dimensional t-norm arithmetic. Ghadhab, Majdi, et al. (2019) conducted a safety study of dynamic fault trees in vehicle guiding systems in the context of reliability engineering and system safety. Baek and Gyunyoung H [39]. (2021) demonstrated the application of dynamic fault tree analysis to prioritize electric power systems in nuclear power plants. Our work presents

an analytical study of the safety of a project that provides pure water, utilizing fuzzy fault tree analysis (FFT) to identify the minimal cut sets of events that lead to system failure [70].

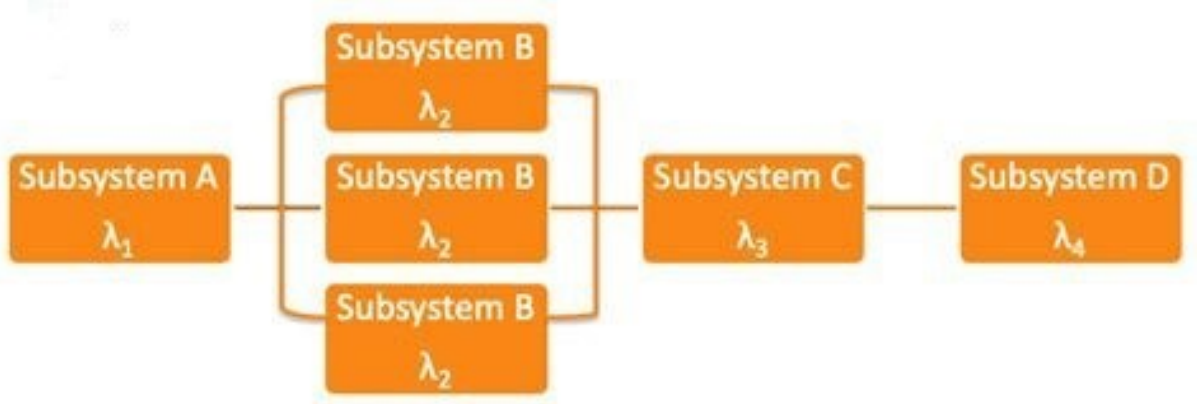

Figure 1.3: system sub-system

1.0.4 Fuzzy Fundamental

Fuzzy Logic is a type of many-valued logic that allows variables to take on any real number between 0 and 1, allowing for the handling of partial truth where truth values can range between completely true and completely false [13]. In contrast, Boolean logic only allows for truth values of 0 or 1 [20]. Fuzzy logic was first introduced in 1965 with the proposal of fuzzy set theory by mathematician Lotfi Zadeh, although the concept of infinite-valued logic had been studied since the 1920s by Łukasiewicz and Tarski [55]. Fuzzy logic is based on the idea that humans make decisions based on imprecise and non numerical information. Fuzzy models or sets are mathematical representations of vagueness and imprecise information, which is why they are referred to as "fuzzy" [57]. These models have the ability to recognize, represent, manipulate, interpret, and utilize data and information that lack certainty or are vague.

1.0.5 Optimization Reliability

The term "Reliability Optimization" first appeared in the late 1940s and was initially applied to communication and transportation systems. Early works in this field focused on analyzing specific performance aspects of operating systems [45]. One of the primary objectives of reliability engineers is to determine the most effective approach to increasing system reliability. System reliability is defined as the probability that a system will operate successfully for a specified period (i.e., mission time) under specific conditions [13]. As systems have become increasingly complex, the consequences of their unreliable performance have become more severe in terms of cost and effort [19]. There is a growing interest in assessing system reliability and improving the reliability of products and systems to address these challenges [76].

1.0.6 Fuzzy Optimization

In many cases, it is not possible or necessary to precisely quantify various performance criteria, decision variables, and system parameters. When such variables cannot be precisely specified, they are considered to be fuzzy or uncertain [78]. If these variables are uncertain, probability distributions may be used to quantify them. Conversely, if they are better described by qualitative adjectives such as hot or cold, dry or wet, clean or dirty, or high or low, fuzzy membership functions are used to quantify them [34]. Both probability distributions and fuzzy membership functions can be included in quantitative optimization models to represent uncertain or qualitative variables [90]. Reliability engineering has been a crucial aspect of technical system design and development since the 1960s. The primary objective of reliability engineers is to identify the most effective means of increasing system reliability [25]. Due to the diverse range of system resources, constraints, and options for reliability improvement, several optimization models have been constructed and analyzed in the literature [79]. Many reliability optimization models have been discussed in various works. For instance,

Misra (1971) explored the use of integer programming to address reliability optimization problems.Kuo and Prasad (2000) and Kuo et a [54]. (2001) introduced effective methods for solving reliability optimization models. In more recent times, Hao et al [40]. (2017) proposed a robust and efficient algorithm for non probabilistic reliability-based design optimization (NRBDO). Subsequently, Hao et al. (2019) established an accurate and efficient RBDO framework based on iso-geometric analysis (IGA) for complex engineering problems. In real-life systems, it is often not feasible to obtain precise data for reliability optimization models due to the inherent uncertainty in the decision maker's judgments. Coefficients and parameters may be imprecise due to the vagueness of nature, further complicating the problem. To address these issues in multi-objective problems, which can be classified as non stochastic imprecise models, a fuzzy approach can be used to handle the vague judgments of the decision maker [84].Fuzzy techniques have been used by researchers to solve multi-objective reliability optimization problems. In one of the earliest studies, Park (1987) applied fuzzy optimization techniques to solve the problem of reliability apportionment for a series system. Sakawa (1978) presented a multi-objective formulation of reliability allocation problem to maximize system reliability and minimize system cost using surrogate worth trade methods [92]. More recently, Grag (2013) utilized the particle swarm optimization method to solve a fuzzy multi-objective reliability optimization problem. Dancese et al. (2014) investigated the reliability and cost analysis of a series system model using fuzzy parametric geometric programming [59]. Wang et al. (2021) developed a fuzzy multi-objective reliability optimization model for a pump system and employed a hybrid algorithm to obtain the optimal solution.In 2017, a novel reliability-based optimization model and method for thermal structure design in a fuzzy environment were proposed [53]. There are various optimization techniques available to solve nonlinear optimization problems. Goal programming (GP) is a particularly effective method for solving certain types of nonlinear programming problems and has been widely applied to real-world problems involving multiple objectives. By using GP, the decision maker can obtain a satisfactory

solution and analyze aspiration levels [93]. Dhingra (1992) applied GP to solve a multi-objective reliability apportionment problem in a fuzzy environment. Gen and Ida (1993) discussed the use of large-scale 0-1 fuzzy goal programming to solve reliability optimization problems. Hwang and Lee (2009) developed an algorithm for nonlinear integer goal programming using the branch-and-bound method and demonstrated its application in solving reliability problems with single and multiple objectives [49].

1.0.7 Fuzzy Reliability

Reliability-Based Design Optimization (RBDO) is a methodology employed in engineering design to achieve an optimal balance between safety and manufacturing costs. Deterministic design optimization methods, in general, may lead to suboptimal design solutions, often at the limits of constraints, resulting in a final product with a high probability of failure. This is because of the uncertainties inherent in the manufacturing process, material properties, and other factors that depend on operating conditions. Furthermore, these uncertainties can cause significant variations in system performance and ultimately result in catastrophic failure.Hence, it is essential to consider such uncertainties during the design process. The RBDO method utilizes probability theory and statistics to model uncertainties and requires optimization to identify the optimal design that meets an acceptable probability of failure. The primary challenge of an RBDO problem lies in evaluating the failure probability of a design, which demands significant computational effort.

1.0.8 Optimization Algorithms

Up to this point, various optimization algorithms have been utilized to train deep learning models. These algorithms were instrumental in updating model parameters and minimizing the value of the loss function, as determined on the training set [31]. While some individuals may be satisfied with treating optimization as a black box device that minimizes objective functions in a straightforward setting, it is worth noting that

numerous implementations of such procedures exist, with names such as "SGD" and "Adam," both bounded and unboundedWhile basic optimization knowledge can be useful, a deeper understanding is required to excel in the field of deep learning. Optimization algorithms play a critical role in deep learning [23]. Training a complex deep learning model can be a time-consuming process, taking hours, days, or even weeks. Therefore, the efficiency of the optimization algorithm directly impacts the model's training time and overall performance. Moreover, comprehending the principles underlying various optimization algorithms and the role of their hyperparameters can enable us to fine-tune these parameters in a targeted manner, thereby improving the performance of deep learning models [92].

CHAPTER 2

BACKGROUND MATHEMATICAL(BASIC CONCEPT)

The basics of mathematics involve a set of fundamental concepts and principles that form the building blocks for more advanced mathematical topics. These concepts include arithmetic, algebra, geometry, and calculus, among others. Basic concepts are essential for success in mathematics and for many other disciplines that rely on quantitative analysis. In this context, understanding the background mathematical concepts is crucial to solving the complex problems and making informed decisions, in this chapter and discuss the mathematical concepts that will be needed later in this dissertation.

2.0.1 Definitions and Properties

This section covers the definitions, notation, and essential results of euclidean Space and provides the mathematical background necessary to solve the problem at hand. Introduce some fundamental concepts and facts in mathematical analysis. For further information on this topic, the interested can see [7, 9, 14, 45, 49, 54, 89].

Definition 2.0.1. [2] Let $f : \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$. Then the partial derivative of f at x with respect to x_i is defined by:

$$\frac{\partial f(x)}{\partial x_i} = lim_{t\to 0}\frac{f(x + te_i) - f(x)}{t}, \tag{2.1}$$

where e_i is ith unit vector. The gradient of f at x is defined by the column vector

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f(x)}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(x)}{\partial x_n} \end{bmatrix}.$$

The Hessian matrix is defined as the $n \times n$ symmetric matrix:

$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

The directional derivative of the function f at x in the direction d given by:

$$f'(x,d) = lim_{t \to 0^+} \frac{f(x+td) - f(x)}{t}. \tag{2.2}$$

The function f is differential at x if and only if the gradient $\nabla f(x)$ exists and satisfies $\langle \nabla f(x), d \rangle = f'(x,d), \forall d \in \mathbb{R}^n$. Moreover, the function f is differentiable over a subset S of $\mathbb{R}^n$ if it is differentiable at every $x \in S$. Otherwise, is non-differentiable, and f is continuously differentiable over S, if

$$lim_{d \to 0} \frac{f(x+d) - f(x) - \langle \nabla f(x), d \rangle}{\parallel d \parallel} = 0, \forall x \in S, \tag{2.3}$$

where $\parallel . \parallel$ is an arbitrary vector norm.

Definition 2.0.2. The Objective Function

The objective function represents the model primary goal, which is to be either minimized or maximized [1].

A point x is feasible if it is in $\mathbb{R}^n$ and satisfies the constraints of problem (2.4). The set F of all feasible points defines the feasible region of the optimization problem, i.e.,

$$S := \{x \in \mathbb{R}^n | h_j(x) = 0 \quad for \quad j = 1, 2, ..., L, \quad g_i(x) \leq 0 \quad for \quad i = 1, 2, ..., M\}.$$

.

Definition 2.0.3. Linear Programming Problems A linear programming (LP) problem is the most basic form of an optimization problem. In an LP problem, all variables are continuous, and both the objective function and constraints are linear in nature [38].

$$(GP) \begin{cases} \underset{x}{\text{Minimize}} & f(x) \\ \text{subject to} & h_j(x) = 0, \quad j = 1, 2, ..., L \\ & g_i(x) \leq 0, \quad i = 1, 2, ..., M \\ & x \in \mathbb{R}^n. \end{cases} \tag{2.4}$$

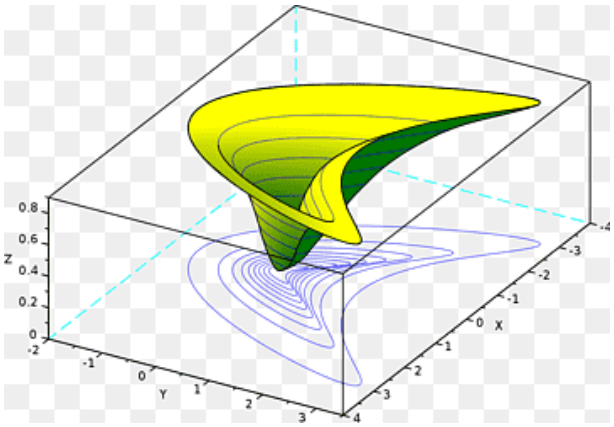

Figure 2.1: Feasible region under constraint

Definition 2.0.4. Nonlinear Programming Problems A nonlinear programming

(NLP) problem refers to an optimization problem where the objective function and/or some of the constraints are nonlinear in nature [38].

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & g_j(x) \leq 0 \quad \text{for} \quad j = 1, \ldots, p \\ & h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m \\ & x \in \mathtt{R^n} \end{cases} \tag{2.5}$$

Unconstrained non-linear programming refers to the process of identifying a vector x that represents a local minimum for the non-linear scalar function f(x). There are several common algorithms used in unconstrained non-linear programming, including Quasi-Newton, Nelder Mead, and Trust-region, as cited by Gopal et al. (2023). Constrained non-linear programming is the process of finding a vector x that minimizes a non-linear function f(x) while adhering to one or more constraints. Afolabi et al. (2023) mention several common algorithms used in constrained non-linear programming, including interior-point, sequential quadratic programming, and trust region reflective.

Definition 2.0.5. Karush-Kuhu-Tucker(KKT) condition for a non-linear programming problem [60].

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & g_j(x) \leq 0 \quad \text{for} \quad j = 1, \ldots, p \\ & h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m \\ & x \in \mathtt{R^n} \end{cases} \tag{2.6}$$

the Lagrangian $L(x, u, v) = f(x) + \sum_{i=1}^{m} v_i h_i(x) + \sum_{j=1}^{p} u_j g_j(x)$

$\Rightarrow L(x, u, v) = f(x) + v^\top + u^\top g(x)$

1. Gradient conditions

$$\frac{\partial(x^*, u^*, v^*)}{\partial x_i} = \frac{\partial f(x^*)}{\partial x_i} + \sum_{j=1}^{m} {v^*}_j \frac{\partial h_j(x^*)}{\partial x_i} + \sum_{j=1}^{p} {u^*}_j \frac{\partial g_j(x^*)}{\partial x_i} = 0 \quad \text{where} \quad i = 1, \ldots, n$$

2. Feasibility check

$$g_j(x) \leq 0 \quad \text{for} \quad j = 1, \ldots, p$$
$$h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m$$

The inequality and equality constrained have to be satisfied, the candidate optimal solution will satisfy the equality constrained as will as inequality constrained.

3. Switching Condition ${u^*}_j g_j = 0, j = 1, \ldots, p$ The non-negativity of Lagrange multiplier for inequality constraints ${u_j}^* \geq, j = 1, \ldots, p$

4. Regularity Check

Gradient of active constraints must be linearly independent

The points that satisfy all these KKT conditions are known as KKT point, it is likely candidate for optimum and the KKT condition " first order conditions for optimality"

Theorem 2.0.1. If $f(x)$ is convex and $g_i(x) \quad \forall i \in I$ are convex function then a feasible KKT point is optimal.

Theorem 2.0.2. KKT Necessary Optimality Conditions [21] If x^o is a local maximum , there is multipliers $\mu_i \geq 0 \quad \forall i \in I$ and $\lambda_j \geq 0 \quad \forall j \in J$ such that

$$\nabla f(x^o) - \sum_{i \in I} \mu_i \nabla g_i(x^o) - \sum_j \lambda_j \nabla h_j(x^o) = 0$$

Theorem 2.0.3. KKT Sufficient Optimality Conditions [21]: If $f(x)$ is concave $g_i(x) \quad \forall i \in I$ are convex functions and $h_j \quad \forall j \in J$ are affine (linear) then the feasible KKT point is optimal .

Example 2.0.1. Locate all of the point of the KKT point for the following problem . Are

these point local solution ? Are the global solution ?

$$\begin{aligned} \textit{minimize} \quad & x_1^2 + x_2^2 - 4x_1 - 4x_2 \\ \textit{subject} \quad & x_1^2 \leq x_2 \\ & x_1 + x_2 \leq 2 \\ & x_1, x_2 > 0 \end{aligned}$$

solution: First write the problem in the standard from required for the application of the KKT theory :

$$\begin{aligned} f_o(x_1, x_2) &= x_1^2 + x_2^2 - 4x_1 - 4x_2 \\ f_1(x_1, x_2) &= x_1^2 - x_2 \\ f_2(x_1, x_2) &= x_1 + x_2 - 2 \end{aligned}$$

$L((x_1, x_2), (\mu_1, \mu_2)) = x_1^2 + x_2^2 - 4x_1 - 4x_2 + \mu(x_1^2 - x_2) + \mu_2(x_1 + x_2 - 2)$ Let us now write the KKT condition

1. (Primal Feasibility) $x_1^2 \leq x_2$ and $x_1 + x_2 \leq 2$
2. (Dual Feasibility) $0 \leq \mu_1$ and $0 \leq \mu_2$
3. (Complementary) $\mu_1(x_1^2 - x_2) = 0$ and $\mu_2(x_1 + x_2 - 2) = 0$
4. (Stationarity of the Lagrangian)$0 = \nabla_x L((x_1, x_2), (\mu_1, \mu_2))$

$$\begin{aligned} 4 &= 2x_1 + 2\mu_1 x_1 + \mu_2 \\ 4 &= 2x_2 - \mu_1 + \mu_2 \end{aligned}$$

The global minimizer for the object function is $(x_1, x_2) = (2, 2)$ if this point are feasible , it will be the global solution and the multiples would both be zero .but it is not feasible ,

both constraint are active at the solution . in this case KKT pair$((x_1, x_2), (\mu_1, \mu_2))$ satisfy

$$\begin{aligned} x_2 &= x_1^2 \\ 2 &= x_1 + x_2 \\ 4 &= 2x_1 + 2\mu_1 x_1 + \mu_2 \\ 4 &= 2x_2 - \mu_1 + \mu_2 \end{aligned}$$

$x_1^2 + x_1 - 2 = (x_1 + 2)(x_1 - 1) = 0$ so $x_1 = -2$ or $x_1 = 1$. thus , either$(x_1, x_2) = (-2, 4)$ or $(x_1, x_2) = (1, 1)$since(x_1, x_2) is closer the global minimizer of the object f_0, to see (x_1, x_2) if it KKT point we plugging $(1, 1) = (x_1, x_2)$ we getting

$2 = 2\mu_1 + \mu_2$ and $2 = -\mu_1 + \mu_2$ then we getting $\mu_1 = 0$ and $\mu_2 = 2$ sine $\mu_1, \mu_2 \geq 0$ then $(1, 1)$ KKT point and by the convexity it is global solution .

Example 2.0.2. Use KKT to solve constraint optimization problems:

$$\begin{cases} \min_x & \mathtt{f(x)} = 4(x_1)^2 + 2(x_2)^2 \\ \text{subject to} & f_1(x) = 3x_1 + x_2 - 8 = 0 \\ & f_2(x) = 15 - 2x_1 - 4x_2 \geq 0 \\ & x_1, x_2 \geq 0 \end{cases} \tag{2.7}$$

solution: The Lagrangian

$$L(x,\lambda) = f(x) + \lambda_1 f_1(x) - \lambda_2 f_2(x) = 4(x_1)^2 + 2(x_2)^2 + \lambda_1(3x_1 + x_2 - 8) - \lambda_2(15 - 2x_1 - 4x_2)$$

$$\nabla L(x,\lambda) = \begin{pmatrix} 8x_1 + 3\lambda_1 + 2\lambda_2 \\ 4x_2 + \lambda_1 + 4\lambda_2 \\ 3x_1 + x_2 - 8 \\ -2x_1 - 4x_2 + 15 \end{pmatrix} = 0 \Rightarrow \begin{pmatrix} 8 & 0 & 3 & 2 \\ 0 & 4 & 1 & 4 \\ 3 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 8 \\ 15 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 1.7 \\ 2.9 \\ -3.12 \\ -2.12 \end{pmatrix}$$

2.0.2 Multiple objective function

Following the discovery of a new disease with novel symptoms in the Chinese city of Wuhan, it rapidly spread within the city and eventually across the world, becoming a pandemic and claiming numerous lives. In response, many researchers began studying the behavior of disease transmission, with mathematicians attempting to develop a mathematical formula to model the spread process. In most of these studies, society was classified into three types: 1- Not infected 2- An injured under treatment, and this type is divided into two types a- Survivor of disease b- Not a survivor (death) By following the recommended guidelines, such as city closures and social distancing, the number of infections has decreased. Recently, a vaccine for this disease was discovered in the United States of America. The optimization approach has become essential in solving the problem of efficiently transferring the vaccine from the producer to the consumer while simultaneously reducing the price of the dose and finding appropriate means of transportation.

Initially, the infected population grows exponentially but after implementing lockdowns in cities and adhering to health guidelines, the number of infected individuals begins to decrease in subsequent iterations. The COVID-19 virus can infect individuals who

may subsequently either die, infect others, or recover from the disease. This process is commonly modeled using an "SIR" model consisting of three types of individuals: susceptible (S), infected (I), and recovered (R).

2.0.3 Mathematics Concept

Let $S \subset R^p$ denote a hyperparallelepiped of $x = (x_1 \dots x_p)^\top \in R^p$ the vector $x \in S \subset R^p$ is called deduction vector, its entries are the deduction variables, and it is bounded by

$$x_i^{inf} \leq x_i \leq x_i^{sup}$$

for every $i = 1 \dots n$ the domain S is known as the deduction variables. Suppose that $f : S \subset R^p \to R^q$ is a multiple-objective function, where $q \geq 2$ and $\lambda(x) = (\lambda^1(x), \dots, \lambda^q(x))^\top$, and $\lambda^k : S \to R$ for $k = 1, \dots, q$ are the objectives. The vector x is referred to as a point or a solution candidate, as it lies within the set of elements in the decision variable space that potentially represent an optimal solution for the optimization problem.

Let $g : S \to R^p$ and $h : S \to R^q$ represent the set of equality and inequality constraints, respectively, that bound the subspace to be searched for the optimal solution, such that $g(x) \leq 0$ and $h(x) \leq 0$, where $g^i(x) \leq 0$ for $i = 1, \dots, p$ and $h^j(x) \leq 0$ for $j = 1, \dots, q$. The set of solutions that satisfy both equality and inequality constraints is denoted as the feasible set $\omega = \{x \in R^n | x^{inf} \leq x \leq x^{sup}, g(x) = 0, h(x) \leq 0\}$.

Let $V = Im(f|_\omega) = \{y = f(x) \subset R^m \mid \forall x \in \omega\}$, which represents the image of the function f over the feasible set.

$$\begin{cases} \text{minimize} & \lambda(x) = (\lambda^1(x) \dots \lambda^m(x))^\top \\ \text{subject to} & h^j(x) \leq 0, \quad j = \{1, \dots, p\}, \\ & g^i(x) = 0, \quad i = \{1, \dots, q\}, \\ & x^{inf} \leq x \leq x^{sup}. \end{cases} \tag{2.8}$$

In a multiple-objective problem, there is no single point that can minimize all the functions simultaneously. This implies that improving one objective function may come at the cost of delaying progress towards at least one other objective. To address this issue, we can use the concept of Pareto optimality.

The concept of dominance provides a way to compare candidate solutions in multiple-objective optimization problems. A solution x^1 is said to dominate another solution x^2, denoted by $x^1 \prec x^2$, if both of the following conditions are satisfied:

1. Solution x^1 is worst than x^2 s.t $\lambda^k(x^1) \leq \lambda^k(x^2) \forall k = 1, \ldots, q$
2. at least one object satisfy $k(x^1) < \lambda^k(x^2) \forall k = 1, \ldots, q$.

The Pareto-optimal which in turn are denoted by $x^8 \in \omega$, that represents the best poise related the minimization of all objective together. An objective vector $z^* \in V$ is called to be a Pareto-optimal if there is no other objective-vector $z \in V$ s.t $z_i \leq z_i^* \quad \forall k = 1, \ldots, m$ and $z_i < z_i^*$.

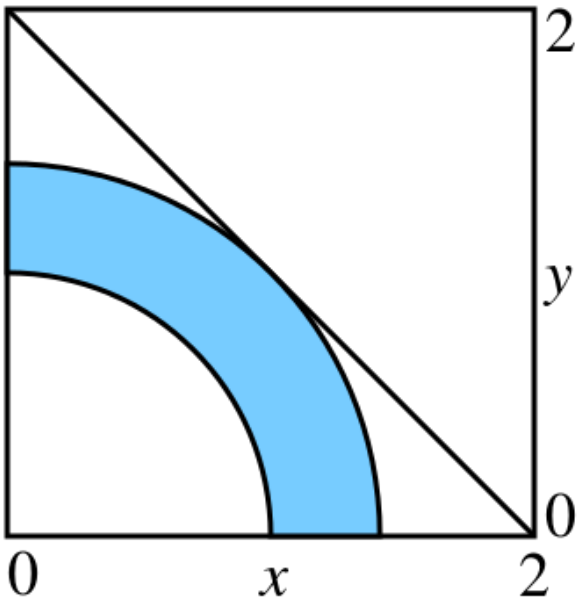

Figure 2.2: feasible region under constraint

Definition 2.0.6. **Convex** [12, 14, 32]

A set K is convex (see Figure 2.3) if the line segment between any two points in the set

K lies entirely within K. In other words, for any $x, y \in K$ with $x \neq y$ and any λ with $0 \leq \lambda \leq 1$, $\lambda x + (1 - \lambda)y \in K$.

Definition 2.0.7. a concave function is the negative of a convex function. A concave function is also synonymously called concave downwards, concave down, convex upwards, convex cap, or upper convex.

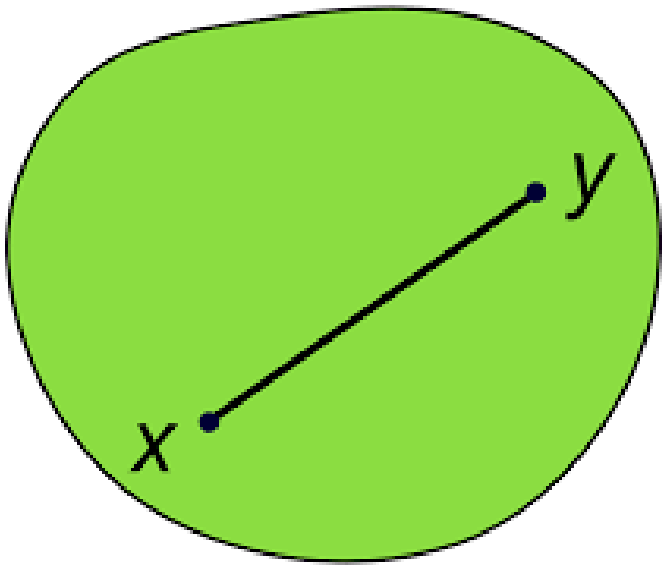

Figure 2.3: set K is convex

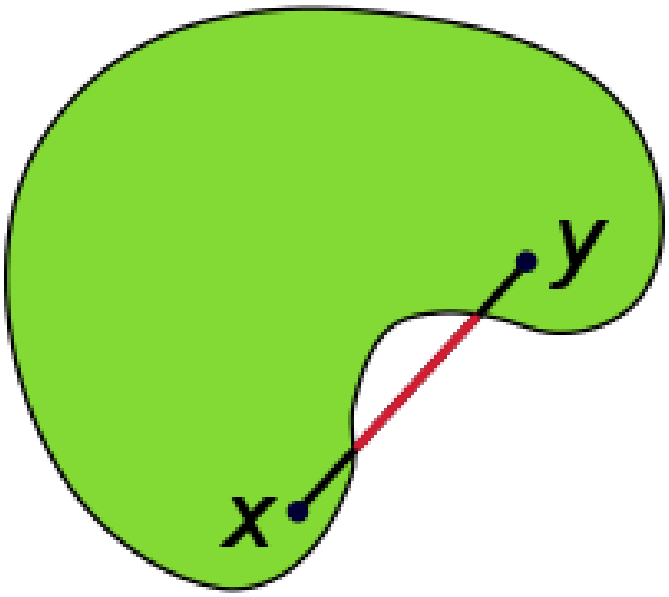

Figure 2.4: set K is concave

- A set is convex : if every point in the set can be seen by every point other point , along an unobstructed straight path between them .
- Every affine set is also convex.
- Convex combination point of the form $\lambda_1 x_1 + ... + \lambda_k x_k$, where $\lambda_1 + ... + \lambda_k$, $\lambda_i \geq 0$ $i = 1, ..., k$
- Set is convex iff it contains every convex combination of it is points .
- Convex hull of the set S denote by Conv

$$C = \{\lambda_1 x_1 + ... + \lambda_k x_k, \lambda_1 + ... + \lambda_k, x_i \in C, \lambda_i \geq 0, i = 1, ..., k, \lambda_1 + ... + \lambda_k = 1\}$$

- The **Conv C** is always convex . Is the smallest convex set that contains C , if H any convex set such that $H \subseteq C$ then **canv**$C \subseteq H$
- Affine function $f(x) = a^\intercal x + b$ (for any $a \in R^n, b \in R$) . The are also concave .

$\forall \lambda \in [0,1]$

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &= a^\intercal(\lambda x + (1-\lambda)y) + b \\ &= \lambda a^\intercal x + (1-\lambda)a^\intercal y + \lambda b + (1-\lambda)b \\ &= \lambda f(x) + (1-\lambda)f(y). \end{aligned}$$

- The quadratic function is convex iff $\mathbb{Q} \geq 0$ (positive semidefinite)

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &= f(y + \lambda(x-y)) \\ &= \frac{1}{2}(y + \lambda(x-y))^\intercal \mathbb{Q}(y + \lambda(x-y)) + c^\intercal(y + \lambda(x-y)) \\ &= \frac{1}{2}y^\intercal \mathbb{Q}y + \lambda(x-y)^\intercal \mathbb{Q}y + \frac{1}{2}\lambda^2(x-y)^\intercal + \mathbb{Q}(x-y) + \lambda c^\intercal x + (1-\lambda)c^\intercal y \\ &\leq \frac{1}{2}y^\intercal \mathbb{Q}y + \lambda(x-y)^\intercal \mathbb{Q}y + \frac{1}{2}\lambda(x-y)^\intercal \mathbb{Q}(x-y) + \lambda c^\intercal x + (1-\lambda)c^\intercal y \\ &= \frac{1}{2}\lambda x^\intercal \mathbb{Q}x + \frac{1}{2}(1-\lambda)y^\intercal \mathbb{Q}y + \lambda c^\intercal x + (1-\lambda)c^\intercal y \\ &= \lambda f(x) + (1-\lambda)f(y) \end{aligned}$$

Definition 2.0.8. [12] **Line**:suppose $x_1 \neq x_2$ are two points in $R^n \quad y = \lambda x_1 + (1-\lambda)x_2$ where $\lambda \in R$, form the line segment passing through x_1 and x_2 , the parameter $\lambda = 1$ corresponds to $y = x_1$, and the parametric value $\lambda = 0$ corresponds to $y = x_2$

Definition 2.0.9. [12] **Line segment**:suppose $x_1 \neq x_2$ are two points in $R^n \quad y = \lambda x_1 + (1-\lambda)x_2$ where $\lambda \in R, \quad 0 \leq \lambda \leq 1$, is the line segment between x_1 and x_2

Example 2.0.3. Is $f(x) = \ln(x)$ convex or concave ?

Solution: by using Python(version 6.1.7601), we are got $\ln(x)$ convex.

Code 1 :f(x)=In(x)

```
import␣numpy␣as␣np
import␣matplotlib.pyplot␣as␣plt
x=␣np.linspace(1,8,50)
#␣objective␣function
f=np.log(x)
plt.plot(x,␣f,␣color=(1,␣0,␣1))
plt.grid()
plt.xlabel('$x␣$␣␣␣␣nofl')
plt.ylabel('$\ln␣x␣$␣␣␣␣nofl')
#␣convexity/␣concavity
a=2
b=7
lamda␣=␣0.4
c=lamda*a␣+(1-lamda)*b
f_a=np.log(a)
f_b=np.log(b)
f_c=np.log(c)
f_c_hat=lamda*f_a␣+␣(1-lamda)*f_b
plt.plot([a,a],[0,f_a],␣color=(1,0,0)␣,␣marker='o'␣,␣label="$f(a)$")
plt.plot([b,b],[0,f_b],color=(0,0,1),␣marker='o',label=
"$f(\lambda␣a␣+␣(1-\lambda)b)$")
plt.plot([c,c],[0,f_c_hat],color=(1/2,2/3,3/4)␣,␣marker='o',label=
"$\lambda␣f(a)␣+␣(1-\lambda)␣f(b)$")
plt.plot([a,b],[f_a,f_b],␣color=(0,1,1))
plt.legend(loc=2)
```

```
plt.show()
```

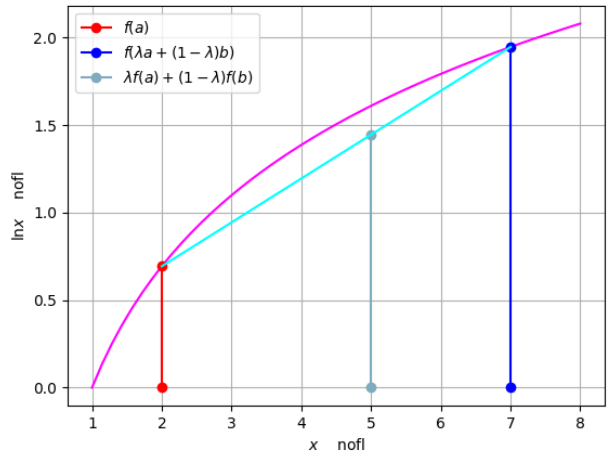

Figure 2.5: The figure shows that. The line segment joining any two points on the curve of the function lie below the curve. Then function is concave

Definition 2.0.10. [12] **Affine set**:A set $S \subseteq R^n$ is affine if the line through two distinct point in S is belong S. this means $x_1 \neq x_2$ are two points in S $y = \lambda x_1 + (1-\lambda)x_2$ where $\lambda \in R$ we have $y \in S$. We can say $x_1, x_2, ..., x_k$ are distinct point where $\lambda_1 + \lambda_2 + ... + \lambda_k = 1$ as an affine combination of the point is $\lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_k x_k$. Indeed every affine set can be expressed as the solution set of system of liner equations, and the converse is also true. the solution set of system of liner equation S=$\{x : Ax = b\}$ where$A \in R^{m\times n}$ and $b \in R^m$ is an affine st, $x_1, x_2 \in S$ such that$Ax_1 = b$, $Ax_2 = b$, for$\lambda \in R$;

$$\begin{aligned} A(\lambda x_1 + (1 - \lambda x_2) &= \lambda A x_1 + (1-\lambda) A x_1 \\ &= \lambda b + (1-\lambda) b \\ &= b \end{aligned}$$

This shown that the affine combination $\lambda x_1 + (1 - \lambda x_2) \in C$

Definition 2.0.11. [94] **Affine hull**: The set of all affine combination of points in some set $S \subseteq R^n$, and denoted**aff C** $= \{\lambda_1 x_1 + ... + \lambda_k x_k | x_1, ..., x_k \in C, \lambda_1 + ... + \lambda_k = 1\}$. The affine set hull is the smallest affine set that contains C, this means if S is any set with $C \subseteq S$, then **aff C** $\subseteq S$. **Affine dimension** of a set S as the dimension of it is **affine hull**

KKT Conditions for Quadratic Programming Problems

$$\begin{cases} \min_x & q(x) = c^\top x + \frac{1}{2} x^\top H x \\ \text{subject to} & A^\top x \leq b \\ & B^\top x = e \\ & x \geq 0 \end{cases} \tag{2.9}$$

where $c = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $e = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ we introduce slack variables S to convert inequalities qualities constraint $A^\top x + s = bb$, $S \geq 0$ and $S \in R^n$. Define Lagrangian : $L = c^\top x + \frac{1}{2} x^\top H x + u^\top (A^\top x + S - b) + v^\top (B^\top x - e) - w^\top x$ note that the non-negativity constraint $x \geq 0$ has been written as $-x \leq 0$

$$\begin{cases} \dfrac{\partial L}{\partial x} = c^t op + H x^\top + Au + Bv - w = 0 \\ \text{subject to} \\ A^\top x + S - b = 0 \\ B^\top x - e = 0 \\ x \geq 0 \end{cases} \tag{2.10}$$

Indeed this linear problem, solve deterioration conditions for $x, u.v, S \text{and} w$

complimentary slackness:

$$\begin{cases} u_i s_i = 0, & i = 1, \ldots, m; s_i, u_i \geq, i = 1, \ldots, m \\ w_i x_i = 0, & i = 1, \ldots, n; w_i \geq, i = 1, \ldots, n \end{cases} \tag{2.11}$$

There are n number of inequality constraint, for each of them we need slack variables, these leads to $u_i s_i = 0$ and n decision variables needs $w_i x_i = 0, i = 1, \ldots, n$, note that $s_i u_i \geq 0$ and $w_i \geq 0$. Because the slack variables and Lagrangian multipliers for the inequality constraints are non-negative, we can solve this conditions for x, u, v, s and w that will give me the KKT point and that maybe a candidate for optimal point. Lagrangian multiplier for equality constraint is free in sign. write it as $v = y - z$ with $y, z \geq 0$, we can define the following matrix and vectors:

$$N = \begin{bmatrix} H & A & -I_{(n)} & O_{(n\times m)} & B & -B \\ A^\top & O_{(n\times m)} & O_{(n\times m)} & I_{(n)} & O_{(m\times n)} & O_{(m\times n)} \\ B^\top & O_{(p\times m)} & O_{(p\times n)} & O_{(p\times m)} & O_{(p\times p)} & O_{(p\times p)} \end{bmatrix}_{(n+m+p)\times(2n+2w+2p)}$$

Now the KKT conditions can be written as : $NX = D$ complimentary slackness conditions reduces to

$$\begin{cases} X_i X_{n+m+i} = 0, & i = 1, \ldots, (n+m) \\ X_i \geq 0, & i = 1, \ldots, (2n+2m+2p) \end{cases} \tag{2.12}$$

where $X = \begin{bmatrix} x \\ u \\ w \\ s \\ y \\ z \end{bmatrix}_{(2n+2w+2p)}$, $D = \begin{bmatrix} -c \\ b \\ e \end{bmatrix}_{(n+m+p)}$

Note only complimentary slackness condition is non-linear in variable x_i, The solution to $NX = D$[which is set of liner equation complemtray slack is non-linear]. If that solution satisfies the complementary slackness condition then it becomes a solution to the KKT, point or solution to the original Quadratic programming problem. The complimentary slackness conditions is non-linear in variable $x_i, i = 1, \ldots, (n + m)$. Thus, the simplex procedure need to be changed to accommodate this, A procedures developed by $wolfe(1959)$ and later refined by $Hadley(1964)$ can be used and the simplex method will converge in a finite number of steps. Provided H is a positive definite matrix. This method is based on the phase I of the two-phase simplex method.

2.1 Reliability

is the probability that a component, device, or system will perform its task well for a specified period of time under certain conditions [6]. The definition can be formulated in a mathematical expression in terms of the random variable T, the time of system or device failure.

$$R(t) = Pr(T > t)$$

The definition of reliability includes all four aspects of the product, unlike quality, which only talks about specifications [56]. This means that reliability is quality over time, which is subject to the influence of time and operating environment. There is another difference between reliability and quality, which is that reliability devices can be manufactured using less reliability components, while it is very difficult to manufacture high-quality devices with lower quality components [30].

2.1.1 Reliability, Hazard and Failure Functions

Let T be a random variable representing the time of the system failure , and t be the interval of the operation of the system [63]. The reliability for the system at the time t is denoted by R (t) , where t ¡ T can be defined as the probability of the operation of that system within the interval [0 , t] according to the certain conditions . We can express that mathematically by $R(t) = Pr(T > t)$where Pr [95]. The term above is used in engineering studies. In the biological problems the term above is called survival function which is denoted by $\alpha(t)$, referees to the probability. Practically the probability of system failure can be defined as follows. $F(t) = Pr(T \leq t)$. We can realize that $F(t)$ is the distribution function of the random variable T , where $R(t) = 1–F(t)$, A mathematical expression of the failure probability density function is

$$F(t) = \lim_{\delta_t \to 0} \frac{Pr(t \leq T \leq +\delta_t)}{\delta_t}$$

[61].

The properties of the above function are :

- F (t) = 0 , t $\leq$ 0
- $\int_0^\infty f(t) = 1, t > 0$

where T is continuous, if T is discrete then the failure probability density function is $F(ti) = P(T = ti)$ where $t_1 < t_2 < \ldots$ The reliability function $R(t)$ can be expressed as :

$R(t) = \int_t^\infty f(t)dt$ which have the properties :

- $R(0) = 1$.
- $R(\infty) = 0$

Every system after a period of operation can be failed. The reliability function is continuous decreasing function. Now the unreliability function $F(t$, can be expressed mathematically by $F(t) = \int_t^0 f(t)dt$, when T is continuous, and if T is discrete then $F(t)$ is of the form. When T is continuous ,and if T is discrete then $F(t)$ is of the form $(t) = \sum_{k=1}^{n} P(t)$ [62].

2.1.2 The Hazard Function h (t)

The rate at which failures occur in a certain time interval $[t, t + \Delta t]$ is called the failure rate . It is defined as the probability that a failure per unit time occurs in the interval $[t, t + \Delta t]$, given that a failure has not occurred prior to t [62]. Thus the failure rate is: $\frac{R(t) - R(t + \Delta t)}{\Delta t \times R(t)}$. The hazard function h(t) is the instantaneous failure rate, $h(t)$ is defined by :

$$h(t) = \lim_{\Delta t \to 0} \frac{R(t) - R(t + \Delta t)}{\Delta t \times R(t)} = \frac{-\frac{d}{dt}R(t)}{R(t) = \frac{f(t)}{R(t)}}$$

The reliability function R(t) can written from the hazard function as follows : $h(t) = \frac{f(t)}{1 - F(t)}$, since $R(t) = 1 - F(t)$. By integrating of both sides we get: $\int_t^0 h(x)dx = \int_t^0 \frac{F(x)}{1 - F(x)}dx = -\log(1 - F(x))$

$\exp(-\int_t^0 h(x)dx) = 1 - F(x) = R(t)$

$R(t) = \exp(-\int_t^0 h(x)dx) = \exp(-h(t))$. Which represents the cumulative hazard function with the properties:

(a) $h(t) \geq 0$

(b) $\int_t^0 h(t) = \infty$

The hazard function curve is called (Bath – tube)

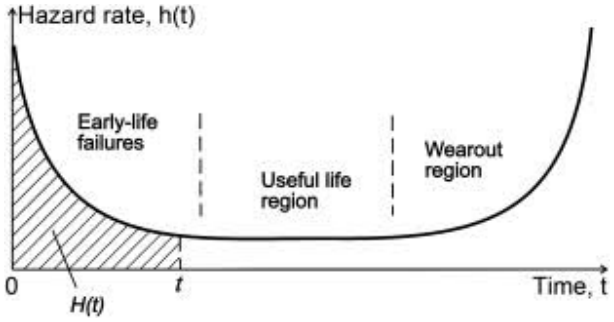

Figure 2.6: Bath – tube

2.1.3 Systems Reliability

Simple system certain types of system frequently arise in practice and serve to illustrate the idea of the structure function [52]. If it is not possible to solve a problem by using the simple structure of this section, it may be possible to solve the problem by viewing it as a combination of simple structure. The ways by which the system element can be combined are the following [36].

In series. In parallel. In parallel – series. In series – parallel. Mixed system. $K–out–of–n$ parallel configuration

2.1.4 Series Model

The series model can only work if all of its components are functioning [36]. This model is depending on both of them.

Figure 2.7: A Series Model

The system's reliability is provided by:$R_s = R_(A_1) \times R_(A_2) \times \ldots \times_(A_n)$

$$R_s = \prod_{(i=1)}^{(n)} R_{(A_i)}$$

Observation: A series configuration system works only if all the components are working . It fails if one or more components fail [62].

Example 2.1.1. Consider a model with four series-connected components, each with a set failure rate.

Figure 2.8: model series consist of 4 components

$R_S = R_1 \times R_2 \times R_3 \times R_4 = 0.2 \times 0.4 \times 0.7 \times 0.8 = 0.0448$

2.1.5 Parallel Model

If n components are linked in parallel so that the model works as long as at least one of them is in good working order, the model will fail only if all of the system's components fail at the same time [87].

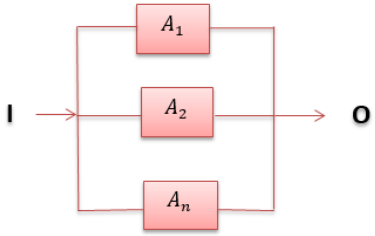

Figure 2.9: A Parallel Model

$$R_s = 1 - P(allfail) \tag{2.13}$$

$$= 1 - [P(A_1(fails) \times P(A_2(fails) \cdots \times P(A_n(fails) \tag{2.14}$$

$$= 1 - (1 - R_1) \times (1 - R_2) \times \ldots (1 - R_n) \tag{2.15}$$

$$R_s = 1 - \prod_{i=1}^{n}(1 - R_i) = (1 - R_i) \tag{2.16}$$

Observation

A parallel configuration system works if at least one of the components is working. It fails only if all of its components fail.

Example 2.1.2. Consider a model with three components connected in a parallel structure, each having a constraint failure rate, such that $R_1 = 0.1, R_2 = 0.3, R_3 = 0.6$ [29].

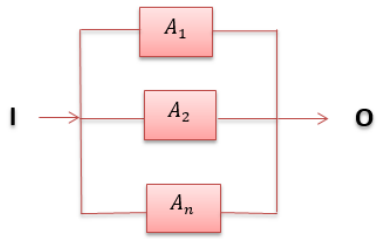

Figure 2.10: Parallel model consist of 3 Components

$$R_s = 1 - \prod_{i=1}^{3}(1 - R_i)$$

$$= 1 - (1 - R_1) \times (1 - R_2) \times (1 - R_3)$$

$$= 1 - (1 - 0.1) \times (1 - 0.3) \times (1 - 0.6)$$

$$= 1 - (0.9)(0.7)(0.4) = 0.748$$

2.1.6 Parallel-Series Model

provides the reliability of n components that are linked in parallel [4]. Where R signifies the reliability of a particular component $R_s = 1 - (1–R)^n$. If m such sets are linked in series, with each set consisting of parallel components as shown in the figure (2.11) (2.10), the system's reliability is defined by: $R_S = [1–(1–R)^n]^m$

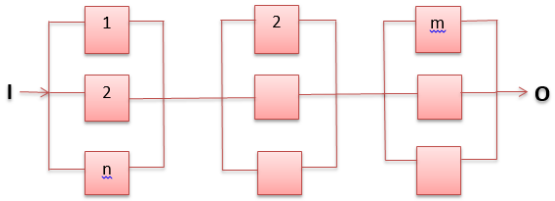

Figure 2.11: Parallel-Series Model

Example 2.1.3. To determine the model's reliability for the links depicted in Figure.

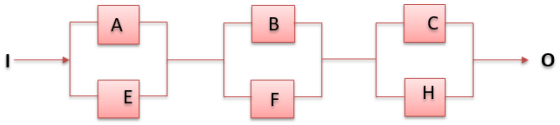

Figure 2.12: A Model of 6 Components

Assuming each component's reliability is 0.8. So we have $n = 2$ and $m = 3$ in this case.

$$R_S = [1–(1–R)^n]^m$$
$$R_S = \left[1–(1–0.8)^2\right]^3 = 0.88$$

2.1.7 Series–Parallel Model

The reliability of a model built of n components joined in parallel redundancy. Where R denotes the reliability of a single component $R_s = 1–(1–R)^n$. If the sets are placed in parallel, each with m components in series [33].

$$R_s = (1 - \prod_{i}^{m}(1 - R_i))^n$$

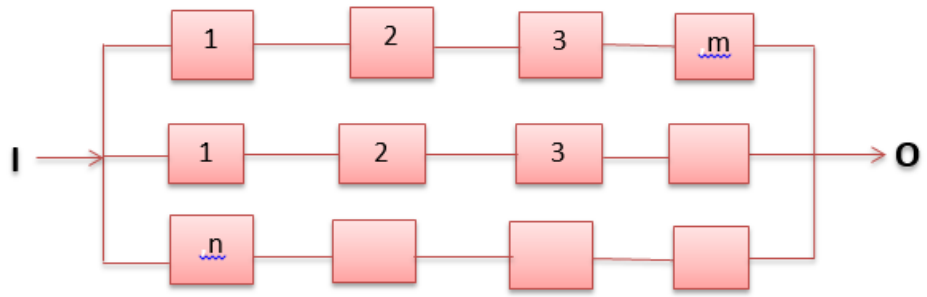

Figure 2.13: Series-Parallel Model

Example 2.1.4. Calculate the system's reliability for the connection shown in Figure below when $m = 3, n = 2$.

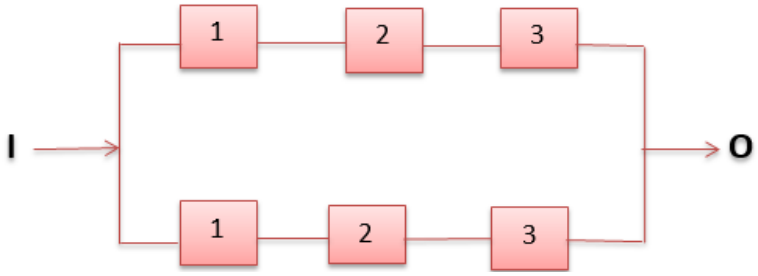

Figure 2.14: A Model Consist of 6 Components

Not that the reliability of $R_1 = 0.76, R_2 = 0.95, R_3 = 0.93$ respectively.

$$R_s = (1 - \prod_{i}^{m}(1 - R_i))^n$$

$$= [1 - (0.76)(0.95)(0.93)]^3 = 0.89$$

2.2 K-out-of-n Model

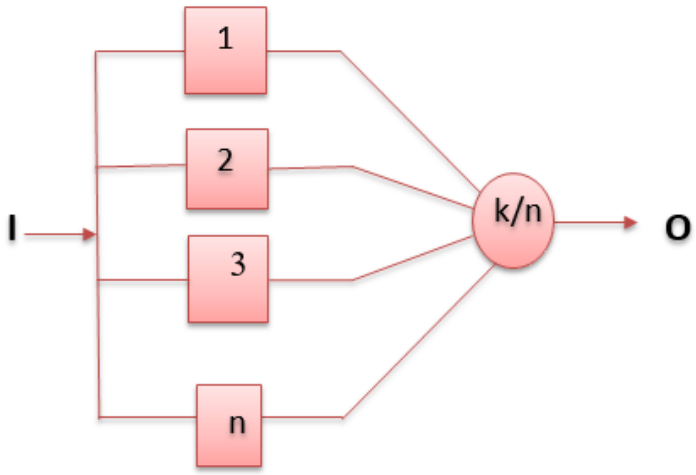

Figure 2.15: A Model $k–out–of–n$

This model is considered as a subset of parallel redundancy in that it must function on at least the first k components of the total n components [67]. Although this model is a special case of parallel redundancy, it is a general configuration in some cases because the number of units required to maintain the model's success is close to the total number of units in the model and the model's behavior is similar to a series model if the number of units requires equal number of units in the model [72]. Noted That

(a) if $k = 1$ then the system becomes parallel and

(b) if $k = n$ then the system becomes series

Example 2.2.1. Consider the $2–out–of–3$model depicted in figure blow:

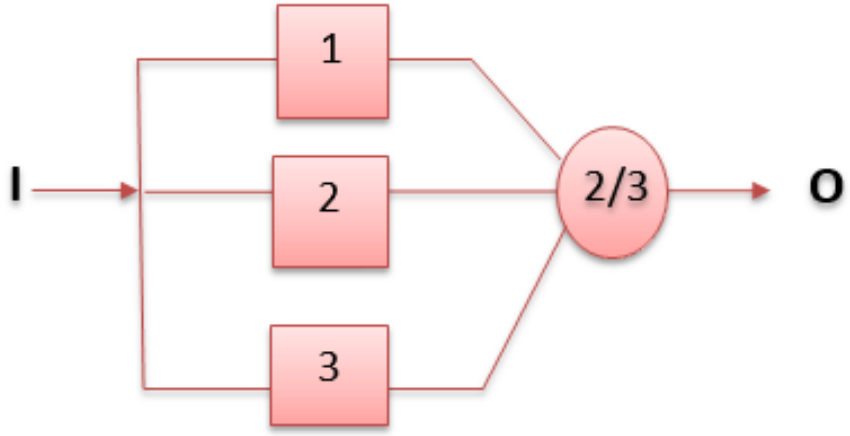

Figure 2.16: A Model 2–*out*–*of*–3 for Example

Not that the reliability of $R_1 = 0.75, R_2 = 0.88, R_3 = 0.91$ respectively

Solution:

Because it must work on at least 2- out- of -3, only one is allowed to fail. The following activities can be performed to ensure model success:

(a) Everyone is working.

(b) The 1 fails, but the 2 and 3 work.

(c) The 2 fails, while the 1 and 3 work.

(d) The 3 fails, but the 1 and 2 work. We can calculate model reliability as follows:

$$\begin{aligned} R_s &= R_1R_2R_3 + (1-R_1)R_2R_3 + R_1(1-R_2)R_3 + R_1R_2(1-R_3) \\ &= R_1R_2R_3 + R_2R_3 - R_1R_2R_3 + R_1R_3 - R_1R_2R_3 + R_1R_2 - R_1R_2R_3 \\ &= R_1R_2 + R_2R_3 + R_1R_3 - 2R_1R_2R_3 \\ &= (0.66) + (0.8008) + (0.6825) - (1.2012) = 0.94 \end{aligned}$$

2.3 Mixed model

This model is a mix of a parallel and a series system, and it is merely divided into series and parallel to determine the reliability of each sub model and complete to determine the system's reliability [3].

Example 2.3.1. Consider the system represented, which consists of four linked components [72].

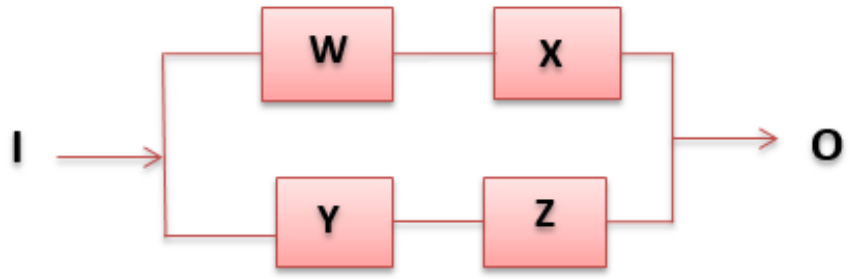

Figure 2.17: A Model Consist of 4 Components

Where $R_W = 0.1, R_X = 0.4, R_Y = 0.3, R_Z = 0.8$ Then:

$R_{WX} = R_W \times R_x = (0.1)(0.4) = 0.04$

$R_S = 1 - (1 - R_{wx})(1 - R_{yz})$

$= 1 - (1 - 0.04)(1 - 0.24)$

$1 - (0.96)(0.76) = 0.27$

2.4 Complex Model

A model that cannot be directly classified into the preceding cases of elementary structures is called a model with complex structure. For the study of systems with complex structure, the concepts of minimal paths and cuts must be introduced. The model in Figure (2.12) is a model with complex structure. This model cannot be directly classified into modules with traditional structures . If the input-output of this

model are located at the extremities of component E, it will then involve a model with elementary structure [3].

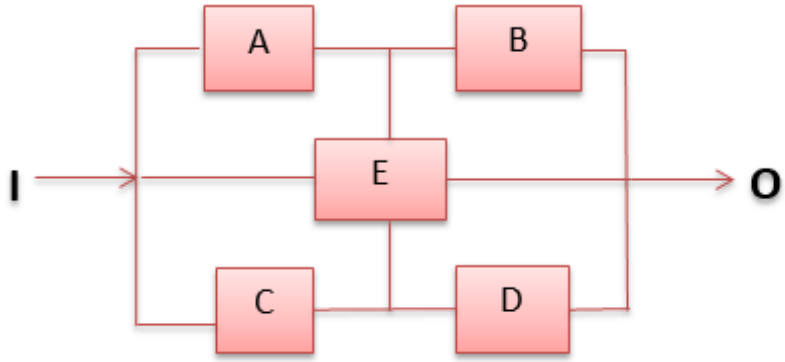

Figure 2.18: A Bridge Model

There several methods for calculating the reliability of mixed and complex systems are exist.

2.4.1 Some Methods Applied to Calculate the Reliability for Different Models

2.4.2 Path-Tracing Method

This method considers any path from a source to a sink. Because of the system's performance, there must be at least one path from one end of the reliability block diagram to the other long as at least one route exists from start to finish, the device is operating, and the structure function will give by the following relationship:

$$R_S = 1 - \prod_{j=1}^{n}(1 - p_j)$$

Where n is the number of minimal path of the model [18, 96].

Example 2.4.1. A model is made up of 9 components that are linked together as indicated to determine reliability.

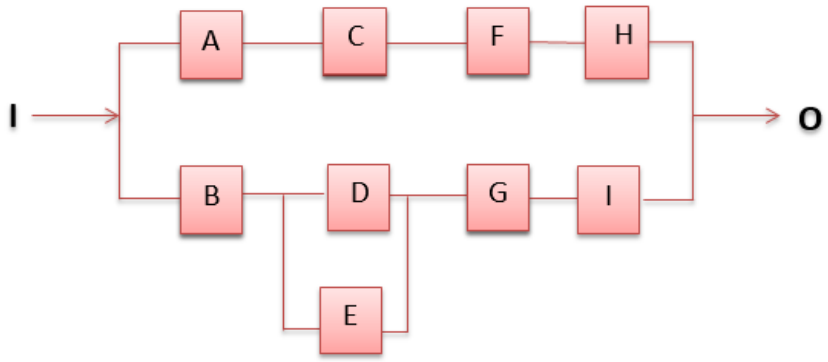

Figure 2.19: A Model Consist of 9 Components

Network representation of the complicated model described above.

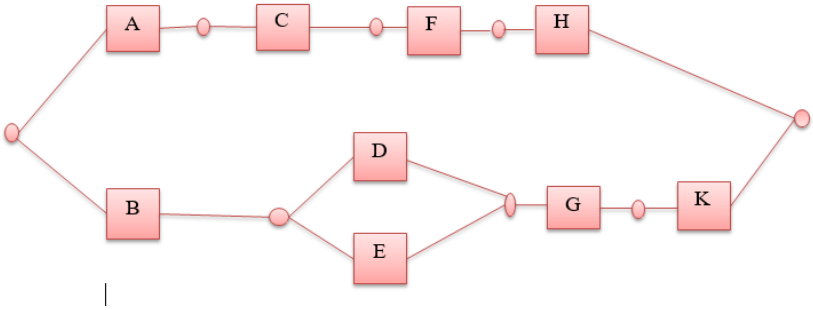

Figure 2.20: Network Representation

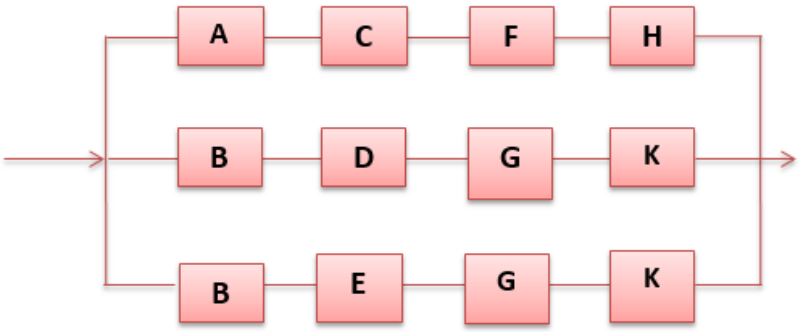

Figure 2.21: Representation

The path sets are $P_1 = \{A, C, F, H\}$, $P_2 = \{B, D, G, K\}$, $P_3 = \{B, E, G, K\}$

$$R_S = 1 - \prod_{j=1}^{3}(1 - p_j)$$

Therefore the model reliability is : $R_s = 1 - \prod_j = (1 - P_j)$

$= 1 - (1 - P_1)(1 - P_2)(1 - P_3)$

$= P_1 + P_2 + P_3 - P_1P_2 - P_1P_3 - P_2P_3 + P_1P_2P_3$

$R_AR_CR_FR_H + R_BR_DR_GR_K + R_BR_ER_GR_K - R_AR_CR_FR_HR_BR_DR_GR_K -$

$R_AR_CR_HR_BR_ER_GR_K - R_BR_DR_GR_K + R_AR_CR_FR_HR_BR_DR_ER_GR_K$

If $R_i = R$

(That is, if the probabilities of all the components are statistically independent and identical) Then we have $R_S = 3R^4 - R^5 - 2R^8 + R^9$, If $R = 0.8$. Then the reliability of the model is $R_S = 0.70$

2.4.3 Reduction to Series Elements

In this method each parallel path is replaced with an equivalent single path , eventually reducing the supplied model to one that only consists of series elements [18].

Example 2.4.2. Consider the system, which consists of 7 linked components, as illustrated in figure below.

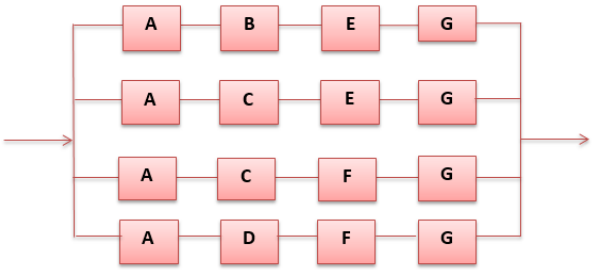

Figure 2.22: Representation for Figure

Solution

Let O, P, Q and S path where

$O = a, b, e, g$

$P = a, c, e, g$

$Q = a, c, f, g$

$S = a, d, f, g$

The reliability for each path

$$R_O = R_a \times R_a \times R_e \times R_g \tag{2.17}$$

$$R_P = R_a \times R_c \times R_e \times R_g \tag{2.18}$$

$$R_Q = R_a \times R_c \times R_f \times R_g \tag{2.19}$$

$$R_S = R_a \times R_d \times R_f \times R_g \tag{2.20}$$

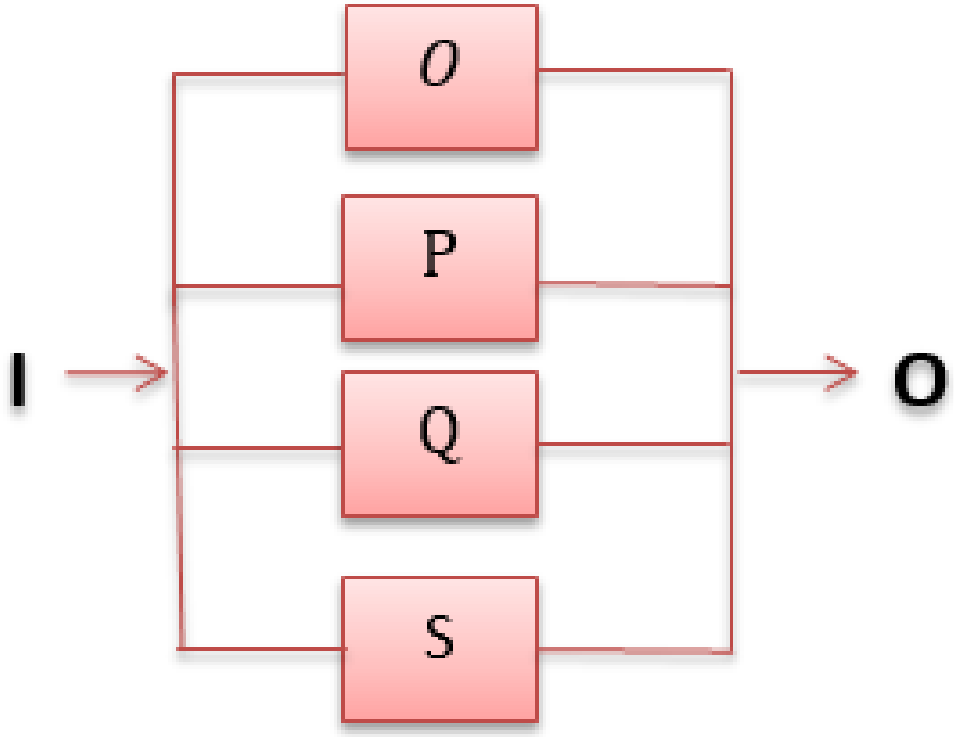

Figure 2.23: Reduction Representation

Therefore the model reliability is : We reduce O, P, Q and S to J

$$
\begin{aligned}
R_j &= 1 - [(1 - R_O)(1 - R_P)(1 - R_Q)(1 - R_S) \\
&= R_O + R_P + R_Q + R_S - R_Q R_S - R_P R_S - R_O R_S - R_P R_Q - R_O R_Q \\
&- R_O R_P + R_P R_Q R_S + R_O R_Q R_S + R_O R_P R_S + R_O R_P R_Q \\
&- R_O R_Q R_S
\end{aligned}
$$

2.4.4 Composite Method

This method is similar to the path tracing method but it does not list the entire parallel paths. The probability factor is derived in a comprehensive manner starting from the input end. When constructing the probability factor expressions, the series components are allocated "and " and the parallel elements are assigned "or " [38].

Example 2.4.3. Consider the following system, with reliabilities$R1, R2$ and $R3$.

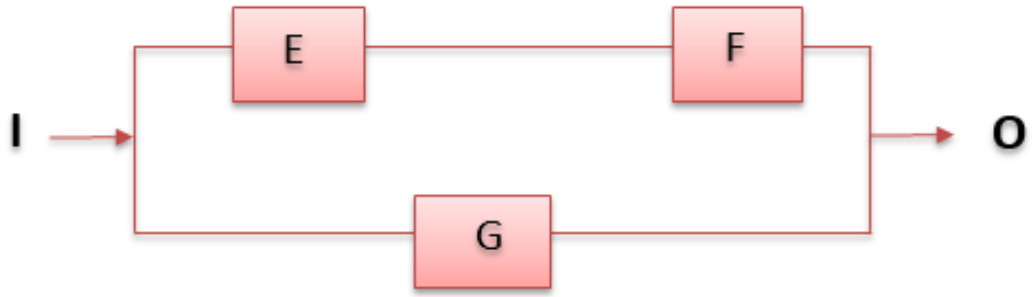

Figure 2.24: A Model Consist of 3 Components

Suppose that the components are independent. Hence $P(S) = P\,[(E(and)F)orG]$

$$
\begin{aligned}
P(S) &= P(E)P(F) + P(G) - P(E)P(F)P(G) \\
R_S(t) &= R_E R_F + R_G - R_E R_F R_G \\
&\text{If} R_E = 0.93, R_F = 0.85 \text{and} R_G = 0.91
\end{aligned}
$$

Then $R_S(t) = (0.93)(0.85) + (0.91) - (0.93)(0.85)(0.91) =$

$= (0.7905) + (0.91) - (0.719355) = 0.98$

2.5 Fuzzy set

Let X be a collection of objects denoted by the generic symbol x. A fuzzy set $\bar{A}$ in X is defined as a collection of ordered pairs:

$$\bar{A} = \{(x, f_A(x)) : x \in X\}$$

The function $f_A(x)$, called the membership function, grade of membership, degree of compatibility, or degree of truth of x in $\bar{A}$, assigns a real number $f_A(x)$ in the interval [0,1] to each element $x \in X$. The value of $f_A(x)$ at x indicates the degree to which x belongs to A.

- $\bar{o} = [(x, 0), \forall x \in X]$is the null fuzzy set.
- $\bar{1} = [(x, 1), \forall x \in X]$ is the absolute fuzzy set.

Definition 2.5.1. Let $\bar{A} = \{(x, f_A(x)) : x \in X\}$ and $\bar{B} = \{(x, f_B(x)) : x \in X\}$ be two fuzzy sets in X. Their union, intersection, and complement are also sets with the membership functions defined as follows:
The union of $\bar{A}$ and $\bar{B}$, denoted by $\bar{A} \cup \bar{B}$, is a fuzzy set defined as:

$$(\bar{A} \cup \bar{B}) = \{(x, \max(f_A(x), f_B(x))) : x \in X\}$$

The intersection of $\bar{A}$ and $\bar{B}$, denoted by $\bar{A} \cap \bar{B}$, is a fuzzy set defined as:

$$(\bar{A} \cap \bar{B}) = \{(x, \min(f_A(x), f_B(x))) : x \in X\}$$

The complement of $\bar{A}$, denoted by $\bar{A}^c$, is a fuzzy set defined as:

$$\bar{A}^c = \{(x, 1 - f_A(x)) : x \in X\}$$

:

- $f_{A\cup B} = (\max\, f_A(x), f_B(x)), \forall x \in X$
- $f_{A\cap B} = (\min\, f_A(x), f_B(x)), \forall x \in X$
- $[f_{\bar{A}}(x]^c = 1 - f_{\bar{A}}, \forall x \in X$

Theorem 2.5.1. Let $\bar{A}, \bar{B}$and $\bar{C}$, be fuzzy sets in a set X, the fallowing statement hold:

(a) (Comparatively):$\bar{A} \vee \bar{B} = \bar{B} \vee \bar{A}$; $\bar{A} \wedge \bar{B} = \bar{b} \wedge \bar{A}$

(b) (Assertively): $(\bar{A} \vee \bar{B}) \vee \bar{C} = \bar{A} \vee (\bar{B} \vee \bar{C})$
And $(\bar{A} \wedge \bar{B}) \wedge \bar{C} = \bar{A} \wedge (\bar{B} \wedge \bar{C})$

(c) (Idem-potency):$\bar{A} \vee \bar{A} = \bar{A}$

(d) (Distributive): $(\bar{A} \vee \bar{B}) \wedge \bar{C} = (\bar{A} \vee \bar{B}) \wedge (\bar{A} \wedge \bar{B})$
$(\bar{A} \wedge \bar{B}) \vee \bar{C} = (\bar{A} \wedge \bar{B}) \vee (\bar{A} \wedge \bar{B})$

(e) (Absorption): $\bar{A} \vee o_x = \bar{A}$; $\bar{A} \vee 1_x = \bar{A}$

(f) (De Morgan's): $(\bar{A} \vee \bar{B})^c = \bar{A}^c \wedge \bar{B}^c$
$(\bar{A} \wedge \bar{B})^c = \bar{A}^c \vee \bar{B}^c$

(g) (Involution):$(\bar{A}^c)^c = \bar{A}$

(h) (Equivalence formula):$(\bar{A}^c \vee \bar{B}) \wedge (\bar{A} \vee \bar{B}^c) = (\bar{A}^c \wedge \bar{B}^c) \vee (\bar{A} \wedge \bar{B})$

(i) (Symmetrical difference formula): $(\bar{A}^c \wedge \bar{B}) \vee (\bar{A} \vee \bar{B}^c) = (\bar{A}^c \vee \bar{B}^c) \wedge (\bar{A} \vee \bar{B})$

(j) (Difference): $(\bar{A}\ \bar{B}) = \bar{A} \wedge \bar{B}^C$

(a) Let $\bar{A}$ and $\bar{B}$ be fuzzy sets in a set , then $\bar{A} \wedge \bar{B} \leq \bar{A}$ and $\bar{A} \wedge \bar{B} \leq \bar{B}$

(b) The excluded-middle law is no longer true:$\bar{A} \wedge \bar{A}^c \neq o + x$, $\bar{A} \vee \bar{A}^c \neq 1_x$

(c) For any family $A = \{\bar{A}_j : j \in J\}$of fuzzy sets in X, the union $\vee_j \bar{A}_J$ and the intersection $\wedge_j \bar{A}_J$are defined by:

$\vee_j \bar{A}_J = \{x, \sup \bar{A}_j : j \in J\}, \forall x \in X$

$\wedge_j \bar{A}_J = \{x, \inf \bar{A}_j : j \in J\}, \forall x \in X$

Definition 2.5.2. Let U be the universal of discourse. A vague set $\bar{v}$ over U is characterized by a truth membership function$t_{\bar{v}}$, $t_{\bar{v}} : U \to [0, 1]$ and a false membership function$f_{\bar{v}}$, $f_{\bar{v}} : U \to [0, 1]$. If generic element of U is denoted by x_i then the lower bound on the membership grade of x_i derived from evidence for x_i is denoted by $t_{\bar{v}}(x_i)$and the lower bound on the negation of x_i is denoted by $f_{\bar{v}}$. $t_{\bar{v}(x_i)}$ and $f_{\bar{v}(x_i)}$ both associate a real number in $[0, 1]$ with each point x_i in X , where $t_{\bar{v}(x_i)} + f_{\bar{v}(x_i)} \leq 1$

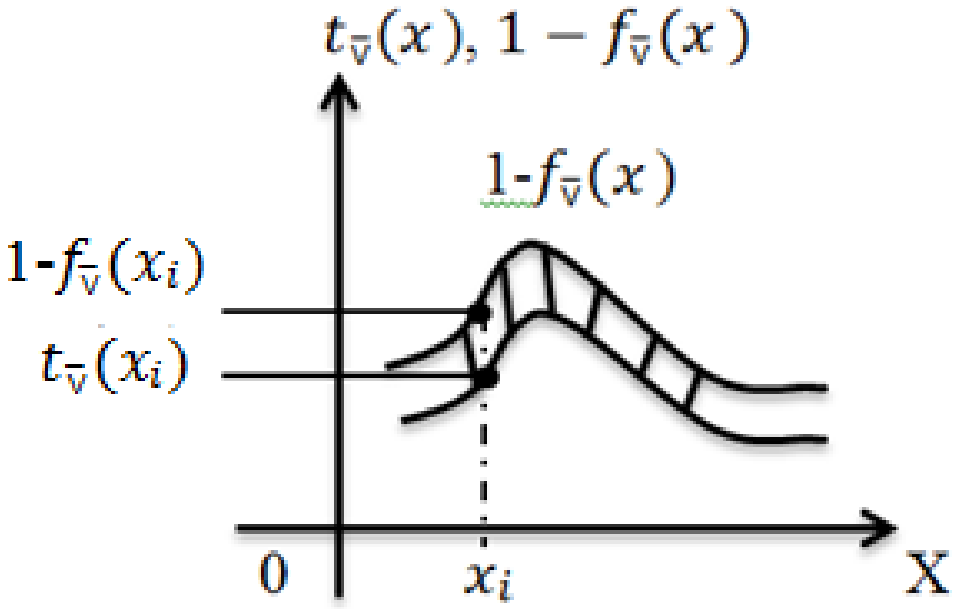

Figure 2.25: Avague Set the universal of discourse

Applications of Fuzzy

Fuzzy Logic has several important applications, such as speech recognition and facial characteristic recognition. In the aerospace industry, Fuzzy Logic is used to control the altitude of aircraft and satellites, as well as to regulate the flow and mixture of ice during

anti-icing and deicing operations of flights [102]. Additionally, Fuzzy Logic is used to regulate airflow in air conditioning rooms. In the automotive industry, Fuzzy Logic is used to control traffic and manage highway systems for vehicle transportation. It is also used to efficiently manage automatic transmissions and shift schedules [65]. Fuzzy Logic can be used to train systems that control the speed of vehicles. In the business sector, entrepreneurs can use Fuzzy Logic to aid in decision-making processes, especially in evaluating employees or workers when the company is large and manual management is difficult [41]. Fuzzy Logic also plays a vital role in target recognition, whether it be underwater or above the ground in the defense sector. Infrared images that are difficult for the human eye to capture can be recognized easily using Fuzzy Logic, making it a valuable support mechanism for the navy and other defense mechanisms. Fuzzy Logic is also known for its use in setting models for NATO decision-making processes. The hypervelocity of the interceptor can be easily controlled using Fuzzy Logic, and video cameras can be set to automatically capture scenes with their automatic exposure [47]. Fuzzy Logic plays a significant role in various applications. For example, it helps cameras to automatically adjust the exposure in different lighting conditions. Fuzzy Logic can also be used to control humidity levels in a well-maintained but humid room. In washing machines, Fuzzy Logic enables housewives to set the time for washing clothes, making the process more efficient. Fuzzy Logic can be used in stock market predictions due to its ability to perceive data in a way similar to the human brain. It is also used in ATMs and other government service agencies where banknotes need to be transferred smoothly to ensure a smooth flow of the process [88]. Fuzzy Logic has many applications, including the transfer and counting of banknotes for various enterprises. With efficient management of Fuzzy Logic, it can also be used to manage funds for daily needs, vacations, and kids' schooling. Fuzzy Logic can aid in the working of microwave ovens by helping set the time and increase the speed of vacuum cleaner motors based on the amount of dust. Fuzzy Logic can be used in rotary cement kilns to control heat exchange and in wastewater treatment plants to control aeration for treatment by integrating incoming oxygen and

oxidation-reduction information [10]. Fuzzy Logic can also be used to measure parameters affecting water quality and treat water accordingly for purification purposes. The industry has various quality checks available to perform qualitative analysis of products, with Fuzzy Logic Controllers (FLCs) used to measure these parameters quantitatively. In the structural design field, Fuzzy Logic can be used to efficiently evaluate and satisfy various design constraints. Water purification plants also use FLCs to control aeration and oxygen levels, while the cheese industry optimizes cheese and milk production with Fuzzy Logic. FLCs are also used in ship control, allowing for measurement of different quantities and control of the autopilot mechanism. This is particularly useful for managing ships' routes, which can be challenging to manage manually [44]. Fuzzy Logic can also aid in selecting efficient routes for ships and steering them effectively. The Fuzzy Logic mechanism can easily handle controls for underwater vehicles. In the medical field, FLC can be used as a diagnostic support system, and it is also used to control arterial pressure while administering anesthesia, which can be fatal in rare cases. FLC can handle various control mechanisms in anesthesia. FLC can be used to draw various neuro pathological models that are helpful in treating Alzheimer's patients. Fuzzy diagnosis is also helpful in the radiology field for disease diagnosis, as well as in diabetes and cancer treatment. Security appliances are also made using FLC to automate systems and raise alarms in the event of something significant happening. In railways, automatic braking and stopping of trains are done with the help of FLC. Underground train operation is managed using FLC. Train schedules are controlled with the help of FLC. The information is made precise and Fuzzy Logic is similar in human decision-making methods. The problems in the world are made simple with the help of this mechanism. It helps to know the fuzziness in decisions.

2.5.1 Applications of Fuzzy Reliability Optimization Problem

Fuzzy optimization problems have been developed to tackle real-world problems encountered in various fields. In agriculture, fuzzy optimization is used for analyzing water use, feed mix, regional resource allocation, and water supply planning [35]. In

banking and finance, fuzzy optimization is used for capital asset pricing models and project investment. In environmental management, fuzzy optimization is used for air pollution regulation problems and energy emission models [52]. In manufacturing and production, fuzzy optimization is used for aggregate production planning problems, machine optimization, magnetic tape production, optimal allocation of metal production, system design, crude oil manufacturing, production mix selection, and production scheduling [36]. In personnel management and transportation, fuzzy optimization is used for coordinating personnel demand and available personnel structure, transportation problems, and truck fleets.

Theorem 2.5.2. If $f(x)$ is convex and $g_i(x) \quad \forall i \in I$ are convex function then a feasible KKT point is optimal.

Definition 2.5.3. **Feasible Region** [14, 16]
The feasible region or the feasible set is the set of all possible values of the problem that satisfies all the constraints of the problem. Otherwise, the region is said to be infeasible if no solution exists which satisfies all the constraints. The constraints could be equalities and inequalities. The feasible points is a substitution in the objective point.

Definition 2.5.4. **Critical Point** [14]
A critical point is a point within the domain of a function where the derivative is either equal to zero or undefined (i.e., the function is not differentiable)..

Definition 2.5.5. [66] For any feasible set x, the active set $A(x)$ of the inequality constraints denoted by E:

$$A(x) = E \cup \{j \in L | h_j(x) = 0\}.$$

Mathematical Formulation

$$\begin{cases} \min(\max) & \mathtt{Q(x)} \\ \text{subject to} & x \in \mathtt{X}. \end{cases} \tag{2.21}$$

where $\mathtt{x} \in \mathtt{R}^{\mathtt{n}}$parameters and $\mathtt{Q}$ is objective function we requirement minimize or maximize.we want find x^* such that $Q(x^*) \leq Q(x)$ for all $\mathtt{x} \in \mathtt{X}$, Always can say maximize$Q(x) \equiv -minimize Q(x)$ the feasible set X of linear programming, And the regain boundaries illustrate the constraints. The general optimization problem in the form

$$\begin{cases} \text{minimize} & f^0(x) \\ \text{subject to} & f^i(x) \leq 0, \quad i = \{1, \ldots, I\}, \\ & g^i(x) = 0, \quad i = \{1, \ldots, J\}, \\ & x \in \Omega. \end{cases} \tag{2.22}$$

Where I and J are sets of indicator for equality and inequality constraints. objective function and equality and inequality constraints are linear. Some problem from an LP

(a) Finding the shortest way between the two cites.

(b) Location problem Finding a suitable location between two cities to build an airport, so that the distance between the airport and the two cities is the minimal.

(c) Transportation Problem. Find the best way to satisfy the requirement of demand points using the capacities of supple point.

Definition 2.5.6. **global minimum and local minimum** [5, 7, 14, 28]: A point x^* is called a local minimize if $f(x^*) \leq f(x)$ for all $x \in \mathcal{B}(x^*, \delta)$ where $\delta > 0$, A point x^* is called a strict local minimum if $f(x^*) < f(x)$ for all ,$x \in \mathcal{B}(x^*, \delta)$ where $\delta > 0$ with $x \neq x^*$ A point x^* is called a global minimize if $f(x^*) \leq f(x)$ for all $x \in R$, A point x^* is called a strict global minimum if $f(x^*) < f(x)$ for all $x \in R^n$, with $x \neq x^*$.

2.6 Numerical Optimization Theory

Numerical optimization problems are a fundamental tool in quantitative decision-making processes. Suppose a system can be described by a set of mathematical equations that adequately encompasses the impact of decision variables in objectives and constraints. In

that case, one might search for the values of those decision variables that produce the best possible outcome using optimization algorithms [13].

The presence or not of nonlinear functions to describe how the decision variables influence the objectives and constraints is used to divide optimization problems into two major categories: Linear and Nonlinear Programming [20]. Management sciences and operations research make extensive use of linear models, whereas nonlinear programming problems tend to arise naturally in the physical sciences and engineering (Nocedal Wright, 2006). Throughout this article, the focus will be on linear models [55]. In some problems, it makes sense that decision variables assume only integer values. For instance, in a personnel scheduling problem, it does not make sense to assign 2.57 employees to a shift; or in a vehicle routing problem, one can not choose to take half a route to go from a to b. These are denoted as either integer or mixed-integer problems [57].

Example: Branch and Bound Search with Examples and Implementation in Python The following code uses the A* algorithm to solve the 8-puzzle problem. It takes an initial state of the puzzle and a final state and finds the shortest path to reach the final state.

```
import copy
from heapq import heappush, heappop
n = 3
row =&&[1, 0, -1, 0]
col =&&[0, -1, 0, 1]

class priorityQueue:
    def __init__(self):
        self.heap = []
    def push(self, k):
        heappush(self.heap, k)
    def pop(self):
        return heappop(self.heap)
```

```
    def empty(self):
        if not self.heap:
            return True
        else:
            return False
class node:
    def __init__(self, parent, mat, empty_tile_pos, cost, level):
        self.parent=&&parent
        self.mat    =     mat
        self.empty_tile_pos=&&empty_tile_pos
        self.cost=&&cost
        self.level=&&level
    def __lt__(self, nxt):
        return self.cost < nxt.cost
def calculateCost(mat, final) -> int:
    count   =    0
    for i in range(n):
        for j in range(n):
            if ((mat[i][j]) and (mat[i][j]!=&&final[i][j])):
                count +=&&1
    return count
def newNode(mat, empty_tile_pos, new_empty_tile_pos, level, parent, final) -> node:
    new_mat=&&copy.deepcopy(mat)
    x1=&&empty_tile_pos[0]
    y1= && empty_tile_pos[1]
    x2 =&& new_empty_tile_pos[0]
    y2 = && new_empty_tile_pos[1]
    new_mat[x1][y1], new_mat[x2][y2]= &&  new_mat[x2][y2], new_mat[x1][y1]
```

```
    cost =&&calculateCost(new_mat, final)
    new_node =&&node(parent, new_mat, new_empty_tile_pos, cost, level)
    return new_node
def printMatrix(mat):
    for i in range(n):
        for j in range(n):
            print("%d " % (mat[i][j]), end=" ")
        print()
def isSafe(x, y):
    return x >=& 0 and x < n and y >= 0 and y < n
def printPath(root):
    if root ==     None:
        return
    printPath(root.parent)
    printMatrix(root.mat)
    print()
def solve(initial, empty_tile_pos, final):
    pq = priorityQueue()
    cost = calculateCost(initial, final)
    root = node(None, initial, empty_tile_pos, cost, 0)
    pq.push(root)
    while not pq.empty():
        minimum = pq.pop()
        if minimum.cost == 0:
            printPath(minimum)
            return
        for i in range(4):
            new_tile_pos = [
```

```
                minimum.empty_tile_pos[0] + row[i],
                minimum.empty_tile_pos[1] + col[i],
            if isSafe(new_tile_pos[0], new_tile_pos[1]):
                child = newNode(
                    minimum.mat,
                    minimum.empty_tile_pos,
                    new_tile_pos,
                    minimum.level + 1,
                    minimum,
                    final,
                pq.push(child)
initial = [[2, 8, 3],
           [1, 6, 4],
           [7, 0, 5]]
final = [[1, 2, 3],
         [8, 0, 4],
         [7, 6, 5]]
empty_tile_pos = [2, 1]
solve(initial, empty_tile_pos, final)
```

solution of the example

```
2  8  3
1  6  4
7  0  5


2  8  3
1  0  4
7  6  5
```

```
2 0 3
1 8 4
7 6 5

0 2 3
1 8 4
7 6 5

1 2 3
0 8 4
7 6 5

1 2 3
8 0 4
7 6 5
```

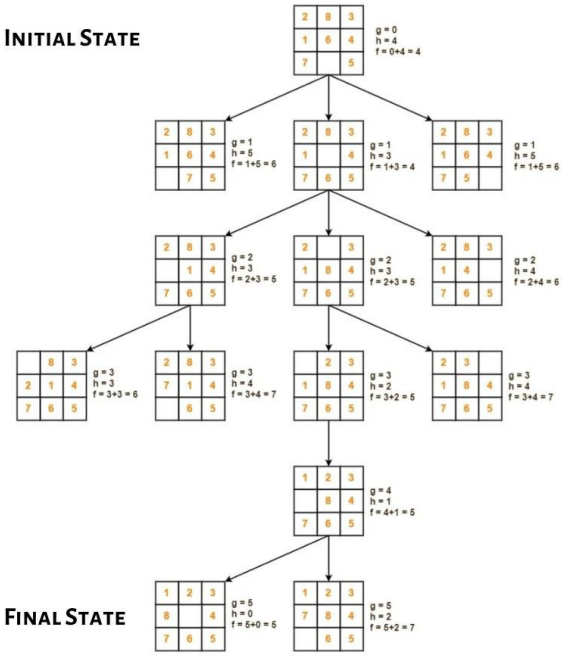

Figure 2.26: Branch and Bound Search in Python

CHAPTER 3

SOLVING MULTI OBJECTIVE LINEAR PROGRAMMING PROBLEMS USING VAGUE OPTIMIZATION METHOD: A COMPARATIVE STUDY

INTRODUCTION

In numerous optimization malfunctions, it takes remained experiential that unimportant of a assumed restraint or constraints can principal to additional well-organized solution [45]. This situation arises frequently in applied modeling, particularly in optimization glitches; it is often unreasonable to specify precise parameters, meanwhile abundant of them are found by guesstimate or around method of anthropological statement [13]. For specimen, in a creation optimization badly-behaved, it is not compulsory that all harvests be of lofty class to be traded at a motionless price.It is possible that jumpily products are broken-down and cannot be sold at the fixed price [19]. Also, due to overpowering conditions, raw substantial prices and over development marketplace amounts may vary due to shop spare/deficiency [76]. Then, it is clear that expenses in addition/or manufacturing be present not indecently deterministic, but imprecise or non-deterministic, and non classical methods requirement be used to determination such optimization glitches.Of progression, most

real-world difficulties linking optimization events are modeled as multi-objective program writing problems.In general, such multi-objective programming hitches may have conflicting objects [90].For example, in agricultural manufacture planning problems, the optimal model should aim at maximizing income while minimalizing input and planting costs [76].Due to the contradictory nature of these objectives, the solution to such problems is frequently a compromise that satisfies each objective role to some extent, and it is in this state that the ideas of connection and non-membership arise. Zimmermann [78], was the first to use Zadeh fuzzy sets. Multi-objective exact programming for solving fuzzy problems.Optimization in fuzzy environments has been studied and applied by several canvassers together with Tanaka [90], Luhandjula [78], Sakawa et al. The labor of Sahinidis provides a brief impression of optimization under uncertainty by numerous researchers.Various extensions of fuzzy sets have emerged due to their increasing When the available information is ambiguous, ambiguous, or uncertain, use it in modeling glitches. An extension of fuzzy sets is used in such a case [90]. In his research, Atanasoff emphasizes that attribution and non-attribution should be treated as distinct, rather than complementary, attributes when dealing with information fuzziness, ambiguity, or doubt. Angelov [34] judiciously well thought-out the concepts of relationship and non-membership in optimization glitches and presented a fuzzy method for resolving optimization hitches. Jana and Roy [90] inspected the multi-objective incoherent in lines software design problem and applied it to the challenging transportation badly behaved. Luo solved multi-criteria decision-making problems by incorporating fuzzy sets. Several other researchers, Several researchers, including Mahapatra et al. [90], investigated linear software proposal hitches in fuzzy environments using fuzzy numbers and interlude qualm in fuzzy numbers. The stimulus for this reading is the faith that it will be responsible for a computational algorithm for disentangling multi-objective linear software design teething troubles using fuzzy optimization methods. In this optimization process, we as well needed to investigate the waves of countless types of membership and non-membership occupations, so we

conducted a virtual scholarship of lined affiliation and non-membership title role with nonlinear involvement and non-membership gatherings. [34]. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. the name Adam is derived from adaptive moment estimation [34]. When introducing the algorithm, the authors list the attractive benefits of using Adam on non-convex optimization problems, as follows: Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

This code uses the NumPy library in Python to define an objective function, its derivative, and the Adam optimization algorithm. The Adam algorithm is a popular optimization technique that iteratively updates variables to minimize the objective function. The algorithm takes the objective function, derivative, bounds, number of iterations, learning rate, beta1, beta2, and epsilon as inputs. It generates a random initial point within the given bounds, initializes the first and second moments to 0, and updates the variables iteratively using the Adam equations.

Code 2 :code for gradient descent optimization

```
code for gradient descent optimization
with Adam for a two-dimensional test function:
from math import sqrt
import numpy as np
def objective(x):
return x[0]**2 + x[1]**2
def derivative(x):
return np.array([2*x[0], 2*x[1]])
```

```
def adam(objective, derivative, bounds, n_iter, alpha, beta1, beta2, eps=1e-8):
np.random.seed(1)
x = bounds[:, 0] + np.random.rand(len(bounds)) * (bounds[:, 1] - bounds[:, 0])
score = objective(x)
m = np.zeros(bounds.shape[0])
v = np.zeros(bounds.shape[0])
t = 0
for i in range(n_iter):
t += 1
grad = derivative(x)
m = beta1 * m + (1 - beta1) * grad
v = beta2 * v + (1 - beta2) * grad**2
m_hat = m / (1 - beta1**t)
v_hat = v / (1 - beta2**t)
delta = alpha * m_hat / (np.sqrt(v_hat) + eps)
x = np.maximum(np.minimum(x - delta, bounds[:, 1]), bounds[:, 0])
score = objective(x)
return x
while t < n_iter:
t += 1
grad = derivative(x)
m = beta1 * m + (1 - beta1) * grad
v = beta2 * v + (1 - beta2) * (grad**2)
m_hat = m / (1 - beta1**t)
v_hat = v / (1 - beta2**t)
step = alpha * m_hat / (sqrt(v_hat) + eps)
x = x - step
x = asarray([min(max(x[i], bounds[i, 0]), bounds[i, 1]) for i in range(x.size)])
```

```
score = objective(x)
print("Iteration %d - Score: %.5f" % (t, score))
return x, score
bounds = asarray([[-5.0, 5.0], [-5.0, 5.0]])
alpha = 0.01
beta1 = 0.9
beta2 = 0.999
eps = 1e-8
x, score = adam(objective, derivative, bounds,
n_iter=100, alpha=alpha, beta1=beta1, beta2=beta2, eps=eps)
print("Final Point:", x)
print("Final Score: %.5f" % score)
```

This code imports NumPy as "np" instead of importing "asarray" and "rand" functions separately. It uses NumPy's built-in "zeros" function instead of initializing first and second moments with a list comprehension. It uses NumPy's "maximum" and "minimum" functions instead of using array indexing to ensure that the updated variables remain within the given bounds. It also calculates the gradient of the objective function using NumPy's "array" function instead of using "asarray". Note that the objective function and its derivative have been changed to a two-dimensional test function, and the while loop has been changed to a for loop that iterates over n_{iter}. Also, the point update has been changed to use element-wise subtraction instead of matrix subtraction, and the point has been clipped to the bounds of the search space. Finally, the progress has been printed after each iteration, and the final point and score have been printed at the end.

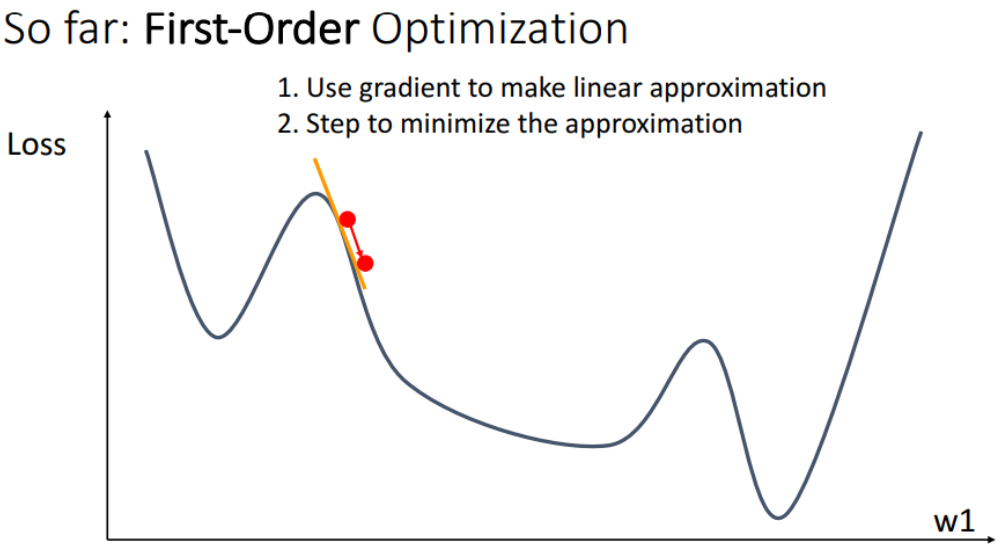

Figure 3.1: A vague set gradient descent optimization with Adam for a two-dimensional test function

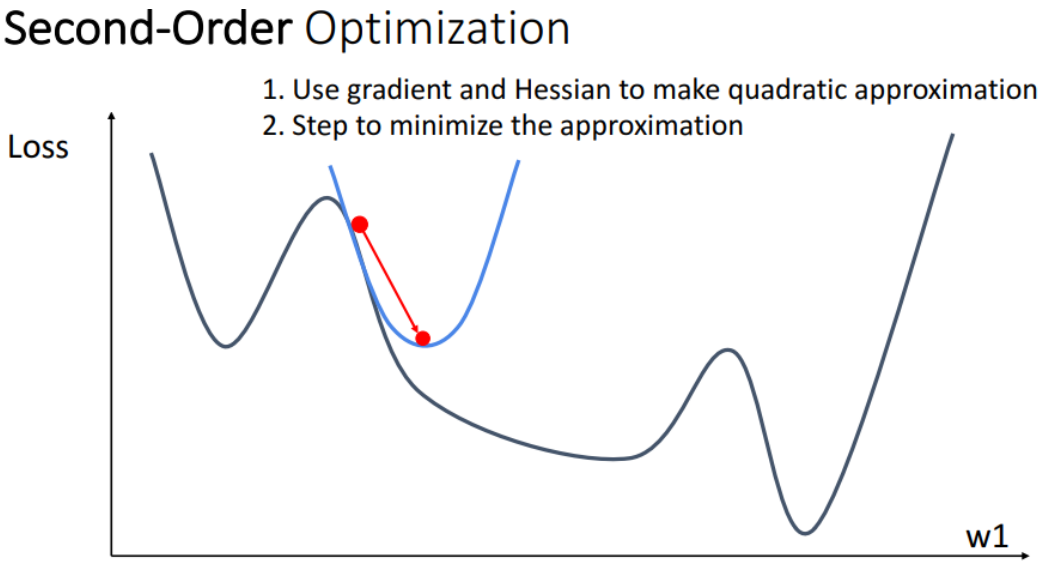

Figure 3.2: A vague set two-dimensional test function

The Adam optimizer, introduced in 2014, is an extended version of stochastic gradient descent that can be utilized in various deep learning applications such as computer vision and natural language processing in the future . It was first presented at ICLR 2015, a renowned conference for deep learning researchers, and is an optimization algorithm that can serve as an alternative to stochastic gradient descent. The name

Algorithm	Tracks first moments (Momentum)	Tracks second moments (Adaptive learning rates)	Leaky second moments	Bias correction for moment estimates
SGD	✗	✗	✗	✗
SGD+Momentum	✓	✗	✗	✗
Nesterov	✓	✗	✗	✗
AdaGrad	✗	✓	✗	✗
RMSProp	✗	✓	✓	✗
Adam	✓	✓	✓	✓

Figure 3.3: A vague set using Adam alone

"Adam" is derived from adaptive moment estimation [94]. The optimizer is named Adam because it employs estimations of the first and second moments of the gradient to adjust the learning rate for each weight of the neural network. It is worth noting that Adam is not an acronym. This optimization technique is considered to be the most efficient stochastic optimizer, as it only requires first-order gradients and has low memory requirements [8]. Prior to the introduction of Adam, several adaptive optimization techniques such as AdaGrad and RMSP were proposed, which demonstrated good performance compared to SGD. However, in some cases, these techniques exhibited certain drawbacks, such as poorer generalization performance than SGD [8]. Adam was introduced as an optimization technique that demonstrated better generalization performance compared to other adaptive optimization techniques [1]. Furthermore, Adam's hyperparameters have intuitive interpretations, which means that less tuning is required [8]. While Adam performs well in many cases, researchers have observed that it may not always converge to the optimal solution. In some diverse deep learning tasks, the generalization performance of Adam may be lower than that of SGD [8]. Nitish Shirish Keskar and Richard Socher have suggested that switching to SGD in certain cases may lead to better generalization performance than using Adam alone [84]. The Adam optimization algorithm is an alternative to traditional stochastic gradient descent for iteratively updating network weights based on training data. It is derived

from the computation of evolutionary moments and is commonly used in deep learning applications.

3.1 Mathematical Aspect of Adam Optimizer

Taking the formulas used in the above two methods, we get

$$m_t = \beta_1 m_{t-1} + (1 - \delta_1) \left[\frac{\delta L}{S_{wt}} \right]$$
$$m_t = \beta_1 v_{t-1} + (1 - \delta_2) \left[\frac{\delta L}{S_{wt}} \right]$$

```
Parameters Used :
1.\epsilon  = a
small +ve constraint to avoid 'division by 0' error when (vt -> 0). (10-8)
2. \beta_1 &
\beta_2 = decay rates of average of gradients in the above two methods.
(\beta_1 = 0.9 & \beta_1 = 0.999)
3. \alpha | Step size parameter / learning rate (0.001)
```

As both m_t and v_t are initialized as 0 using the above methods, they tend to become biased towards 0 when both β_1 and β_2 are close to 1. To address this issue, the Adam optimizer computes 'bias-corrected' m_t and v_t. This correction helps to control the weights when approaching the global minimum and prevent high oscillations when near it. The formulas used for the bias correction are as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.1}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t} \tag{3.2}$$

The name "Adam" comes from the fact that, during each iteration of the gradient descent process, we adapt our parameters in a controlled and unbiased manner. To achieve this,

we use bias-corrected weight parameters, denoted as ($m_{\hat{t}}$) and ($v_{\hat{t}}$), instead of the normal weight parameters m_t and v_t. By plugging these bias-corrected parameters into the general equation, can effectively update our parameters in a more accurate and reliable way.

Code 3 :The optimization technique of Gradient Descent with Adam is used to minimize a test function with two dimensions.

This code imports NumPy and the "sqrt" function from the "math" library. It defines the objective function and its derivative using NumPy's "array" function. It initializes the first and second moments to 0 using NumPy's "$zeros_likе$" function. It updates the variables iteratively using the Adam equations and NumPy's "maximum" and "minimum" functions to ensure that the updated variables remain within the given bounds. It prints the updated variables and the objective function value for each iteration.

```
 iteration.
import numpy as np
from numpy.random import rand, seed
from math import sqrt
def objective(x):
return x[0]**2 + x[1]**2
def derivative(x):
return np.array([2*x[0], 2*x[1]])
def adam(objective, derivative, bounds, n_iter, alpha, beta1, beta2, eps=1e-8):
seed(1)
x = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:, 0])
score = objective(x)
m = np.zeros_like(x)
v = np.zeros_like(x)
for t in range(n_iter):
grad = derivative(x)
for i in range(x.shape[0]):
```

```
m[i] = beta1 * m[i] + (1 - beta1) * grad[i]
v[i] = beta2 * v[i] + (1 - beta2) * grad[i]**2
m_hat = m[i] / (1 - beta1**(t+1))
v_hat = v[i] / (1 - beta2**(t+1))
delta = alpha * m_hat / (sqrt(v_hat) + eps)
x[i] = np.maximum(np.minimum(x[i] - delta, bounds[i, 1]), bounds[i, 0])
score = objective(x)
print('>%d f(%s) = %.5f' % (t, x, score))
return [x, score]
np.random.seed(1)
bounds = np.array([[-1.0, 1.0], [-1.0, 1.0]])
n_iter = 60 ,alpha = 0.02 , beta1 = 0.8 , beta2 = 0.999
best, score = adam(objective, derivative, bounds, n_iter, alpha, beta1, beta2)
print('Done!')
print('f(%s) = %f' % (best, score))
```

k	x_1	x_2	$f(x_1, x_2)$
0	-0.14595599	0.42064899	0.19825
1	-0.12613855	0.40070573	0.17648
2	-0.10665938	0.3808601	0.15643
3	-0.08770234	0.3611548	0.13812
4	-0.06947941	0.34163405	0.12154
5	-0.05222756	0.32234308	0.10663
6	-0.03620086	0.30332769	0.09332
7	-0.02165679	0.28463383	0.08149
...	...	...	...
...	...	...	...
...	...	...	...
48	-0.00048176	-0.00389929	0.00002
49	-0.00055861	-0.00356777	0.00001
50	-0.00056912	-0.00321961	0.00001
51	-0.00052452	-0.00286514	0.00001
52	-0.00043908	-0.00251304	0.00001
53	-0.0003283	-0.00217044	0.00000
54	-0.00020731	-0.00184302	0.00000
55	-8.95352320e-05	-1.53514076e-03	0.00000
56	1.43050285e-05	-1.25002847e-03	0.00000
57	9.67123406e-05	-9.89850279e-04	0.00000
58	0.00015359	-0.00075587	0.00000
59	0.00018407	-0.00054858	0.00000

Table 3.1: The optimization technique of Adam is used to minimize a test function with two variables

Code 4:The one max optimization problem is being solved using a search using a genetic algorithm

```
from numpy.random import randint, rand
def objective(x):
    return -sum(x)
def selection(pop, scores, k=3):
    selection_ix = randint(len(pop))
    for ix in randint(0, len(pop), k-1):
        if scores[ix] < scores[selection_ix]:
            selection_ix = ix
    return pop[selection_ix]
def crossover(p1, p2, r_cross):
    c1, c2 = p1.copy(), p2.copy()
    if rand() < r_cross:
        pt = randint(1, len(p1) - 2)
        c1 = p1[:pt] + p2[pt:]
        c2 = p2[:pt] + p1[pt:]
    return [c1, c2]
def mutation(bitstring, r_mut):
    for i in range(len(bitstring)):
        if rand() < r_mut:
            bitstring[i] = 1 - bitstring[i]
def genetic_algorithm(objective, n_bits, n_iter, n_pop, r_cross, r_mut):
    pop = [randint(0, 2, n_bits).tolist() for _ in range(n_pop)]
    best, best_eval = 0, objective(pop[0])
    for gen in range(n_iter):
        scores = [objective(c) for c in pop]
        for i in range(n_pop):
```

```
            if scores[i] < best_eval:
                best, best_eval = pop[i], scores[i]
                print(">%d, new best f(%s) = %.3f" % (gen, pop[i], scores[i]))
        selected = [selection(pop, scores) for _ in range(n_pop)]
        children = list()
        for i in range(0, n_pop, 2):
            p1, p2 = selected[i], selected[i + 1]
            for c in crossover(p1, p2, r_cross):
                mutation(c, r_mut)
                children.append(c)
        pop = children
    return [best, best_eval]
n_iter = 100
n_bits = 20
n_pop = 100
r_cross = 0.9
r_mut = 1.0 / float(n_bits)
best, score = genetic_algorithm(objective, n_bits, n_iter, n_pop, r_cross, r_mut)
print('Done!')
print('f(%s) = %f' % (best, score))
```

The following code solves the one max optimization problem using a genetic algorithm. The objective function in this case is the negative sum of the input bit string. The genetic algorithm uses selection, crossover, and mutation operations to evolve a population of bit strings towards the optimal solution. The population is randomly initialized with n_pop bit strings of length n_bits. The genetic algorithm runs for n_iter generations, during which the fitness of each bit string is evaluated using the objective function. The best bit string and its fitness are updated if a better solution is found. The genetic algorithm selects parents by tournament selection, performs crossover with a probability of r_cross, and performs

mutation with a probability of r_mut. The best bit string found by the genetic algorithm and its fitness are returned.

```
new best f([1.00, 0.00, 1.00, 1.00, 1.00, 1.00, 1.00,
 0.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
  0.00, 0.00, 0.00, 1.00]) = -15.001
>0, new best f([1.00, 0.00, 1.00, 1.00, 1.00, 0.00, 0.00,
 0.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
 1.00, 1.00, 1.00, 1.00]) = -16.001
>1, new best f([1.00, 1.00, 1.00, 0.00, 1.00, 1.00, 1.00,
1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 0.00, 1.00, 0.00,
 1.00, 1.00, 1.00, 1.00]) = -17.001
>3, new best f([1.00, 1.00, 1.00, 1.00, 1.00,1.00, 1.00,
1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 0.00, 0.00, 1.00,
 1.00, 1.00, 1.00, 1.00]) = -18.001
>4, new best f([1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
0.00,1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
1.00, 1.00, 1.00, 1.00]) = -19.001
>5, new best f([1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
 1.00, 1.00, 1.00, 1.00]) = -20.001
Done!
f([1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
1.00, 1.00, 1.00]) = -20.000011
```

CHAPTER 4

A VISUAL EXPLANATION OF GRADIENT DESCENT METHODS AND ADAM METHOD

Introduction

Gradient descent is a widely used optimization algorithm in machine learning and deep learning, used to minimize the cost function of a model by iteratively adjusting its parameters in the direction of steepest descent [92]. However, understanding the underlying mathematics and mechanics of gradient descent can be challenging, particularly for beginners in the field [59]. To address this challenge, researchers have developed various visual explanations of gradient descent methods, which make use of interactive animations and intuitive visualizations to illustrate the workings of the algorithm [49]. One such popular visualization is the "Loss Landscape" visualization, which provides a 3D representation of the cost function and the path taken by the gradient descent algorithm to reach the global minimum [93]. Another popular variant of gradient descent is the Adam optimizer, which is a more advanced optimization algorithm that combines the benefits of both the momentum. It utilizes adaptive learning rates for each parameter, which allows it to converge faster and with higher accuracy than traditional gradient descent methods [53].

4.1 Gradient Descent Method [42]

The problem have involves finding the minimum of the function $f(x)$, where $f(x)$ is a continuously differentiable function of the first order with respect to each coordinate variable x_i, $i = 1, \ldots, n$. To solve this problem, we need to minimize $f(x)$ with respect to one coordinate variable x_i, while keeping all other coordinate variables x_j, $j \neq i$, constraint.

Algorithm 1 : Gradient Descent Method

1. Initialize x^o,ϵ; set k:=0
2. while $\left\|g^k\right\| > \epsilon$
   (a) for $i = 1, \ldots, n$
   (b) ${x_i}^{new} = arg \min_{x_i} f(x)$
   (c) $x_i = {x_i}^{new}$ end for

output output: $x^* = x^k$ a stationary point of $f(x)$

globally convergent method if a search along any coordinate direction yields a unique minimum point.

Algorithm 2 : Augmented Lagrangian Genetic Algorithm

By default,the genetic algorithm uses the Augmented lagrangian Genetic Algorithm (ALGA)to solve nonlinnear constraint problems without integer constraints.The optimization problemsolved by the ALGA algorithm is:

$$\begin{cases} \text{minimize} & f^0(x) \\ \text{subject to} & c_i(x) \leq 00, \quad i = \{1, \ldots, m\}, \\ & ceq_i(x) = 0, i = m + 1 \ldots mt, \\ & xA.xb \\ & Aeq.x = beq \\ lb \leq x \leq ub. \end{cases} \tag{4.1}$$

where $c(x)$ represent the non linear inequality constraints $ceq(x)$represents the equality constraints,$m$is the number of nonlinear inequality constraints,and mt is the total number of nonlinear constraints. The Augmented Lagrangian Genetic algorithm (ALGA)attempts to solve nonlinear optimization problem with nonlinear constraints,linear constraints and bounds.in this approach,bounds and ,linear constraints are handled separately from nonlinear constraints.A sub problem is formulated by combining the fitness function and nonlinear constraints function using the Lagrangian and the penalty parameters.A sequence of such optimization problem are approximately minimized using the genetic algorithm such that the linear constraints and bounds are satisfied A sub problem formulation is defined as

$$\theta(x, \lambda, s,) = f(x) - \sum_{1}^{n} [\lambda_i, s_i log] (s_i - c_i(x)) + /2, \sum_{i=m+1}^{mt} = ceq_i(x)^2$$

where •$The components$ $\theta$of the vectorθ are non-negative and are known as Lagrange multiplier estimates

•$The elements s_i$ of the vector s are non negative shifts

•$Is the positive penalty parameter. the algorithm begins by using an initial value for the penalty parameter$

Algorithm 3 : Gradient Descent

Require:Learning rate a_n.

Require:The first parameter θ While stop criteria not met do

sample of m examples from the training set $.x_1, \cdots, x_m$ with corresponding

objectives y_i

comput gradient descent estimate: $\hat{g} \leftarrow + \frac{1}{m} \delta \sum_i (h_{(x_i)-y_i)}$

Applying updates:$\theta \leftarrow \theta - a\hat{g}$

End while

Algorithm 4: What is the Adam optimization algorithm?.

1. Input : $\alpha, \beta_1, \beta_2, f_t$
2. Output:θ_t

3. Initialize parameters ($\alpha = 0.001, \beta_1 = \beta_2 = 0.9, m_0 = 0, v_0 = 0, t = 0$)
4. For $t = 1$ to T do.
5. Get a stochastic gradient objective at time step t: $g_t = \delta_\theta f_t(\theta_{(t-1)})$
6. Update biased first-order moment estimation: $m_t = \theta_1 m_{t-1} + (1 - \theta_1) g_t$
7. Update biased second -order moment estimation: $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
8. Get bias-corrected first-order moment estimation:$\hat{m}_t = m_t/(1 - \beta_1^t)$
9. Get bias-corrected second-order moment estimation:$\hat{v}_t = v_t/(1 - \beta_2^t)$
10. Update the parameter : $\theta_t = \theta_{t-1} - \alpha \hat{m}_t / \hat{v}_t{}^{\frac{1}{2}}$
11. End For

Definition 4.1.1. **NumPy**, NumPy is a Python library that provides multidimensional array objects and a set of functions for manipulating those arrays. NumPy enables users to perform mathematical and logical operations on arrays efficiently. This tutorial provides an overview of NumPy, including its architecture and environment [92]. It covers various array functions, indexing techniques, and also introduces Matplotlib for visualization. The concepts are explained using examples to aid in understanding. In addition, NumPy allows developers to perform a range of operations, including but not limited to the following:

- Mathematical operations on arrays, such as addition, subtraction, multiplication, and division
- Trigonometric and logarithmic functions
- Linear algebra operations, such as matrix multiplication and inversion
- Statistical operations, such as mean, standard deviation, and variance
- Indexing and slicing of arrays

- Broadcasting, which allows operations on arrays with different shapes and sizes

Code 5 :Adam objective derivative [99]

```
import numpy as np
from math import sqrt
from numpy.random import rand, seed
from numpy import meshgrid, arange
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
def objective(x, y):
    return x**2 + y**2
def derivative(x, y):
    return np.array([2*x, 2*y])
def adam(objective, derivative, bounds, n_iter, alpha, beta1, beta2, eps=1e-8):
    solutions = []
    x = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:, 0])
    score = objective(x[0], x[1])
    m = np.zeros(bounds.shape[0])
    v = np.zeros(bounds.shape[0])
    for t in range(n_iter):
        g = derivative(x[0], x[1])
        for i in range(bounds.shape[0]):
            m[i] = beta1 * m[i] + (1 - beta1) * g[i]
            v[i] = beta2 * v[i] + (1 - beta2) * g[i]**2
            mhat = m[i] / (1 - beta1**(t+1))
            vhat = v[i] / (1 - beta2**(t+1))
            x[i] = x[i] - alpha * mhat / (sqrt(vhat) + eps)
        score = objective(x[0], x[1])
        solutions.append(x.copy())
```

```
        print(f'>{t} f({x}) = {score:.5f}')
    return solutions
def plot_contour(objective, bounds, solutions):
    xaxis = arange(bounds[0, 0], bounds[0, 1], 0.1)
    yaxis = arange(bounds[1, 0], bounds[1, 1], 0.1)
    x, y = meshgrid(xaxis, yaxis)
    results = objective(x, y)
    plt.contourf(x, y, results, levels=50, cmap='jet')
    solutions = np.asarray(solutions)
    plt.plot(solutions[:, 0], solutions[:, 1], '.-', color='w')
    plt.show()
seed(1)
bounds = np.array([[-1.0, 1.0], [-1.0, 1.0]])
n_iter = 30
alpha = 0.02
beta1 = 0.8
beta2 = 0.999
solutions = adam(objective, derivative, bounds, n_iter, alpha, beta1, beta2)
plot_contour(objective, bounds, solutions)
```

The code is now divided into several functions: ‘objective‘, ‘derivative‘, ‘adam‘, and ‘$plot_contour$‘. The ‘objective‘ function calculates the value of the objective function for a given point ‘(x, y)‘. The ‘derivative‘ function calculates the gradient of the objective function at a given point ‘(x, y)‘. The ‘adam‘ function performs the Adam optimization algorithm to find the minimum of the objective function.

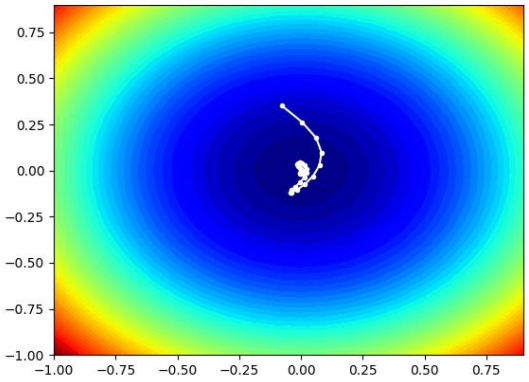

Figure 4.1: the Adam optimization algorithm

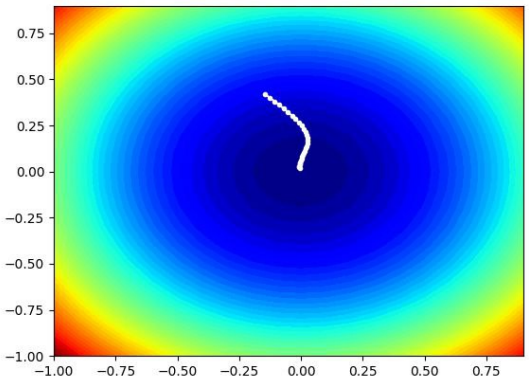

Figure 4.2: the Adam optimization algorithm divided into several functions

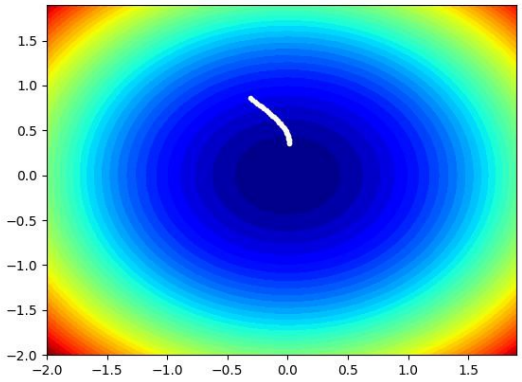

Figure 4.3: function performs the Adam optimization algorithm

k	x_1	x_2	gradient$f(x_1, x_2)$	Adam $f(x_1, x_2)$
1	−0.14595599	0.42064899	0.19825	0.19825
2	−0.12613855	0.40070573	0.17648	0.17647
3	−0.10665938	0.3808601	0.15643	0.15640
4	−0.08770234	0.3611548	0.13812	0.13810
5	−0.06947941	0.34163405	0.12154	0.12154
6	−0.05222756	0.32234308	0.10663	0.10663
7	−0.03620086	0.30332769	0.09332	0.09432
8	−0.02165679	0.28463383	0.08149	0.08140
9	−0.00883663	0.26630707	0.07100	0.07200
21	0.00532366	0.07867522	0.00622	0.00622
22	0.00193919	0.06900318	0.00477	0.00477
23	−0.00094677	0.06005154	0.00361	0.00370
24	−0.00324034	0.05181012	0.00269	0.00300
25	−0.00489722	0.04426444	0.00198	0.00122
26	−0.00591902	0.03739604	0.00143	0.00130
27	−0.00634719	0.0311828	0.00101	0.00100
28	−0.00625503	0.02559933	0.00069	0.00050
29	−0.00573849	0.02061737	0.00046	0.00040

Table 4.1: Adam objective derivative k x_1 x_2 gradient$f(x_1, x_2)$Adam $f(x_1, x_2)$

k	x_1	x_2	gradient$f(x_1, x_2)$	Adam $f(x_1, x_2)$
1	−0.07595599	0.35064899	0.12872	0.12873
2	0.00492189	0.26235065	0.06885	0.06887
3	0.06165083	0.17750892	0.03531	0.03538
4	0.08317613	0.09845384	0.01661	0.01660
5	0.0749628	0.0280328	0.00641	0.00640
6	0.04867756	−0.03072457	0.00331	0.00335
7	0.01508978	−0.07525774	0.00589	0.00577
8	−0.01617629	−0.10420301	0.01112	0.01111
9	−0.03721359	−0.11777423	0.01526	0.01534
10	−0.04396757	−0.11760788	0.01576	0.01565
11	−0.03723776	−0.10628817	0.01268	0.01260
12	−0.02134717	−0.08686837	0.00800	0.00870
23	−0.00359812	0.0156427	0.00026	0.00021
24	0.00316497	0.00500456	0.00004	0.00008
25	0.00750817	−0.00431422	0.00007	0.00007
26	0.0082094	−0.01135138	0.00020	0.00025
27	0.00562061	−0.01558072	0.00027	0.00025
28	0.00131057	−0.01692221	0.00029	0.00030
29	−0.00272043	−0.01569108	0.00025	0.00020

Table 4.2: Adam objective derivative gradient$f(x_1, x_2)$Adam $f(x_1, x_2)$

k	x_1	x_2	gradient$f(x_1, x_2)$	Adam $f(x_1, x_2)$
0	-0.14595599	0.42064899	0.19825	0.19825
1	-0.12523753	0.40039005	0.17600	0.17660
2	-0.10332941	0.37977213	0.15490	0.15450
3	-0.07963871	0.35874092	0.13504	0.13500
4	-0.05310417	0.33726003	0.11656	0.11666
5	-0.02111463	0.31529413	0.09986	0.09996
6	0.02607848	0.29280034	0.08641	0.08601
7	0.06464942	0.2697221	0.07693	0.07633
8	0.07347443	0.24598217	0.06591	0.06599
9	0.07454267	0.22147221	0.05461	0.05401
10	0.07092461	0.19603532	0.04346	0.04366
11	0.06373215	0.16943472	0.03277	0.03207
12	0.0534012	0.14129185	0.02282	0.02280
20	-0.03959691	-0.12012963	0.01600	0.01606
21	-0.02872136	-0.11640715	0.01438	0.01430
22	-0.01371166	-0.10936681	0.01215	0.01217
23	0.00950346	-0.09946959	0.00998	0.00999
24	0.03543143	-0.0869337	0.00881	0.00888
25	0.03870565	-0.07173569	0.00664	0.00665
26	0.03680966	-0.05348735	0.00422	0.00424
27	0.03127849	-0.03097202	0.00194	0.00190
28	$2.23921658e-02$	$-4.41267289e-05$	0.00050	0.00055
29	0.00943589	0.05527443	0.00314	0.0031

Table 4.3: Adam objective derivative gradient$f(x_1, x_2)$Adam $f(x_1, x_2)$ with other

4.2 Fuzzy Reliability

Reliability, denoted by $\bar{R}$, refers to the probability of a device successfully performing its intended purpose under specified operating conditions for a given period of time. In traditional reliability theory, this probability is calculated using a probabilistic approach. However, this approach has limitations, including the issue of obtaining accurate source data that reflects the real conditions of the system's operation [92]. Furthermore, statistical data used in probabilistic reliability models only record the occurrence of failures, without providing information on their causes. The causes of these failures are often related to variables such as temperature, humidity, and voltage, which may exceed or fall below certain critical levels. To address these limitations, other approaches such as fuzzy logic and chaos theory have been used for reliability modeling and optimization, it can be concluded that the probabilistic theory of reliability models the occurrence of failures, which is the space of event effects, and is not suitable for modeling the causes of failures, which is the space of event causes such as variables. An alternative approach to probabilistic modeling of reliability is based on fuzzy logic and possibility theory [11]. In this approach, the concept of "failure probability" is replaced by "failure possibility" which is determined by the membership function of the system or element variables in the reliable state.

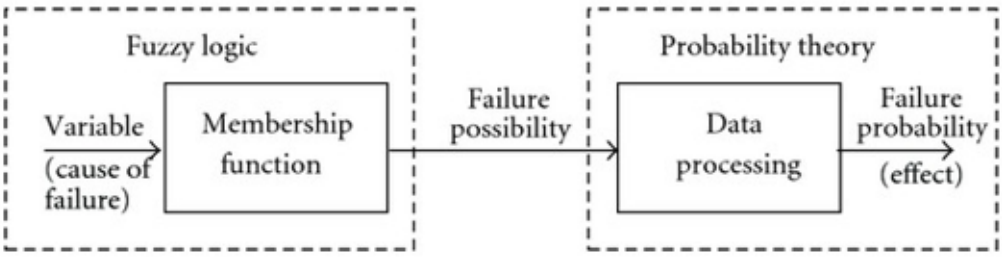

Figure 4.4: Failure possibility

and so is a promotion of a undefined set. Here merger and fitting together of dual unclear

sets are clear as

$$\bar{A} \cap \bar{B} = \left\{ \left[x, \min(t_{\bar{A}(x)}, t_{\bar{B}(x)}), \max(f_{\bar{A}(x)}, f_{\bar{B}(x)}) \right] \right\}$$

$$\bar{A} \cup \bar{B} = \left\{ \left[x, \max(t_{\bar{A}(x)}, t_{\bar{B}(x)}), \min(f_{\bar{A}(x)}, f_{\bar{B}(x)}) \right] \right\}$$

Fuzzy optimization Technique The maximum-minimum method Zimmerman first rummage-sale Bellman and Zadeh's [18] max-min worker to solve multi-objective lined problem design (MOLP) glitches, and he defined the problematic (1) as: Find X

$$\begin{aligned} & Z_k(x) \geq g_k, k = 1, 2, \ldots, p \\ g_j \leq 0, j = 1, 2, \ldots, p \qquad & x \geq 0 \end{aligned}$$

anywhere g_k and x mean goal mouth, and all unprejudiced gatherings are expected to remain exploited. In this case, impartial goals are regarded as fuzzy restraints. We could first get a table of optimistic ideal answers in order to establish association functions of impartial purposes (PIS). The possible response set is clear by the message of the fuzzy objective set in the min-operator idea. This feasible solution set is then defined by its membership $A(x)$, which is as follows: $t_D(x) = \min(t_1(x), \ldots, t_k(x))$. A decision maker then brands a decision by the maximumt D worth in the likely choice set. The choice answer can be got by solving the tricky of maximizet $D(x)$ below the given restraints, this means:

$$\begin{cases} \max \quad \min \quad (t_k(x)) \\ \text{subject to} \quad g_i(x) \leq 0, \quad i = \{1, \ldots, q\}, \\ \qquad\qquad\quad x \geq 0. \end{cases} \tag{4.2}$$

Now, if pretentious $\alpha = \min t_k(x)$, be the overall acceptable equal of collaboration, previously we obtain the next equivalent perfect Max α Such that $t_k(x) \geq \alpha, \forall k \qquad g_i(x) \leq 0,$

4.3 Unclear Optimization Method

Consider the ambiguous optimization routine as promotion of the as the crow flies above problematic a below demanding by Angelo [3]

$$\begin{cases} \min & f_i(x) \\ \text{subject to} & g_i(x) \leq 0, \quad j = \{1, \ldots, q\}, \\ & x \geq 0. \end{cases} \tag{4.3}$$

Where x signifies the choice variables, $f_i(x)$ signifies the impartial drives, $g_j(x)$ signifies the limit meanings, and p and q represent the number of impartial purposes and restraints. This problem's best answer must encounter all restraints accurately. Accordingly, an analogous ambiguous optimization prototypical of the problem is used to feat the degree of reception of bits and pieces and restraints as follows:

$$\begin{cases} \min^{\sim} & f_i(x) \\ \text{subject to} & g_i(x) \leq 0, \quad j = \{1, \ldots, q\}, \\ & x \geq 0. \end{cases} \tag{4.4}$$

Where (min) means unclear minimization and typifies fuzzy discrimination. Bellman and Zadeh [4] rummage-sale vague set exploit for the step of association of the decisiveness and shrinking to solve this order (5).

$$\max t_k(x), x \in X, k = 1, 2, \ldots, p+q \qquad 0 \leq t_k(x) \leq 1 \qquad x \geq 0$$

wherever $t_k(x)$ means the notch of gratification with the particular indefinite sets. It is critical to appreciate that the grade of non-membership in a ambiguous usual is the second of membership, so growth of the relationship drive determination automatically minimalize non-membership. However, in a vague set, the degree of rebuff is sure concurrently with the score of receiving, and because these grades are not

complementary, VS may deliver a more wide-ranging tool for describing this uncertainty-based optimization model. As a size, the dim unsure optimization (VO) picture-perfect for awkward(3) is if as for every [8].

$$\begin{cases} \max \quad \min & \\ & t_{k(x)} \qquad k=1,2,\ldots,p+q \\ & f_{k(x)} \qquad k=1,2,\ldots,p+q \\ \text{subject to} & \\ & f_k(x)\geq 0, \quad k=\{1,\ldots,p+q\} \\ & t_k(x)\geq f_k(x), \quad k=\{1,\ldots,p+q\} \\ & t_k(x)+f_k(x)\leq 1, \quad k=\{1,\ldots,p+q\} \\ & x\geq 0. \end{cases} \tag{4.5}$$

someplace $t_k(x)$ represents the degree to which x is predictable by the k^th VS and $f_k(x)$ denotes the degree to which x is rejected by the th^k IFS. Vague objects and fetters are among the IFS. The excellent set D is now clear as a mixture of vague objectives and restraints.

$$\bar{F}\cap\bar{C}=\begin{cases}[x,\min(t_{\bar{F}}(x),t_{\bar{c}}(x))] \\ \max(f_{\bar{F}}(x),f_{\bar{C}}(x))\end{cases} \tag{4.6}$$

where,$\bar{F}$ is combined vague uncertain impartial and $\bar{C}$ means combined vague fetters and

is distinct as.

$$\bar{F} = \{[x, (t_{\bar{F}}(x), f_{\bar{F}}(x), x \in X)]\} = \cap_{i=1}^{p} \bar{F}(i) \tag{4.7}$$

$$= \left\{ [x, \min_{i=1}^{p} t_i f(x), \max_{i=1}^{p} t_i f(x)], x \in X \right\} \tag{4.8}$$

$$\bar{c} = \{[x, (t_{\bar{c}}(x), f_{\bar{c}}(x), x \in X)]\} = \cap_{i=1}^{p} \bar{c}(i) \tag{4.9}$$

$$= \left\{ [x, \min_{j=1}^{q} t_j f(x), \max_{i=1}^{q} t_i f(x)], x \in X \right\} \tag{4.10}$$

$$\tag{4.11}$$

Further, the vague choice set (VDS) denoted as $\bar{D}$:

$$\bar{D} = \bar{F} \cup \bar{C} = \{x, t_{\bar{D}}(x), f_{\bar{D}}(x), x \in X\} \tag{4.12}$$

$$\bar{D}(x) = \min[t_{\bar{F}}(x), t_{\bar{c}}(x)] = \min_{k=1}^{p+q} t_k(x) \tag{4.13}$$

$$f_{\bar{D}}(x) = \min[f_{\bar{F}}(x), f_{\bar{c}}(x)] = \min_{k=1}^{p+q} t_k(x) \tag{4.14}$$

anywhere $t_{D(x)}$ designates the degree of VDS getting and $f_D(x)$ means the grade of VDS snub. Now, for a possible answer, the equal of receipt of VDS is continuously fewer than or equal to the level of receipt of any separate and constraint.

$$t_{D(x)} \leq t_{k(x)} \tag{4.15}$$

$$f_{D(x)} \geq f_{k(x)} \tag{4.16}$$

$$\forall, k = 1, 2, \ldots, p + q \tag{4.17}$$

As a result the above group can be transmuted to the following union of disparities:

$$\alpha \leq t_k(x) \qquad k = 1, 2, \ldots, p+q \tag{4.18}$$

$$\beta \geq f_k(x) \qquad k = 1, 2, \ldots, p+q \tag{4.19}$$

$$\alpha + \beta \leq 1 \tag{4.20}$$

$$\alpha \geq \beta, \beta \geq 0, x \in X \tag{4.21}$$

someplace represents the lowest acceptable notch of independent(s) and limits, and signifies the maximum notch of criticism of objective(s) and irons. By means of the unclear optimization, problematic (1) is now biased to the lined programming problematic assumed as:

$$\text{exploit}(\alpha - \beta) \qquad \text{subject} \quad \text{to} \quad \alpha \leq t_k(x)$$

$$\beta \geq f_k(x) \qquad k = 1, 2, \ldots, p+q \tag{4.22}$$

$$\alpha + \beta \leq 1 \tag{4.23}$$

$$\alpha \geq \beta, \beta \geq 0, x \in X \tag{4.24}$$

Now, this lined software design problem can be effortlessly solved using a simplex technique to deliver an optimization response to a multiobjective linear programming problematic (1). Figure 1 portrays the lined membership and nonmembership purposes.

COMPUTATIONAL ALGORITHM

Algorithm 2 :(Linear Association Function)

1. Select the first detached role from the hitch's set of k determinations and solve it as a on its own unprejudiced surrounded by dint of the constraints as short as [50]. Control the significance of middle-of-the-road functions and decision variables.

2. Using the judgment variables' standards, compute the values of the remaining $(k-1)$ objects.

3. For the right-hand over $(k-1)$ objective senses, reappearance Rankings 1 and 2.

4. Create the PIS bench by tabulating the standards of the dispassionate purposes obtained in Classifications 1 and 2 and 3.

5. Using the figures from Step 4, determine the higher and upper leaps for each disinterested common sense.

6. TABEL;POSITEVE IDEAL SOLUTION (PIS)

.	.	.
$\max f_1$	$f_1^{/}(x_1) \times f_2^{/}(x_1) \times f_3^{/}(x_1) \ldots f_k^{/}(x_1)$	x_1
$\max f_2$	$f_1^{/}(x_2) \times f_2^{/}(x_2) \times f_3^{/}(x_2) \ldots f_k^{/}(x_2)$	x_2
$\max f_3$	$f_1^{/}(x_3) \times f_2^{/}(x_3) \times f_3^{/}(x_3) \ldots f_k^{/}(x_3)$	x_3
$\vdots$	$\vdots$	$\vdots$
$\max f_k$	$f_1^{/}(x_k) \times f_2^{/}(x_k) \times f_3^{/}(x_k) \ldots f_k^{/}(x_k)$	x_k
	$f_1^{/} \times f_2^{/} \times f_3^{/} \cdots \times f_k^{/}$	

Table 4.4: POSITEVE IDEAL SOLUTION

where $f_{k^*}$and $f_{k'}$are the highest, smallest values respectively.

Stage 6. Set $U_k^{\mu} = \max(Z_k(X_r))$ and
$L_k^{\mu} = \min(Z_k(X_r)), 1 \leq r \leq p$ connotation and for non-membership resolutions.

$$U_k^{\mu} = U_k^{\mu} - \lambda(U_k^{\mu} - L_k^{\mu}) \tag{4.25}$$

$$L_k^{\mu} = L_k^{\mu} \tag{4.26}$$

$$0 \leq 1 \tag{4.27}$$

7. Use following lined association occupation($t_k(f_k)(x)$ and non-membership drive ($f_k(f_k)(x)$ for each objective functions:

$$t_k(f_k)(x) = \begin{cases} 0 & \text{if} \quad f_k(x) \leq L_k^t \\ \dfrac{f_k(x) - L_k^t}{U_k^t} & \text{if} \quad L_k^t \leq f_k(x) \leq U_k^t \\ 1 & \text{if} \quad f_k(x) \geq U_k^t \end{cases}$$

$$f_k(f_k)(x) = \begin{cases} 0 & \text{if} \quad f_k(x) \leq U_k^t \\ \dfrac{U_k(x) - f_k^t}{U_k^t - L_k^t} & \text{if} \quad L_k^t \leq U_k(x) \leq U_k^t \\ 1 & \text{if} \quad f_k(x) \geq L_k^t \end{cases}$$

8. Today the vague optimization system for MOLP unruly (1) with lined association and non relationship drives gives a equivalent linear software enterprise difficult as :

$$\begin{cases} \text{Maximize} \\ \qquad (\alpha - \beta) \\ \text{subject to} \\ \qquad \alpha \leq t_k(f_k(x)) \\ \qquad \beta \geq (f_k(f_k(x)) \\ \qquad (\alpha + \beta) \geq 1 \\ \qquad \alpha \geq \beta, \qquad \beta \geq 0 \\ \qquad g_j(x) \leq b_j, \qquad x \geq 0 \\ \qquad k = 1, 2, \ldots, p; j = 1, 2, \ldots, q \end{cases}$$

9. Firmness the directly above crumpled software development problematic(14) by means of the simplex technique. B. Procedure II (Nonlinear Contribution Purpose) Stages 1–6 must be boring to make a bench of positive perfect answers.

10. Undertake that the process computed keys review a hyperbolic purpose for relationship and an exponential job for non-membership

$$t_k(f_k)(x) = \begin{cases} 0 & \text{if} \quad f_k(x) \leq L_k^t \\ 1 - \exp\left\{-\psi \dfrac{f_k(x) - L_k^t}{U_k^t - L_k^t}\right\} & \\ \text{if} \quad U_k^t \leq f_k(x) \leq L_k^t & \\ 1 \quad \text{if} \quad f_k(x) \geq U_k^\mu \quad \text{and} \psi \to \infty \end{cases} \tag{4.28}$$

$$f_k(f_k)(x) = \begin{cases} 1 & \text{if} \quad f_k(x) \leq L_k^\mu \\ \dfrac{1}{2} - \dfrac{1}{2}\tanh\left\{\delta \dfrac{U_k(x) - L_k^t}{2}\right\} & \\ \text{if} \quad L_k^t \leq f_k(x) \leq U_k^t & \\ 0 \quad \text{if} \quad f_k(x) \geq U_k^f \end{cases} \tag{4.29}$$

anywhere are non-zero boundaries fixed by the conclusion fabricator. Besides, the unknown optimization performance for MOLP unruly (1) by way of exponential envelopment and hyperbolic non production livings profits the direct software

scheme badly-behaved: (-) Takings full disadvantage of

$$\begin{cases} & \alpha \leq t_k(f_k)(x) \\ & 1 - \exp\left\{-\psi \dfrac{f_k(x) - L_k^t}{U_k^t - L_k^t}\right\} \\ & \text{if} \qquad U_k^t \leq f_k(x) \leq L_k^t \\ & \beta \geq f_k(f_k(x)) \\ \text{subject to} \quad & \dfrac{1}{2} - \dfrac{1}{2}\tanh\left\{\delta \dfrac{U_k(x) - L_k^t}{2} - f_k(x)\right\} \leq \beta \\ & \alpha + \beta \leq 1, \alpha \geq \beta \\ & \beta \geq 0 \\ & g_i(x) \leq b_i, x \geq 0 \\ & k = 1, 2, \ldots, p; j = 1, 2, \ldots, q \end{cases} \tag{4.30}$$

For result suitability the above your head problem (15) is partial to

$$\begin{cases} \text{maximize} \quad \gamma - \eta \\ \qquad\qquad f_k(x) - \dfrac{\gamma(U_k^t - L_k^t)}{4} \geq L_k^t) \\ \qquad\qquad \text{where} \qquad \gamma - \log(1 - \alpha) \\ \qquad\qquad f_k(x) - \dfrac{\eta}{\delta_k} \geq \dfrac{U_k^f - L_k^f}{2} \\ \text{subject to} \quad \text{where} \qquad \eta = \tanh^{-1}(2\beta - 1) \qquad \psi = 4 \\ \qquad\qquad \delta_k = \dfrac{6}{U_k^f - L_k^f} \\ \qquad\qquad g_j(x) \leq b_j \\ \qquad\qquad k = 1, 2, \ldots, p; j = 1, 2, \ldots, q \end{cases} \tag{4.31}$$

Application

Building Preparation Problem Deliberate a joint of six mechine systems whose measurements are to be whole-hearted to construction of three goods and chattels. A contemporary dimensions portfolio is available , measured in mechine hours each weedy for individually mechine amounts unit valued giving to apparatus type. Essential information is recorded as in less Table II.

	Product				
Machine type	Machi hours	Unit price (100 per hour)	x_1	x_1	x_1
Milling machine	1400	0.75	12	17	0
Lathe	1000	0.60	3	9	8
Grinder	1750	0.35	10	13	15
Jig saw	1325	0.50	6	0	16
Drill press	900	1.15	0	12	7
Band saw	1075	0.65	9.5	9.5	4

Table 4.5: Essential information is recorded as in less Table II

Total capacity cost 4658.75,

	f_1	f_2	f_3	X
max f_1	8041.14	10020.33	9319.25	X_1
max f_2	5452.63	10950.59	5903.00	X_2
max f_3	7983.60	10056.99	9355.90	X_3

Table 4.6: POSITIVE IDEAL SOLUTION

Let X_1, X_2, X_3 indicate three crops, then the ample in your own time formulation of the above stated tricky as a Multi-objective. Undeviating Software plan (MOLP) problematic

is known as:

$$\begin{cases} \text{maximize} f_1(x) = 50x_1 + 100x_2 + 17.5x_3 \\ f_2(x) = 92x_1 + 75x_2 + 50x_3 \\ f_3(x) = 25x_1 + 100x_2 + 75x_3 \\ \text{subject to the restraints} \\ 12x_1 + 17x_2 \leq 1400 \\ x_1 + 9x_2 + 8x_3 \leq 1000 \\ 10x_1 + 13x_2 + 15x_3 \leq 1750 \\ 6x_1 + 16x_2 \geq 1325 \\ x_1, x_2, x_3 \geq 0 \end{cases} \tag{4.32}$$

Rejoinder of the overhead tricky is well thought-out by the means I and method II revealed in preceding components. For policy of the events almost of steps stand showing as

Step 1: Disentangle a wrinkly software proposal problem good-looking one and only objective

$$\begin{cases} \text{Expoit} & f_1 = 50x_1 + 100x_2 + 17.5x_3 \\ & 12x_1 + 17x_2 \leq 1400 \\ & x_1 + 9x_2 + 8x_3 \leq 1000 \\ & 10x_1 + 13x_2 + 15x_3 \leq 1750 \\ \text{Topic to the fetters} & 6x_1 + 16x_2 \geq 1325 \\ & 12x_2 + 7x_3 \leq 900 \\ & 9.5x_1 + 9.5x_2 + 4x_3 \leq 1075 \\ & x_1, x_2, x_3 \geq 0 \end{cases} \tag{4.33}$$

VALUES OF OPTIMAL DECISION VECTORS

	λ	β	x_1	x_2	x_3
1	65.2571	26.9187	49.8324	0.5899	0.4101
2	58.4833	34.5907	47.6992	0.8525	0.1475
3	65.2600	26.9155	49.8328	0.8847	0.2417
4	65.2585	26.9172	49.8328	0.8847	0.1153
5	66.1947	25.8441	49.2978	1.000	0.0000
6	71.1362	22.6184	44.8504	1.000	0.0000
7	71.7199	25.7841	35.6084	1.000	0.0000
8	75.3355	14.2823	45.3258	1.000	0.00000
9	82.1131	9.12270	46.1075	1.000	0.00000

Table 4.7: Vague optimization Technique when association and Non- memberships are lined.

vague optimization Technique when membership and Non- memberships are Non-linear

	λ	β	x_1	x_2	x_3
1	49.8906	471360	42.5550	0.6321	0.3345
2	64.6968	36.6846	41.7421	0.6321	0.0073
3	62.1896	38.0097	41.8452	0.6321	0.0009
4	62.8180	38.0109	41.5300	0.6321	0.0001
5	62.8157	38.0125	41.8454	0.6321	0.0000
6	62.8163	38.0120	41.8454	0.6321	0.0000

Table 4.8: Vague optimization Technique when association and Non- memberships are lined

Definition 4.3.1. Birnbaum's measure of reliability importance of component t time t is the probability that the system is in such a state at time t that component i is critical for the system. Now we using Birnbaum's importance , IB equation below to calculate

for the seven components [24].

$$IB_{ci} = \frac{\partial R_s(R_1, R_2, R_3, \cdots, R_7)}{\partial R_i}) \tag{4.34}$$

Where C_i represent the component i, $i = 1, 2, 3, \times, \partial R_i$ is the reliability of i compound i

Definition 4.3.2. The importance potential (IPC_i) with respect to component i at time t as follows:

$$(IPC_i)(i/t) = IBM(i, T)(1 - P_i(t)) \tag{4.35}$$

where $1 - P_i(t)$ is the unreliability of component i [24].

Example 4.3.1. Let a complex system which is consist of seven components, its a components have the same initial reliability of $o.91$ at a given time.

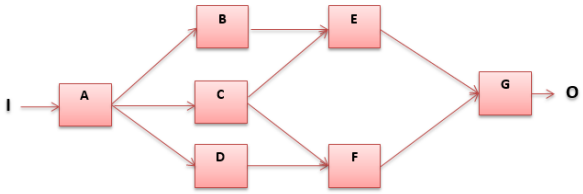

Figure 4.5: Types of systems and the methods for solving the systems reliability

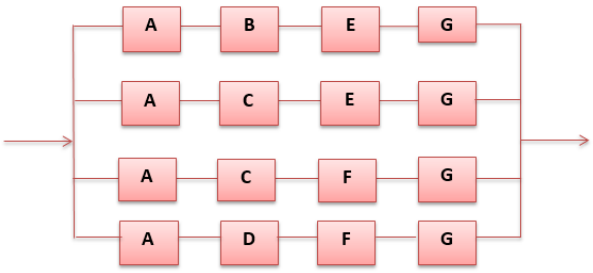

Figure 4.6: Representation for Figure

Let O, P, Q, S paths where, $O = \{a, b, e, g\}$, $p = \{a, c, e, g\}$

$Q = \{a, c, f, g\}$, $S = \{a, d, f, g\}$ the reliability for each path.

$$R_O = R_a \times R_b \times R_e \times R_g$$

$$R_P = R_a \times R_c \times R_e \times R_g$$

$$R_Q = R_a \times R_c \times R_f \times R_g$$

$$R_S = R_a \times R_d \times R_f \times R_g$$

Therefore the model reliability is : $R_{\text{system}} = 1 - \prod_{j=1}^{4} = (1 - R_j)$

$= 1 - (1 - O)(1 - P)(1 - Q)(1 - S)$

$= O + P + Q + S - OP - OQ - PQ - PS - OS - QS + OPQ + OPS + OQS + PQS - OPQS$

$R_a R_b R_e R_g + R_a + R_c R_e R_g + R_a R_c R_f R_g$

$R_a R_b R_c R_e R_g - R_a - R_b R_c R_f R_g - R_a R_b R_d R_e R_f R_g - R_a R_c R_e R_f R_g - R_a + R_c R_d R_e R_f R_g$

$-R_a R_c R_d R_f R_g - R_a + R_b R_c R_e R_f R_g$

$+R_a R_b R_c R_d R_e - R_f R_g + R_a R_b R_c R_d R_e R_f R_g + R_a R_c R_d R_e R_f R_g$

$-R_a R_b R_c R_d R_e R_f - R_g$

Substituting $R_s = 0.91$ for each component we are get $R_{\text{system}} = 0.9903$ we reduce o, p, q, r, to, i

$$R_{\text{system}} = 1 - [(1 - R_O)(1 - R_P)(1 - R_Q)(1 - R_R)]$$

$$= 1 - [1 - R_i]^m$$

$$R_{\text{system}} = 1 - [1 - P(t)]^m$$

$$R_O = R_P = R_Q = R_S = 0.9$$

$$= 0.91^4 = 0.685$$

$$R_{\text{system}} = 1 - (1 - 0.6857)^4 = 0.9903$$

Now using Birnhaums importance IB, equation $4,34$, we get

$\text{IBM}_{C_a} = 0.9007$

$IBM_{C_b} = 0.0116$

$IBM_{C_c} = 0.0177$

$IBM_{C_d} = 0.0116$

$IBM_{C_e} = 0.0794$

$IBM_{C_f} = 0.0794$

$IBM_{C_g} = 0.9007$

Finding the important potential (IP), for components by substation in equation (4.34) we get

$$C_i : i = a, b, c, d, e, f, g$$

1. $IP_{C_a} = 0.9007(1 - 0.91) = 0.09$

2. $IP_{C_b} = 0.0116(1 - 0.91) = 0.09$

3. $IP_{C_c} = 0.0177(1 - 0.91) = 0.0015$

4. $IP_{C_d} = 0.0116(1 - 0.91) = 0.0010$

5. $IP_{C_e} = 0.0794(1 - 0.91) = 0.00071$

6. $IP_{C_f} = 0.0794(1 - 0.91) = 0.00071$

7. $IP(C_g) = 0.4007(1 - 0.91) = 0.018$

Main results

1. The components with avery row value of IBM will have a negligible effect on the system reliability.

2. According to IBM, the component with the highest reliability is the most important in a parallel system so we can say that the most reliable component is most important or the optimal one.

3. Classification of importance that is to allocate the components into two or most groups according to some preset criterias

Note. 1. The components with a very row value of IBM will have a negligible effect on the system reliability.

2. According to IBM, the component with the highest reliability is the most important in a parallel system, we can say that the most reliable component is most important

4.4 Fuzzy optimization of the Reliability of complex system

Fault tree analysis, reliability optimization, and risk analysis are just a few of the many fields in which fuzzy reliability theory has found use. Fuzzy mathematical techniques can successfully be used to improve conventional reliability theory without requiring fuzzy reliability theory. In order to address the reliability mapping problem for systems

with two connected components in series, Park initially demonstrated how to employ fuzzy optimization approaches. Reliability and redundancy allocation for Level 4 and 5 overspeed avoidance systems later became the focus of Dhingra and Rao. Use explicit or ambiguous multi-objective optimization techniques. In summary, the non-compensated min operator is used as an aggregator in all the previously described works. While Dhingra and Rao utilized linear membership functions for the dependability function, Park uses a linear membership function for the objective. Stochastic search based optimization methods are not used in any of these publications. As a result, a vast array of metaheuristic algorithms capable of handling mixed integer optimization problems that are highly nonlinear and nonconvex have been disregarded.

4.4.1 Preliminaries

Definition 4.4.1. The overall structure under consideration is called a system, and subsystems or components are the lower-level structures that make up a system. Multiple connected components make up a complex system [26].

Definition 4.4.2. neither in a sequence nor a parallel fashion A system's dependability is the The likelihood that a system will effectively complete its intended purpose within the given environmental parameters for a given amount of time [101].

Definition 4.4.3. The ability of a device to carry out its intended function with varying degrees of success for the intended amount of time under the actual operating conditions is known as fuzzy reliability [100].

Definition 4.4.4. If U is the set of objects It is a fuzzy set if x is the genetic representation of it. A is a collection of ordered pairs in U: $\bar{A} = \{(x, f_{\bar{A}}(x)) : x \in U\}$ $f_{\bar{A}}$ is also known as the degree of membership of x in A and specifies a real number $f_{\bar{A}}(x)$ in the range $[0, 1]$ [17].

Remark. Using the two following presumptions, fuzzy reliability theory

1. Fuzzy State Assumptions: System successes and failures are represented by fuzzy states. A system can be considered to be partially in one of two state at any given time. So system failures are more ambiguous than absolutes (success or failure).

2. Possibility premise: The probability measure completely characterizes the system failure behavior. An explicit reliability theory based on the following assumptions:- The system can only be entirely failed, according to the binary state assumption. Probability assumptions: Probability measures completely describe the system failure behavior. The membership functions for fuzzy number A are:

$$\begin{cases} 0 & x < a_1, x \geq a_3 \\ \dfrac{x - a_1}{a_2 - a_1} & a_1 \leq x \leq a_2 \\ \dfrac{a_3 - x}{a_3 - a_2} & a_2 \leq x \leq a_3 \end{cases} \tag{4.36}$$

Then A is referred to as, $a_1, a_2, a_3 \in R$, number dented by $A = (a_1, a_2, a_3)$

Example 4.4.1. Based on [5]. If coft is a real number that is somewhat larger than 2m, then consider the membership function of the fuzzy set A.$A(x) = \dfrac{1}{1 + 10(x-2)^2}$ The graph of the membership is as in below.

Example 1

Taken from[5] to maximizing the system reliability as close to 1 as possible, the approximate cost of the system cost unit) The linear membership function is as follows:

$$\begin{cases} 0 & \text{where} \quad R_s \leq 0.99 \\ \dfrac{R_s - 0.99}{0.9905 - 0.99} & 0.99 \leq R_s \leq 0.9905 \\ 1 & \text{where} \quad R_s \leq 0.9905 \end{cases}$$

And

$$\mu_{c_s} = \begin{cases} 1 & \text{where} \quad C_s \leq 5.0 \\ \dfrac{5.02047 - C_s}{5.02047 - 0} & 0.99 \leq R_s \leq 0.9905 \\ 1 & \text{where} \quad C_s \leq 5.02047 \end{cases}$$

Assume that each component of the bridge system in figure (2) has a reliabilityR_i where $iof 1, 2, 3, or 4$. The following pathways are available:
$P1 = 1, 4; P2 = 2, 5; P3 = 1, 3, 5; andP4 = 2, 3, 4$ The structure func. allows one to calculate R s for the bridge system.

$$R_s = (R_1 \wedge R_4) \vee (R_2 \wedge R_5) \vee (R_1 \wedge R_3 \wedge R_5) \vee (R_2 \wedge R3 \wedge R_4)$$

Therefore $R_s = 1 - (1 - P_1)(1 - P_2)(1 - P_3)(1 - P_4)$
$= R_1R_4 + R_2R_5 + R_1R_3R_5 + R_2R_3R_4 + 2R_1R_2R_3R_4R_5 - R_1R_2R_4R_5 - R_1R_2R_3R_4 - R_1R_3R_4R_5 - R_2R_3R_4R_5 - R_1R_2R_3R_5$. To identify the choice variables R_i that minimize, set $i = 1, 2, \ldots, 5$. To identify the choice variables R_i that minimize, set $i = 1, 2, \ldots, 5$.
$C_s = \sum_{i=1}^{5} a_i \exp \dfrac{bi}{1 - R_i}$ The system casts the global lowest reported5.0199 Ravietal [73].

4.5 applications fuzzy mathematics of reliability analysis

One of fuzzy mathematics' most important contributions to human dependability is its ability to reflect the phenomenon of uncertainty, which is connected to information sources and the inherently unpredictable nature of human-machine interaction. As Zio points out, a major problem in reliability analysis is the unpredictability of failure occurrences and their consequences. Depending on the quantity of information available, risk assessments can handle uncertainty in one of three ways [14].
Developing a human dependability model may involve subjective data; as a result, an efficient mathematical approach for this kind of data is necessary. Operators'

comprehension of system variables and their interrelationships, according to Sheridan is vague. An essential source of information for job analysis is this knowledge [28]. Uncertainty is brought on by the imprecision and ambiguity of natural language. For instance, a system can withstand any plausible accident if it is designed to survive the worst accident situations. However, using the phrase "worst case scenario" suggests subjectivity and arbitrary decision-making, which might prompt the consideration of incredibly implausible scenarios. Giving this linguistic phrase a flexible definition should fix that problem [37].

4.6 Hybrid algorithm [68, 97]

A hybrid algorithm is a combination of two or more different algorithms that work together to solve a problem. The goal of a hybrid algorithm is to take advantage of the strengths of each individual algorithm while minimizing their weaknesses. Here's an example of a hybrid algorithm:

1. Start with a simple heuristic algorithm that provides a good initial solution to the problem.

2. Use a genetic algorithm to improve upon the initial solution by generating a population of potential solutions and selecting the best ones for the next generation.

3. Use a local search algorithm to fine-tune the best solutions found by the genetic algorithm by exploring the local search space around each solution.

4. Repeat steps 2 and 3 until a satisfactory solution is found or a stopping criterion is met.

This hybrid algorithm combines the strengths of each individual algorithm – the heuristic algorithm provides a good starting point, the genetic algorithm is good at exploring a

large search space and finding good solutions, and the local search algorithm is good at refining solutions to find the best possible solution. By combining these algorithms, the hybrid algorithm is able to find high-quality solutions in a reasonable amount of time.

Example 2

In this example, we first define the input variables (‘X‘, ‘Y‘, and ‘Z‘) and output variable (‘Reliability‘) of the fuzzy system. We then define the membership functions for each variable and the rules for the fuzzy system. We create a control system simulation object (‘fuzzy objective function‘) to run the fuzzy system and compute the output value (‘Reliability‘) given an input solution.

We then define a hybrid algorithm function (‘hybrid algorithm‘) that generates an initial solution and uses the fuzzy system to compute the fitness of the solution (i.e., the system reliability). The function returns the fitness of the solution, which can be used for further optimization (e.g., using a genetic algorithm or local search algorithm).

Finally, we test the hybrid algorithm by calling the‘hybrid algorithm‘ function and printing the output value.

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import random

# Define the input variables
X = ctrl.Antecedent(np.arange(0, 11, 1), 'X')
Y = ctrl.Antecedent(np.arange(0, 11, 1), 'Y')
Z = ctrl.Antecedent(np.arange(0, 11, 1), 'Z')

# Define the output variable
Reliability = ctrl.Consequent(np.arange(0, 101, 1), 'Reliability')
```

```
# Define the membership functions for the input variables
X['low'] = fuzz.trimf(X.universe, [0, 0, 5])
X['medium'] = fuzz.trimf(X.universe, [0, 5, 10])
X['high'] = fuzz.trimf(X.universe, [5, 10, 10])

Y['low'] = fuzz.trimf(Y.universe, [0, 0, 5])
Y['medium'] = fuzz.trimf(Y.universe, [0, 5, 10])
Y['high'] = fuzz.trimf(Y.universe, [5, 10, 10])

Z['low'] = fuzz.trimf(Z.universe, [0, 0, 5])
Z['medium'] = fuzz.trimf(Z.universe, [0, 5, 10])
Z['high'] = fuzz.trimf(Z.universe, [5, 10, 10])

# Define the membership functions for the output variable
Reliability['low'] = fuzz.trimf(Reliability.universe, [0, 0, 50])
Reliability['medium'] = fuzz.trimf(Reliability.universe, [0, 50, 100])
Reliability['high'] = fuzz.trimf(Reliability.universe, [50, 100, 100])

# Define the rules for the fuzzy system
rule1 = ctrl.Rule(X['low'] & Y['low'] & Z['low'], Reliability['low'])
rule2 = ctrl.Rule(X['medium'] & Y['medium'] & Z['medium'], Reliability['high'])
rule3 = ctrl.Rule(X['high'] & Y['high'] & Z['high'], Reliability['high'])

# Create the fuzzy system
system = ctrl.ControlSystem([rule1, rule2, rule3])
fuzzy_objective_function = ctrl.ControlSystemSimulation(system)

# Define the hybrid algorithm
```

```
def hybrid_algorithm():
    # Define the initial solution
    solution = [random.randint(0, 10), random.randint(0, 10), random.randint(0, 10)

    # Set the input values for the fuzzy system
    fuzzy_objective_function.input['X'] = solution[0]
    fuzzy_objective_function.input['Y'] = solution[1]
    fuzzy_objective_function.input['Z'] = solution[2]

    # Compute the output value for the fuzzy system
    fuzzy_objective_function.compute()

    # Return the output value as the fitness of the solution
    return fuzzy_objective_function.output['Reliability']

# Test the hybrid algorithm
print(hybrid_algorithm())
 # Output: a number between 0 and 100 representing the system reliability of the in
```

4.7 The fuzzy genetic algorithm [55, 81]

The fuzzy genetic algorithm will then continue to iterate, generating new chromosomes and evaluating their fitness. The best chromosomes will be selected to create new chromosomes, and the process will continue until the stopping criteria are met.

1. Initialize the population. The population is a set of chromosomes, where each chromosome represents a possible solution to the problem. The chromosomes are initialized randomly.

2. Evaluate the fitness of each chromosome. The fitness of a chromosome is a measure

of how good it is at solving the problem. The fitness function is a fuzzy function that takes into account the different factors that are important for the problem.

3. Select the parents. The parents are chromosomes that are selected from the population to be used to create new chromosomes. The parents are selected using a selection algorithm, such as roulette wheel selection or tournament selection.

4. Crossover. Crossover is a genetic operator that combines two chromosomes to create a new chromosome. The crossover point is a random position in the chromosome. The new chromosome is created by exchanging the genes between the two parents at the crossover point.

5. Mutation. Mutation is a genetic operator that randomly changes the genes in a chromosome. The mutation rate is a probability that a gene will be mutated.

6. Repeat steps 2-5 until the stopping criteria are met. The stopping criteria can be a maximum number of generations, a minimum fitness value, or a maximum time limit.

Here are some of the advantages of using fuzzy genetic algorithms:

1. They can be used to solve problems that are difficult to solve with traditional optimization methods.

2. They can handle uncertainty and imprecision.

3. They can be used to find multiple solutions to a problem.

Here are some of the disadvantages of using fuzzy genetic algorithms:

1. They can be computationally expensive.

2. They can be difficult to tune.

3. They may not find the optimal solution.

Example 3

In this example, we're using a fuzzy logic system to evaluate the fitness of each solution in the population. We're then using fuzzy logic to select the parents for the next generation based on their fitness. Finally, we're using fuzzy logic to perform crossover and mutation on the selected parents to create the next generation. Note that this is just one example of how to implement a fuzzy genetic algorithm. The specific implementation may vary depending on the problem you're trying to solve and the tools you're using

Example of Python code that implements a fuzzy genetic algorithm:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import random

# Define the input variables
X = ctrl.Antecedent(np.arange(0, 11, 1), 'X')
Y = ctrl.Antecedent(np.arange(0, 11, 1), 'Y')
Z = ctrl.Antecedent(np.arange(0, 11, 1), 'Z')

# Define the output variable
Reliability = ctrl.Consequent(np.arange(0, 101, 1), 'Reliability')

# Define the membership functions for the input variables
X['low'] = fuzz.trimf(X.universe, [0, 0, 5])
X['medium'] = fuzz.trimf(X.universe, [0, 5, 10])
X['high'] = fuzz.trimf(X.universe, [5, 10, 10])

Y['low'] = fuzz.trimf(Y.universe, [0, 0, 5])
```

```
Y['medium'] = fuzz.trimf(Y.universe, [0, 5, 10])
Y['high'] = fuzz.trimf(Y.universe, [5, 10, 10])

Z['low'] = fuzz.trimf(Z.universe, [0, 0, 5])
Z['medium'] = fuzz.trimf(Z.universe, [0, 5, 10])
Z['high'] = fuzz.trimf(Z.universe, [5, 10, 10])

# Define the membership functions for the output variable
Reliability['low'] = fuzz.trimf(Reliability.universe, [0, 0, 50])
Reliability['medium'] = fuzz.trimf(Reliability.universe, [0, 50, 100])
Reliability['high'] = fuzz.trimf(Reliability.universe, [50, 100, 100])

# Define the rules for the fuzzy system
rule1 = ctrl.Rule(X['low'] & Y['low'] & Z['low'], Reliability
['low'])
rule2 = ctrl.Rule(X['medium'] & Y['medium'] & Z['medium'],
Reliability['high'])
rule3 = ctrl.Rule(X['high'] & Y['high'] & Z['high'], Reliability['high'])

# Create the fuzzy system
system = ctrl.ControlSystem([rule1, rule2, rule3])

# Define the FGA parameters
population_size = 10
mutation_rate = 0.1
max_generations = 100

# Define the fitness function using the fuzzy system
```

```
def fitness(solution):
    # Set the input values for the fuzzy system
    system.input['X'] = solution[0]
    system.input['Y'] = solution[1]
    system.input['Z'] = solution[2]

    # Compute the output value for the fuzzy system
    system.compute()

    # Return the output value as the fitness of the solution
    return system.output['Reliability']

# Define the fuzzy mutation function
def fuzzy_mutation(solution):
    for i in range(len(solution)):
        # Compute the fuzzy mutation value using a triangular membership function
        mutation_value = fuzz.trimf(np.arange(0, 11, 1), [0,
        random.randint(0, 5), 10])[random.randint(0, 10)]

        # Apply the mutation with a certain probability
        if random.random() < mutation_rate:
            solution[i] = mutation_value

    return solution

# Define the fuzzy crossover function
def fuzzy_crossover(parent1, parent2):
    # Create a new child solution using a weighted average of the parents
```

```
    child = [parent1[i] * random.random() + parent2[i] *
     (1 - random.random()) for i in range(len(parent1))]

    # Apply the mutation to the child solution
    child = fuzzy_mutation(child)

    return child

# Initialize the population with random solutions
population = [[random.randint(0, 10), random.randint(0, 10),
 random.randint(0, 10)] for i in range(population_size)]

# Start the evolution loop
for generation in range(max_generations):
    # Evaluate the fitness of each individual in the population
    fitnesses = [fitness(solution) for solution in population]

    # Convert the fitness values to fuzzy sets
    fitness_fuzzy_sets = [fuzz.trimf(Reliability.universe,
     [max(0, f - 25), f, min(100, f + 25)]) for f in fitnesses]

    # Select the parents for the next generation using fuzzy logic
    parents = []
    for i in range(population_size):
        # Compute the fuzzy membership values for each individual
        membership_values = [fuzzy_sets[i].membership(fitnesses[i])
         for fuzzy_sets in fitness_fuzzy_sets]
```

```
        # Use the fuzzy membership values to select the parents
        parent_indices = np.random.choice(population_size, size=2, replace=False,
        p=membership_values)
        parents.append([population[index] for index in parent_indices])

    # Create the next generation by performing crossover and mutation
    next_generation = []
    for i in range(population_size):
        # Perform crossover between the selected parents
        child = fuzzy_crossover(parents[i][0], parents[i][1])

        # Add the child to the next generation
        next_generation.append(child)

    # Replace the old population with the new generation
    population = next_generation

    # Print the best solution found in this generation
    best_index = np.argmax(fitnesses)
    best_solution = population[best_index]
    best_fitness = fitnesses[best_index]
    print(f'Generation {generation}: Best solution = {best_solution},
     Best fitness = {best_fitness}')
```

.

CHAPTER 5

CONCLUSION

In this study, it is desirable to analyze the reliability of components in fuzzy systems with different membership functions.

1. The definition of fuzzy logic is briefly discussed up front, Due to its historical development and use. Previously, The achievement of system dependability for components with various failure probability density functions was not always simple.

2. Containing and integrating different PDFs was necessary, but also difficult. We consider these issues and explain the ambiguity and inaccuracy of data in the real world. some comments are provided on the problems encountered in integrating fuzzy logic into structural reliability assessment.

3. In order to compare the vague optimization method with the fuzzy optimization method, we obtained the solution of the numerical problem by fuzzy optimization.

4. The optimization scheme according to Zimmermann was superfluous and the superlative domino effect attained were leaf through and sold for appraisal with the present-day scholarship.

5. We have cautiously considered the preeminent perseverance achieved by the two established algorithms in Board VI Mandate, and in each case additionally compared them with the results gained by the fuzzy optimization means.The aim of the recent exploration is to arrange for an active algorithm for unclear optimization methods to find prime way out to multi-objective linear programming teething troubles.The advantage of this line of attack is that it provides resolution makers with a solution that is of wavering degrees of interest.Decision makers can choose the appropriate worst solution according to the prerequisites of the current situation.In addition, a comparison of our underperforming effects clearly shows that fuzzy optimization underperforms fuzzy optimization.The achieved results also explain why the fuzzy optimization system II with nonlinear true value and nonlinear false value gives better results than fuzzy optimization algorithm I using line association function and line non-membership gathering.

6. Fuzzy logic can be used effectively to solve complex optimization problems that involve imprecise and uncertain information. The objective function of system reliability optimization can be formulated in a fuzzy framework, taking into account various factors such as the failure rate, repair time, and cost of components. Fuzzy optimization techniques, such as fuzzy linear programming, can be used to find the optimal solution for the fuzzy objective function.

7. The optimal solution obtained from the fuzzy optimization approach provides a trade-off between system reliability, cost, and other performance measures. The results of the optimization can be used to make informed decisions about the design and maintenance of the system, and to improve the overall system reliability.

8. In summary, solving a fuzzy objective function for system reliability optimization is a powerful tool that can help engineers and decision-makers to improve the reliability and performance of complex systems.

FUTURE WORK

1. Integration of Machine Learning Techniques: One potential area of future work is the integration of machine learning techniques into the optimization process. This could involve using machine learning algorithms to learn the optimal values of the fuzzy parameters that define the objective function, or to identify patterns in the data that can be used to guide the optimization process.

2. Multi-Objective Optimization: Another area of future work is the extension of fuzzy optimization techniques to multi-objective optimization problems. In such problems, there are multiple conflicting objectives that need to be optimized simultaneously. Fuzzy optimization techniques can be used to find a set of solutions that are optimal with respect to all objectives.

3. Application to Real-World Problems: While fuzzy optimization has been successfully applied to a wide range of problems, there is still a need for more research on its application to real-world problems. Future work could involve the application of fuzzy optimization techniques to problems in areas such as engineering, finance, and healthcare.

4. Hybrid Approaches: Another area of future work is the development of hybrid approaches that combine fuzzy optimization techniques with other optimization techniques such as linear programming or nonlinear programming. This could lead to more efficient and effective optimization algorithms that can handle a wider range of problems.

5. Uncertainty Quantification: Finally, there is a need for more research on uncertainty quantification in fuzzy optimization. This involves quantifying the uncertainty in the optimization process and providing measures of confidence in the results. This is particularly important in applications where the results of the optimization process are used to make critical decisions.

REFERENCES

[1] Taofeek Afolabi and Hooman Farzaneh. Optimal design and operation of an off-grid hybrid renewable energy system in nigeria's rural residential area, using fuzzy logic and optimization techniques. *Sustainability*, 15(4):3862, 2023.

[2] Ahmed Al-Jilawi. *Solving the Semidefinite Programming Relaxation of Max-cut Using an Augmented Lagrangian Method*. PhD thesis, Northern Illinois University, 2019.

[3] Hussein Alahmer, Ali Alahmer, Malik I Alamayreh, Mohammad Alrbai, Raed Al-Rbaihat, Ahmed Al-Manea, and Razan Alkhazaleh. Optimal water addition in emulsion diesel fuel using machine learning and sea-horse optimizer to minimize exhaust pollutants from diesel engine. *Atmosphere*, 14(3):449, 2023.

[4] George J Anders, Alfredo Vaccaro, et al. *Innovations in power systems reliability*. Springer, 2011.

[5] Tadeusz Antczak. The exact absolute value penalty function method for identifying strict global minima of order m in nonconvex nonsmooth programming. *Optimization Letters*, 10(7):1561–1576, 2016.

[6] Ibrahim Aqel and Mohamed Arezki Mellal. Optimal reliability allocation of heterogeneous components in pharmaceutical production plant. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, pages 1–10, 2023.

[7] Kenneth J Arrow and Leonid Hurwicz. Reduction of constrained maxima to saddle-point problems. In *Traces and Emergence of Nonlinear Programming*, pages 61–80. Springer, 2014.

[8] Kenneth Joseph Arrow, Floyd J Gould, and Stephen Mills Howe. A general saddle point result for constrained optimizations. Technical report, North Carolina State University. Dept. of Statistics, 1971.

[9] Adil Bagirov, Napsu Karmitsa, and Marko M Mäkelä. *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer, 2014.

[10] Barnabás Bede and Sorin G Gal. Generalizations of the differentiability of fuzzy-number-valued functions with applications to fuzzy differential equations. *Fuzzy sets and systems*, 151(3):581–599, 2005.

[11] Alexandre Belloni. Lecture notes for iap 2005 course introduction to bundle methods. *Operation Research Center, MIT, Version of February*, 11, 2005.

[12] Dimitri Bertsekas. *Convex optimization theory*. Athena Scientific Belmont, 2009.

[13] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[14] Dimitri P Bertsekas and Athena Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[15] Piero P Bonissone, Raj Subbu, Neil Eklund, and Thomas R Kiehl. Evolutionary algorithms+ domain knowledge= real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280, 2006.

[16] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[17] Kai-Yuan Cai. *Introduction to fuzzy reliability*, volume 363. Springer Science & Business Media, 1996.

[18] Sampa ChauPattnaik, Mitrabinda Ray, and Mitalimadhusmita Nayak. Fuzzy set-based reliability estimation. *International Journal of Software Innovation (IJSI)*, 11(1):1–14, 2023.

[19] Alice Chiche and Jean Charles Gilbert. How the augmented lagrangian algorithm can deal with an infeasible convex quadratic optimization problem. *Journal of Convex Analysis (to appear).[pdf]*, 3(4):5, 2014.

[20] Andrew R Conn, Nick Gould, Annick Sartenaer, and Ph L Toint. Convergence properties of an augmented lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, 6(3):674–703, 1996.

[21] William Cook, WH Cunningham, WR Pulleyblank, and A Schrijver. Combinatorial optimization. *Oberwolfach Reports*, 5(4):2875–2942, 2009.

[22] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[23] Stephan Dempe and Jonathan F Bard. Bundle trust-region algorithm for bilinear bilevel programming. *Journal of Optimization Theory and Applications*, 110(2):265–288, 2001.

[24] P Dimitri Bertsekas. Nonlinear programming. 1999. *Athena, Scientific, Belmont CA*.

[25] Yu Du and Andrzej Ruszczyński. Rate of convergence of the bundle method. *Journal of Optimization Theory and Applications*, 173(3):908–922, 2017.

[26] Charles E Ebeling. *An introduction to reliability and maintainability engineering*. Waveland Press, 2019.

[27] Yohanna Ejieji. Nigerian journal of mathematics and applications v olume 24,(2015), 216- 227 c nig. j. math. appl. http://www. njmaman. com.

[28] Mahmoud Mahmoud El-Alem. *A global convergence theory for a class of trust region algorithms for constrained optimization*. PhD thesis, 1988.

[29] Benjamin Epstein and Ishay Weissman. *Mathematical models for systems reliability*. Crc Press, 2008.

[30] Farzad Esmaeili, Saeid Shabanlou, and Mohsen Saadat. Novel reliable model by integrating the adaptive neuro-fuzzy inference systems with wavelet transform and firefly algorithms for rainfall forecasting in the north of iran. *Applied Water Science*, 13(2):46, 2023.

[31] Stefan Feltenmark and Krzysztof C Kiwiel. Dual applications of proximal bundle methods, including lagrangian relaxation of nonconvex problems. *SIAM Journal on Optimization*, 10(3):697–721, 2000.

[32] Carlos Fernández González. *Métodos matemáticos en problemas de entrelazamiento: convertibilidad de estados, medidas conjuntas y halmiltonianos en PEPS*. PhD thesis, Universidad Complutense de Madrid, 2014.

[33] Ernst G Frankel. *Systems reliability and risk analysis*, volume 1. Springer Science & Business Media, 2012.

[34] Michael R Garey and David S Johnson. Computers and intractability, vol. 29, 2002.

[35] Harish Garg, Monica Rani, SP Sharma, and Yashi Vishwakarma. Intuitionistic fuzzy optimization technique for solving multi-objective reliability optimization problems in interval environment. *Expert Systems with Applications*, 41(7):3157–3167, 2014.

[36] Harish Garg and SP Sharma. Multi-objective reliability-redundancy allocation problem using particle swarm optimization. *Computers & Industrial Engineering*, 64(1):247–255, 2013.

[37] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[38] Nand Gopal and Dilbagh Panchal. A structured framework for performance optimization using jblto, fcopras and fcodas methodologies. *Journal of Quality in Maintenance Engineering*, 29(1):220–248, 2023.

[39] Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[40] JB Hiriart-Urruty and C Lemaréchal. Convex analysis and minimization algorithms. no. 305-306 in grund. der math. wiss, 1993.

[41] Kaoru Hirota. *Industrial applications of fuzzy technology*. Springer Science & Business Media, 2012.

[42] Minhui Huang, Shiqian Ma, and Lifeng Lai. A riemannian block coordinate descent method for computing the projection robust wasserstein distance. In *International Conference on Machine Learning*, pages 4446–4455. PMLR, 2021.

[43] Ching-Lai Hwang, Kwangsun Yoon, Ching-Lai Hwang, and Kwangsun Yoon. Methods for multiple attribute decision making. *Multiple attribute decision making: methods and applications a state-of-the-art survey*, pages 58–191, 1981.

[44] Abraham Kandel. *Fuzzy mathematical techniques with applications*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[45] Christian Kanzow, Daniel Steck, and Daniel Wachsmuth. An augmented lagrangian method for optimization problems in banach spaces. *SIAM Journal on Control and Optimization*, 56(1):272–291, 2018.

[46] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[47] Peter J King and Ehmat H Mamdani. The application of fuzzy control systems to industrial processes. *Automatica*, 13(3):235–242, 1977.

[48] RP King. Necessary and sufficient conditions for inequality constrained extreme values. *Industrial & Engineering Chemistry Fundamentals*, 5(4):484–489, 1966.

[49] Krzysztof C Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical programming*, 46(1-3):105–122, 1990.

[50] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):1–23, 2017.

[51] Anand Kulkarni, Ganesh Krishnasamy, and Ajith Abraham. *Introduction to Optimization*, volume 114, pages 1–7. 09 2017.

[52] Way Kuo and V Rajendra Prasad. An annotated overview of system-reliability optimization. *IEEE Transactions on reliability*, 49(2):176–187, 2000.

[53] Claude Lemarechal. An extension of davidon methods to non differentiable problems. In *Nondifferentiable optimization*, pages 95–109. Springer, 1975.

[54] Claude Lemarechal. Methods of descent for nondifferentiable optimization (krzysztof c. kiwiel). *SIAM Review*, 30(1):146, 1988.

[55] Sven Leyffer and Charlie Vanaret. Augmented lagrangian filter method. 2016.

[56] Zeshang Li, Lei Wang, and Tangqi Lv. Additive manufacturing-oriented concurrent robust topology optimization considering size control. *International Journal of Mechanical Sciences*, page 108269, 2023.

[57] Bruno F Lourenço, Ellen H Fukuda, and Masao Fukushima. Optimality conditions for nonlinear semidefinite programming via squared slack variables. *Mathematical Programming*, 168(1-2):177–200, 2018.

[58] Michael James Lowe. *Nonlinear programming: augmented Lagrangian techniques for constrained minimization*. PhD thesis, Montana State University-Bozeman, College of Engineering, 1974.

[59] Marko Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization methods and software*, 17(1):1–29, 2002.

[60] Jérôme Malick. The spherical constraint in boolean quadratic programs. *Journal of Global Optimization*, 39:609–622, 2007.

[61] Santiago Martinez-Boggio, Davide Di Blasio, Tom Fletcher, Richard Burke, Antonio García, and Javier Monsalve-Serrano. Optimization of the air loop system in a hydrogen fuel cell for vehicle application. *Energy Conversion and Management*, 283:116911, 2023.

[62] Mohamed Arezki Mellal, Sameer Al-Dahidi, Rajkumar Bhimgonda Patil, Basavraj S Kothavale, and Rajendra S Powar. System reliability-redundancy optimization with high-level of subsystems. *Materials Today: Proceedings*, 77:627–630, 2023.

[63] Debiao Meng, Shiyuan Yang, Abílio MP de Jesus, and Shun-Peng Zhu. A novel kriging-model-assisted reliability-based multidisciplinary design optimization

strategy and its application in the offshore wind turbine tower. *Renewable Energy*, 203:407–420, 2023.

[64] Angelo Miele, EE Cragg, RR Iyer, and AV Levy. Use of the augmented penalty function in mathematical programming problems, part 1. *Journal of Optimization Theory and Applications*, 8(2):115–130, 1971.

[65] Constantin Virgil Negoiţă and Dan A Ralescu. *Applications of fuzzy sets to systems analysis*. Springer, 1975.

[66] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[67] Reza Ghasemi Pirbalouti, Mohammadreza Karimi Dehkordi, Javad Mohammadpour, Esmaeil Zarei, and Mohammad Yazdi. An advanced framework for leakage risk assessment of hydrogen refueling stations using interval-valued spherical fuzzy sets (iv-sfs). *International Journal of Hydrogen Energy*, 2023.

[68] Nikolaos Ploskas, Nikolaos Samaras, et al. *Linear Programming Using MATLAB®*, volume 127. Springer, 2017.

[69] Michael JD Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969.

[70] Michael JD Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, pages 144–157. Springer, 1978.

[71] Pierre Ramond. Dual theory for free fermions. *Physical Review D*, 3(10):2415, 1971.

[72] Marvin Rausand and Arnljot Hoyland. *System reliability theory: models, statistical methods, and applications*, volume 396. John Wiley & Sons, 2003.

[73] V Ravi, BSN Murty, and J Reddy. Nonequilibrium simulated-annealing algorithm applied to reliability optimization of complex systems. *IEEE Transactions on Reliability*, 46(2):233–239, 1997.

[74] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121:307–335, 2010.

[75] R Tyrell Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and applications*, 12(6):555–562, 1973.

[76] R Tyrrell Rockafellar. *Convex analysis*. Number 28. Princeton university press, 1970.

[77] R Tyrrell Rockafellar. Penalty methods and augmented lagrangians in nonlinear programming. In *IFIP Technical Conference on Optimization Techniques*, pages 418–425. Springer, 1973.

[78] R Tyrrell Rockafellar. Augmented lagrange multiplier functions and duality in nonconvex programming. *SIAM Journal on Control*, 12(2):268–285, 1974.

[79] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[80] RT Rockafellar. New applications of duality in convex programming. In *Proceedings of the 4th Conference of Probability, Brasov, Romania*, pages 73–81, 1971.

[81] Judah Ben Rosen, Olvi L Mangasarian, and Klaus Ritter. *Nonlinear Programming: Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin, Madison, May 4-6, 1970*. Number 25. Elsevier, 2014.

[82] Bernard Roy. Classement et choix en présence de points de vue multiples. *Revue française d'informatique et de recherche opérationnelle*, 2(8):57–75, 1968.

[83] Stephan Russenschuck. Mathematical optimization techniques. Technical report, CERN, 1999.

[84] Andrzej Ruszczynski. *Nonlinear optimization*. Princeton university press, 2011.

[85] Thomas L Saaty. â€ the analytic hierarchy processâ€™ mcgraw-hill. *New York*, 1980.

[86] Aman Sharma and Rinkle Rani. A systematic review of applications of machine learning in cancer prediction and diagnosis. *Archives of Computational Methods in Engineering*, 28(7):4875–4896, 2021.

[87] Chanan Singh and Roy Billinton. *System reliability, modelling and evaluation*, volume 769. Hutchinson London, 1977.

[88] Toshiro Terano, Kiyoji Asai, and Michio Sugeno. *Fuzzy systems theory and its applications*. Academic Press Professional, Inc., 1992.

[89] Fellar Tyrrell. Convex analysis, 1970.

[90] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[91] Angelika Wiegele. Biq Mac Library—A collection of Max-Cut and quadratic 0-1 programming instances of medium size. Technical report, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria, 2007.

[92] Angelika Wiegele. Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. *Preprint*, 2007.

[93] Philip Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. In *Nondifferentiable optimization*, pages 145–173. Springer, 1975.

[94] Xin-She Yang. *Optimization Algorithms*, volume 356, pages 13–31. 01 1970.

[95] He Yao, Cunbao Zhao, Pengyu Chen, Yue Zhang, and Shengnan Zhao. A truncated reliability analysis method with the fuzzy boundary. In *Structures*, volume 48, pages 1808–1816. Elsevier, 2023.

[96] Abbas Yazdinejad, Ali Dehghantanha, Reza M Parizi, and Gregory Epiphaniou. An optimized fuzzy deep learning model for data classification based on nsga-ii. *Neurocomputing*, 522:116–128, 2023.

[97] Changjun Yu. *A study of optimization and optimal control computation: exact penalty function approach*. PhD thesis, Curtin University, 2012.

[98] Lotfi A Zadeh. Fuzzy sets, information and control, 8: 338-353. *MathSciNet zbMATH*, 1965.

[99] Xiaopeng Zhao and Jen-Chih Yao. Linear convergence of a nonmonotone projected gradient method for multiobjective optimization. *Journal of Global Optimization*, pages 1–18, 2021.

[100] H-J Zimmermann. Fuzzy set theory. *Wiley interdisciplinary reviews: computational statistics*, 2(3):317–332, 2010.

[101] Hans-Jürgen Zimmermann. *Fuzzy set theory—and its applications*. Springer Science & Business Media, 2011.

[102] Hans-Jürgen Zimmermann. *Practical applications of fuzzy technologies*, volume 6. Springer Science & Business Media, 2012.

جمهورية العراق
وزاره التعليم الهالي & والبحث العلمي
جامعة بابل
كلية التربية للعلوم الصرفة
قسم الرياضيات

حل مسائل الامثلية المعولية الضبابية باستعمال البرمجة اللخطية الضبابية

.

اطروحة مقدمة
,الى مجلس كلية التربية للعلوم الصرفة
جامعة بابل / كلية التربية
كجزء من متطابات]دكتوراه فلسفة في التربية / الرياضيات
من قبل احمد عبد الحسين جبار حمود
باشراف ا.د. عدي صبري عبد الرزاق

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

المستخلص

كل من طريقة آدم وطريقة التدرج هي خوارزميات التحسين المستخدمة لحل مشاكل تحسين القيد. ومع ذلك ، هناك بعض الاختلافات الرئيسية بين هاتين الطريقتين.

طريقة التدرج هي خوارزمية تحسين من الدرجة الأولى تقوم بتحديث المعلمات من خلال اتخاذ خطوات صغيرة في اتجاه التدرج السلبي لوظيفة الهدف. يتمثل العيب الرئيسي لطريقة التدرج اللوني في أنها قد تتعثر في الحدود الدنيا المحلية ، وقد يستغرق الأمر وقتًا طويلاً لتتقارب مع الحد الأدنى العالمي.

من ناحية أخرى ، فإن طريقة آدم هي خوارزمية تحسين من الدرجة الثانية تجمع بين مزايا طريقة التدرج والنهج القائم على الزخم. يستخدم آدم معدلات التعلم التكيفية ، مما يسمح لها بالتقارب بشكل أسرع من طريقة التدرج ، ومن غير المرجح أن تتعثر في الحدود الدنيا المحلية.

من حيث معدل التقارب ، فإن طريقة آدم بشكل عام أسرع من طريقة التدرج ، خاصة بالنسبة للمشاكل ذات الأبعاد العالية أو البيانات الصاخبة. ومع ذلك ، يمكن أن يكون آدم حساسًا لاختيار المعلمات الفائقة ، وقد يتقارب مع الحلول دون المستوى الأمثل إذا لم يتم ضبط المعلمات الفائقة بشكل صحيحباختصار ، في حين أن كلتا الطريقتين مفيدتان لحل مشاكل تحسين القيد ، قد تكون طريقة آدم أكثر فاعلية للمشكلات ذات البيانات عالية الأبعاد أو الصاخبة ، بينما قد تكون طريقة التدرج أكثر ملاءمة للمشكلات الأبسط. يعتمد اختيار الطريقة في النهاية على المشكلة المحددة والموارد الحسابية المتاحة.تواجه الهندسة تحديًا بالغ الأهمية في تحسين موثوقية النظام ،

والذي يهدف إلى تعظيم إمكانات النظام للعمل بشكل صحيح طوال عمره الافتراضي. ومع ذلك ، في مواقف العالم الحقيقي ، قد يكون هدف الموثوقية الدقيق غير مؤكد وغير واضح. نتيجة لذلك ، تم تقديم نهج جديد في هذه الرسالة لمعالجة تحسين وظيفة الهدف الضبابي لموثوقية النظام باستخدام خوارزمية هجينة. تجمع هذه الخوارزمية بين اثنين من تقنيات التحسين التلوي ، تحسين سرب الجسيمات (عصو) والتليين المحاكي (صا) ، للاستفادة من مزايا كل منهما. يتم استخدام عصو للبحث عن أفضل أداء عالمي للوظيفة الموضوعية ، بينما يتم استخدام صا للتحرر من اوبتم المحلي.

يتم تقييم فعالية الخوارزمية المقترحة على نظامين معياريين ، وتثبت النتائج تفوقها على الخوارزميات الأخرى الموجودة. يمكن تطبيق الطريقة المقترحة في مجالات مختلفة ، بما في ذلك تصميم النظام وجدولة الصيانة وتعزيز الموثوقية لتحسين موثوقية النظام في بيئة ضبابية.تجديد الاستجابة. نحن نتعامل مع موضوع تحسين موثوقية النظام المعقد نظرًا لأن مشكلة التحسين الغامض متعددة الأهداف ، يتم أيضًا تضمين موثوقية النظام والتكلفة والحجم والوزن ، كلها أهداف غامضة. باستخدام أمثلة عملية ، تحسب دراستنا تحسينات الموثوقية للأنظمة المعقدة مع قيود التكلفة والوزن. ولتحديث إجراء حسابي لحل مشكلة البرمجة متعددة الأهداف المبنية عن طريق تقنية تحسين غير واضحة. كما أنه يغطي بعض المقتنيات الهامة لمجموعة غير واضحة ، صحية مثل العمليات عليها. يعتمد نمو الإجراء على أفضل كود مجموعة الاختيار ، والذي يتم الحصول عليه من خلال مجموعات الاختيار الغامضة المتباينة التي تم الحصول عليها لكل محرك موضوعي. أيضًا ، نظرًا لأن طريقة التحسين غير الواضحة تستخدم درجات من الملاءمة وعدم الملاءمة ، فقد قمنا بعمل مدرسي نسبي من وظائف التلميح الخطي وغير الخطي للتركيب وعدم الانتماء لإدراك كيفية لمس التحسين واكتساب الفهم في مثل هذه العملية. لتوضيح الإجراء المتقدم ، تم توفير مثيل رقمي.