

**Computational impact of Mixture Distribution
on multi-modal continuous functions using Real-
coded Genetic Algorithms**



by

Anum Touqeer Shah

Reg.No. 187-FBAS/MSST/F21

**Department of Mathematics and Statistics
Faculty of Sciences
International Islamic University, Islamabad
2023**

TH-2704/ Vh

MS
519.6
ANC

Mathematical optimization

Genetic algorithms

Multimodal functions (Statistics)

Numerical analysis

Computational mathematics

**Computational impact of Mixture Distribution
on multi-modal continuous functions using Real-
coded Genetic Algorithms**



by

Anum Touqeer Shah

Rg.No. 187-FBAS/MSST/F21

Supervised by

Dr . Ehtasham ul Haq

Department of Mathematics and Statistics

Faculty of Sciences

International Islamic University, Islamabad

2023

**Computational Impact of Mixture Distribution
on multi-modal continuous functions using Real-
coded Genetic Algorithms**

By

Anum Touqeer Shah

Reg. No. 187-FBAS/MSST/F21

A thesis

Submitted to the Department of Mathematics & Statistics

International Islamic University Islamabad

In fulfillment of the requirements for the degree of

MASTER IN SCIENCES

in

STATISTICS

Supervised by

Dr . Ehtasham ul Haq

Department of Mathematics and Statistics

Faculty of Sciences

International Islamic University, Islamabad

2023

Certificate

Computational Impact of Mixture Distribution on Multi-Modal Continuous Functions using Real-coded Genetic Algorithms

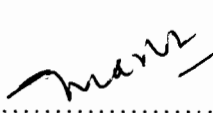
By

Anum Touqeer Shah

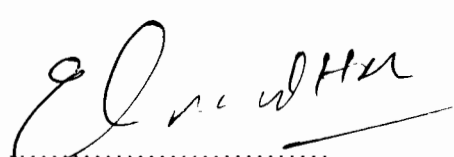
Reg.No.187-FBAS/MSST/F2¹

A THESIS SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF THE MASTER OF SCIENCE IN STATISTICS

We accept this as conforming to the required standard.

1. 
.....

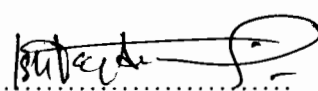
Prof Dr. Nasir Ali
(Chairperson)

2. 
.....

Dr. Ehtasham ul Haq
(Supervisor)

3. 
.....

Dr. Shabbir Ahmad
(External Examiner)

4. 
.....

Dr. Ishfaq Ahmad
(Internal Examiner)

Department of Mathematics and Statistics

Faculty of Sciences

International Islamic University, Islamabad

2023

Declaration

I hereby declare that the dissertation titled **Computational impact of Mixture Distribution on multimodal continuous functions using Real-coded Genetic Algorithms**, neither as a whole nor as a part there of has been copied out from any source. It is further declared that I have prepared this thesis entirely on the basis of my personal effort made under the sincere guidance of my supervisor. No portion of the work presented in this thesis has been submitted in the support of any application for any degree or qualification of this or any other institute of learning.

Signature: _____

Anum Touqeer Shah

MS-Statistics

Reg No. 187-FBAS/MSST/F21

Department of Mathematics and Statistics

Faculty of Sciences

International Islamic University, Islamabad

Dedication

I humbly dedicated this effort to my loving parents Mr.& Mrs Syed Touqeer Hussain Shah for their endless guidance, support raising me to believe that anything is possible.

Along with all the respected teachers and my husband ,sisters and brother for their love, couragement, support and affection that make me able to get the success.

Above all, to **Almighty Allah** who always give me strength ,Knowledge and Wisdom in everything I do.

Thesis Completion Certificate

The Thesis entitled **Computational impact of Mixture Distribution on multimodal continuous functions using Real-coded Genetic Algorithms** the research work of MS degree in Statistics submitted by Anum Touqeer Shah , 187-FBAS/MSST/F-21 in partial fulfillment of MS degree in Statistics has been completed under my guidance and supervision. I am satisfied with the quality of her work and allow her to submit this thesis for further process o graduate with Master of Sciences from Deparment of Mathematics and Statistics, as per IIUI rules and regulation.

Dr . Ehtasham ul Haq

Department of Mathematics and Statistics

Faculty of sciences

International Islamic University, Islamabad, Pakistan

Date _____

Acknowledgement

First and foremost, I would like to thank Allah for being with me at every moment for giving me the strength and the will to succeed, for being my support and sole purpose in life.

I would like to thank my supervisor Dr. Ehtasham-ul-haq for his innovative guidance, dedication knowledge, tremendous patience, constructive suggestion and support throughout this reach and writing of this thesis. His inside and words of encouragement have often inspired and renewed my hopes for completing my MS Statistics research. I couldn't have taken this project forward without him.

I am very much thankful to respected teacher of MS Statistics Dr. Farzana Abbasi, Dr. Maryam Saddiqa, and Dr. Shamim Alam innovative guidance, dedication knowledge giving me.

I would like to thanks my father late Syed Touqeer Hussain Shah and mother Parveen Akhtar for their love and support during my studies and for enriching my thought. Special thanks to my brother Syed Touseef Hussain Shah and my husband Syed Kamran Ajmal for their big support and four sisters Asia, Shabnam, Sadaf and Sobia for their encouragement towards MS Statistics..

Anum Touqeer Shah

Table of Contents

- 1 Introduction**
 - 1.1 Introduction 1
 - 1.2 Evolutionary Algorithms..... 2
 - 1.3 Genetic Algorithms.....2
 - 1.3.1 The Selection Operators..... 5
 - 1.3.1.1 Tournament selection (TS)..... 5
 - 1.3.2 The Crossover Operators..... 5
 - 1.3.2.1 Mechanism for crossover..... 5
 - 1.3.2.2 Crossover with the binary string..... 6
 - 1.3.2.3 Path-based crossover (Permutation-based string)..... 6
 - 1.3.2.4 The real coded crossover..... 7
 - 1.3.3 The Mutation Operators 7
 - 1.3.3.1 Machanism for mutation8
 - 1.4 Parameters in genetic algorithms9
 - 1.4.1 Population size 9
 - 1.4.2 Generation gap..... 10
 - 1.5 Benchmarks function..... 10
 - 1.6 Objective of the study.....11
- 2 Literature Review13**
- 3 Materials And Methodology17**
 - 3.1 Tournament seslection (TS)17
 - 3.2 Crossover Operators17
 - 3.2.1 Double Pareto Crossover 17

3.2.2 Laplace Crossover	18
3.2.3 Simulated binary Crossover	19
3.3 Mutation Operators	20
3.3.1 Power Mutation	20
3.3.2 Makinen Periaux And Toivanen Mutation	21
3.3.3 Non-Uniform Mutation	21
3.4 The Proposed Probability Distribution	22
3.4.1 Exponential Crossover Operator	22
3.4.2 Laplace Crossover Operator	24
3.4.3 The Fisk Crossover	26
3.4.4 Gompertz Crossover Operator	28
3.4.5 Weibull Crossover Operator	30
3.4.6 Double Pareto Crossover Operator	32
3.4.7 Rayleigh Crossover Operator	33
3.4.8 Lindley Crossover Operator	35
3.5 Mixture Distribution Based Real-Coded Crossover.....	37
3.5.1 Background.....	37
3.5.2 Newly Suggested Mixture Distribution Based Real-coded.....	37
Crossover Operator	
3.5.2.1 The Proposed Mixture of Lindley Probability Distribution.....	37
3.5.2.2 The Proposed Mixture of Rayleigh Probability Distribution.....	39
3.6 Benchmark Functions	40

4 Results And Discussion

4.1 Experimental setup.....43

4.2 Simulation study.....44

4.3 Experimental result and Comparison.....45

5 Conclusion64

References65

List of Abbreviations

2PX	2-Point Crossover
BTS	Binary Tournament Selection
DPX	Double Pareto Crossover
EX	Exponential Crossover
FX	Fix crossover
GAs	Genetic Algorithms
GX	Gompertz Crossover
LX	Laplace Crossover
LDX	Lindely Crossover
MDX	Mixture Distribution Crossover
MIX-LDX	Mixture Of Lindely Crossover
MIX-RHX	Mixture of Rayleigh Crossover
MPTM	Makinen, Periaux and Toivanen
NUM	Non-uniform Mutation
PI	Performance Index
RHX	Rayleigh Crossover
SBX	Simulated Binary Crossover
SD	Standard Deviation
SRS	Stochastic Remainder Selection
TS	Tournament Selection
WBX	Weibull Crossover

Abstract

Genetic algorithms (GAs) are stochastic based heuristic search methods which are inspired by the Darwin's theory of evolution. This random search optimization technique using three main operators that are selection, crossover and mutation. These operators aid in obtaining the best solutions for both constraint and unconstraint optimization problems. In the current thesis, a new aspect of genetically focused research is being pursued that offers comprehensive operating theory for multi-offspring crossover operators. In the context of real-coded crossover, we will now put a greater emphasis on how effectively designed crossover operators and can improve the performance of the GA process. Currently, the probability based recoded crossover operators included Exponential, Laplace, Fisk, Gompertz, Weibull, Double pareto, Rayleigh, Lindely are used in the study. The performance of these operators are assessed using a number of well-known benchmark functions. Moreover, some real-coded crossover operators are also presented with unique idea of mixture distribution including Rayleigh and Lindely probability distributions. These probability distributions are co-integrated with a variety of parameters. Overall, simulated results and performance indices will show that the new crossover procedures are quite effective for obtaining optimal solutions. As a result, the proposed genetic operators are built to utilize the right qualities of optimization theory, which direct the GA process for the convergence to a practicable solutions. This thesis is composed of five chapters.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The technique of optimization spans various fields of study and finds relevance in situations where choosing the best among given alternatives is crucial. These diverse areas encompass engineering, management science, medicine, computer science, applied mathematics, and more. Naturally, each field perceives optimization from its unique standpoint, but the ultimate objective remains the same – making an optimal decision. Due to its wide-ranging applicability in different disciplines, a concise single definition of optimization becomes challenging to establish. For instance, mathematicians seek to determine the maximum or minimum of a real function within a defined set of variables. In the realms of computing and engineering, the aim is to maximize system or application performance while minimizing runtime and resource consumption as much as possible. (Chu & Beasley, 1995).

The best solutions are frequently needed in various real-life decision-making situations. Any engineering, scientific, economic, or financial problems could be one of these. When the quality of potential answers can be objectively characterized, finding a superior and occasionally optimal response via an algorithm may be possible. In this situation, choices are made by building optimization models that capture the essence of the issue and then resolving those using mathematical techniques. Constrained optimization problems thus form the core of the most basic optimization scenario, despite the fact that the current research study only takes into account unconstrained optimization problems. Mathematically, an unconstrained nonlinear optimization problem is:

$$\text{Min } f(y), \text{ where } f: R^n \rightarrow R,$$

$$\text{and } S = \{y \in P^n \mid g_i(y) \leq 0, h_i(y) = 0, i = 1, 2, \dots, k; j = 1, 2, \dots, m; m \leq n\}$$

where the function “ f ” is the objective function and more precisely known as the cost function and $y \in S$, the elements of set S are the feasible solutions. A feasible solution, $y \in S$ which maximizing/minimizing the objective function according to the nature of the problem. Hence, the f is known as the optimal solution. Generally, the

basic aim of the optimization is to find the most suitable values of the decision variables which minimize or maximize objective function concerning some given restrictions.

The gradient information is always employed by optimization strategies for unconstrained problems to find the optima. As a result, the objective functions with non-differentiable components can be solved using the gradient-based optimization technique known as proximal gradient descent. Due to their inability to handle discrete variables effectively, these strategies are most likely stalled at a local optimum for multimodal objective functions. Since the local optima are frequently attained at the global optima, gradient-based approaches are frequently assured of this.

1.2 Evolutionary Algorithms

A population-based stochastic optimization approach is known as an evolutionary algorithm (EA). This dissertation accepts the idea that EAs and Evolutionary Computation (EC) algorithms are interchangeable terms. Metaheuristic algorithms or simply metaheuristics have also been used to describe these algorithms. Evolutionary optimization is the practice of using EC algorithms for optimization.

In the initial stage, the goal of Evolutionary Algorithms is to connect the algorithms to the real-life issue. This establishes a link between the ecology of the original issue and the area of problem-solving where progress takes place. This process is described as the representation and storage of potential solutions in a fashion that allows for computer manipulation. They can typically be expressed as binary codes consisting of 0s and 1s. The encoding technique that encodes potential search candidates must be carefully chosen, and each chromosome is represented by a vector whose length depends on the number of choice variables that define the search space's dimensions.

1.3 Genetic Algorithms

Population-based metaheuristic optimization techniques called genetic algorithms (GAs) examine randomly selected individuals and chromosomes across the search space. John Holland developed the fundamental concept of genetic algorithms in the 1960s, drawing inspiration from Darwin's notion of evolution's "survival of the fittest" (Holland, 1992). These algorithmic techniques work well for resolving issues with and without restrictions in optimization in a search space that has all conceivable solutions. Strong, workable solutions will be passed down to the next generation, but ineffective ones will be abandoned. These workable answers are assessed based on their fitness ratings and are referred to as the population of individuals. Strong, workable answers will be passed down to the following poor solutions and will be replaced in the

following generation. To give fitness values and initialize the starting point in the search space is the most crucial component of the genetic algorithm framework (Sivanandam & Deepa, 2008). Before going into further detail regarding GAs, there are a few key terms that are frequently used in the literature on GAs. Genes are a set of parameters that define an individual. By connecting genes, a chromosome is produced (solution). A genetic algorithm represents a person's genetic makeup as a string or an alphabet. The most common type of value is binary (a string of 1s and 0s). The closeness of the chromosomes from the specified solution is summed together and measured using the fitness value. Each of these chromosomes would undergo a fitness test based on the fitness function after producing the solutions represented by the chromosomes.

GA always attempts to produce better people by causing the population to develop from an initial randomly produced population. By exchanging information with one another to create new individuals or by altering existing ones, the evolution process is finished. The people who exchange information are known as parents, and the new people created as a result of the information exchange are known as children (offspring). As shown in Figure 1.1, GA is an iterative process with three primary operators: crossover, mutation, and selection.

The GA algorithm will keep running until the termination requirement is satisfied. To create a robust search space, the theoretical foundation of GA is based on an artificial evolutionary process. Initializing the population and finding potential answers to the challenges at hand are the first steps in the algorithmic process. A collection of genes, referred to as chromosomes, serves as the population's initialization, and each gene holds all of the properties of the data set. It is crucial to choose the right objective function because each chromosome has a fitness value that depends on the complexity of the issue. Using genetic operators like selection, crossover, and mutation, GAs evolve the population.

The primary goal of the choice operator is to raise the standard of answers by preferring the most suited chromosomes and preventing those with low fitness values. The mating pool will then be chosen from these chromosomes. The crossover operator mixes the solutions' chosen components during the recombination phase to create new offspring. It is the process of genetic material being transferred between two chromosomes that were chosen during the reproductive stage. In the meantime, the mutation process seeks to promote variety in the new population with a very low likelihood (Haq et al., 2019a).

Through the generations, genetic operators can preserve genetic diversity. Genetic variety or genetic variability is essential to the evolutionary process.

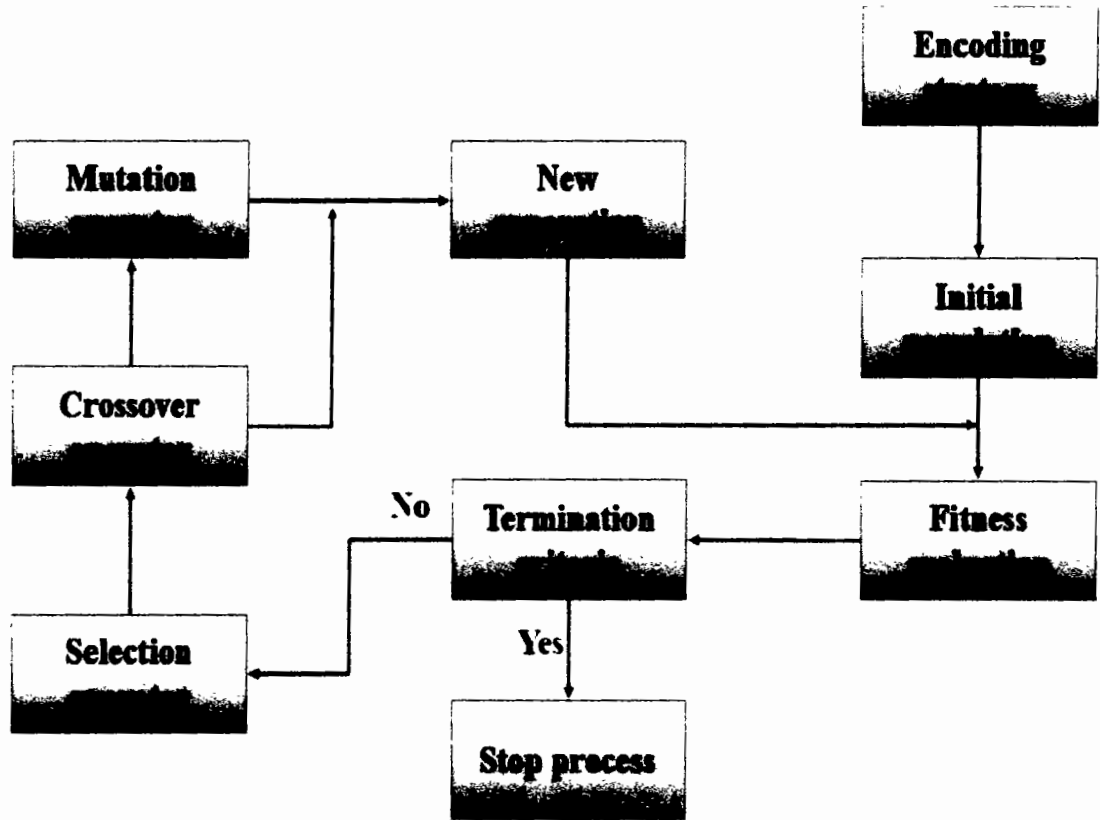


Figure 1.1: Flowchart of the basic steps in GA (Haq et al., 2019)

In all evolutionary algorithms, GA is the best algorithmic technique because it strikes the right balance between two opposing metrics for performance: exploitation (selection pressure) and exploration (population diversity). Exploitation is the act of fast getting to the best result, while exploration is the capacity of an algorithm to look up or investigate each component of the available lookup field. The GA performs better when exploration and exploitation are balanced properly.

1.3.1 The Selection Operators:

Selection for reproduction and selection for survival are two separate groups of selection operators in the EA. It happens frequently that the operator being mentioned depends on the context of the conversation. The evolutionary theories of biology provide a foundation for both operators.

The method of selecting the next generation from populations of parent and child individuals is called survival selection. According to the conventional generational theory, the parent population will be entirely replaced by the kid population.

1.3.1.1 Tournament Selection (TS)

The TS method is an easy-to-use ranking-based selection process. For the binary tournament selection, two individuals are picked at random, and the winning chromosome is then selected for the mating pool based on having the highest fitness value (Haq et al., 2019a). The weakest chromosomes, on the other hand, will not be able to mate because of their low fitness value when compared to a predetermined selection probability, p_i . Therefore, Equation 3.1 gives the predetermined selection probability " p_i " of the (i^{th}) sorted individual:

$$p_i = \frac{1}{Z^t} ((Z - i + 1)^t - (Z - i)^t); \quad i \in \{1, 2, \dots, Z\} \quad (1.3)$$

where t is the size of the competition and Z is the population size. $t=2$ and $t>2$ are the conditions for the binary tournament. If more than two people are wanted, the TS can also be expanded (Sivanandam & Deepa, 2008).

1.3.2 The Crossover Operators:

The most complicated EA operator, crossover, is also occasionally referred to as the recombination operator in the literature. A reproduction selection process that selects at least two parent strings is necessary for crossover. It draws inspiration from the genetic recombination seen during the reproduction process known as meiosis. Keep in mind that there are significant differences between the specifics of biological crossover between diploid organisms and the crossover operator of the EA. The fundamental concept is to somehow integrate the genetic information from two or more parents to roughly imitate the consequences of breeding in wild populations.

1.3.2.1 Mechanism for crossover

Crossover is the process of recombining chosen people to produce new offspring that share some distinguishing characteristics with their parents. The genotype's characteristics and how well the parents pass on their traits to the offspring determine the crossover technique's operation. Therefore, Parkinson (2004) investigated how recombination passed inherited traits from parents to offspring.

1.3.2.2 Crossover with the binary string

The most straightforward way to represent a chromosome in GA is by binary encoding. The following crossover operators with binary string (Haq et al., 2019c) are thus broadly applicable to address a variety of real-world issues:

- One-point crossover (*see Figure 1.2*)
- Two-point crossover
- Multi-point crossover (*see Figure 1.3*)
- Uniform crossover etc.

1.3.2.3 Path-based crossover (Permutation-based string)

A path-based crossover is frequently employed in algorithmic studies to effectively resolve combinatory-type difficulties. Permutation-based crossover is most famously recognized for its framework, which guarantees that each individual's tracks have a valid permutation (Haq et al., 2019c). The following are a few examples of path-based crossover operators:

- Partially-mapped crossover
- Cycle crossover
- Order crossover

Parent 1	0	1	0	1	0	0	1	1
Parent 2	1	0	1	0	1	1	0	0
Offspring 1	0	1	0	1	0	1	0	0
Offspring 2	1	0	1	0	1	0	1	1

Figure 1.2: Visual description of one-point crossover

Parent 1	0	1	0	1	0	0	1	1
Parent 2	1	0	1	0	1	1	0	0
Offspring1	0	0	0	0	0	1	1	1
Offspring 2	1	1	1	1	1	0	0	0

Figure 1.3: Visual description of multi-point crossover

1.3.2.4 Real-coded crossover

Real-coded genetic algorithms are a basic idea that Lucasius and Kateman (1989) put into the context of evolutionary optimization approaches. According to Yu and Kuang (2010), Wang et al. (2011), and Haq et al. (2020), real-coded genetic algorithms provide many benefits, such as appropriate precision, no requirement for Encoding, and a thorough random space search, low-cost computing, quick convergence, and enclosed danger of becoming trapped at local rather than universal optimums. Here are a couple of crossover operators with genuine coding.

- Arithmetic crossover
- Flat crossover
- Blend crossover
- Fuzzy Connectives Based crossover
- Heuristic crossover
- Simulated binary crossover
- Laplace real-coded crossover
- XLM real-coded crossover
- Double-Pareto crossover

1.3.3 The Mutation Operator

The EA's mutation operator is arguably its most basic counterpart. It draws inspiration from the fundamental mutation seen in biological genetics as a result of different transcriptional effects and spontaneous chromosome changes seen in early evolutionary studies. Using a bit-flip probability applied to each bit separately is the most typical mutation strategy for binary genomes. Assuming an expected single-bit flip but

allowing for variation, a typical mutation rate is $1/n$. Others have used a rate of $1/mn$, which predicts that one bit will be flipped in each generation over the entire population.

1.3.3.1 Mechanism for mutation

The GA recombination phase's final algorithmic step is a mutation, which guards against the new population becoming locked at local optimal circumstances and maintains population diversity. The process of mutation will help the population produce better offspring than the previous generation. With some aspects in the solution that will be mutated or modified with a minor probability, $P_m=0.001$ is typically indicated and is beneficial in making the algorithmic process more dependable (Bandyopadhyay & Maulik, 2002). The mutation is characterized by a little, arbitrary change in the chromosome, leading to a new solution.

The mutation clock created by Goldberg (1989) addresses several issues with computing complexity that can arise during the mutation process. A mutation in a constrained environment expands the search space. A tiny chromosomal variation could have a big effect on the next generation. By addressing the issue of local optima, new search spaces are intended to be obtained. The mutation is a procedure that involves flipping, reversing, swapping, and exchanging bits in the binary strings that Hong et al. (2000) have explored. The idea of flipping, swapping, and inverting binary bit values is shown in Figure 1.4.

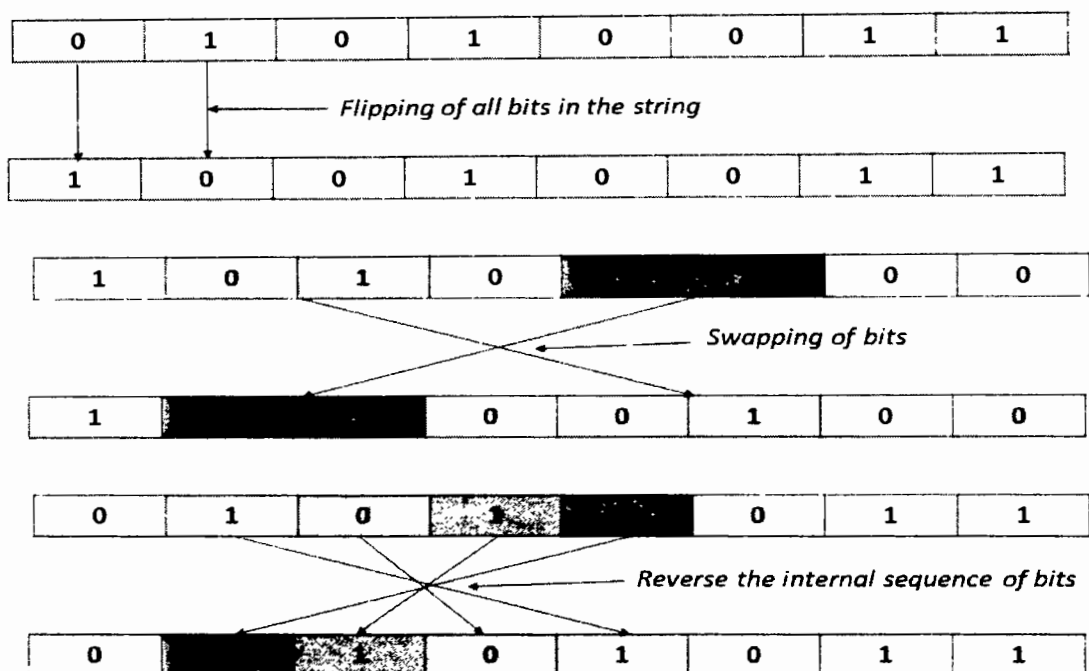


Figure 1.4: Visual description of flipping, swapping, and inverting binary bit

1.4 Parameters in genetic algorithms:

One of the most crucial elements in an effective algorithmic process is parameter selection. In GA, selecting the initial population (population size), crossover probability, and mutation rate are crucial for arriving at the best solution. De Jong (1980) proposed a balanced GA parameter adjustment that was beneficial for the efficient genetic process. Mitchell (1998) discussed how suitable population size, low mutation rate, and a comparatively high crossover probability affected algorithmic investigations. Furthermore, Hopgood (2011) proposed that to achieve the best outcomes, the values of genetic parameters should be modified in accordance with the nature of the issues.

1.4.1 Population size:

The population is chosen for the seamless application of GA in a fully random manner, but the size, complexity, and accessibility of the problem also play important roles in the efficient search for the best solution. Goldberg (1989), Schmitt, and Amini (1998), and jointly proposed that the maximum number of generations and the minimum population size have a better-controlled search space for global optimization. However, the issue of a small population size raises the possibility of premature convergence. Reeves (1993) and Chatterjee et al. (1996), among others, have found that a large population size increases population diversity. The drawback of the huge population

size, however, causes a long convergence time and high computational load. Therefore, Goldberg (2002) indicated that choosing the right population size is important for overall optimization.

1.4.2 Generation gap:

In certain circumstances, it's possible that comparably fit candidates won't be chosen for the following generation during the selection process. A generation gap is defined as a parameter in the GA process to replace the fixed ratio of the population in each generation to address these issues. As a result, the crossover process maintains a constant percentage of the generally better fitness values, and this replaces the new one across the board. As a result, the De Jong and Sarma (1993) study's findings about the generation gap are acknowledged.

1.5 Benchmark function

Different guided random search algorithmic techniques have been created in the past to address various optimization issues. Genetic algorithms (GAs), simulated annealing (SA), ant colony optimization (ACO), and particle swarm optimization (PSO) are subcategories of these algorithmic techniques. Although these algorithmic methods have effective search capabilities, they may have trouble with challenging optimization issues. As a result, it is also noted that different optimization issues suffer from the curse of dimensionality. According to the algorithms' performance degrades as the search space's dimensions grow. This flaw is primarily caused by two factors: first, the aspects of the problem might change depending on the problem's magnitude and nature, and second, the solution space of optimization problems grows exponentially as the dimension rises. A better example is the Rosenbrock benchmark function, which is unimodal for two dimensions but multimodal for multiple dimensions.

By using several well-known benchmark functions from the standard literature that have more varied properties, optimization issue complexity can be evaluated (Molga & Smutnicki, 2005). Benchmarks might be different well-known continuous test issues. These benchmarks can be grouped based on the following characteristics.

Convex and unimodal with several dimensions: It is possible to evaluate the algorithmic performance and gradual convergence to the ideal solution using this combination of benchmark properties.

Multimodal, bi-dimensional, and with few local optima: This combination in the benchmark is used to assess the effectiveness of conventional optimization methods in a hostile environment.

Bi-dimensional, multimodal, and having a high number of local optimums: It is utilized for intelligent resistance-based optimization methods like the genetic algorithm and simulated annealing, among others. These algorithmic techniques are built on a fictitious framework for two-dimensional optimization techniques, which are extremely uncommon in actual use.

Multidimensional and multimodal with a large number of local optimums: This set of characteristics can be used to assess intelligent resistant optimization issues. It is also advised for discrete optimization issues because they practically enable higher dimensions.

The extensive list of benchmark procedures includes Rosenbrock, Cosine mixture, Schwefel, Sphere Function, Styblinski-Tang, Sum of Power, Drop-wave, Rastrigin, Levy and Mantalvo and New function the effectiveness and viability of suggested evolutionary approaches are assessed (Jamil & Yang, 2013).

1.6 Objectives of the study

Designing and creating selection and crossover systems based on probability utilizing statistical computing techniques is the study's main goal. Several combinatorial optimization problems and well-known benchmark functions with a range of properties are used to evaluate these approaches. With the use of appropriate statistical approaches, our work on genetic algorithms has the following precise objectives:

- To analytically investigate the elements and steps involved in genetic algorithms.
- To maintain population diversity through a process of crossover that can avoid premature convergence.
- To exploit promising regions in the solution space where good solutions are likely to be found.
- To create a fine balance between population diversity and selection pressure for exploring the better solution space.
- To enhancement the functionality of complex multimodal benchmarks.
- To validate and optimize the effectiveness of the proposed crossover operators against existing alternatives via simulation research.
- To implement statistical evaluation approaches while conducting simulation-based comparative evaluations of the current versus newly proposed crossover schemes.

Chapter 2

Literature Review

The British naturalist Charles Darwin first proposed the fundamental theory of natural selection in 1859; see, for instance, Holland (1992). According to the principle of natural selection, it is more likely for people who have particular advantageous features to reproduce and pass on their distinguishing traits to their progeny. As a result, people with less desirable traits will eventually disappear from the population. Chromosomes, which are formed of genes, are where the genetic inheritance is stored, according to a natural phenomenon (Haq et al., 2019a). The traits of each person are determined by their genes, which are passed on to their offspring when they mate. Sometimes throughout the genetic process, the shape of the chromosomes might alter as a result of crossing and mutation (Goldberg, 1989). The average population will continue to grow as a result of the natural process of reproduction increasing the number of people with desirable traits.

Thus, the crossover operator uses exploring the new search space using genetic information across chromosomes, whereas the mutation operator uses genetic information between chromosomes to preserve population diversity and prevent early convergence (Haq et al., 2019b). Numerous population-based probabilistic optimization strategies exist that are non-continuous or non-differentiable objective functions to maintain population diversity and avoid local optima. These methods include genetic algorithms (Holland, 1991; Deb, 2000); particle swarm optimization (Eberhart et al., 2001); simulated annealing (Kirkpatrick et al., 1983); differential evolution (Price et al., 2006); ant colony optimization (Dorigo, 1991); and tabu search (Glover, 1986; Knox, 1994), among others. Thus, guided random search methods constitute the umbrella term for all of these optimization strategies (Goldberg, 1991). GA is the most effective method for comprehending and resolving issues with little information.

A real-coded genetic algorithm has attracted a lot of interest in recent years due to its unique and remarkable performance. As a result, numerous academics conducted in-depth analyses and produced useful findings. The GA performance is significantly impacted by the crossover and mutation operators. As a result, several researchers focus heavily on improving these operators' performance.

A heuristic crossover (HX) operator was first presented by Wright (1991), and it is more effective at handling both constrained and unconstrained optimization issues. The HX crossover operator's generation of children is situated along a line that connects the parents who place the most priority on fitness. By integrating the theoretical idea of interval schemata, Eshelman et al. (1993) proposed a blend crossover operator (BLX-) in the early 1990s. Because of how little the parents differ from one another, generations of kids are fairly similar, but when parents differ greatly, the offspring are identical to a random sample of space. Blend crossover is a parameter represented by the symbol, with 0.5 being the recommended value for the best outcomes. Determining the need for a modified single-point binary crossover, A simulated binary crossover (SBX) was developed by Deb et al. in 1995. Given that is the SBX distribution index, SBX generates two offspring for each of the two selected parents, and this distance between the two offspring is determined by SBX. If the value is significant, there is a greater chance that the created offspring will be located closer to the two parents; if the value is relatively small, there is a greater chance that the created offspring will be located farther from both parents. The drawback of SBX is that it is unable to regulate the separation between two offspring created by the crossover process since it is unable to adaptively control the magnitude of the parameter's value. To increase the efficiency of the evolutionary process.

The Ellipsoidal probability distribution and the uni-modal normal distribution crossover operator (UNDX) were created by Ono et al. in 2003. Deb et al. PCX, or self-adaptive multi-parent crossover, was proposed in 2002. For the generation of N offspring with different parents, this PCX computes various vectors using a high probability. Laplace crossover (LX), which can be used to locate the offspring and is associated with the Laplace probability distribution, was proposed by Deep et al. in 2007. As a result, the LX operator's two children are symmetrical in terms of their parental position and are not always found close to their best parents. To find the global optimum solution of multi-modal complicated optimization issues in 2014, Thakur devised a self-adaptive real-coded crossover. This crossover operator is related to the Pareto crossover (DPX) double Pareto probability distribution. Along with Using three mutation operators through integration concurrently, DPX also has a significant competitive advantage over other well-known crossover operators. A direction-based crossover (DBX) that might investigate $(2n - 1)$ different search paths was proposed by

Chuang et al. (2016). However, DBX's search options are constrained. Although it may be able to produce a crossover-based route that leads the chromosomes in the direction of the best outcome, this is not very suitable. When the dimensionalities of the variables are low, the null vector solution is more likely to be produced. Similarly to this, the literature lists several different mutation techniques, including Michalewicz's (1996) Non-Uniform Mutation (NUM) operator. The started and the end of the simulation process both benefit from the robust global search capacity of NUM. Thus, the NUM operator's primary flaw is the imposition of a maximum iteration number for all optimization jobs. Wang et al. (1996) once more the creation of a mutation operator that underwent mutations as measured by the gradient of the goal function. However, the performance of the mutation operator is insufficient if the goal function is not able to differentiate.

Deb et al. (2014) suggested Polynomial Mutation (PM), another real-coded mutation operator with applications in many different disciplines. Powerful random search capabilities for reaching a universal optimum solution with gradual convergence are the primary characteristic of PM operators. Dynamic random mutation (DRM) is a mutation operator that was recently proposed by Chuang et al. (2016). Therefore, the step size's mathematical formulation runs counter to its justification. It is difficult to determine how many rounds the mutation operator will require in total.

A new parent-centric real-coded crossover operator is developed in the simulation-based study by Haq et al. (2022) using an original probabilistic component of the mixed distribution. He also used the mixture distribution. This results from combining the double Pareto and Laplace probability distributions with different parameter values. Finding the most effective answers to difficult multimodal optimization problems is the major objective of the newly presented methodology. Therefore, for to assess the recently suggested mixed distribution crossover operator (MDX), three mutation operators (MTPM, PM, and NUM) are combined with the double Pareto (DPX), Laplace (LX), and simulated binary (SBX) crossover operators. In terms of computational complexity, durability, scalability, and capability for exploration and exploitation, the mixture-based crossover operator greatly surpassed all previous crossover operators, according to the empirical findings of the simulation-based study.

According to Fakhra Naqvi et al. (2020), well-known crossover operators known as the Logistic crossover (LogX) integrated with mutation operators known as the Makinen, Periaux, and Toivanen mutation (MPTM). She used the actual encoded crossover with the mutation operator while creating an algorithm. She employed a group of fifteen test problems from the global optimization literature to gauge the viability of the suggested strategy. She compared the results to some popular genetic algorithms (GAs) that have been reported in the literature. By contrasting the results of the various crossover operators, this study shows that the logistic crossover operator (LogX) with three mutation operators works better.

Chapter 3

MATERIALS AND METHODOLOGY

In this chapter, we integrated some newly suggested real coded crossover operators with several mutation operators, such as non-uniform mutation (NUM), Makinen, Periaux, and Toivanen mutation (MPTM), and power mutation operator (PM). We also theoretically outline the fundamental principles of a few recently used real coded GA operators including Laplace crossover (LX), double Pareto crossover (DPX), and simulated binary crossover (SBX), and discuss their advantages and disadvantages.

3.1 Tournament Selection (TS)

This way of selection operator falls under the category of Binary Tournament Selection. The Binary Tournament Selection is based on selecting two individuals at random, then competing to determine (Haq et al., 2019a). The weakest chromosomes, on the other hand, will not be able to mate because of their low fitness value when compared to a predetermined selection probability, z_i . Therefore, Equation 3.1 gives the predetermined selection probability " z_i " of the (i^{th}) sorted individual:

$$t_i = \frac{1}{z^u} ((Z - i + 1)^u - (z - i)^u); \quad i \in \{1, 2, \dots, S\} \quad (3.1)$$

where u is the size of the competition and S is the population size. $u=2$ and $u > 2$ are the conditions for the binary tournament. If more than two people are wanted, the TS can also be expanded (Sivanandam & Deepa, 2008).

3.2 Crossover operators

3.2.1 Double Pareto crossover (DPX)

A different kind of parent-centric crossover operator, the DPX operator (Thakur 2014), uses the Double Pareto probability distribution, whose CDF is provided by Equation 3.2.

$$F(x) = \begin{cases} \frac{1}{2} \left(1 - \frac{x}{\alpha\beta}\right)^{-\alpha}, & x < 0 \\ \frac{1}{2} \left[1 - \left(1 - \frac{x}{\alpha\beta}\right)^{-\alpha}\right], & x \geq 0 \end{cases} \quad (3.2)$$

where $b > 0$ is the distribution's scaling parameter and R is the location parameter.. By using DPX, pair of offspring $w^{(1)} = (w_1^{(1)}, w_2^{(1)}, \dots, w_n^{(1)})$ and $w^{(2)} = (w_1^{(2)}, w_2^{(2)}, \dots, w_n^{(2)})$ are generated from two parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ followed by a sequential process.

Step 1: Using a uniform probability distribution and a value between 0 and 1, generate a random number in step one.

Step 2: Equation 3.3 is used to calculate a cumulative distribution's opposite function of the double Pareto distribution, which is used to produce using the double Pareto probability distribution, generate random integers. The parameter value will be produced by this.

$$\beta_i = \begin{cases} \left(\alpha \beta (1 - (2r_i)^{-\frac{1}{\alpha_i}}) \right), & r_i \leq 0.5 \\ \left(\alpha \beta \left(1 - (2r_i)^{-\frac{1}{\alpha_i}} \right) - 1 \right), & r_i > 0.5 \end{cases} \quad (3.3)$$

Step-3: The descendants are now in Equations 3.4 and 3.5. $i = 1, 2, \dots, n$.

$$w_i^{(1)} = \frac{(x_i^{(1)} + x_i^{(2)}) + \beta_i |x_i^{(1)} - x_i^{(2)}|}{2} \quad (3.4)$$

$$w_i^{(2)} = \frac{(x_i^{(1)} + x_i^{(2)}) - \beta_i |x_i^{(1)} - x_i^{(2)}|}{2} \quad (3.5)$$

If the offspring generation in DPX is over the variable bound, that is, $x_i < x_i^l$ or $x_i < x_i^{ri}$, then random values are presented by x_i in a range $[x_i^l, x_i^{ri}]$.

3.2.2 Laplace crossover (LX)

According to Deep and Thakur (2007), the LX Among self-adaptive crossover operators, the crossover operator is one of them with real encoding. This crossover is related to the Laplace probability distribution, and Equation 3.6 afterward expresses the cumulative distribution function.

$$F(x) = \begin{cases} \frac{1}{2} e^{\left(\frac{|x-a|}{b}\right)}, & x \leq a \\ \left[1 - \frac{1}{2} e^{\left(\frac{|x-a|}{b}\right)} \right], & x > a \end{cases} \quad (3.6)$$

The location parameter in the Laplace probability distribution is known as “a” and belongs to the real number, whereas the scale parameter is known as $b > 0$. Utilizing LX, from two parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$, two

offspring $t^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_n^{(1)})$ and $t^{(2)} = (t_1^{(2)}, t_2^{(2)}, \dots, t_n^{(2)})$ which are produced by step-wise access.

Step 1: A number at random with a range of one should be generated using a uniform distribution.

Step 2: To create random numbers, use the Laplace probability distribution. To determine the parametric value β_i , invert the cumulative distribution function as indicated in Equation 3.7.

$$\beta_i = \begin{cases} a - b \log_e(r_i), & r_i \leq \frac{1}{2} \\ a + b \log_e(r_i), & r_i > \frac{1}{2} \end{cases} \quad (3.7)$$

Step-3: Now, Equations 3.8 and 3.9, respectively, produce the offspring for $i = 1, 2, \dots, n$.

$$t_i^{(1)} = x_i^{(1)} + \beta_i |x_i^{(1)} - x_i^{(2)}| \quad (3.8)$$

$$t_i^{(2)} = x_i^{(2)} + \beta_i |x_i^{(1)} - x_i^{(2)}| \quad (3.9)$$

When the number of descendants is external to the permissible range i.e. $x_i < x_i^l$ or $x_i > x_i^u$ Random values were then set to various bounds in LX and its location parameter was given the value "0".

3.2.3 Simulated Binary Crossover (SBX)

With real coding, the SBX is a significant crossover operator. Deb and Agrawal (1995) first introduced this operator, It stands out for having a continuous search space by binary transformation. The random integer is developed in the first stage using a uniform distribution with a limit of 0 to 1 consequently, the following mathematical equation in Equation 3.10 is used to determine the parametric value β_i .

$$\beta_i = \begin{cases} (2r_i)^{\frac{1}{(n_c+1)}}, & \text{if } r_i \leq \frac{1}{2} \\ \frac{1}{(2-2r_i)^{\frac{1}{(n_c+1)}}}, & \text{otherwise} \end{cases} \quad (3.10)$$

where $n_c \in [0, \infty]$ called the distribution index.

Hence, both parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$, an offspring $z = (z_1, z_2, \dots, z_n)$ is produced in the following Equation 3.11

$$t_i = \frac{1}{2} \left((x_i^{(1)} + x_i^{(2)}) - \beta_i |x_i^{(1)} - x_i^{(2)}| \right) \quad (3.11)$$

3.3 Mutation operators

3.3.1 Power mutation (PM)

A sufficient degree of population variation is maintained in the genetic process thanks to the mutation operator, which also prevents the simulation process from being stuck at the local optimum. In other words, the genetic process will aid in producing the population's most compatible kids. To address the computational complexity issues with the mutation process, Goldberg (1989) developed the mutation clock. By shifting the string's location, he used the exponential probability distribution to determine the next place. Exploring the new search space and effectively resolving challenging optimization issues are the fundamental goals of the mutation process. In order to prevent local minimum solutions, the mutation operator helps by utilizing larger search spaces. Here, we provide a theoretical definition of the Deep and Thakur (2007) proposed PM mutation operator.

The power distribution, from which the PM is derived, has the following p.d.f. and CDF in Equations 3.12 and 3.13.

$$f(x) = px^{p-1} \quad 0 \leq x \leq 1 \quad (3.12)$$

$$F(x) = x^p \quad 0 \leq x \leq 1 \quad (3.13)$$

where PM is employed to produce offspring, and p is a representation of the distribution index. $t = (t_1, t_2, \dots, t_n)$ from a parent $x = (x_1, x_2, \dots, x_n)$ the following methodical process.

Step 1: r_i should be chosen at random from a Uniform Distribution with the range $[0, 1]$.

Step 2: By utilizing the power distribution in Equation 3.14, determine a random value k_i .

$$k_i = (r_i)^{\frac{1}{p}} \quad (3.14)$$

utilize the following mathematical formulas to produce offspring in Equation 3.15.

$$t_i = \begin{cases} x_i - k_i(x_i - x_i^l), & \text{if } \frac{x_i - x_i^l}{x_i^{r_i} - x_i^l} < r_i \\ x_i + k_i(x_i - x_i^l), & \text{if } \frac{x_i - x_i^l}{x_i^{r_i} - x_i^l} \geq r_i \end{cases} \quad (3.15)$$

Where x_i^l and $x_i^{r_i}$ the smallest and higher bounds of the i^{th} choice variable, respectively. According to the aforementioned mathematical formulation. The "p" parameter is negatively correlated with the perturbation in offspring. Consequently, the probability

of producing mutant offspring grows with distance from the parametric constraint, which always yielded a suitable result.

3.3.2 Makinen, Periaux, and Toivanen mutation (MPTM)

The MPTM mutation operator was developed by Makinen et al. (1999), and it is used to address several broad-based shape-related optimization problems in GA, particularly in the fields of airflow and electromagnetics. Meittinen et al.'s 2003 work is also helpful for GA-based restricted optimization situations. On multimodal nonlinear optimization issues, Deep and Thakur (2007) examined and assessed its findings. From a point $x = (x_1, x_2, \dots, x_n)$ distorted point $\hat{x} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$ is obtained in the following way. Let's say that (r_i) comes from a uniform distribution and falls between 0 and 1. Equations 3.16 and 3.17 include the muffled answer as a result.

$$\hat{x}_i = (1 - \hat{z})x_i^l + (\hat{z})x_i^{r_i} , \quad (3.16)$$

Where

$$z = \begin{cases} z_i - z_i \left(\frac{z_i - r_i}{z_i} \right)^b , & \text{if } r_i < z_i \\ z_i , & \text{if } r_i = z_i \\ z_i + (1 - z_i) \left(\frac{z_i - r_i}{1 - z_i} \right)^b , & \text{if } r_i \geq z_i \end{cases} \quad (3.17)$$

And

$$z = \frac{x - x_i^l}{x_i^{r_i} - x}$$

Hence, the x_i^l and $x_i^{r_i}$ are the highest and lower range of i^{th} choice as well as variable.

3.3.3 Non-Uniform Mutation (NUM)

In real-coded GAs, Michalewicz's NUM mutation operator is the most often employed. Michalewicz et al.'s (1994, 1996) work on the non-uniform mutation's mechanism may have been the starting point. For implementation purposes, the power of the mutation may be decreased by increasing the number of generations so that it appears uniformly for the simulation process' first generations while looking place for subsequent origination . For a point $x^{(1)} = (x_1^{(v)}, x_2^{(v)}, \dots, x_n^{(v)})$ and muted point $x^{(v+1)} = (x_1^{(v+1)}, x_2^{(v+1)}, \dots, x_n^{(v+1)})$ is created after that:

Step 1: Make a random variable with the range [0, 1] by using a uniform probability distribution.

Step 2: Follow the mathematical expression in Equation 3.18 to get a muted solution.

$$x_i^{p+1} = \begin{cases} \left(x_i^p + (y_i^u - x_i^p) \left(1 - w_i^{(1-\frac{p}{P})^b} \right) \right), & r_i \leq 0.5 \\ \left(x_i^p - (x_i^p - y_i^l) \left(1 - w_i^{(1-\frac{p}{P})^b} \right) \right), & \text{Otherwise} \end{cases} \quad (3.18)$$

where "b" is a parametric parameter that establishes the mutation operator's functional range.

the present generation and the most generations possible are denoted by P and p, respectively. The ith decision variable's upper and lower bounds, respectively, are denoted by the letters y_i^u and y_i^l .

3.4 The proposed probability distribution

3.4.1 Exponential crossover operator (EX)

The exponential distribution is a continuous probability distribution that is applied to time until a certain event happens in probability theory and statistics. Throughout this process, things happen continuously, separately, and on average at a constant speed. The exponential distribution's lack of memory is a key trait. The exponential random variable could consist of more small values or fewer large values. A customer's overall grocery shop spending, for instance, follows an exponential distribution.

The exponential distribution is one of the most frequently used continuous distributions. It aids in calculating the interval of time between the events. Numerous fields, including physics, dependability theory, queuing theory, and others use it. The following list includes some of the fields that the exponential distribution models. Finding the separation between mutations on a DNA strand is made easier by exponential distribution figuring out the decay period of the radioactive particle. The use of an exponential distribution allows us to calculate the heights of different molecules in a gas under stable parameters of temperature, pressure, and gravitational field.

The probability density function shown below in Equation 3.19 is used to create the mathematical underpinnings of the exponential crossover operator

$$f(x) = \lambda e^{-\lambda x}. \quad (3.19)$$

Aids in computing the highest monthly and annual amounts of normal rainfall and river outflow volumes we use exponential distribution. Equation 3.20 also features the cumulative distribution function in the second term.

$$F(x) = \begin{cases} e^{-\lambda x}, & x < 0 \\ 1 - e^{-\lambda x}, & x > 0 \end{cases} \quad (3.20)$$

By taking eX , a pair of offspring $v^{(1)} = (v_1^{(1)}, v_2^{(1)}, \dots, v_n^{(1)})$ and $v^{(2)} = (v_1^{(2)}, v_2^{(2)}, \dots, v_n^{(2)})$ are generated from a couple of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in the following sequential process.

Step-1: Produce a random number u between 0 and 1.

Step-2: the region beneath the curve going from to the randomly chosen number u is used to build the parameter β_i by inverting the CDF of Laplace Distribution.

$$y = e^{-\lambda x} \quad (3.21)$$

$$\ln(y) = (-\lambda x) \quad (3.22)$$

$$\frac{\ln(y)}{\lambda} = x \quad (3.23)$$

If $x > 0$, now:

$$y = 1 - e^{-\lambda x} \quad (3.24)$$

$$y-1 = -e^{-\lambda x} \quad (3.25)$$

$$\ln(y-1) = \lambda x \quad (3.26)$$

$$\frac{\ln(y-1)}{\lambda} = x \quad (3.27)$$

$$Bq = \begin{cases} \frac{\ln(y)}{\lambda}, & r_i \leq 0.5 \\ \frac{\ln(y-1)}{\lambda}, & r_i > 0.5 \end{cases} \quad (3.28)$$

Step 3: Following Equations 3.29 and 3.30, respectively, generate the offspring for $i=1, 2 \dots n$.

$$t_i^{(1)} = \frac{(x_i^{(1)} + x_i^{(2)}) + \beta_q |x_i^{(1)} - x_i^{(2)}|}{2}, \quad (3.29)$$

And,

$$t_i^{(2)} = \frac{(x_i^{(1)} + x_i^{(2)}) - \beta_q |x_i^{(1)} - x_i^{(2)}|}{2}. \quad (3.30)$$

3.4.2 Laplace crossover operator (LX)

The distribution of differences between two independent variants with identical exponential distributions is known as the Laplace distribution, sometimes known as the double exponential distribution (Abramowitz and Stegun 1972, p. 930). It has uses in finance, ocean engineering, hydrology, and voice and image recognition. Its primary feature is the way it represents the likelihood of errors, also known as deviations from a central value.

The Laplace crossover operator's mathematical foundation is built using the probability density function described in Equation 3.31 below:

$$f(x) = \frac{1}{2b} e^{-|x-\mu|/b} \quad (3.31)$$

Equation 3.32 also includes a subsequent representation of the cumulative distribution function.

$$F(x) = y = \begin{cases} 1/2 e^{[x-\mu]/b}, & x \leq \alpha \\ 1 - \frac{1}{2} e^{[x-\mu]/b}, & x > \alpha \end{cases} \quad (3.32)$$

By taking LX, a pair of offspring $v^{(1)} = (v_1^{(1)}, v_2^{(1)}, \dots, v_n^{(1)})$ and $v^{(2)} = (v_1^{(2)}, v_2^{(2)}, \dots, v_n^{(2)})$ are generated from a couple of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in the following sequential process.

Step-1: Produce a U is a chance number between 0 and 1.

Step-2: The parameter is constructed through CDF inversion of the Laplace distribution, which employs the region beneath the curve from to to the arbitrary number u.

$$y = 1/2 e^{[x-\mu]/b} \quad (3.33)$$

$$2y = e^{[x-\mu]/b} \quad (3.34)$$

$$\ln 2y = [x - \mu]/b \quad (3.35)$$

$$b * \ln 2y = [x - \mu] \quad (3.36)$$

$$\mu + b * [\ln 2y] = x \quad (3.37)$$

If $x > \alpha$, now:

$$y = 1 - 1/2 e^{[x-\mu]/b} \quad (3.38)$$

$$y-1 = - 1/2 e^{[x-\mu]/b} \quad (3.39)$$

$$1-y = 1/2 e^{[x-\mu]/b} \quad (3.40)$$

$$2[1-y] = e^{[x-\mu]/b} \quad (3.41)$$

$$\ln 2[1-y] = x - \mu/b \quad (3.42)$$

$$b * \ln 2[1-y] = x - \mu \quad (3.43)$$

$$\mu - b * \ln 2[1-y] = x \quad (3.44)$$

$$\mu - b * [\ln 2(1-y)] = x \quad (3.45)$$

Now,

$$\beta_q = \begin{cases} \mu + b * [\ln 2y], & r_i \leq 0.5 \\ \mu - b * [\ln 2(1-y)], & r_i > 0.5 \end{cases} \quad (3.46)$$

Step 3: Following Equations 3.47 and 3.48, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$t_i^{(1)} = \frac{(x_i^{(1)} + x_i^{(2)}) + \beta_q |x_i^{(1)} - x_i^{(2)}|}{2}, \quad (3.47)$$

And,

$$t_i^{(2)} = \frac{(x_i^{(1)} + x_i^{(2)}) - \beta_q |x_i^{(1)} - x_i^{(2)}|}{2}. \quad (3.48)$$

3.4.3 The Fisk Crossover Operator (FX) Based On Log-Logistic Probability Distribution

The Log-Logistic probability distribution is useful for modeling the transmission times of numerical information in software and networking, as well as for estimating stream flow and precipitation in hydrology. The creation of economic wealth or income models is another distinctive feature of log-logistic distribution. This probability distribution effectively creates accurate simulations of transmission delays related to sensory data obtained from a networked telerobotic, allowing us to predict future arrival times and

guarantee the timeliness of these systems (Gago-Bentez et al., 2013). Additionally, the log-logistics distribution is very effective in obtaining accurate estimates of the low-flow frequency during the course of an event involving extremely low flows within a given period (Ashkar & Mahdi, 2006).

Equation 3.49, which is provided below, uses the probability density function of the Log-logistics distribution to establish the mathematical underpinnings of the Fisk crossover operator:

$$f(x) = \frac{\left(\frac{\beta}{\alpha}\right)\left(\frac{x}{\alpha}\right)^{\beta-1}}{\left(1+\left(\frac{x}{\alpha}\right)^{\beta}\right)^2}. \quad (3.49)$$

The subsequent diagram represents the cumulative distribution function: in Equation 5.50.

$$F(x) = y = \begin{cases} \frac{1}{1+\left(\frac{x}{\alpha}\right)^{-\beta}}, & x > 0 \\ 1 - \frac{1}{1+\left(\frac{x}{\alpha}\right)^{-\beta}}, & x \leq 0 \end{cases} \quad (3.50)$$

Where $\alpha > 0$ is the scale parameter and $\beta > 0$ is the shape parameter

By taking FX, a pair of offspring $v^{(1)} = (v_1^{(1)}, v_2^{(1)}, \dots, v_n^{(1)})$ and $v^{(2)} = (v_1^{(2)}, v_2^{(2)}, \dots, v_n^{(2)})$ are generated from a couple of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in the following sequential process.

Step-1: Create (r_i) , a random integer generated using a uniform probability distribution with r_i between 0 and 1.

Step-2: By generating random integers from the distribution and determining the cumulative distribution function's inverse of the Fisk/Log-logistic distribution as described in Equation 3.51, you can calculate the parameter's value.

$$\beta_q = \begin{cases} \alpha \left(\frac{1-y}{y}\right)^{\frac{1}{\beta}}, & r_i \leq 0.5 \\ \alpha \left(\frac{y}{1-y}\right)^{\frac{1}{\beta}}, & r_i > 0.5 \end{cases} \quad (3.51)$$

Step-3: Following Equations 3.52 and 3.53, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$z_i^{(1)} = \frac{(y_i^{(1)} + y_i^{(2)}) + \beta_q |y_i^{(1)} - y_i^{(2)}|}{2}, \quad (3.52)$$

And,

$$z_i^{(2)} = \frac{(y_i^{(1)} + y_i^{(2)}) - \beta_q |y_i^{(1)} - y_i^{(2)}|}{2} . \quad (3.53)$$

Concerning the two mentioned Equations 3.52 and 3.53, it is evident that for the fixed parametric value in Figure 3.2, the lowest parametric value generates offspring who are close to their parents the larger Parametric value of offspring produced apart from the parents. As a result, Figure 3.3 illustrates how the production of offspring is farther distance from the parents for the predetermined parametric value while the offspring created by the lower parametric value are closer to their parents.

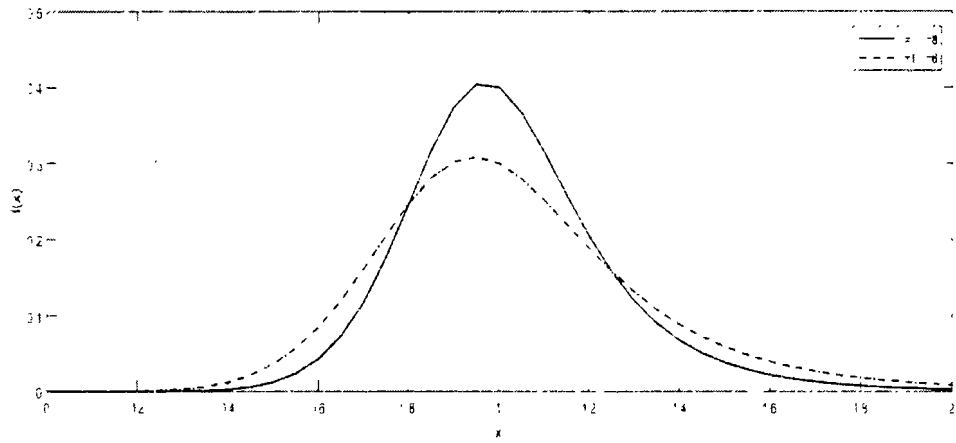


Figure 3.2: *p.d.f* of Fisk/log-logistic distribution for fix α

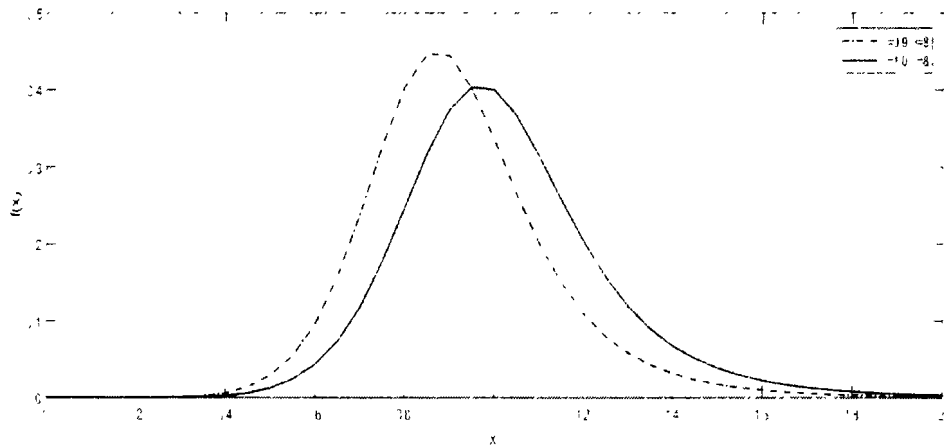


Figure 3.3: *p.d.f* of Fisk/log-logistic distribution for fix β

3.4.4 Gompertz crossover operator (GX)

The pattern of adult deaths is best described by the Gompertz distribution (Wetterstrand 1981; Gavrilov and Gavrilova 1991). The Gompertz force of mortality extends to the entire life span of populations with no documented mortality slowing for low baby (and young adult) death rates (Vaupel 1986).

Demographers and actuaries have given the Gompertz distribution a lot of consideration. The Gompertz distribution was first thoroughly examined by Pollard and Valkovics in 1992. Their findings, however, are valid only when the initial mortality level is very near zero. Similar conclusions were reached by Kunimura (1998). They both described the Gompertz distribution's moment-generating function in terms of the imperfect. The Gompertz force of mortality was reformulated by Willemse and Koppelaar in 2000, and correlations were derived for this new formulation. The negative Gompertz distribution a Gompertz distribution with a negative rate of the aging parameter was later described by Marshall and Olkin (2007).

Equation 3.54, which is provided below, uses the probability density function of the Gompertz distribution to establish the mathematical underpinnings of the Gompertz crossover operator. $F(x)$

$$f(x) = b\eta e^{bx} e^{\eta} \exp(-\eta e^{bx}). \quad (3.54)$$

Following is another representation of the cumulative distribution function: in Equation 3.55.

$$F(x) = \begin{cases} \exp(-\eta(e^{bx} - 1)), & x < 0 \\ 1 - \exp(-\eta(e^{bx} - 1)), & x > 0 \end{cases} \quad (3.55)$$

Where parameter b and $\eta > 0$.

By using GX, a pair of offspring $u^{(1)} = (u_1^{(1)}, u_2^{(1)}, \dots, u_n^{(1)})$ and $u^{(2)} = (u_1^{(2)}, u_2^{(2)}, \dots, u_n^{(2)})$ are generated from a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in a subsequent sequential process.

Step-1: Create (r_i) , a number chosen at random from a range of 0 to 1 using a uniform probability distribution

Step-2: By creating random integers from the distribution and finding the Gompertz distribution's inverse cumulative distribution function as described in Equation 3.55, you can calculate the parameter's value:

$$y = 1 - \exp(-\eta(e^{bx} - 1)) \quad (3.56)$$

$$\exp(-\eta(e^{bx} - 1)) = 1 - y \quad (3.57)$$

$$e^{bx} - 1 = -\frac{1}{\eta} \ln(1 - y) \quad (3.58)$$

$$e^{bx} = 1 - \frac{1}{\eta} \ln(1 - y) \quad (3.59)$$

$$bx = \ln \left[1 - \frac{1}{\eta} \ln(1 - y) \right] \quad (3.60)$$

$$x = \frac{1}{b} \ln \left[1 - \frac{1}{\eta} \ln(1 - y) \right] \quad (3.61)$$

If $x > 0$, now:

$$y = \exp(-\eta(e^{bx} - 1)) \quad (3.62)$$

$$\ln y = -\eta(e^{bx} - 1) \quad (3.63)$$

$$-\frac{\ln y}{\eta} = e^{bx} - 1 \quad (3.64)$$

$$1 - \frac{\ln y}{\eta} = e^{bx} \quad (3.65)$$

$$\ln \left(1 - \frac{\ln y}{\eta} \right) = bx \quad (3.66)$$

$$\frac{1}{b} \ln \left(1 - \frac{\ln y}{\eta} \right) = x \quad (3.67)$$

Now,

$$B_q = \begin{cases} \frac{1}{b} \ln \left(1 - \frac{\ln y}{\eta} \right), & r_i < 0 \\ \frac{1}{b} \ln \left[1 - \frac{1}{\eta} \ln(1 - y) \right], & r_i > 0 \end{cases} \quad (3.68)$$

Step-3: Following Equations 3.67 and 3.68, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$z_i^{\{1\}} = \frac{(y_i^{\{1\}} + y_i^{\{2\}}) + \beta_q |y_i^{\{1\}} - y_i^{\{2\}}|}{2}, \quad (3.67)$$

And,

$$z_i^{\{2\}} = \frac{(y_i^{\{1\}} + y_i^{\{2\}}) - \beta_q |y_i^{\{1\}} - y_i^{\{2\}}|}{2}. \quad (3.68)$$

3.4.5 Weibull crossover operator (WBX)

The Weibull Distribution is a continuous probability distribution that is used to examine product reliability, model failure rates, and analyze life statistics. It may also accommodate a broad range of data from numerous other disciplines, including engineering sciences, hydrology, biology, and economics. It is a probability distributional extreme value that is widely used to describe data such as reliability, survival, wind velocity, and other variables. Weibull distribution should only be used due to its adaptability. Because it can imitate several distributions, including normal and exponential distributions. With the aid of parameters, the dependability of the Weibull distribution is assessed.

Due to its flexibility, the Weibull distribution is frequently used in reliability analysis and life data analysis. This distribution is used to model the range of behaviors for a specific function depending on the parameter values. The Weibull has been used extensively throughout the engineering and scientific communities. Its acceptance, breadth of applications, and theoretical advancement all continue to rise by Stephen. Luko (1999).

Hence, Equation 3.69, which is provided below, uses the probability density function of the Weibull distribution to establish the mathematical underpinnings of the Weibull crossover operator:

$$f(x) = \frac{t}{\lambda} \left(\frac{x}{\lambda}\right)^{t-1} e^{-\left(\frac{x}{\lambda}\right)^t} \quad (3.69)$$

Following is another representation of the cumulative distribution function in Equation 3.70.

$$F(x) = y = \begin{cases} e^{-\left(\frac{x}{\lambda}\right)^t}, & x < 0 \\ 1 - e^{-\left(\frac{x}{\lambda}\right)^t}, & x > 0 \end{cases} \quad (3.70)$$

By taking WBX, a pair of offspring $u^{(1)} = (u_1^{(1)}, u_2^{(1)}, \dots, u_n^{(1)})$ and $u^{(2)} = (u_1^{(2)}, u_2^{(2)}, \dots, u_n^{(2)})$ are generated from a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in a subsequent sequential process.

Step-1: Produce a random number u between 0 and 1.

Step-2: By creating random integers from the distribution and finding the opposite of the Weibull distribution's cumulative distribution function as described in Equation 3.70, you can calculate the parameter's value:

$$y = e^{(-\frac{x}{\lambda})^k} \quad (3.71)$$

$$\ln y = (-\frac{x}{\lambda})^k \quad (3.72)$$

$$(\ln y)^{\frac{1}{k}} = -\frac{x}{\lambda} \quad (3.73)$$

$$-\lambda(\ln y)^{\frac{1}{k}} = x \quad (3.74)$$

If $x > 0$, now:

$$y = 1 - e^{(-\frac{x}{\lambda})^k} \quad (3.75)$$

$$1 - y = e^{(-\frac{x}{\lambda})^k} \quad (3.76)$$

$$\ln(1 - y) = (-\frac{x}{\lambda})^k \quad (3.77)$$

$$-\lambda(\ln(1 - y))^{\frac{1}{k}} = x \quad (3.78)$$

Now,

$$B_q = \begin{cases} -\lambda(\ln y)^{\frac{1}{k}}, & r_i < 0 \\ -\lambda(\ln(1 - y))^{\frac{1}{k}}, & r_i > 0 \end{cases} \quad (3.79)$$

Step 3: Following Equations 3.80 and 3.81, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$t_i^{\{1\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) + \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}, \quad (3.80)$$

And,

$$t_i^{\{2\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) - \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}. \quad (3.81)$$

3.4.6 Double Pareto crossover operator (DPX):

Combining the Pareto distribution and the distribution of a Pareto random variable will result in the double Pareto distribution, which has power tail behavior at zero and infinity and occurs as an exponential function of a double exponential distribution. (Kotz et al., 2001; Reed, 2001). For heavy-tailed phenomena, the double Pareto distribution has been suggested as a model. When modeling finance, physics, and

engineering (Embrechts et al., 1997; Fama, 1965; Jansen and Varies, 1991; Loretan and Phillips, 1994; Mandelbrot, 1963; Madfal and Raw, 1996; Rachev, 2003; Rachev and Mitnik, 2000), heavy-tailed distributions are crucial (Barkai et al., 2000; Rachev and Mitnik, 2000). By reducing mean squared error fit on a probability plot and least squares fitting on a plot of the tail distribution function, Burroughs and Tebbens (2001a; 2001b) estimated the parameters of the truncated distribution. Equation 3.82, which is provided below, uses the probability density function of the Double Pareto distribution to establish the mathematical underpinnings of the Double Pareto crossover operator:

$$f(x) = \frac{\theta}{2\beta} \left(\frac{x}{\beta}\right)^{\theta-1} \quad (3.82)$$

Where parameters $\theta < x$ and $\beta > x$

Equation 3.83 also shows the CDF as the next.

$$F(x) = y = \begin{cases} \frac{1}{2} \left(\frac{x}{\beta}\right)^{\theta} , & \theta \leq x < \beta \\ 1 - \frac{1}{2} \left(\frac{\beta}{x}\right)^{\theta} , & \theta \geq \beta \end{cases} \quad (3.83)$$

By using DPX, a pair of offspring $v^{(1)} = (v_1^{(1)}, v_2^{(1)}, \dots, v_n^{(1)})$ and $v^{(2)} = (v_1^{(2)}, v_2^{(2)}, \dots, v_n^{(2)})$ are generated from a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in a subsequent sequential process.

Step-1: Make a number u at random, between 0 and 1.

Step-2: You can calculate the value of the parameter by producing random integers from the distribution and determining the inverse of the Double Pareto Distribution's CDF, as stated in Equation 3.83:

$$y = 1 - \frac{1}{2} \left(\frac{\beta}{x}\right)^{\theta} \quad (3.84)$$

$$y - 1 = -\frac{1}{2} \left(\frac{\beta}{x}\right)^{\theta} \quad (3.85)$$

$$(y - 1) = -\left(\frac{\beta}{x}\right)^{\theta} \quad (3.86)$$

$$(2(y - 1))^{\frac{1}{\theta}} = -\frac{\beta}{x} \quad (3.87)$$

$$\frac{(2(y - 1))^{\frac{1}{\theta}}}{\beta} = -\frac{1}{x} \quad (3.88)$$

$$\frac{\beta}{(2(y - 1))^{\frac{1}{\theta}}} = x \quad (3.89)$$

If $\theta \geq \beta$, now:

$$y = \frac{1}{2} \left(\frac{x}{\beta} \right)^\theta \quad (3.90)$$

$$2(y) = \left(\frac{x}{\beta} \right)^\theta \quad (3.91)$$

$$[2(y)]^{\frac{1}{\theta}} = \frac{x}{\beta} \quad (3.92)$$

$$\beta[2(y)]^{\frac{1}{\theta}} = x \quad (3.93)$$

Now,

$$B_q = \begin{cases} \beta[2(y)]^{\frac{1}{\theta}}, & r_i \leq 0.5 \\ \frac{\beta}{(2(y-1))^{\frac{1}{\theta}}}, & r_i > 0.5 \end{cases} \quad (3.94)$$

Step 3: Following Equations 3.95 and 3.96, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$t_i^{\{1\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) + \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}, \quad (3.95)$$

$$t_i^{\{2\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) - \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}. \quad (3.96)$$

3.4.7 Rayleigh Crossover Operator (RX)

The Rayleigh distribution is one of the most widely applied probability distributions. The Rayleigh distribution is a particular example of the Weibull distribution that Rayleigh described in 1880. In areas including project effort loading modeling, survival and reliability analysis, communication theory, physical sciences, technology, diagnostic imaging, applied statistics, and clinical research, it is crucial for modeling and analyzing lifetime data by N H Al-Noor and N K Assi 2020. Equation 3.98, which is provided below, uses the probability density function of the Rayleigh distribution to establish the mathematical underpinnings of the Rayleigh crossover operator:

$$f(x) = \frac{x}{\beta} e^{-\frac{1}{2} \left(\frac{x}{\beta} \right)^2}. \quad (3.98)$$

The cumulative distribution function is also depicted as subsequent in Equation 3.99

$$F(x) = y = \begin{cases} e^{-\frac{1}{2}(\frac{x^2}{\beta^2})}, & x < 0 \\ 1 - e^{-\frac{1}{2}(\frac{x^2}{\beta^2})}, & x > 0 \end{cases} \quad (3.99)$$

By using RHX, a pair of offspring $u^{(1)} = (u_1^{(1)}, u_2^{(1)}, \dots, u_n^{(1)})$ and $u^{(2)} = (u_1^{(2)}, u_2^{(2)}, \dots, u_n^{(2)})$ are generated from a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in a subsequent sequential process.

Step-1: Produce a random number u between 0 and 1.

Step-2: By creating random integers from the distribution and finding the inverse of the cumulative distribution function of the Rayleigh distribution as described in Equation 3.99, you may calculate the parameter's value.

$$y = e^{-\frac{1}{2}(\frac{x^2}{\beta^2})} \quad (3.100)$$

$$\ln y = -\frac{1}{2}(\frac{x^2}{\beta^2}) \quad (3.101)$$

$$-2 \ln y = \frac{x^2}{\beta^2} \quad (3.102)$$

$$\beta^2(-2 \ln y) = x^2 \quad (3.103)$$

$$\beta(-2 \ln y)^{\frac{1}{2}} = x \quad (3.104)$$

If $x > 0$, now:

$$y = 1 - e^{-\frac{1}{2}(\frac{x^2}{\beta^2})} \quad (3.105)$$

$$1-y = e^{-\frac{1}{2}(\frac{x^2}{\beta^2})} \quad (3.106)$$

$$\ln (1-y) = -\frac{1}{2}(\frac{x^2}{\beta^2}) \quad (3.107)$$

$$-2 \ln (1-y) = \frac{x^2}{\beta^2} \quad (3.108)$$

$$\beta^2(-2 \ln (1-y)) = x^2 \quad (3.109)$$

$$\beta(-2 \ln (1-y))^{\frac{1}{2}} = x \quad (3.110)$$

$$B_q = \begin{cases} \beta(-2\ln y)^{\frac{1}{2}} & , \quad x < 0 \\ \beta(-2\ln(1-y))^{\frac{1}{2}} & , \quad x > 0 \end{cases} \quad (3.111)$$

Step 3: Following Equations 3.112 and 3.113, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$t_i^{\{1\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) + \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}, \quad (3.112)$$

$$t_i^{\{2\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) - \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}. \quad (3.113)$$

3.4.8 Lindley crossover operator

Equation 3.114, which is provided below, uses the probability density function of the Lindley distribution to establish the mathematical underpinnings of the Lindley crossover operator:

$$f(x) = \frac{\theta^2}{\theta+1} (1+x) e^{-\theta x}. \quad (3.114)$$

The cumulative distribution function is also depicted as subsequent in Equation 3.115

$$F(X) = y = \begin{cases} \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x}, & x < 0 \\ 1 - \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x}, & x > 0 \end{cases} \quad (3.115)$$

By using LDX, a pair of offspring $u^{(1)} = (u_1^{(1)}, u_2^{(1)}, \dots, u_n^{(1)})$ and $u^{(2)} = (u_1^{(2)}, u_2^{(2)}, \dots, u_n^{(2)})$ are generated from a pair of parents $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)})$ in a subsequent sequential process.

Step-1: Produce a random number u between 0 and 1.

Step-2: By creating random integers from the distribution and finding the inverse of the cumulative distribution function of the Lindley distribution as described in Equation 3.115, you can calculate the parameter's value.

$$y = \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x} \quad (3.116)$$

$$y - 1 = \frac{\theta x}{\theta+1} e^{-\theta x} \quad (3.117)$$

$$\frac{(y-1)(\theta+1)}{\theta} = x e^{-\theta x} \quad (3.118)$$

$$\ln \left[\frac{(y-1)(\theta+1)}{\theta} \right] = x(-\theta x) \quad (3.119)$$

$$\ln \left[\frac{(y-1)(\theta+1)}{\theta} \right] = -\theta x^2 \quad (3.118)$$

$$\left[\frac{-\ln \frac{(y-1)(\theta+1)}{\theta}}{\theta^2} \right]^{\frac{1}{2}} = x \quad (3.119)$$

If $x > 0$, now:

$$y = 1 - \left[1 + \frac{\theta x}{\theta+1} \right] e^{-\theta x} \quad (3.120)$$

$$y - 1 = \left[1 + \frac{\theta x}{\theta+1} \right] e^{-\theta x} \quad (3.121)$$

$$y-2 = \left[1 + \frac{\theta x}{\theta+1} \right] e^{-\theta x} \quad (3.122)$$

$$-\frac{\ln(y-2)(\theta+1)}{\theta} = x^2 \quad (3.123)$$

$$\left[-\frac{\ln(y-2)(\theta+1)}{\theta} \right]^{\frac{1}{2}} = x \quad (3.124)$$

Now,

$$B_q = \begin{cases} \left[\frac{-\ln(y-1)(\theta+1)}{\theta^2} \right]^{\frac{1}{2}}, & x > 0 \\ \left[-\frac{\ln(y-2)(\theta+1)}{\theta} \right]^{\frac{1}{2}}, & x < 0 \end{cases} \quad (3.125)$$

Step 3: Following Equations 3.126 and 3.127, respectively, generate the offspring for $i=1, 2, \dots, n$.

$$t_i^{\{1\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) + \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}, \quad (3.126)$$

And

$$t_i^{\{2\}} = \frac{(x_i^{\{1\}} + x_i^{\{2\}}) - \beta_q |x_i^{\{1\}} - x_i^{\{2\}}|}{2}. \quad (3.127)$$

3.4 Mixture distribution-based real-coded crossover

3.5.1 Background

The current section presents the mixed distribution crossover (MDX) operator, a unique real-coded crossover operator, as a result of the co-integration of the two probability distributions. In real-coded genetic algorithms, the idea of two-component mixture probability models is used to create this novel crossover operator. To achieve the optimum results, three mutation operators power mutation (PM), Makinen, Periaux, and Toivanen Mutation (MPTM), and non-uniform Mutation (NUM) are combined with the MDX (Thakur, 2014). A comparative computational analysis is conducted to look at the mixture distribution crossover's overall performance.

3.5.2 Newly Suggested Mixture Distribution based Real-Coded Crossover Operator

In the present part, a unique real-coded crossover operator is developed based on ten well-known existing real-coded crossover operators that have been hybridized. The strength of optimality from which it is achieved is retained by the design of this innovative technique. A mixture distribution crossover (MDX) is a newly developed crossover operator that combines the Lindley crossover operator with parameter (θ) and with Lindley crossover operator with parameter (ϕ) and the Rayleigh crossover operation (α) with Rayleigh crossover with parameter (β) to produce two offspring. Real-coded crossover operators are used to initiate the probabilistic-based Exploitation and research process as required by the hybridization concept. In other words, the best genetic process would result from the right ratio of exploitation and exploration

3.5.2.1 The proposed mixture of Lindley probability distribution

As a result, the finite mixture models have drawn increasing attention in the field of mathematical statistics over the years from both theoretical and practical viewpoints. According to Sultan and Al-Moisheer (2015), the finite mixture models are also used directly in a variety of applied sciences and engineering domains. Using the two-component mixture as a base of Lindley probability distribution with parameter θ and Lindley probability distributions with parameter Φ , In the following section, we recommend a parent-centric crossover operator (Thakur, 2014).

Equations 3.128 and 3.127 provide the probability density function (pdf) and cumulative distribution function (cdf) of the first component (Lindley with parameter θ).

$$f_1(t) = \frac{\theta^2}{\theta+1} (1+x) e^{-\theta x} . \quad (3.128)$$

$$F_1(t) = \begin{cases} \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x}, & t \leq 0 \\ 1 - \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x}, & t > 0 \end{cases} \quad (3.127)$$

The pdf and cdf of the second component (Lindley with parameter Φ) in Equations 3.128 and 3.127.

$$f_2(t) = \frac{\Phi^2}{\Phi+1} (1+x) e^{-\Phi x} . \quad (3.128)$$

$$F_1(t) = \begin{cases} \left[1 + \frac{\Phi x}{\Phi+1}\right] e^{-\Phi x}, & t \leq 0 \\ 1 - \left[1 + \frac{\Phi x}{\Phi+1}\right] e^{-\Phi x}, & t > 0 \end{cases} \quad (3.129)$$

Additionally, Equations 3.128 and 3.129 provide the pdf and cdf of the two-component mixture probability model.

$$f(t) = \sigma_1 f_1(t) + \sigma_2 f_2(t), \quad (3.130)$$

where θ_1 and θ_2 are defined as weights, hence $0 < \sigma_1, \sigma_2 < 1$, and $\sigma_1 + \sigma_2 = 1$

$$f(t) = \sigma_1 \frac{\theta^2}{\theta+1} (1+x) e^{-\theta x} + \sigma_2 \frac{\phi^2}{\phi+1} (1+x) e^{-\phi x}. \quad (3.131)$$

$$F(t) = \sigma_1 F_1(t) + \sigma_2 F_2(t) \quad (3.132)$$

$$F(t) = \begin{cases} \sigma_1 \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x} + \sigma_2 \left[1 + \frac{\phi x}{\phi+1}\right] e^{-\phi x}, & t \leq 0 \\ \sigma_1 \left[1 - \left[1 + \frac{\theta x}{\theta+1}\right] e^{-\theta x}\right] + \sigma_2 \left[1 - \left[1 + \frac{\phi x}{\phi+1}\right] e^{-\phi x}\right], & t > 0 \end{cases} \quad (3.133)$$

Combination cumulative distribution function, θ & $\phi > 0$ are parameters. Using MDX, from both of your parents $t^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_n^{(1)})$ and $t^{(2)} = (t_1^{(2)}, t_2^{(2)}, \dots, t_n^{(2)})$, two offspring $u^{(1)} = (u_1^{(1)}, u_2^{(1)}, \dots, u_n^{(1)})$ and $u^{(2)} = (u_1^{(2)}, u_2^{(2)}, \dots, u_n^{(2)})$ are acquired by following a step-by-step process.

Step-1: Use a uniform distribution to generate a random number with a range of units.

Step-2: Using the Mixture probability distribution to generate random numbers, find the parametric value by simply reversing the CDF as shown in Equation 3.133:

$$\beta_t = \begin{cases} \sigma_1 \left[\frac{-\ln(y-1)(\theta+1)}{\theta^2} \right]^{\frac{1}{2}} + \sigma_2 \left[\frac{-\ln(y-1)(\phi+1)}{\phi^2} \right]^{\frac{1}{2}}, & r_j \leq 0.5 \\ \sigma_1 \left[-\frac{\ln(y-2)(\theta+1)}{\theta} \right]^{\frac{1}{2}} + \sigma_2 \left[-\frac{\ln(y-2)(\phi+1)}{\phi} \right]^{\frac{1}{2}}, & r_j > 0.5 \end{cases} \quad (3.134)$$

Step-3: Finally, the two off-springs for $j = 1, 2, \dots, n$ are determined through subsequent Equations 3.135 and 3.136 respectively.

$$u_j^{\{1\}} = \frac{(t_j^{\{1\}} + t_j^{\{2\}}) + \beta_t |t_j^{\{1\}} - t_j^{\{2\}}|}{2}, \quad (3.135)$$

$$u_j^{\{2\}} = \frac{(t_j^{\{1\}} + t_j^{\{2\}}) - \beta_t |t_j^{\{1\}} - t_j^{\{2\}}|}{2}. \quad (3.136)$$

Whenever children in MDX are born outside of the variable limits i.e. $u_j < u_j^l$ or $u_j < u_j^{rj}$, then, at an interval, random values are presented $[u_j^l, u_j^{rj}]$.

3.5.2.2 The proposed mixture of Rayleigh probability distribution

We suggest a parent-centric crossover operator in the next section based on the two-component mixture of Rayleigh probability distributions with parameters and Rayleigh probability distributions with parameters (Thakur, 2014).

Equations 3.137 and 3.138 provide the probability density function (pdf) and cumulative distribution function (cdf) of the first component (Rayleigh with parameter β).

$$f_1(t) = \frac{x}{\beta} e^{-\frac{1}{2}(\frac{x}{\beta})^2}. \quad (3.137)$$

$$F_1(t) = y = \begin{cases} e^{-\frac{1}{2}(\frac{x}{\beta})^2}, & x < 0 \\ 1 - e^{-\frac{1}{2}(\frac{x}{\beta})^2}, & x > 0 \end{cases} \quad (3.138)$$

Equations 3.139 and 3.140 provide the pdf and cdf for the second component (Rayleigh with parameter α) of the mixing function.

$$f_2(t) = \frac{x}{\alpha} e^{-\frac{1}{2}(\frac{x}{\alpha})^2}. \quad (3.139)$$

$$F_2(t) = y = \begin{cases} e^{-\frac{1}{2}(\frac{x}{\alpha})^2}, & x < 0 \\ 1 - e^{-\frac{1}{2}(\frac{x}{\alpha})^2}, & x > 0 \end{cases} \quad (3.140)$$

Additionally, Equations 3.142– 3.144 provide the pdf and cdf of the two-component mixture probability model.

$$f(t) = \sigma_1 f_1(t) + \sigma_2 f_2(t), \quad (3.141)$$

where θ_1 and θ_2 are defined as weights, hence $0 < \sigma_1, \sigma_2 < 1$, and $\sigma_1 + \sigma_2 = 1$

$$f(t) = \sigma_1 \frac{x}{\beta} e^{-\frac{1}{2}(\frac{x}{\beta})^2} + \sigma_2 \frac{x}{\alpha} e^{-\frac{1}{2}(\frac{x}{\alpha})^2}. \quad (3.142)$$

$$F(t) = \sigma_1 F_1(t) + \sigma_2 F_2(t) \quad (3.143)$$

$$F(t) = \begin{cases} \sigma_1 [e^{-\frac{1}{2}(\frac{x}{\beta})^2}] + \sigma_2 [e^{-\frac{1}{2}(\frac{x}{\alpha})^2}], & t \leq 0 \\ \sigma_1 [1 - e^{-\frac{1}{2}(\frac{x}{\beta})^2}] + \sigma_2 [1 - e^{-\frac{1}{2}(\frac{x}{\alpha})^2}], & t > 0 \end{cases} \quad (3.144)$$

Cumulative distribution function in a mixture, α & $\beta > 0$ is the parameters. Using MDX, from both of your parents $t^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_n^{(1)})$ and $t^{(2)} = (t_1^{(2)}, t_2^{(2)}, \dots, t_n^{(2)})$, two offspring $u^{(1)} = (u_1^{(1)}, u_2^{(1)}, \dots, u_n^{(1)})$ and $u^{(2)} = (u_1^{(2)}, u_2^{(2)}, \dots, u_n^{(2)})$ are acquired in a step-by-step process.

Step-1: Use a uniform distribution to generate a random number with a range of units.

Step-2: Find the parametric value by simply inverting the cumulative distribution function using the Mixture probability distribution to produce random numbers as stated in Equation 3.137:

$$\beta_t = \begin{cases} \sigma_1 [\beta(-2 \ln y)^{\frac{1}{2}}] + \sigma_2 [\alpha(-2 \ln y)^{\frac{1}{2}}], & r_j \leq 0.5 \\ \sigma_1 [\beta(-2 \ln (1 - y))^{\frac{1}{2}}] + \sigma_2 [\alpha(-2 \ln (1 - y))^{\frac{1}{2}}], & r_j > 0.5 \end{cases} \quad (3.138)$$

Step-3: Finally, the two off-springs for $j = 1, 2, \dots, n$ are established by the ensuing Equations 3.139 and 3.140, respectively.

$$u_j^{\{1\}} = \frac{(t_j^{\{1\}} + t_j^{\{2\}}) + \beta_t |t_j^{\{1\}} - t_j^{\{2\}}|}{2}, \quad (3.139)$$

$$u_j^{\{2\}} = \frac{(t_j^{\{1\}} + t_j^{\{2\}}) - \beta_t |t_j^{\{1\}} - t_j^{\{2\}}|}{2}. \quad (3.140)$$

If MDX offspring are produced beyond of the variable limits i.e. $u_j < u_j^l$ or $u_j < u_j^r$, then, at an interval, random values are presented $[u_j^l, u_j^r]$.

3.6 Benchmark functions:

There isn't a set formula for determining the effectiveness of GA by selecting the right optimization function. As a result, the algorithmic performance depends on the problem's nature, including the objective function's variation rate, the number of local optima, etc. (Srinivas & Patnaik, 1994). Multiple local optima make up a multimodal function. When looking for global optimal locations, the effective search method must be adept at removing the area surrounding the local optimum. In the search space, the case is more complicated when there are random local optima patterns. Another important element that significantly complicates the challenge is the search space's dimensionality. Friedman (1994) conducted a thorough investigation into the dimensionality problem and its characteristics.

We now offer a set of ten unconstrained optimization test issues that can be used to gauge how well optimization works. We now offer a set of 10 unconstrained optimization test issues that may be used to assess how well optimization methods perform.

1- Rosenbrock Function

$$f_1(y) = \sum_{i=1}^{D-1} [100(y_{i+1} - y_i^2)^2 + (y_i - 1)^2]$$

subject to $-30 \leq x_i \leq 30$. The global minima is located at $x^* = f(1, \dots, 1)$,
 $f(x^*) = 0$

2- Cosine Mixture Function

$$f_2(y) = -0.1 \sum_{i=1}^n \cos(5\pi y_i) - \sum_{i=1}^n y_i^2$$

subject to $-1 \leq y_i \leq 1$. The global minimum is located at $y^* = f(0, 0)$,
 $f(y^*) = (0.2 \text{ or } 0.4)$ for $n = 2$ and 4 respectively.

3- Schwefel Function

$$f_3(y) = (\sum_{i=1}^D x_i^2)^\alpha$$

where $\alpha \geq 0$, subject to $-100 \leq y_i \leq 100$. The global minima is located at

$$y^* = f(0, \dots, 0), f(y^*) = 0.$$

4- Sphere Function

$$f_4(y) = \sum_{i=1}^D y_i^2$$

subject to $0 \leq y_i \leq 10$. The global minima are located $y^* = f(0, \dots, 0), f(y^*) = 0$

5- Styblinski-Tang Function

$$f_5(y) = \frac{1}{2} \sum_{i=1}^n (y_i^4 - 16y_i^2 + 5y_i)$$

subject to $-5 \leq y_i \leq 5$. The global minimum is located $y^* = f(-2.903534, -2.903534), f(y^*) = -78.332$.

6- Sum of Power

$$f_6(y) = \sum_{i=1}^n |y_i|^{(n+1)}$$

subject to $-1 \leq y_i \leq 1$. The global minimum is $f(y^*) = 0$.

7- Drop-wave

$$f_7(y) = \sum_{i=1}^n \frac{1 + \cos\left(12\sqrt{y_i^2 + y_{i+1}^2}\right)}{0.5(y_i^2 + y_{i+1}^2) + 2}$$

subject to $-5.12 \leq y_i \leq 5.12$. The global minimum is $f(y^*) = -1$.

8- Rastrigin

$$f_8(y) = 10n + \sum_{i=1}^n [y_i^2 - 10 \cos(2\pi y_i)]$$

subject to $-5.12 \leq y_i \leq 5.12$. The global minimum is $f(y^*) = 0$

9- Levy and Mantalvo

$$f_9(y) = 0.1$$

subject to $-5 \leq y_i \leq 5$. The global minimum is $f(y^*) = 0$

10- New function

$$f_{10}(y) = \sum_{i=0}^n [0.2y_i^2 + 0.1x_i^2 \sin(2y_i)]$$

subject to $-10 \leq y_i \leq 10$. The global minimum is $f(y^*) = 0$.

Chapter 4

Results and Discussion

4.1 Experimental setup

In the current chapter, a newly suggested set of crossover operators, namely exponential, Laplace, Fisk, Gompertz, Weibull, Double Pareto, Rayleigh, Mix-Rayleigh, Lindley, and Mix-Lindley crossovers are closely compared to improve the performance of the genetic process. The performance of these operators is assessed using several well-known benchmark functions. To assess their overall effectiveness, these ten crossover operators were merged alongside MPTM, PM, and NUM mutation operators. The findings of hypothetical research into algorithmic combinations are compiled in Table 4.1, together with the related crossover and mutation probability and final parametric values. By properly changing parametric parameters, the empirical inquiry can produce the best findings.

Thirty independent runs of each method were used to simulate the results. Each of these algorithms has a population size that is ten times the number of decision factors. The entire GA algorithmic method uses tournament selection and elitism with size one. All studies come to an end once the number of generations reaches 300 and the best outcomes for the GA process have been discovered using trial runs and screening experiments.

Table 4.1: Information about the parameters for all algorithms

Crossover Operators	Mutation Operators	Selection operator	Crossover Probability	Mutation Probability
EX	MTPM	Tournament	0.70	0.02
LX	MTPM	Tournament	0.70	0.02
FX	MTPM	Tournament	0.70	0.02
GX	MTPM	Tournament	0.70	0.02
WBX	MTPM	Tournament	0.70	0.02
DPX	MTPM	Tournament	0.70	0.02
RHX	MTPM	Tournament	0.70	0.02
MIX-RHX	MTPM	Tournament	0.70	0.02
LDX	MTPM	Tournament	0.70	0.02
MIX-LDX	MTPM	Tournament	0.70	0.02
EX	PM	Tournament	0.65	0.0
LX	PM	Tournament	0.65	0.005
FX	PM	Tournament	0.65	0.005
GX	PM	Tournament	0.65	0.005
WBX	PM	Tournament	0.65	0.005
DPX	PM	Tournament	0.65	0.005
RHX	PM	Tournament	0.65	0.005
MIX-RHX	PM	Tournament	0.65	0.005
LDX	PM	Tournament	0.65	0.005
MIX-LDX	PM	Tournament	0.65	0.005
LX	NUM	Tournament	0.70	0.01
FX	NUM	Tournament	0.70	0.01
GX	NUM	Tournament	0.70	0.01
WBX	NUM	Tournament	0.70	0.01
DPX	NUM	Tournament	0.70	0.01
RHX	NUM	Tournament	0.70	0.01
MIX-RHX	NUM	Tournament	0.70	0.01
LDX	NUM	Tournament	0.70	0.01
MIX-LDX	NUM	Tournament	0.70	0.01

The mean, standard deviation, and average time to complete in seconds are utilized as the final results after thirty runs of each algorithm to guarantee compatibility and effectiveness. Using MATLAB (version R2018a), the effectiveness of the recently suggested real-coded crossover system connected with multiple probability distributions is assessed on 10 benchmark functions.

4.2 Simulation study

Our primary contribution to the current empirical study is the introduction of new real-coded crossover operators, and the major goal of our chapter is to assess the effectiveness of suggested crossover operators in light of a simulation's output. In light of the findings in Tables 4.2, 4.3, and 4.4 we now compare the recently proposed crossover operator Exponential (EX), Laplace(LX), Fisk(FX), Gompertz(GX), Weibull(WBX), Double Pareto(DPX), Rayleigh(RHX), Mix-Rayleigh(MIX-RH), Lindley(LDX), and Mix-Lindley (MIX-LDX) based on mean, standard deviation (SD). The empirical findings are often seen as being close to the theoretical optimum value in the majority of the multi-modal test situations, which indicates the increased performance of the recently introduced crossovers system. The performance of MIX-LDX is extraordinarily perfect in terms of mean values, standard deviation, and average execution time. It also aids in overcoming the drawbacks of the GA process, such as Exploitation and research. As a result, the lowest mean values smaller SD, and the shortest typical execution time show superior adequate ability to influence the loss of population diversity and exert control over the selection pressure.

The results for the majority of the benchmark functions, such as Brown, Levy-Mont, Schwefel etc., have the lowest mean, standard deviation, and average execution time. In ten benchmark functions, MIX-LDX performs best overall, completely dominating the other nine crossover operators.

Mutation and crossover operators can be combined to improve can have better control over the limitations of the GA process. The substantial agreement between empirical findings and theoretically ideal values indicates improved control overpopulation diversity decline, and the short execution time reveals sustained selection pressure. MIX-LDX generally has a higher rate of success in achieving the best outcomes.

4.3 Experimental result and comparison Tables 4.2, 4.3, and 4.4 show the experimental results of a simulation study. After that, we now compare the recently proposed crossover operators on the mean, and standard deviation (SD).

Table 4.2 Statistical results with average execution time for real-coded crossover operators under non-uniform (NUM) operator

Benchmark function			EX_NUM		LX_NUM		FX_NUM		GX_NUM	
			Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	5.93×10^3		167.684288	4.13×10^2	175.766024	1.50×10^5	179.1205	1.30×10^7	180.713089
	S.D	6.33×10^3			225.1067		9.50×10^4		5.33×10^6	
Sum of power	Mean	2.64×10^{-22}		169.459301	1.62×10^{-26}	167.481237	9.25×10^{-16}	171.548068	8.51×10^{-10}	179.893017
	S.D	7.10×10^{-22}			4.89×10^{-26}		2.86×10^{-15}		1.01×10^{-9}	
Comix	Mean	-3.0301		163.800367	-3.4776	163.626646	-0.1463	167.298261	4.6664	179.684954
	S.D	0.6281			0.4646		1.0744		0.6538	
Dropwave	Mean	-0.0815		176.965646	-0.0971	181.789447	-0.0222	186.088342	-0.0166	197.394081
	S.D	0.0191			0.0289		0.0018		0.0029	
Rastrigin	Mean	53.4877		153.113432	28.1633	159.385567	93.0881	152.803468	154.6082	158.76067
	S.D	20.1275			10.7808		21.637		21.7449	
Schwefel_1	Mean	-7.1×10^{73}		391.594946	-2.09×10^{298}	438.874478	0	955.511013	0	987.167061
	S.D	1.94×10^{74}			0		0		0	
LevyMont	Mean	4.00×10^3		154.837905	1.48×10^{-4}	157.851578	1.24×10^{-1}	161.698655	3.96×10^{-9}	164.60878
	S.D	1.60×10^{-3}			2.21×10^{-4}		4.05×10^{-2}		8.50×10^{-2}	
Sphere	Mean	0.8647		203.45434	0.1456	169.203228	24.5545	219.837855	146.5608	197.697561
	S.D	0.2296			0.0715		5.2293		35.077	
New function	Mean	4.88×10^0		166.145614	4.29×10^0	168.729321	93.9093	177.259798	109.8264	178.589035
	S.D	3.5149			2.6293		15.7903		18.448	
Stybin	Mean	-1.58×10^3		1	-1.63×10^3	195.390545	-1.31×10^3	194.121858	-1.11×10^3	197.382513
	S.D	167.7244		94.169457	176.0682		41.605		57.7408	

Benchmark function		WBX_NUM		DPX_NUM		RHX_NUM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	1.72×10^5	258.799823	277.0033	155.29667	3.44×10^4	158.780714
	S.D	2.81×10^4		175.7451		3.80×10^4	
Sum of power	Mean	2.00×10^{-16}	274.165459	3.16×10^3	170.60072	2.17×10^{-16}	167.882228
	S.D	3.14×10^{-16}		6.69×10^{-30}		6.79×10^{-16}	
Comix	Mean	0.0658	291.306924	-4.5413	227.985766	0.8474	175.373032
	S.D	0.8721		0.3039		1.3015	
Dropwave	Mean	-0.0491	294.353192	-0.1876	171.760614	-0.0251	169.9984
	S.D	0.0136		0.04		0.004	
Rastrigin	Mean	174.6423	228.315932	21.9114	145.468766	87.5759	148.541292
	S.D	36.0267		7.7962		18.0486	
Schwefel_1	Mean	-2.06×10^4	412.075176	-3.90×10^{103}	487.433046	-2.9×10^{167}	471.992942
	S.D	96.0052		1.23×10^{104}		0	
LevyMont	Mean	6.84×10^2	248.033451	8.87×10^{-5}	167.118023	8.15×10^{-2}	172.412453
	S.D	1.45×10^2		8.70×10^{-5}		2.81×10^{-2}	
Sphere	Mean	5.9335	283.432103	0.1007	170.93084	1.06×10^1	172.180698
	S.D	0.504		0.0548		3.69	
New function	Mean	16.7138	266.099538	0.95	167.962267	88.6951	176.502337
	S.D	11.1814		0.9477		13.4932	
Stybin	Mean	-1.15×10^3	286.913724	-1.74×10^3	203.012492	-1.35×10^3	201.73396
	S.D	57.9452		294.6934		62.2708	

Benchmark function		MIX-RHX_NUM		LDX_NUM		MIX-LDX_NUM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	1.86×10^4	171.05885	-8.96085	289.459665	-0.0612	120.160925
	S.D	1.15×10^4		1.23×10^7		1.03×10^6	
Sum of power	Mean	8.47×10^{-3}	169.99411	5.86×10^{-7}	265.470348	-0.0568	159.596366
	S.D	1.77×10^{-31}		1.54×10^6		87.0381	
Comix	Mean	-3.7917	171.402312	-0.95956	365.566905	-0.2416	244.126287
	S.D	0.1979		0.3348		1.33×10^{-8}	
Dropwave	Mean	-0.4126	167.266836	-0.9563	307.07506	-23.11596	699.15023
	S.D	0.0561		0.0021		1.55×10^{-6}	
Rastrigin	Mean	45.0196	150.686225	-0.036	313.097998	-37.19823	166.734431
	S.D	8.5521		2.9841		4.4425	
Schwefel_1	Mean	-2.11×10^4	193.626539	1.23×10^7	453.097295	-173.364	158.631295
	S.D	1.72×10^3		4.2954		3.7857	
LevyMont	Mean	1.09×10^{-3}	159.146871	-0.2177	334.510294	-0.4049	155.496675
	S.D	2.00×10^3		0.0037		0.9173	
Sphere	Mean	2.7462	169.266415	2.7462	287.39287	-0.2728	218.008955
	S.D	0.4435		82.3281		2.35×10^{-6}	
New function	Mean	2.0259	169.342671	-24.2396	299.190393	-0.2198	150.480026
	S.D	0.3601		103.6608		62.9765	
Stybin	Mean	-1.92×10^3	198.130373	-0.5028	307.887905	-0.4037	259.340812
	S.D	5.3789		36.4346		1.1707	

Table 4.3 Statistical results with average execution time for real-coded crossover operators under PM operator

Benchmark function		EX_PM		LX_PM		FX_PM		GX_PM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	6.25×10^4	163.46210	2.5×10^3	162.08191	2.65×10^7	165.543659	2.66×10^7	170.152305
	S.D	4.94×10^4		1.62×10^3		7.69×10^6		1.14×10^7	
Sum of power	Mean	5.2×10^{-15}	171.78532	2.77×10^{-18}	181.466446	4.16×10^{-7}	178.498651	5.39×10^{-7}	186.022443
	S.D	1.6×10^{-14}		8.68×10^{-18}		7.30×10^{-7}		9.48×10^{-7}	
Comix	Mean	-2.2474	171.53317	-3.3336	165.57412	7.2081	176.990036	7.6209	181.640508
	S.D	0.3748		0.5606		0.8713		0.6504	
Dropwave	Mean	-0.0686	166.87929	-0.0622	167.79054	-0.0159	185.72633	-0.0148	190.204647
	S.D	0.0194		0.0152		0.0019		0.0013	
Rastrigin	Mean	53.3957	151.25310	34.0673	153.241901	99.268	153.713715	146.6936	160.84213
	S.D	13.7044		13.7028		32.0707		14.591	
Schwefel_1	Mean	-4.2×10^{72}	429.38897	-1.26×10^{293}	471.975257	0	452.048628	0	471.938488
	S.D	8.4×10^2		0		0		0	
LevyMont	Mean	5.7×10^{-3}	161.52567	4.27×10^{-4}	160.60487	6.23×10^{-1}	161.329326	6.44×10^1	193.663728
	S.D	2.7×10^{-3}		3.75×10^{-4}		7.84×10^{-2}		1.73×10^{-1}	
Sphere	Mean	11.984	172.07388	3.7819	164.353894	203.3505	179.107724	204.6509	179.682187
	S.D	3.4141		1.7236		24.0231		34.8761	
New function	Mean	1.2×10^1	164.28504	6.30	163.602792	134.9782	174.781195	138.5133	181.08835
	S.D	5.1007		2.2658		15.0404		11.6234	
Stybin_30	Mean	-1.4×10^3	205.71562	-1.63×10^3	202.9256	-9.26×10^2	196.63524	-9.28×10^2	201.961715
	S.D	102.121		117.4468		64.0525		85.0232	

Benchmark function		WBX-PM		DPX-PM		RHX-PM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	5.35×10^6	245.06581	1.67×10^3	151.837394	2.54×10^7	163.740098
	S.D	2.85×10^{-6}		875.4831		8.59×10^6	
Sum of power	Mean	4.89×10^{-10}	279.529353	1.48×10^{-25}	178.358576	2.51×10^{-7}	179.149916
	S.D	8.11×10^{-10}		3.09×10^{-25}		3.83×10^{-7}	
Comix	Mean	2.3989	264.57751	-4.3631	178.003804	6.5902	178.125272
	S.D	1.3552		0.3268		0.957	
Dropwave	Mean	-0.0244	307.026164	-0.1618	178.773241	-0.0163	186.771657
	S.D	0.0044		0.0376		0.0019	
Rastrigin	Mean	170.1377	261.71645	19.3824	150.625601	218.7732	153.441858
	S.D	18.853		5.3466		22.2148	
Schwefel_1	Mean	-2.57×10^4	352.536309	-8.89×10^{103}	437.845064	-1.5×10^{159}	441.826076
	S.D	1.26×10^4		2.20×10^{104}		0	
LevyMont	Mean	1.43×10^{-1}	260.177683	4.16×10^{-4}	154.668478	5.89×10^{-1}	157.910846
	S.D	4.71×10^{-2}		4.24×10^{-4}		7.85×10^{-2}	
Sphere	Mean	75.0158	274.032859	1.7235	165.052555	2.15×10^3	179.75951
	S.D	25.6199		0.7079		17.7523	
New function	Mean	56.1569	281.284003	2.1566	173.187237	121.9712	176.598544
	S.D	8.7151		1.0898		11.8028	
Stybin	Mean	-1.11×10^3	280.143056	-1.73×10^3	201.30569	-9.7×10^2	197.383047
	S.D	49.4334		153.693		97.0128	

Benchmark function		MIX-RHX-PM		LDX-PM		MIX-LDX-PM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	1.9×10^4	157.845271	-8.1221	315.1730323	-2	141.646719
	S.D	9.0×10^3		2.88×10^7		1.98×10^6	
Sum of power	Mean	5.78×10^{-30}	176.832127	2.91×10^{-6}	329.213969	-0.8913	160.577198
	S.D	1.7×10^{-29}		3.12×10^{-6}		100.4508	
Comix	Mean	-3.5697	170.535981	-4.884	361.906268	-0.2408	282.147397
	S.D	0.2093		0.1791		0.3031	
Dropwave	Mean	-0.4343	170.284267	-1284.816	382.071542	-0.00529	314.546248
	S.D	0.0561		5.84×10^{-7}		8.88×10^{-8}	
Rastrigin	Mean	38.8096	148.803958	-0.0686	329.127048	-0.137	323.347333
	S.D	11.0502		3.8743		7.90×10^{-8}	
Schwefel_1	Mean	-2.1×10^4	182.916744	-0.10745	467.651344	-0.01121	163.492746
	S.D	6.1×10^3		7.3876		3.71	
LevyMont	Mean	8.0×10^{-3}	155.380448	-0.581126	349.754331	-1.985	164.685817
	S.D	2.50×10^{-3}		0.0038		1.0179	
Sphere	Mean	3.5523	179.193524	-463.1427	285.515849	-2.91	210.041438
	S.D	0.9818		31.5251		4.29×10^{-5}	
New function	Mean	2.1722	180.589433	-91.71	313.001526	-0.4787	173.687994
	S.D	0.4613		63.2155		123.1715	
Stybin	Mean	-1.73×10^3	201.30569	-75.37	315.559643	-0.7278	295.164013
	S.D	153.693		117.31		1.728	

Table 4.4 Statistical results with average execution time for real-coded crossover operators under MTPM operator

Benchmark function		EX_MTPM		LX_MTPM		FX_MTPM		GX_MTPM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	8.0919	147.593903	1.16×10^1	149.162019	23.1062	149.810878	16.8994	149.868177
	S.D	11.272		25.581		18.9519		11.5221	
Sum of power	Mean	1.22×10^{-52}	198.955852	9.53×10^{-65}	185.790405	4.80×10^{-63}	181.555035	1.74×10^{-65}	179.517906
	S.D	3.86×10^{-52}		2.85×10^{-64}		1.52×10^{-62}		5.48×10^{-65}	
Comix	Mean	-4.9986	174.074515	-4.9984	174.141233	3.4883	184.711101	5.2474	181.987489
	S.D	0.0011		0.0027		5.8862		5.4952	
Dropwave	Mean	-0.8745	169.925627	-0.9212	163.659595	-0.3569	179.846841	-0.7509	170.349722
	S.D	0.1093		0.0476		0.4465		0.2881	
Rastrigin	Mean	51.2472	158.518858	26.7216	157.591976	29.9069	161.707928	41.1907	152.458018
	S.D	26.7388		14.519		55.4497		39.7542	
Schwefel_1	Mean	-2.03×10^{73}	425.326862	0	437.154413	0	505.769228	0	452.33055
	S.D	3.51×10^{73}		0		0		0	
LevyMont	Mean	1.03×10^{-4}	158.149296	2.23×10^{-5}	158.609777	2.50×10^{-4}	162.159026	2.57×10^{-4}	165.413622
	S.D	8.67×10^{-5}		3.63×10^{-5}		4.54×10^{-4}		2.81×10^{-4}	
Sphere	Mean	0.0232	163.639832	0.0037	160.197695	165.7529	171.183592	65.047	167.44013
	S.D	0.0351		0.0088		89.7803		105.4691	
New function	Mean	1.05	156.200746	5.10×10^{-3}	156.603277	105.8369	174.901432	95.3769	164.627476
	S.D	3.3165		0.0074		61.0151		66.2115	
Stybin_30	Mean	-1.96×10^3	215.16127	-1.96×10^3	204.104035	-1.85×10^3	205.664437	-1.96×10^3	202.910138
	S.D	0.0372		0.035		326.7217		0.2395	

Benchmark function		WBX_MTPM		DPX_MTPM		RHX_MTPM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	16.6257	226.183681	11.0969	162.504817	1.49×10^1	161.948849
	S.D	12.9211		15.6503		16.0384	
Sum of power	Mean	3.3510^{-61}	292.24881	5.92×10^{-62}	196.121417	2.52×10^{-63}	192.492527
	S.D	1.0310^{-60}		1.87×10^{-61}		7.76×10^{-63}	
Comix	Mean	-4.8588	271.685064	-4.9944	165.627418	1.5848	173.229963
	S.D	0.2385		0.0087		6.9299	
Dropwave	Mean	-0.8007	283.605003	-0.9044	166.930912	-0.4398	172.865099
	S.D	0.2644		0.0673		0.4503	
Rastrigin	Mean	169.9814	248.729509	21.5264	151.498334	85.7084	157.250182
	S.D	21.4868		7.6258		70.4028	
Schwefel_1	Mean	-5.6410^4	456.608099	-3.44×10^{102}	446.011425	-6.85×10^{156}	426.419999
	S.D	2.4710^4		9.13×10^{102}		0	
LevyMont	Mean	1.82	280.128907	3.84×10^{-5}	158.14717	2.00×10^{-4}	162.496964
	S.D	1.70×10^{-4}		7.28×10^{-5}		3.05×10^{-4}	
Sphere	Mean	0.0081	262.941077	0.0026	172.294767	3.96×10^1	185.763037
	S.D	0.0089		0.0046		83.5382	
New function	Mean	3.4436	252.372534	0.006	161.992285	75.0585	180.015291
	S.D	7.2689		0.0122		65.2743	
Stybin	Mean	-1.96×10^3	301.709394	-1.96×10^3	202.766787	-1.96×10^3	202.058918
	S.D	0.151		0.0224		0.2752	

Benchmark function		MIX-RHX-MTPM		LDX-MTPM		MIX-LDX-MTPM	
		Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)	Statistical measures	Average execution time(sec)
Rosenbrock	Mean	23.946	162.406372	-2.643	265.000467	-88.7	125.863363
	S.D	32.3466		1.04×10^3		6.39×10^3	
Sum of power	Mean	1.3×10^{-55}	193.457046	2.36×10^{-34}	278.488179	-0.35	161.943644
	S.D	4.1×10^{-55}		7.43×10^{-34}		34.0683	
Comix	Mean	-4.989	174.805932	-0.4828	364.302135	-3.50	251.500144
	S.D	0.0123		0.4257		2.64×10^{-7}	
Dropwave	Mean	-0.8286	159.797017	-0.000688	333.87314	-4.19×10^1	282.925485
	S.D	0.1634		1.15×10^{-6}		5.30×10^8	
Rastrigin	Mean	40.4273	154.974261	-21.711	349.089728	-0.0838	291.882632
	S.D	7.4103		3.1526		2.29×10^{-8}	
Schwefel_1	Mean	-7.1×10^5	367.987236	-0.09	457.393249	-0.22	160.624501
	S.D	1.03×10^6		6.4195		3.5942	
LevyMont	Mean	3.2×10^{-4}	167.933571	-676	339.902982	-1505.44	149.692485
	S.D	4.7×10^{-4}		0.0056		0.4497	
Sphere	Mean	0.0138	174.679748	-1785.148504	285.210813	-0.158	211.042559
	S.D	0.2		100.5998		3.76×10^{-6}	
New function	Mean	0.011	163.943794	-0.77	298.111614	-0.032	154.433302
	S.D	0.006		66.3839		56.545	
Stybin	Mean	-1.9×10^3	200.83355	-180.24	234.345678	-0.032	280.074874
	S.D	0.1873		77.1234		1.789	

4.3.1 EX-NUM VS LX-NUM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for LX that are lower than those for EX-NUM, except for Comix, Sphere, and Stybin benchmark functions. Compared to EX-NUM, LX-NUM has a smaller mean and standard deviation (Table 4.2).

4.3.2 LX-NUM VS FX-NUM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, LX-NUM is superior to FX-NUM. It implies that the LX outperforms the FX in terms of close-to-ideal results (Table 4.2).

4.3.3 FX-NUM VS GX-NUM

GX-NUM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to FX-NUM, the suggested algorithm is superior.

4.3.4 GX-NUM VS WBX-NUM

Ten benchmark functions were successfully solved by ten algorithms. All problems have smaller mean values and standard deviations for WBX-NUM than for GX-NUM, except for Rosenbrick, Schwefel-1, and Levy Mont benchmark functions. Compared to GX-NUM, WBX-NUM has a smaller mean and standard deviation (Table 4.2).

4.3.5 WBX-NUM VS DPX-NUM

DPX-NUM displays slightly better results than the 10 benchmark functions, with a lower mean value. The suggested method performs significantly better than WBX-NUM.

4.3.6 DPX-NUM VS RHX-NUM

In comparison to the 10 benchmark functions, the results for DPX-NUM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to RHX-NUM.

4.3.7 RHX-NUM VS MIX-RHX-NUM

Ten benchmark functions were successfully solved by ten algorithms. MIX-RHX consistently has a mean value and standard deviation that are lower than RHX-NUM, except for the Sum of Power and New function. MIX-RH-NUM has a lower mean and standard deviation than RHX-NUM (Table 4.2).

4.3.8 MIX-RHX-NUM VS LDX-NUM

In comparison to the 10 benchmark functions, the results for LDX-NUM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to MIX-RHX-NUM.

4.3.9 LDX-NUM VS MIX-LDX-NUM

When compared to the 10 benchmark functions, MIX-LD-NUM's mean values are somewhat lower, which suggests a marginally better outcome. Direct comparison shows that the suggested method outperforms LDX-NUM.

4.3.10 EX-NUM VS MIX-LDX-NUM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for MIX-LDX that are lower than those for EX-NUM, except for Levy Mont, and Stybin benchmark function. Compared to EX-NUM, MIX- LX-NUM has a smaller mean and standard deviation (Table 4.2).

4.3.11 LX-NUM VS MIX-LDX-NUM

In comparison to the 10 benchmark functions, the results for MIX- LDX-NUM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to LX-NUM.

4.3.12 FX-NUM VS MIX-LDX-NUM

When compared to the 10 benchmark functions, MIX-LD-NUM's mean values are somewhat lower, which suggests a marginally better outcome. Direct comparison shows that the suggested method outperforms FX-NUM.

4.3.13 GX-NUM VS MIX-LDX-NUM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for MIX-LDX that are lower than those for GX-NUM, except for the Stybin benchmark function. Compared to GX-NUM, MIX-LDX-NUM has a smaller mean and standard deviation (Table 4.2).

4.3.14 WBX-NUM VS MIX-LDX-NUM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, MIX-LDX-NUM is superior to WBX-NUM. It implies that the MIX-LDX outperforms the WBX in terms of close-to-ideal results (Table 4.2).

4.3.15 DPX-NUM VS MIX-LD-NUM

MIX-LDX-NUM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to DPX-NUM, the suggested algorithm is superior.

4.3.16 RHX-NUM VS MIX-LDX-NUM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for MIX-LDX-NUM that are lower than those for RHX-NUM, except for the Stybin benchmark function. Compared to RHX-NUM, MIX-LDX-NUM has a smaller mean and standard deviation (Table 4.2).

4.3.17 MIX-RHX-NUM VS MIX-LDX-NUM

MIX-LDX-NUM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to MIX-RHX-NUM, the suggested algorithm is superior.

4.3.18 LDX-NUM VS MIX-LDX-NUM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, MIX-LDX-NUM is superior to LDX-NUM. It implies that the MIX-LDX outperforms the LDX in terms of close-to-ideal results (Table 4.2).

4.3.19 EX-PM VS LX-PM

In comparison to the 10 benchmark functions, the results for LX-PM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to EX.

4.3.20 LX-PM VS FX-PM

FX-PM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to LX-PM, the suggested algorithm is superior.

4.3.21 FX-PM VS GX-PM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for FX that are lower than those for GX-PM, except for the Drop wave and Stybin benchmark function. Compared to GX-PM, FX-PM has a smaller mean and standard deviation (Table 4.3).

4.3.22 GX-PM VS WBX-PM

In comparison to the 10 benchmark functions, the results for GX-PM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to WBX-PM.

4.3.23 WBX-PM VS DPX-PM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, DPX-PM is superior to WBX-PM. It implies that the DPX outperforms the WBX in terms of close-to-ideal results (Table 4.3).

4.3.24 DPX-PM VS RHX-PM

In comparison to the 10 benchmark functions, the results for DPX-PM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to RHX-PM.

4.3.25 RHX-PM VS MIX-RHX-PM

MIX-RHX-PM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to RHX-PM, the suggested algorithm is superior.

4.3.26 MIX-RH-PM VS LDX-PM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, LDX-PM is superior to MIX-RHX-PM. It implies that the LDX outperforms the MIX-RHX in terms of close-to-ideal results (Table 4.3).

4.3.27 EX-PM VS MIX-LDX-PM

In comparison to the 10 benchmark functions, the results for MIX-LDX-PM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to EX-PM.

4.3.28 LX-PM VS MIX-LDX-PM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, MIX-LDX-PM is superior to LX-PM. It implies that the LX outperforms the EX in terms of close-to-ideal results (Table 4.3).

4.3.29 FX-PM VS MIX-LDX-PM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for MIX-LDX that are lower than those for FX-PM, except for the Drop wave Stybin benchmark function. Compared to FX-PM, MIX-LDX-PM has a smaller mean and standard deviation (Table 4.3).

4.3.30 GX-PM VS MIX-LDX-PM

MIX-LDX-PM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to GX-PM, the suggested algorithm is superior.

4.3.31 WBX-PM VS MIX-LDX-PM

In comparison to the 10 benchmark functions, the results for MIX-LDX-PM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to WBX-PM.

4.3.32 DPX-PM VS MIX-LDX-PM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for MIX-LDX that are lower than those for DPX-PM, except for Drop wave, Schwefel-1, and Stybin benchmark function. Compared to DPX-PM, MIX-LDX-PM has a smaller mean and standard deviation (Table 4.3).

4.3.33 RHX-PM VS MIX-LD-PM

MIX-LDX-PM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to RHX-PM, the suggested algorithm is superior.

4.3.34 MIX-RH-PM VS MIX-LD-PM

In comparison to the 10 benchmark functions, the results for MIX- LDX-PM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to MIX- RHX-PM.

4.3.35 LDX-PM VS MIX-LDX-PM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, MIX- LDX-PM is superior to LDX-PM. It implies that the MIX- LDX-PM outperforms the LDX-PM in terms of close-to-ideal results (Table 4.3).

4.3.36 EX-MTPM VS LX-MTPM

In comparison to the 10 benchmark functions, the results for EX-MTPM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to LX-MTPM.

4.3.37 LX-MTPM VS FX-MTPM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for LX that are lower than those for FX-PM, except for the Sum of power and Stybin benchmark function. Compared to FX-PM, LX-PM has a smaller mean and standard deviation (Table 4.4).

4.3.38 GX-MTPM VS WBX-MTPM

In comparison to the 10 benchmark functions, the results for WBX-MTPM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to GX-MTPM.

4.3.39 WBX-MTPM VS DPX-MTPM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for DPX that are lower than those for WBX-MTPM, except for Comix, Sphere, and Stybin benchmark functions. Compared to WBX-MTPM, DPX-MTPM has a smaller mean and standard deviation (Table 4.4).

4.3.40 DPX-MTPM VS RHX-MTPM

DPX-MTPM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to RHX-MTPM, the suggested algorithm is clearly superior.

4.3.41 RHX-MTPM VS MIX-RHX-MTPM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, MIX-RHX-MTPM is superior to RHX-MTPM. It implies that the MIX-RHX outperforms the RHX in terms of close-to-ideal results (Table 4.4).

4.3.42 MIX-RHX-MTPM VS LDX-MTPM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for LDX that are lower than those for MIX-RHX-MTPM, except for Comix, Sphere, and Stybin benchmark functions. Compared to MIX-RHX-MTPM, LDX-MTPM has a smaller mean and standard deviation (Table 4.4).

4.3.43 EX-MTPM VS MIX-LDX-MTPM

MIX-LDXX-MTPM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to EX-MTPM, the suggested algorithm is superior.

4.3.44 LX-MTPM VS MIX-LDX-MTPM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark functions, MIX-LDX-MTPM is superior to LX-MTPM. It implies that the MIX-LDX outperforms the LX in terms of close-to-ideal results (Table 4.4).

4.3.45 FX-MTPM VS MIX-LDX-MTPM

MIX-LDX-MTPM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to FX-MTPM, the suggested algorithm is superior.

4.3.46 GX-MTPM VS MIX-LDX-MTPM

In comparison to the 10 benchmark functions, the results for MIX-LDX-MTPM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to GX-MTPM.

4.3.50 WBX-MTPM VS MIX-LDX-MTPM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for LX that are lower than those for MIX-LDX-

MTPM, except for Comix, Sphere, and Stybin benchmark functions. Compared to WBX-MTPM, MIX-LDX-MTPM has a smaller mean and standard deviation (Table 4.4).

4.3.51 DPX-MTPM VS MIX-LDX-MTPM

In comparison to the 10 benchmark functions, the results for MIX- LDX-MTPM are somewhat superior in terms of reduced mean values. The suggested algorithm is superior to DPX-MTPM.

4.3.52 RHX-MTPM VS MIX-LDX-MTPM

In terms of success rate and decreased mean and standard deviation value across 10 benchmark function, MIX-LDX-MTPM is superior to FX-MTPM. It implies that the MIX-LDX outperforms the RHX in terms of close-to-ideal results (Table 4.4).

4.3.53 MIX-RHX-MTPM VS MIX-LDX-MTPM

Ten methods were successful in solving ten benchmark functions. All problems have mean values and standard deviations for MIX-LDX that are lower than those for MIX-RHX-MTPM, except for Comix, Sphere, and Stybin benchmark functions. Compared to MIX-RHX-MTPM, MIX-LDX-MTPM has a smaller mean and standard deviation (Table 4.4).

4.3.54 LDX-MTPM VS MIX-LDX-MTPM

MIX-LDX-MTPM performs a little bit better than the 10 benchmark functions in terms of a lower mean value. Compared to LDX-MTPM, the suggested algorithm is superior.

Chapter 5

Conclusion

The main objective of our experiment-based study is to evaluate the effectiveness of the mixture distribution crossover (LDX and RHX) in resolving complex unconstrained optimization problems. We assess ten distinct crossover operators, which are combined with three mutation operators. Presently, we propose a new set of crossover operators, namely exponential, Laplace, Fisk, Gompertz, Weibull, Double Pareto, Rayleigh, Mix-Rayleigh, Lindley, and Mix-Lindley crossovers. The performance of these operators is assessed using several well-known benchmark functions like Rosenbrock, Sum of power, Comix, Dropwave, Rastrigin, Schwefel-1, Levy mont, Sphere, New function and Stybin.

The novel real-coded GA is created by integrating the proposed crossover operators with MPTM, NUM, and PM mutation operators. We compare the performance of the suggested operator with ten real-coded algorithms that are proposed by the operator. The implementations of each method are comparable, and the simulation conditions are the same. The suggested operator outperforms the other three crossover operators in terms of providing results close to the ideal values and having a higher success rate.

Furthermore, we introduce an innovative parent-centric real-coded crossover operator that combines Rayleigh and Lindley distributions with different parameters in mixture distributions. The empirical findings show that newly proposed operators achieve significant dominance in terms of statistical significance, computing difficulty, resilience, durability, and exploration and exploitation potential.

To compare the algorithms further, we consider the success rate and minimal median values of the objective functions for each algorithm. Mix-LDX with MPTM, NUM, or PM mutation operator outperforms all other algorithms in terms of success rate and minimal mean objective function values.

References

- Abramowitz, & Stegan. (1972). *An introduction to exponential distribution* . MIT press
- Beasley, D., Bull, D. R., & Martin, R. R. (1993). *An Overview of Genetic Algorithms: Part 1, fundamentals*. *University Computing*, 15(2), 56-69.
- Beasley, J. E., Sonander, J., & Havelock, P. (2001). Scheduling aircraft landings at London Heathrow using a population heuristic. *Journal of the operational Research Society*, 52(5), 483-493.
- Bandyopadhyay, S., & Maulik, U. (2002). Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35(6), 1197- 1208.
- Chatterjee, S., Carrera, C., & Lynch, L. A. (1996). Genetic algorithms and traveling salesman problems. *European Journal of Operational Research*, 93(3), 490-510.
- Chuang, Y. C., Chen, C. T., & Hwang, C. (2016). A simple and efficient real-coded genetic algorithm for constrained optimization. *Applied Soft Computing*, 38, 87-105.
- De Jong, K. (1980). Adaptive system design: a genetic approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(9), 566-574.
- De Jong, K. A., & Sarma, J. (1993). Generation gaps revisited. In *Foundations of genetic algorithms* (Vol. 2, pp. 19-28. Elsevier.
- Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4), 311-338.
- Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. *In Proceedings of the first European conference on artificial life*, 42, 134-142.
- Deb, K., & Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9(2), 115-148.
- Deep, K., & Thakur, M. (2007). A new mutation operator for real coded genetic algorithms. *Applied mathematics and Computation*, 193(1), 211-230.

- Deb, K., & Deb, D. (2014). Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4(1), 1-28
- Eberhart, R. C., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. Elsevier.
- Eshelman, L. J., & Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms*, 2, 187-202. Elsevier.
- Gago-Benítez, A., Fernández-Madrigal, J. A., & Cruz-Martín, A. (2013). Log-logistic modeling of sensory flow delays in networked telerobots. *IEEE Sensors Journal*, 13(8), 2944-2953.
- Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, 69-93. Elsevier.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533-549.
- Goldberg, D. (2002). *The Design of Innovation (Genetic Algorithms and Evolutionary Computation)*. Springer New York.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison.
- Haq, E.U., Ahmad, I., & Hussain, A. (2019). Performance evaluation of novel selection processes through hybridization of k-means clustering and genetic algorithm. *Applied Ecology and Environmental research*, 17(6), 14159-77.
- Haq, E.U., Ahmad, I., & Hussain, A. (2019). Development a New Crossover Scheme for Traveling Salesman Problem by aid of Genetic Algorithm. *International Journal of Intelligent Systems and Applications*, 11(12), 46-52.
- Haq, E.U., Ahmad, I., & Almanjahie, I. M. (2020). A Novel Parent Centric Crossover with the Log-Logistic Probabilistic Approach Using Multimodal Test Problems for Real-Coded Genetic Algorithms. *Mathematical Problems in Engineering*.
- Haq, E.U., Ahmad, I., Hussain, A., & Almanjahie, I.M. (2019). A Novel Selection Approach for Genetic Algorithms for Global Optimization of Multimodal Continuous Functions. *Computational Intelligence and Neuroscience*.
- Hong, T. P., Wang, H. S., & Chen, W. C. (2000). Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*, 6(4), 439-455.
- Hopgood, A.A. (2011). *Intelligent Systems for Engineers and Scientists*. CRC press.

- Holland, J. H. (1991). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press.
- Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimization*, 4(2), 150-194.
- Knox, J. (1994). Tabu search performance on the symmetric traveling salesman problem. *Computers & Operations Research*, 21(8), 867-876
- Lucasius, C.B., & Kateman, G. (1989). Application of genetic algorithms in chemometrics. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, 170-176.
- Michalewicz, Z., Logan, T., & Swaminathan, S. (1994). Evolutionary operators for continuous convex parameter spaces. In *Proceedings of the 3rd Annual conference on Evolutionary Programming*, 84-97
- Michalewicz, Z., & Hartley, S.J. (1996). Genetic algorithms+ Data structures= Evolution programs. *Mathematical Intelligencer*, 18(3). 71.
- Mäkinen, R. A., Périaux, J., & Toivanen, J. (1999). Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. *International Journal for Numerical Methods in Fluids*, 30(2), 149-159.
- Michalewicz, Z., Dasgupta, D., Le Riche, R. G., & Schoenauer, M. (1996). Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering*, 30(4), 851-870.
- Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
- Ono, I., Kita, H., & Kobayashi, S. (2003, July). A robust real-coded genetic algorithm using uni-modal normal distribution crossover augmented by uniform crossover: Effects of self-adaptation of crossover probabilities. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, 496-503.
- Price, K., Storn, R. M., & Lampinen, J. A. (2006). *Differential Evolution: A Practical Approach to Global Optimization*. Springer Science & Business Media.

- Parkinson, E. (2004). *Using improvement location and improvement preference to create meta-heuristics* (Doctoral dissertation, City University (London, England),).
- Reeves, (1993). *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory* (Vol. 20). Springer Science & Business Media.
- Sivanandam, S. N., & Deepa, S. N. (2008). Genetic algorithms. In *Introduction to Genetic Algorithms*, Springer, Berlin, Heidelberg.
- Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4), 656-667.
- Thakur, M. (2014). A new genetic algorithm for global optimization of multimodal continuous functions. *Journal of Computational Science*, 5(2), 298-311.
- Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms*, 1, 205-218.