Lagrange Elementary Optimization Algorithm Based on New Crossover Operator

A Dissertation
Submitted to the Council of the College of Erbil Technical Engineering at Erbil Polytechnic University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Systems Engineering

By

Aso Mohammed Aladdin

B.Sc. in Statistics and Computer, University of Sulaimani, Iraq (2010)
M.Sc. in Software Systems and Internet Technology, University of Sheffield, UK (2012)

Supervised by

Dr. Tarik Ahmed Rashid

Professor

Erbil, Kurdistan

December 2023

DECLARATION

I hereby affirm that the Higher PhD Dissertation titled "Lagrange Elementary Optimization Algorithm Based on New Crossover Operator" is entirely my own original work. I certify that all the contents of this dissertation, unless otherwise stated, stem from my independent research efforts and have not been previously submitted for any other degree at any academic institution. Any external sources or references utilized in this dissertation have been duly acknowledged within the text.

Signature:

Student Name: Aso Mohammed Aladdin

Date: / 12 / 2023

SUPERVISOR CERTIFICATE

This dissertation, under my guidance, has been authored and is now being presented for the conferment of the Doctor of Philosophy in Information Systems Engineering, with my full endorsement as the supervisor.

_____ Tarik Ahmad Rashid

Signature Name

___/ 12 / 2023_____

Date

I confirm that all requirements have been fulfilled.

Signature:

Name: Bayad Abdulqadir Ahmad

Head of the Department of Information Systems Engineering

Date: / / 2023

I confirm that all requirements have been fulfilled.

Postgraduate Office

Signature:

Name:

Date: / 12 / 2023

Examining Committee Certification

We certify that we have read this Dissertation: Lagrange Elementary Optimization Algorithm Based on New Crossover Operator and as an examining committee examined the student (Aso Mohammed Aladdin) in its content and what related to it. We approve that it meets the standards of a dissertation for the degree of Doctor of Philosophy in Information Systems Engineering.

Signature:	Signature:
Name: Prof. Dr.Subhi Rafiq M. Zeebaree	Name: Asst. Prof. Dr. Adel Sabry Eesa
Member	Member
Date:	Date:

Signature:	Signature
Name: Asst. Prof. Dr. Shahab W. Kareem	Name: Asst. Prof. Dr. Ismael Abdulrahman
Member	Member
Date:	Date:

Signature:	Signature
Name: Prof. Dr. Tarik Ahmad Rashid	Name: Prof. Dr. Ayad Ghany Ismaeel Barznchy
Member & Supervisor	Chairman
Date:	Date:

Approved by the Dean of the College of Erbil Technical Engineering College

Signature
Name: Prof. Dr. Ayad Zaki Saber Agha
Dean of the College of Erbil Technical Engineering College
Date:

Certificate of Proofreading

This is to certify that this thesis entitled “Lagrange Elementary Optimization Algorithm Based on New Crossover Operator” written by the postgraduate student (Aso Mohammed Aladdin) has been proofread and checked for grammatical, punctuation and spelling mistakes. Therefore, after making all the required corrections by the student for further improvement, I confirm that this last copy of the thesis is ready for submission.

Signature: Name: Dr. Chalak Ali Mohammed Ameen
Qualification: Applied Linguistics
Date: / 12 / 2023

ACKNOWLEDGMENTS

Above all, I wish to convey my heartfelt appreciation to ALLAH for bestowing a smooth and manageable journey upon me. I am also deeply grateful to Prof. Dr. Tarik A. Rashid for his exceptional guidance and unwavering support throughout this endeavor. Furthermore, I extend my thanks to both Erbil Polytechnic University and Charmo University for granting me the opportunity and offering their invaluable assistance along the way.

Finally, I owe a great debt of gratitude to my wife and family for their unwavering support and encouragement during my educational journey. Their steadfast belief in me has played a crucial role in attaining this level of scientific education.

Abstract

The evolutionary sophistication method solves optimization problems; however, its effectiveness and scalability can be challenged as problem complexity increases. Population-based evolutionary metaheuristic algorithms heavily rely on operators that determine their overall performance. These operators enhance exploration and exploitation, crucial for effective search and optimization. The research introduces the crossover operator, Lagrangian Problem Crossover (LPX), to boost evolutionary algorithms' performance in tackling new optimization problems. Additionally, it presents Lagrange Elementary Optimization (LEO), a single-objective algorithm where LPX plays a significant role.

The crossover operator in population-based algorithms is crucial for selecting suitable solutions in optimization processes. Its efficiency saves time, minimizes errors, and reduces costs in engineering applications. The initial phase of the study presents an overview of the current crossover methods utilized in engineering operations and problem representation. Furthermore, presenting LPX, it is a fresh and inventive hybrid technique that draws inspiration from the principles of the Lagrangian Dual Function (LDF). Experimental evaluations compare LPX with other standards such as Simulated Binary Crossover (SBX), Blended Crossover (BX), and Qubit-Crossover (Qubit-X) in real-coded crossovers. The results indicate that LPX generally outperforms other methods and shows comparable performance in remaining cases. Specifically, in TF7, LPX demonstrates superior performance and shorter computation time across all three random values compared to Mean (α=0.2) at 0.0048, Standard Deviation (α=0.2) at 0.0031, and time computation (α=0.2) at 143.005 units. Statistical analysis validates the significance and reliability of LPX compared to other crossover standards.

In the second phase of the research, a novel evolutionary method named Leo is introduced. Leo is inspired by the accurate vaccination process that utilizes the human blood albumin quotient. Leo utilizes a self-adaptive approach, evolving intelligent agents through gene crossover based on fitness function values. The algorithm's accuracy and precision are extensively validated through rigorous testing on diverse benchmark functions, including both traditional and CECC06 2019 benchmarks. Leo's performance is benchmarked against well-known algorithms like Dragonfly, Genetic Algorithm, Practical Swarm Optimization, and others across multiple functions. A comprehensive comparison evaluates Leo's effectiveness and efficiency in solving optimization problems against these established algorithms. In optimizing multimodal test functions (TF8-TF13), particularly TF11, the proposed approach outperformed other algorithms, with an average TF11 value of (2.7393E-08). Notably, across the composite test functions (TF14-TF19), the proposed method exhibited consistently high performance compared to the base algorithms. The statistical analysis supports the research conclusions, and real-world applications of Leo are also showcased. The stability of Leo is confirmed using standard metrics for exploration and exploitation.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Explanation
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AGV	Automated Guided Vehicle
Alb-CSF	Albumin Cerebrospinal Fluid
Alb-Serum	Albumin Serum
BX	Blended Crossover
CA	Cultural Algorithm
CPAMS	Cyber-Physical Attacks on Manufacturing Systems
DA	Dragonfly Algorithm
DE	Differential Evolution
EA	Evolutionary Algorithm
FDO	Fitness Dependent Optimizer
GA	Genetic Algorithm
IgG	Immunoglobulin G
LDF	Lagrangian Dual Function
LPB	Learner Performance-based Behavior
LPX	Lagrangian Problem Crossover
MOO	Multi-Objective Optimization
PN	Petri Net
PSO	Particle Swarm Optimization
Q-Alb	Quotient Albumin
Qubit-X	Qubit Crossover
SBX	Simulated Binary Crossover
SOO	Single-Objective Optimization
SSA	Salp Swarm Algorithm
STD	Standard Division
TF	Test Function
TSP	Traveling Salesman Problem
WOA	Whale Optimization Algorithm

List of Symbols

Symbols	Remarks or Descriptions
Q	Quotient
O	Offspring
C	Chromosome
G	Gene
γ	Gamma
α	Alpha (random number)
η	Eta
μ	Mu
S	Station (position)
λ	Lambda (Lagrange multiplier)
$\mathcal{L}$	Lagrange
g	gradients
$\mathfrak{hg}$	Half Group
$f\mathfrak{g}$	First Group
$\mathfrak{sg}$	Second Group
σ	Sigma (random number)

CHAPTER ONE

1. Introduction

This chapter of the dissertation centers on the categorization and overview of optimization techniques for complex problems, while also delves into the concept of self-adaptation for such complexities. Additionally, it serves to provide clarity regarding the dissertation's problem statement, objectives, and the motivation that drove the investigation of the issues encountered throughout the project development. Ultimately, the chapter outlines the dissertation organizational structure, designed to assist readers in navigating the inquiries effectively.

1.1. An Overview of Optimization Algorithms and their Categorizations

Searching for the unknown and seeking the most effective solution have been priorities since computers were invented. In 1945, Alan Turing utilized a specific search method to decrypt German Enigma ciphers during World War II (Copeland, 2000). Following that, in (Gill et al., 2008) technique for solving linear programming problems (Gill et al., 2008). Since then, a myriad of algorithms has been developed for diverse applications, including optimization and problem-solving. These optimization algorithms play a crucial role in finding suitable solutions to various problems. While multiple approaches might exist for a given situation, the optimal approach is the one that takes a global perspective into account. Typically, optimization problems exhibit non-linearity and possess intricate characteristics.

Moreover, based on the predictability and repeatability of their behavior, algorithms can be categorized into two main groups: deterministic algorithms and non-deterministic algorithms. It is extremely important to understand that non-deterministic algorithms do not mean they are completely arbitrary or random. To explore and discover better answers, they adhere to

predetermined norms or heuristics. However, given the inherent randomness, their output can still fluctuate. Providing consistency and predictability is essential or when an optimal solution can be found without exploration or randomization, deterministic algorithms are often used. Non-deterministic algorithms, on the other hand, come in handy when the issue is complicated and finding approximations or close to ideal solutions can be accomplished by considering many options (Gopalakrishna et al., 2019)(Beloglazov and Buyya, 2012).

These are illustrations of categorization using optimization algorithms. It is crucial to remember that each category contains a wide range of additional specialized algorithms and modifications. The selection of the algorithm depends on the nature of the present problem and the specific requirements of the optimization task. Depending on how they are used throughout searches, optimization algorithms can be divided into numerous groups depending on various criteria (Iqbal et al., 2014). Here are some commonly recognized classifications.

- Discrete Optimization Algorithms: These algorithms solve optimization problems with discrete variables and discrete basic problems. They consist of problems in which variables can only assume specific discrete values.
- Continuous Optimization Algorithms: These algorithms are designed for solving optimization problems with continuous variables and basic counting problems. They have to demonstrate the optimal values of variables within continuous domains and spaces.
- Linear Programming Algorithms: Linear programming approaches have been developed for solving linear optimization problems. In these issues, a linear objective function is maximized or minimized under linear constraints.

- Non-linear Optimization Algorithms: They solve optimization problems with non-linear or complex objective functions or constraints. These algorithms handle more complex optimization scenarios where variables are non-linearly related.
- Heuristic Algorithms: They are problem-solving techniques that may not guarantee an optimal solution but aim to find satisfactory solutions within a reasonable timeframe. They are categorized as traditional algorithms.
- Metaheuristic Algorithms: These are high-level strategies that guide search across a problem space and domain. They are frequently employed to solve complex optimization problems or non-linear optimization problems, where traditional algorithms encounter difficulties or limitations. They are categorized as traditional algorithms. Examples include GAs, ACO, and ABC.
- Stochastic Optimization Algorithms: They incorporate randomness or probabilistic elements into their search process. These algorithms are suitable for uncertain or noisy data problems.
- Gradient-based Optimization Algorithms: They utilize information from the gradients (derivatives) or lagrangians of the objective function to iteratively improve the solution. These algorithms are primarily used for smooth and differentiable optimization problems (Chaparro et al., 2008).

1.2. Optimization Techniques Synopsis

Normally, exploring algorithms and optimization techniques are fascinating voyages into problem-solving and efficiency improvement. To take on difficult computational issues and make wise decisions, humans explore the huge landscape of algorithms and optimization techniques. Algorithms are logical, sequential processes created to solve particular problems or complete particular activities. They offer a methodical methodology for segmenting a problem into more manageable, smaller parts, enabling effective problem-

solving. Thus, one develops a deeper grasp of the fundamental concepts, advantages, and disadvantages of many algorithms by researching a wide variety of them.

Additionally, evolutionary sophistication is the advanced complexity and efficiency achieved in evolutionary algorithms' problem-solving strategies and mechanisms and widely used to resolve global optimization challenges. However, as the initial issue grows more intricate, so does its effectiveness and expandability. Furthermore, evolutionary nature-inspired metaheuristic algorithms are a category of optimization algorithms influenced by natural phenomena and animal intelligence. These algorithms draw inspiration from the principles of evolution and mimic the behaviors observed in nature to solve complex optimization problems. Stochastic parabolic curve optimization is emphasized in a wide range of scientific and technical fields. Global optimization finds applications in engineering, financial services, and management systems by optimizing linear and non-linear objective functions to address structural problems in these fields. Due to this and as a result, two categories of large-scale algorithms are established: the first is traditional algorithms, including gradient-based optimization algorithms or quadratic programming. Secondly, evolutionary algorithms (EAs) are among several artificial intelligence techniques, including heuristic and meta-heuristic algorithms. Traditional algorithms demonstrate efficiency and deterministic behavior throughout their execution. They primarily rely on local searches; which means that achieving global optimality in the majority of optimization problems is not guaranteed. Consequently, these algorithms have restricted solution diversity and are ineffective when confronted with highly non-linear and multimodal problems (Henderson et al., 2003). Although traditional algorithms are efficient, several key aspects can be discussed about their characteristics.

The majority of their algorithms are deterministic, meaning that a given input will consistently produce the same output, except for the Hill-climbing Algorithm that is restarted randomly, which is found in the algorithms that are derived from Lagrange stationary points. Additionally, their reliance on local searches presents uncertainty regarding global optimality in most optimization problems. Therefore, the range of solutions that can be obtained is limited (Cook and Mitchell, 1997). Furthermore, traditional algorithms operate on problem-specific information, tailoring them to precise problem domains. Moreover, their inability to effectively address non-linear problems restricts their effectiveness at cracking multimodal problems.

Evolutionary algorithms, as stochastic methods, overcome limitations by exploring solution spaces more broadly than conventional algorithms, utilizing heuristics and meta-heuristics for experimental and empirical searches. These algorithms integrate specific randomization mechanisms and employ various methods of local search to navigate the problem landscape (Dey et al., 2017)(Dey et al., 2014). In addition, with further research and development, heuristic algorithms have evolved into meta-heuristic algorithms. The "meta" prefix implies a higher level of performance compared to traditional heuristics. However, it is important to highlight that the terms "heuristic" and "meta-heuristic" are often used interchangeably, as their definitions have minimal distinction. Lastly, meta-heuristic algorithms, including EAs, offer advantages over heuristic algorithms in terms of productivity and performance. These algorithms leverage stochastic procedures and auto-adaptive plans to more effectively explore the solution space, theoretically overcoming the limitations encountered by traditional algorithms (Fister Jr et al., 2013)(Hoos and Stützle, 2004).

Nature-inspired or bio-inspired algorithms draw from biological behaviors. Effective categorization involves defining genetic operators based on these

behaviors. Thus, optimization problems can be classified into three main types based on complexity: Single-Objective Optimization (SOO) focuses on optimizing one objective, finding the best solution. Multi-Objective Optimization (MOO) involves optimizing multiple conflicting objectives simultaneously. Many-Objective Optimization (MAOO) extends MOO to scenarios with an unusually large number of conflicting objectives, requiring advanced techniques for effective solution exploration and trade-off analysis.

The effectiveness of the majority of evolutionary metaheuristic algorithms or bio-inspired algorithms is contingent upon the utilization of different operators. An individual within the genetic material of populations (chromosomes, people or animals) during the evolutionary process is called a genetic operator in the context of adaptive algorithms. These operators replicate biological evolution concepts of genetic diversity and natural selection. The common genetic operators used in GAs are classified into selection, crossover (recombination), mutation, and elitism (Haldurai et al., 2016). However, these genetic operators work together to simulate natural selection and genetic variation processes. They play a crucial role in enhancing the fitness of the population across multiple generations, aiming to find optimal or near-optimal solutions for a given problem. Among these operators, the crossover or recombination operator holds particular significance. It is categorized into two types: application-dependent and application-independent crossover operators. As part of the find-best solution procedure, the crossover standard allows the best-fitted point to be chosen during the process of finding the best solution.

SOO, also known as single-objective problems, refers to a type of optimization problem where the objective is to find the optimal solution that meets a specific goal or constraint. The goal is to identify the input variables or parameters that result in the ideal value for the objective.

This may involve maximizing or minimizing a given function. In SOO, the objective function establishes a mathematical relationship between the variables within the solution space, known as choice variables. The main goal is to explore the solution space and find the set of variables that optimally maximize the objective function.

Finally, the EA typically discovers SOOs that are simple and have only one objective. SOO's goal is finding the most efficient solution for a particular principle or metric, such as execution time or performance. This can involve incorporating additional metrics like energy consumption and power dissipation. By defining the single-objective cost function as a weighted sum of normalized costs associated with each metric, multiple criteria can be combined into a single-objective optimization problem.

1.2. Problem Statement

Real-world problems are intricate and challenging to solve comprehensively due to limitations in time, space, and cost. As a result, there is a demand for cost-effective, efficient, and intelligent mechanisms. Mimicking biological behaviors provides effective solutions for tackling intricate problems in various domains. Firstly, unlike evolutionary optimization techniques that discard information after each generation, bio-based techniques retain information about the search agent throughout each iteration. Secondly, biological behavior algorithms have fewer constraints on parameters and a reduced number of operators compared to EAs, making them highly adaptable to diverse problem domains. Therefore, researchers have investigated the effectiveness of viruses and anti-viruses in animals and humans, alongside analyzing animals' behaviors and natural phenomena. The object is to comprehend how these organisms tackle problems and find potential solutions by drawing inspiration from their strategies and behaviors. This study investigates the manner in which vaccines influence the immune system of

the body and the process of immunity development. It draws parallels between this study and previous research on the navigation, predator evasion, group selection, and prey hunting behaviors of ants, animals, fish, birds, and prey. The dissertation primarily focuses on the following core problem issues:

- Identifying the optimal operator, specifically the crossover standard, for gene rejoining, is crucial in achieving a balance between heuristic evaluation and exploration.
- How does research play a significant role in efficiently addressing complex problems?
- What is the importance of identifying the optimal operator, especially the crossover standard, for gene recombination?
- How critical is the operator's role in striking a balance between heuristic evaluation and exploration?
- Can the combination of operators for generating a new generation yield satisfactory levels of accuracy and performance?
- How does research play a significant role in efficiently addressing complex problems?
- What is the importance of identifying the optimal operator, especially the crossover standard, for gene recombination?
- How critical is the operator's role in striking a balance between heuristic evaluation and exploration?
- Can the combination of operators for generating a new generation yield satisfactory levels of accuracy and performance?

1.3. Work Contributions

Optimization algorithms are among the most effective metaheuristics for dealing with multi-case problems. As a result, population-based approaches have emerged as highly efficient methods for creating and combining new algorithms to optimize combinatorial functions. In accordance with this

specification and the dissertation, the subsequent section delineates the key contributions of this research.

1. Listing the evolution from previous standards to binary, real-coded, and ordered-coded forms, with specific emphasis on each and a brief overview of implemented crossover mathematical forms.
2. This dissertation proposes a novel crossover method based on LDF which provides original metaheuristic optimization to build a more efficient optimum solution. Hence, the anticipated LPX is evaluated through a comparison with other previously existed tuning methods. In this assessment, an updated LPB algorithm is employed to compare the LPX standard with other particular real-coded standard form operators. The LPX results are then experimentally evaluated alongside these alternatives.
3. An innovative bio-inspired intelligence algorithm is proposed called Leo. This algorithm resolves SOOs and practical problems. It explores previous standards that contributed to advancements in modulated immunity systems and periodic antenna array designs. To find the most appropriate solution, several factors are taken into account based on the immunity system during vaccination. An illustrative example of this is the utilization of a fitness function to assign appropriate weights; thereby assisting the algorithm in both the exploration and exploitation phases. Consequently, the algorithm achieves rapid convergence towards global optimal population coverage.
4. Because this proposed algorithm use a similar mechanism for updating agent positions, Leo can be considered a GA-based algorithm as a population-based algorithm; however, this newly introduced algorithm employs a distinct fitness function and identifies stationary points based on the principles of LDF, and it is experimental evaluation proven in this

work by comparing this proposed algorithm with PSO, DA, GA, WOA, SSA, FDO, LPB, and FOX which has comparative results on others.

5. Finally, this dissertation introduces two new real-world applications that address significant optimization problems and achieves a balanced approach between different cases or phases.

1.4. Work Motivations and Objectives

A plethora of studies have been conducted in the realm of bio- or nature-inspired metaheuristic algorithms, with a substantial number of effective algorithms identified in the literature. Researchers are motivated by the aspiration to enhance the exploration and exploitation capabilities of bio-inspired algorithms, such as genetic algorithms. As a result, they actively develop and propose novel genetic crossover operators. These operators are crucial components of genetic algorithms and play a significant role in generating new solutions by combining genetic information from different individuals in the population. Thus, several motivations drive the effort to generate new genetic crossover operators. For instance, one motivation is to enhance the algorithm's capacity to explore the search space more efficiently.

Additionally, addressing problem-specific characteristics enables the algorithm to exploit the unique features of the problem at hand. Furthermore, improving the algorithm's performance can be achieved by incorporating domain knowledge and maintaining diversity within the population. Designing adaptive crossover operators becomes crucial to adapt to different problem scenarios effectively. The continuous pursuit of integrating ideas from algorithms like simulated annealing, PSO, or local search into new crossover operators fuels the ongoing improvement of bio-inspired algorithms, addressing diverse optimization challenges effectively.

Whenever evolving algorithms demonstrate comparable or superior performance, they are always welcomed as viable alternatives. The "Fundamental Theorem of Optimization" is a theory about optimization, as stated by Ewen and Lessard in 2015, establishing the prerequisites for a point to be a local minimum (or maximum) of a restricted optimization difficulty, and improving existing optimization problems and creating new ones (Ewens and Lessard, 2015).

As a result, no single global algorithm offers the most accurate or better answer to every optimization problem and all real applications. For instance, there is a reasonable possibility that optimization problem Y will work better with an improved algorithm than on an old algorithm. This is if the new algorithm performs better than the old algorithm with optimization problem X and vice versa. Hence, this dissertation's objective is to propose novel algorithms, named Leo, designed to facilitate real-world applications in the pursuit of global solutions. It is inspired by the behavior of vaccines when it comes to finding new immune systems.

As discussed, optimizing the diverse range of optimization problems effectively remains challenging for a single algorithm. Encouraging results from existing techniques have motivated researchers to propose novel approaches with superior performance and problem-solving capabilities compared to previous algorithms. Therefore, the primary objectives of this achievement are preserved as follow:

- Introduce an innovative crossover operator utilizing the rejoining of parent genes to generate unique genetic variations. This process aims to be implemented in a new algorithm or enhance existing genetic algorithms.

- Aims to propose a novel population-based evolutionary algorithm that achieves a well-balanced trade-off between exploration and exploitation while prioritizing superior performance and accuracy.
- Proposing new real-life applications to showcase the effectiveness and practicality of optimization algorithms. The applications aim to validate and refine the proposed algorithm's performance and applicability to diverse real-world scenarios.

1.5. Dissertation Map

The subsequent sections of the dissertation are organized in the following manner:

i. Chapter two offers a thorough literature review, tracing the evolution of search algorithms from their early versions in computing history. It also provides an overview of standard operators used in population-based algorithms, including genetic recombination.

ii. In Chapter three, the LPX standard operator, enhancing population-based algorithms through genetic recombination, is introduced. The section details the proposed Leo algorithms, inspired by the human immune system, outlining the methodology. The theoretical description precedes the programmatic simulation using pseudocode or graphics.

iii. Chapter four presents experimental evaluations and discussions. The first part assesses LPX through heuristic evaluation, comparing it to other crossover operators using LPB. The second part focuses on testing the Leo algorithm on various benchmarks and real-world applications, comparing results to other algorithms. Nonparametric statistical tests are employed for analysis.

iv. Finally, chapter five reveals the final notes and further potentials of this work. Because of their length and complexity, some created data are presented in the appendix section via tables.

CHAPTER TWO

2. Background and Literature Review

This chapter serves as literature review by providing a complete explanation of the history and theoretical background of pertinent earlier research. First, we examine the history of population-based optimization to understand and shed light on its origins. Then it looks at some genetic-based algorithms that rely heavily on genetic operators. Furthermore, it explores the evolution of several types of crossover strategies over time and makes comparisons between them. Notably, we highlight some typical crossings that have previously been utilized in algorithms. Furthermore, one can delve into the history and background of classical single-objective algorithms and explain the phenomena of cultural algorithms, which is central to this work. Finally, we examine the capability of the immune system to manufacture vaccines following vaccination, citing this process as a useful source of inspiration for our work.

2.1. Population-Based Algorithm

A population-based algorithm is a computer strategy for solving optimization or search issues that require preserving and evolving a population of candidate solutions. It is a metaheuristic approach influenced by natural evolution and social behavior principles. Thus, metaheuristics is obviously a high-level, problem-independent optimization paradigm used to address complex and difficult optimization issues. Metaheuristics function higher than typical optimization techniques, which rely on explicit issue structures. Metaheuristics are adaptable and can solve a variety of problems, including those with nonlinear and non-differentiable objective functions, restrictions, and discrete choice variables (Boussaïd et al., 2013). Furthermore, metaheuristic is renowned for its capability to strike a balance between exploration and exploitation. The exploration phase enables a wide search

across the solution space, while the exploitation phase focuses on fine-tuning within favorable regions. Due to their adaptability, metaheuristics can effectively avoid being confined to local optima and, instead, converge towards global optima. This trait makes them valuable instruments for uncovering near-optimal solutions to complex real-world problems (Zhou et al., 2020). Because of their adaptability, metaheuristic algorithms have the capacity to evade local optima and, instead, converge towards global optima. This attribute renders them valuable tools in the pursuit of identifying near-optimal solutions for intricate real-world scenarios.

Besides, a distinct collection of persons, agents, or genes, often considered as solutions in a population-based algorithm, represent potential answers to the problem at hand. These individuals proceed through a selection, crossover or variation, mutation, and evaluation procedure to enhance the quality of the solutions through generations (Boussaïd et al., 2013; Osuna-Enciso et al., 2022). Typically, these algorithms initiate by creating a population of randomly generated individuals. Each individual represents a potential solution encoded in a suitable representation format, such as binary strings or real-valued vectors. The performance or suitability of these individuals is then quantified using an objective function or fitness metric. After the evaluation, a selection method is employed to determine which individuals will progress to the next generation. The selection process may involve various methods, such as fitness proportionate selection, tournament selection, or other techniques that prioritize individuals with higher fitness values (Boussaïd et al., 2013).

In the initial step, individuals responsible for the next generation are selected, and several operators are employed to generate offspring and promote diversity within the population (Lyakhov et al., 2013). The most common operator is crossover, where two or more individuals exchange information to

create creative solutions. Crossover can be executed in various ways, such as one-point crossover, two-point crossover, or uniform crossover, depending on the encoding scheme used for the individuals. This standard will be discussed further in the next section (Kora and Yadlapalli, 2017). In addition to mutation, another operator introduces random changes in individuals to explore new regions. It supports preventing premature convergence and promotes diversity within the population. It can involve flipping bits in binary representations or introducing small perturbations in real-value representations (McGinley et al., 2011). The selection, crossover, and mutation process continue for several generations until a termination requirement is granted. The termination criterion can be defined based on different factors, such as the maximum number of iterations, the desired level of solution quality, or the fulfillment of a predetermined stopping condition (Gutierrez et al., 2019).

GA, ACO, and ABC are examples of population-based algorithms that have demonstrated their efficacy in tackling various optimization problems, including function optimization, parameter tuning, and combinatorial optimization. They provide a flexible and robust framework for solving complicated problems where traditional optimization techniques may be difficult or impossible to implement (Bao et al., 2020). Metaheuristic optimization encounters greater difficulty when dealing with problems featuring fluctuating objective functions. In such cases, real-world search or self-adaptive optimization methods are commonly employed. The search technique utilized to address these challenges must possess the adaptability to effectively handle the dynamic changes in the objective function during the optimization process (Beyer and Deb, 2001). The effective challenge with population-based optimizers is that after identifying a locally optimal solution, it becomes essential to implement diversity-preserving strategies.

These approaches can include using a substantial level of crossover or incorporating a clustering operator to maintain diversity within the population. As a result, most metaheuristic optimization methods have been refined and enhanced (Mirjalili et al., 2017). In addition, several efficient methods have been presented and improved in several special types of research for improving novel optimizers based on gene crossing. Alternatively, the proposed algorithms are always considered, provided they offer novel enhancements or achieve results that are comparable to existing methods.

One of the most well-known population-based algorithms is GA. John Holland devised and implemented this population-based search algorithm in the 1960s and 1970s. The GA mimics certain evolutionary processes based on Charles Darwin's evolution theory: Selection, fitness, reproduction, crossover, and mutation are all factors to be considered (Sivanandam et al., 2008). GAs have found applications in diverse optimization problems, ranging from function optimization to scheduling and machine learning. ACO, on the other hand, takes inspiration from ant foraging behavior, relying on the indirect communication between ants through chemical pheromone trails. A population of artificial ants is used in ACO, which deposits pheromones on paths to guide the search process effectively (Jalali et al., 2005). ACO has been extensively employed in addressing combinatorial optimization problems such as the traveling salesman problem and vehicle routing challenges. On the other hand, DE is another population-based optimization algorithm that operates with real-valued vectors. It also evolved the population through a combination of mutation, crossover, and selection procedures. DE has proven to be useful in tackling continuous optimization, parameter estimation, and function approximation (Wong and Dong, 2005). Cultural Algorithm (CA) is a population-based algorithm that combines

genetic algorithms with cultural learning mechanisms. It incorporated cultural knowledge, such as beliefs, traditions, and norms, into the evolutionary process. This cultural knowledge guided the evolution and adaptation of individuals in the population (Kuo and Lin, 2013). CA has been utilized in diverse problem domains, encompassing applications in classification, data mining, and optimization. At last, a novel population algorithm, called the LPB algorithm, falls under the category of EAs. This algorithm employs a population of individuals, often referred to as solutions, and undergoes selection, crossover, and mutation operators stimulated by natural evolution (Rahman and Rashid, 2021).

These examples are only a few examples of population-based algorithms proposed and discussed in the literature. Each algorithm has its own characteristics and is suitable for different types of problems. Researchers continue to propose and develop various population-based algorithms to address various real-world optimization challenges. In this basic, this study investigates the interplay between bio-inspired algorithms and EAs in the context of complex mathematical functions. Specifically, it explores the relationship between these two processes using population-based algorithms. The virus optimization algorithm was initially proposed by Liang and Cuevas-Juarez in 2016, and later, Liang et al. further improved it (Liang and Juarez, 2016). Similar to many other metaheuristics, the effectiveness of its application heavily depends on its initial configuration.

2.2. Crossover Overview

Combinatorial optimization stands out as a prominent research area within artificial intelligence, attracting multiple projects each year. To enhance strategic metaheuristic algorithms, these projects adopt a knowledge-based crossover mechanism that focuses on the solution structure rather than its coding (Osaba et al., 2014). Consequently, a group of optimization

algorithms, influenced by natural events and animal intelligence, is classified as evolutionary nature-inspired metaheuristic algorithms. Therefore, they can be considered as nature-inspired algorithms, and the examples discussed in the previous section represent instances of population-based algorithms. Nature-inspired computing, a prominent field in computer science, finds application and relevance in optimization algorithms, computational intelligence, data mining and machine learning (Yang, 2018). This section primarily centers on the creation and assessment of crossover operators and their impact on metaheuristic algorithms. Crossover, also referred to as recombination, represents a genetic operator in which the genetic codes of two parents are utilized to generate offspring (children). Furthermore, the crossover technique is seen as a vital means to stochastically generate novel solutions from the existing population. These crossover operators play a significant role in maintaining a balance between exploitation and exploration, allowing for feature extraction from both parent chromosomes or genes. The ultimate aim is to produce offspring with advantageous qualities inherited from both parent chromosomes (Hassanat and Alkafaween, 2017).

Throughout the years, numerous forms of crossover have been developed, and comparisons between different types have been suggested. It all began with one-point crossover and has since evolved to encompass a variety of techniques catering to different conditions, including uniform crossover (Bäck et al., 2018). Various crossover operator standards have been established based on the mathematical distribution. These standards determine the forms of binary, real-coded, floating-point, and order-coded crossover. Similarly, different standards have been defined for permutation-based problems like the Traveling Salesman Problem (TSP). When using evolutionary algorithms to address the TSP, various representations such as binary, route, closeness, ordinal, and vector are considered. In an effort to reduce the overall distance,

researchers have proposed an enhanced crossover operator for the TSP (Hussain et al., 2017). In 2010, another research study demonstrated that sequential constructive crossover (SCX) was a successful method for solving the TSP. The core concept of this approach involves selecting a random crossover point and applying the SCX technique to enhance edges before this point. After the crossover site, the remaining chromosomes are exchanged between parents to generate two offspring. In the process, any duplicated chromosomes are replaced with unoccupied ones (Ahmed, 2010). Also, Ring Crossover (RC) emerged as an innovative solution to the recombination problem. In this unique approach, parents were brought together in a circular arrangement, and an element of randomness was introduced by selecting a cut point at random. The circular process was thoughtfully designed with the parents' interactions in focus, and the slice point was chosen in a spontaneous manner (Kaya and Uyar, 2011). Nevertheless, when it comes to evolutionary algorithms striving to maximize the ordering of an extensive series, the need for specific crossover operators becomes evident to steer clear of erroneous outcomes. While it is impractical to enumerate all such operators, Table (2-1) presents several exemplary crossovers, each meticulously crafted to cater to distinct global solutions.

During implementation, crossover strategies are frequently classified according to how the gene is represented; the genetic sequence is stored as either a bit matrix or an actual code on the chromosome, depending on the algorithm. Both conventional and illustrative examples of crossover methods, such as genetic recombination, are extensively explained in the following sections. Numerous contemporary techniques guarantee that these strategies can be employed to enhance global numerical optimization and address current practical problems, as demonstrated by recently proposed metaheuristics, likes moth search algorithm (MSA) (Wang, 2018),

Table 2-1 Standard Crossovers Generation Overview

No.	Standard Crossover Operator Name	Initial Abbreviation	Standard Category	Related Work
1	Order Crossover	OX1	exchanging segments	(Hussain et al., 2017)(Dey, 2017)(Puljić and Manger, 2013)
2	Sequential Constructive	SCX	exchanging segments	(Ahmed, 2010)
3	Order-Based Crossover	OX2 - OBX	exchanging segments	(Dey, 2017) (Umbarkar and Sheth, 2015)
4	Maximal Preservation Crossover	MPX	exchanging segments	(Umbarkar and Sheth, 2015)(Pongcharoen et al., 2001)
5	Alternating Edges Crossover	AEX	exchanging segments	(Puljić and Manger, 2013)(Pongcharoen et al., 2001)
6	Edge Recombination Crossover	ERX	exchanging segments	(Dey, 2017)(Puljić and Manger, 2013)
7	Position-Based Crossover	POS	mathematical segments	(Dey, 2017)(Umbarkar and Sheth, 2015)(Gain and Dey, 2020)
8	Voting Recombination Crossover	VR	mathematical segments	(Dey, 2017)(Umbarkar and Sheth, 2015)
9	Alternating Position Crossover	AP	mathematical segments	(Dey, 2017)(Larranaga et al., 1999)
10	Automated Operator Selection	AOS	mathematical segments	(Hilding and Ward, 2005)
11	Complete Sub-tour Exchange	CSEX	exchanging segments	(Umbarkar and Sheth, 2015)(Katayama et al., 2000)
12	Double Masked Crossover	BMX	exchanging segments	(Umbarkar and Sheth, 2015)(Patel et al., 2001)
13	Fuzzy Connectives Based	FCB	fuzzy rule exchanging	(Thapatsuwan et al., 2006)(Herrera et al., 1997)
14	Unimodal Normal Distribution	UNDX	mathematical segments	(Ono, 1997)[(Kita et al., 1999)
15	Discrete Crossover	DC	mathematical segments	(Bosch, 2007)
16	Arithmetical Crossover	AC	mathematical segments	(Kaya and Uyar, 2011)(Herrera et al., 1997)(Tawhid and Ali, 2016)
17	Average Bound Crossover	ABX	mathematical segments	(Ling and Leung, 2007)
19	Heuristic Crossover	HC	heuristic rule exchanging	(Hussain et al., 2017)(Ackora-Prah et al., 2014)
20	Parent Centric Crossover	PCX	mathematical segments	(Umbarkar and Sheth, 2015)(García-Martínez et al., 2008)

slime mould algorithm (SMA) (Li et al., 2020), hunger games search (HGS) (Yang et al., 2021), harris hawks optimization (HHO) (Heidari et al., 2019), and colony predation algorithm (CPA) (Tu et al., 2021). Consequently, there is a need to introduce innovative criteria for advancing evolutionary algorithms.

2.3. Optimization Problem Algorithms

Optimizing an operation is reducing or maximizing an objective function by assigning suitable values to variables from a population of viable values. Optimization problems appear in more of our daily activities than they do in composite science concerns. For example, individuals travel to a location in several directions. Using an objective function for reducing fuel consumption, trip time, etc., the decision in the most logical direction may be obtained. As mentioned, the exploration of nature-inspired metaheuristic algorithms dates back to the 1960s when it was commenced at the University of Michigan (Gandomi and Yang, 2012).

Nevertheless, the field has seen numerous significant improvement signals during the last two decades. S. Kirkpatrick, C. D. Gellar, and M. P. Vecchi developed simulated annealing (SA), an algorithm inspired by metal annealing. Additionally, the significance of swarm intelligence approaches, which emulate the collective intelligence of natural swarms, groups, schools, or flocks of animals, has been well-established in the realm of optimization strategies. In 1989, Gerardo Benny and Joon Wang brought the concept of swarm intelligence to cellular robotics systems. This breakthrough paved the way for significant growth in the field, and the topic gained widespread popularity as a result (Shebin S and Mallikarjunaswamy S, n.d.).

Accordingly, various instances of microbiological intelligence have been observed, including ant colonies, bee colonies, bird flocking, eagle hunting,

mammal herds, bacterial development, fish schooling, and microbial intelligence. These algorithms draw inspiration from the biological expertise or collective behaviors exhibited by organisms. Remarkably, some creatures can ensure the survival of their colony without the need for a centralized control system. In such cases, organisms forage for food individually, even when far from their nests or hives, without any external direction on where to begin or how to search efficiently. Both Swarm intelligence (Kennedy, 2006) and evolutionary sophistication (Jeong et al., 2015) incorporate meta-heuristic algorithms, with EAs emulating the principles of evolution found in nature. Among all the algorithms in this category, the GA is widely recognized as the most effective and highly regarded (Melanie, 1999), which rooted in simulating the Darwinian theory of evolution, draws inspiration from the concepts of natural selection and genetic variation (Fogel, 1994).

Moreover, the algorithms most extensively utilized in this context are fitness dependent optimizer (FDO) (Abdullah and Ahmed, 2019), salp swarm algorithm (SSA) particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), the cuckoo search (CS) (Yang and Deb, 2009), and FOX-inspired optimization algorithm (Mohammed and Rashid, 2023). To find the shortest route from a food source to the nest or hive, the ACO and FDO algorithms mimic the interactions of ants and bees. The PSO algorithm, on the other hand, models the navigation and hunting behavior of bird groups. Other swarm intelligence methods described in the literature include: artificial bee colony (Karaboga et al., 2014), cat swarm optimization (Chu et al., 2006), grey wolf optimizer (Mirjalili et al., 2014), and moth-flame optimization (Mirjalili, 2015). As a result, swarm intelligence finds applications in various fields such as anthropology, industry, technology, and basic research.

In a vector-based approach, several algorithms outperform GA in various applications. Later on, in 2001, Zong WooGeem et al. introduced the

harmony search (HS) algorithm, which has been effectively utilized to solve various optimization problems, including transport models and water distribution (Geem et al., 2001). The honey bee algorithm was created in 2004 by C. Tovey and S. Nakrani. They utilized it to optimize Internet hosting centers (Nakrani and Tovey, 2004). A year later, in 2005, D. Karaboga et al. introduced the ABC algorithm. Subsequently, in 2010, Xin-She Yang proposed a bat-inspired algorithm (Yang, 2010). The dragonfly algorithm (DA) (Meraihi et al., 2020) was proposed in 2015 by Mirjalili A. S., which is PSO-based algorithm inspired by the dragonfly swarm behavior of attraction to food and adversary distraction. Later, in 2016, the whale optimization algorithm (WOA) was also announced (Mirjalili and Lewis, 2016).

Also, Lagrange multipliers, named in honor of Joseph-Louis Lagrange, offer a solution to constrained optimization problems. These problems entail seeking the maximum or minimum value of a function while satisfying one or more constraints. By using the method of Lagrange multipliers, the constraints can be integrated into the objective function, ultimately leading to the discovery of the optimal solution (Naidu, 2002). The Lagrangian dual approach is a widely used technique for resolving optimization problems, and its applicability has been extended to address Bilinear Matrix Inequalities (BMIs) (Tuan et al., 2000).

According to medical-based, vaccination optimization algorithms are computer approaches used in bio-inspired situations to improve the efficiency and efficacy of vaccination programs. These algorithms seek to enhance the overall effect and advantages of vaccination programs by optimizing different parts of the vaccination process (Matrajt et al., 2021). These parts include vaccine allocation, distribution, scheduling, and prioritizing. They can help policymakers, healthcare providers, and public health authorities create and implement efficient and targeted vaccination regimens, contributing to

infectious disease control and prevention. Healthcare optimization algorithms are computational methods used to improve healthcare delivery efficiency, quality, and cost-effectiveness. These algorithms seek to improve overall healthcare system performance by optimizing many areas of healthcare operations. These areas include medical resource allocation, decision-making, immune system care, and patient care. The bed management model based on GA has been generated which is a good example of algorithm (Belciug and Gorunescu, 2016).

The domain of nature-inspired metaheuristics presents numerous intricate challenges and applications that are beyond the scope of conventional solutions in terms of time and processing cost (Dhal et al., 2019). When dealing with problems that involve space complexity or a large number of variables, traditional methods or direct search techniques are often employed. However, in such situations, fundamental algorithmic modifications may be necessary to effectively address the challenges at hand (Arcuri and Briand, 2011). To tackle noisy objectives effectively, the application of efficient stochastic optimization techniques is essential. In this context, the focus is on stochastic single-objective optimization in high-dimensional parameter spaces. For such cases, using higher-order optimization techniques is not appropriate; hence, our discussion will be limited to first-order optimization methods. Over time, numerous researchers have explored and refined these algorithms, seeking to enhance their performance and leverage them to address a wide array of problems across different domains (Xu et al., 2021).

In addition, there are problems with only one objective, known as single-objective problems. MOO problems refer to situations where there are multiple objective functions to be optimized simultaneously. In such cases, a set of conflicting objectives exists, making it challenging to achieve optimal solutions that satisfy all objectives simultaneously. However, this work did

not focus on multi-objective optimization. The objective functions are subject to several minimization or maximization constraints, or both. To solve SOO problems, various algorithms and techniques can be employed, including gradient-based methods as discussed. The objective function can take various forms, such as linear, non-linear, continuous, or discrete, depending on the problem at hand. The constraints, if present, impose additional conditions on feasible solutions. The selection of the appropriate optimization algorithm relies on various factors, including the nature of the problem (e.g., smoothness and convexity of the objective function), the complexity and dimensionality of the search space, the existence of constraints, and the available computational resources. These considerations help in determining the most suitable algorithm to efficiently and effectively address the optimization task at hand. This approach represents an unbiased method for discovering the best possible solution that meets the prerequisites of the problem while maximizing the value of the objective function to its fullest potential. Besides, in the realm of bio-inspired algorithms, not all are strictly derived from biological systems; some are rooted in principles of physics and chemistry. Many of these bio-inspired algorithms do not directly rely on swarming behavior. Due to this distinction, the term "bio-inspired" is favored over "swarm intelligence-based." For example, genetic algorithms draw their inspiration from nature but are not inherently related to swarm intelligence. Differential search algorithm (DSA) (Civicioglu, 2012) and differential evolution (DE) algorithm (Qin and Suganthan, 2005) are two algorithms that present a challenge in terms of classification. DE, in particular, cannot be considered biologically inspired in the true sense of the word, as it lacks an obvious connection to any biological function.

2.4. Vaccination-Induced Immune System

The immune system serves as the body's defense against infections. When viruses or bacteria enter the body, they start to infect and multiply, causing an invasion known as an infection. However, the immune system counters this invasion by deploying white blood cells to attack and neutralize the infectious agents (Abbas, 2020). As indicated in Figure 2-1 (Eli Benjamini et al., 2000), the majority of white blood cells in the immune system are comprised of macrophages, B-lymphocytes, and T-lymphocytes. B-cells are responsible for attacking intruders from outside the cells, while T-cells target infected cells. Macrophages play a crucial role in the immune system by absorbing foreign objects and activating an immune response, aiding in the elimination of invaders from the body (Abbas, 2020).

When assessing the effectiveness of the immune system, it is crucial to take into account the albumin quotient in human blood serum. To demonstrate that the immune system is functioning well, Immunoglobulin G (IgG) levels should be within the restricted range and show an increasing trend. If the albumin quotient (Q_{Alb}) is declining rapidly while serum albumin (Alb_{serum}) is rising significantly, it indicates that the blood level of IgG is increasing and falls within the normal range, while the presence of albumin in cerebrospinal fluid (Alb_{CSF}) influences the albumin quotient (Q_{Alb}). (Reiber, 2003). In such cases, individuals demonstrate a high albumin quotient (Q_{Alb}), which suggests that humans have developed a modality-immunity system according to the function equation (2.1) proposed by (Andersson et al., 1994).

$$Q_{Alb} = \frac{Alb_{CSF}}{Alb_{serum}} \qquad 2.1$$

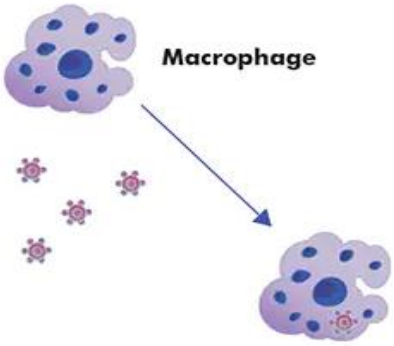	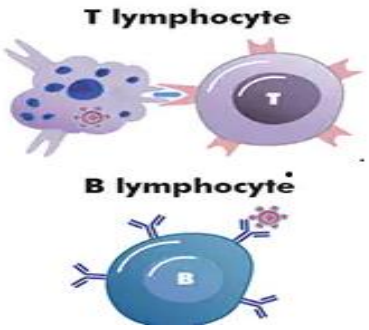
i- Innate Response (After Hours)	ii- Adaptive Response (After Date)

Fig. 2-1 Response cells of the innate and adaptive immune systems. (Macrophages, B-lymphocytes and T-lymphocytes) (Eli Benjamini et al., 2000)

Even when someone is still sick, behavioral traits and infectious diseases can spread through social interactions, especially during the rapid spread of the COVID-19 epidemic over the years. To curb this expansion, effective vaccinations have become necessary to maintain human immunity. Vaccinations work to naturally strengthen the immune system, helping it combat illnesses and reduce their effects. These technologies play a crucial role in preventing the spread of diseases among groups, as individuals often copy their social connections when forming vaccination preferences. This research involves investigating the relationship between these two processes by utilizing bio-inspired algorithms and evolutionary algorithms based on complex mathematical functions as a population-based approach.

Existing models inadequately handle the clustering of vaccination practices within a group, assuming an even distribution of individuals, leading to inaccuracies. Consequently, the concentration of anti-vaccination attitudes can lead to disease outbreaks by compromising protective immunity (Nuwarda et al., 2022)(Ndeffo Mbah et al., 2012). To study the impact of imitation dynamics on vaccination rates and disease outbreaks, algorithms

create models that determine the optimal global decision by replicating individual behavior to create a highly effective immune system for human vaccination. However, it is essential to note that various vaccinations function in different ways to confer protection. For example, COVID-19 vaccines aid in disease prevention by helping our bodies develop immunity to the COVID-19 virus. Different vaccine types, such as $mRNAs$, viral vectors, protein subunits, and inactivated vaccines provide protection in diverse ways (Lundstrom, 2020). Once administered, all vaccines are eventually eliminated from the body, but they leave behind a pool of "memory" T-lymphocytes and B-lymphocytes that have the ability to counter the virus in the future. This process strengthens the immune response by increasing antibody production and generating memory cells that can identify and respond to the actual virus if the body becomes infected. Figure 2-2 visually depicts the sequential steps involved in developing an immune defense system against spike proteins (Fang et al., 2022).

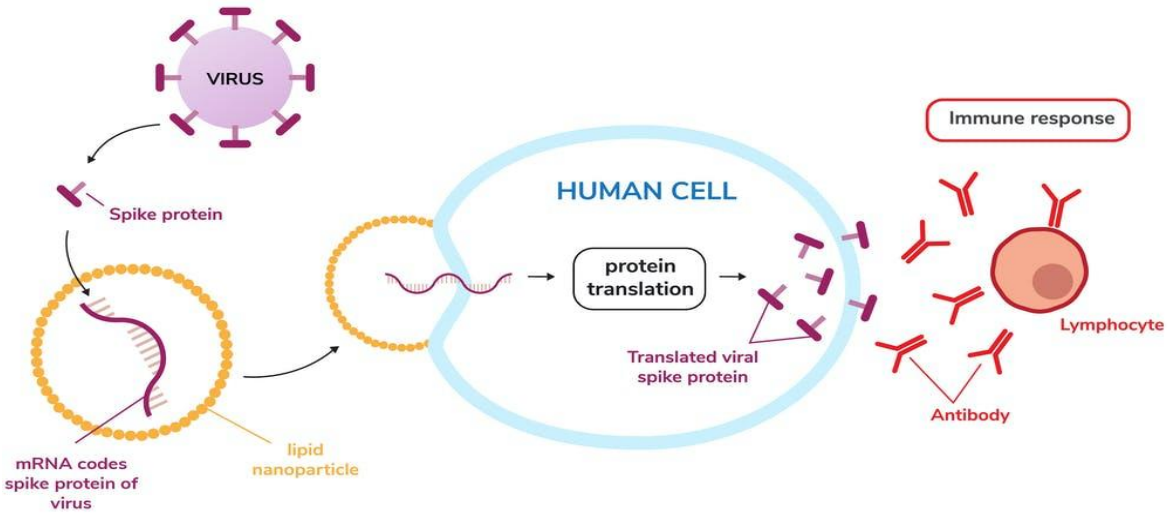

Fig. 2-2 example of spike protein (mRNA) vaccine cycle life. (Fang et al., 2022)

CHAPTER THREE

3. Research Methodology and Design

This work concerns two linked issues, which are discussed in two sections in this chapter. The first section provides an innovative standard generated from several mathematical evaluation processes, which incorporates recently developed methods. This section focuses on the effect of real-coded standard operators on mathematical distribution, and population-based algorithms, exploring inspiration from genetic and biological properties and introducing LPX. These insights are then integrated into the algorithmic structure of the proposed approach, aligning both algorithmic and biological aspects. The second half of this work introduces Leo by offering a thorough mathematical explanation via a set of connected equations and various sequence operators. The practical implementation of Leo is then depicted via pseudocode, as well as extensive explanations. The degree of information in these sections is intended to aid other researchers in replicating our work. This is done by assuring clarity and accessibility in understanding and copying our strategy.

3.1. Inspiration and Exploration of Genetic Recombination

Genetic recombination is the process of exchanging genetic material across distinct gene molecules or chromosomes. It is essential for biological diversity and evolution. In genetic recombination, gene segments (genomes) are swapped, resulting in novel genetic material combinations. As previously mentioned, several standard operators have been covered in earlier chapters, and these operators have emerged after John Holland’s proposal of the GA in the 1970s (Sivanandam et al., 2008). Subsequently, several population algorithms have been introduced, with a notable focus on achieving a balanced combination of exploitation and exploration phases to improve fitness and overall performance. Correspondingly, the DNA recombination

phenomena have long been a source of fascination and research in a variety of domains, including genetics, evolutionary biology, and biotechnology. These have been inspired by genetic recombination mechanics and effects, resulting in advances in a variety of fields. Plant and animal breeding, recombinant DNA technologies, evolutionary, genomic, and synthetic biology, and body systems are all areas of study (Nicholl, 2023).

The exploration of optimal solutions in metaheuristic or inspiration algorithms revolves around generating novel members from existing ones. The crossover process facilitates the exchange of genetic codes between parent individuals, resulting in offspring that may possess exceptional genetic traits inherited from their parents. The study delves into a plethora of crossover techniques, urging researchers to delve into whether the most effective standard strategy has been refined and embraced or not. As highlighted, the crossover operator can be likened to a powerful combination of multiplication and biological recombination (Takahashi and Kita, 2001). According to the data, it is evident that the selection of more than one genome is essential, and children are generated using genetic codes represented by sky blue balls on the parents' chromosomes. Additionally, two more children are produced using two derived offspring genes depicted as dusty pink balls. Figure 3-1 visually presents the probabilistic scale, illustrating the range of potential offspring in a two-dimensional constrained real space between x and y dimensions through the application of a box crossover between genes and new offspring.

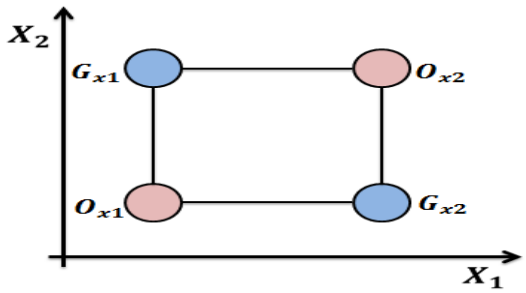

Fig. 3-1 Significant probability in the real-coded crossover

3.2 Crossover Operator Technique

3.2.1. Mathematical Distribution Crossover

Crossover is a widely used operation in metaheuristic algorithms, with a significant emphasis on its implementation in GA; especially in cases involving real-coded or binary-coded algorithms. Crossover plays a crucial role, making it challenging to achieve desirable results without its utilization (Herrera et al., 2005). Therefore, the introduction of crossover probability serves the purpose of preserving genes from the parents, even if the offspring may not outperform them. Crossover methods can be categorized into three groups. The first category encompasses binary crossover techniques, while the second category comprises real-coded or floating-point crossovers. Finally, the third category includes order-coded crossovers.

The first category of crossover operators comprises a wide range of techniques used in binary representations for metaheuristic algorithms. Enhancements to previous results demonstrate the effectiveness of most of these crossovers in addressing current challenges. Some crossover standards are practically implemented, and interesting comparisons between them are also highlighted. In traditional genetic material storage, genes are represented as bit strings in various methods. Crossover procedures for bit-order are prominent, including examples like binary single-point crossover, double-point or n-point crossover, uniform crossover or half-uniform crossover, uniform crossover with crossover mask (UCM), shuffle crossover (SHX) (Haldurai et al., 2016), and three-parent crossover (TPX) (Zhang et al., 2017) and qubit-crossover (Qubit-X) (Zamani et al., 2021).

The UCM operator divides the matrices into several non-overlapping ones, and the logical operator generates a matrix known as the crossover mask (CM) based on this control. A binary crossover mask is derived from protocol rules using the UCM operator to determine which genes are transferred

between the parent individuals. Genes with mask bits set to (1) are copied from the first parent, while genes with mask bits set to (0) are copied from the second parent. This allows for a diverse mix of genetic material between the parents, promoting exploration of the search space by evolutionary algorithms.

In the TPX (Three-Parent Crossover) (Zhang et al., 2017) operator, various probability rate algorithms are used to generate innovative offspring from three parent genes, as per the prior solution approach. The calculation of future generations involves deliberate offspring generated by swapping genes, and this process is illustrated in Figure 3-2, highlighting the problems encountered during this operation, as described in the general pseudocode. It describes a process to generate three offspring (Offspring1, Offspring2, and Offspring3) from a combination of three parents (Chromosome 1, 2, and 3). The goal is to create new individuals by selecting bits from the parent chromosomes based on specific conditions. The offspring are generated based on certain bit-wise conditions between the parents. The process continues for additional offspring following the specified conditions. Additionally, the specific conditions provided in the pseudo code can be extended or modified to suit the requirements of the genetic algorithm or evolutionary process being implemented.

Furthermore, the second category of crossovers involves real-coded or floating-point structures. In these representations, the genes are real-valued without the need for encoding or decoding into binary form, which speeds up the process. Although less intuitive than binary representations, crossover with floating-point formats has shown to perform as well as, if not better than, regular binary strings. Therefore, there is no need to worry about algorithm efficiency when using floating-point encoding.

To generate Offspring1 from (C1, C2, C3) combination

1. *if C1 bit equal to C2 bit*
2. *Then select C1 bit*
3. *Else C3*

To generate Offspring2 from (C1, C2, C3) combination

4. *if C1 bit equal to C3 bit*
5. *then select C1 bit*
6. *Else C2*

To generate Offspring3 from (C1, C2, C3) combination

7. *if C2 bit equal to C3 bit*
8. *then select C2 bit*
9. *Else C1*

the generating new Offspring will be continued
according to the conditions…

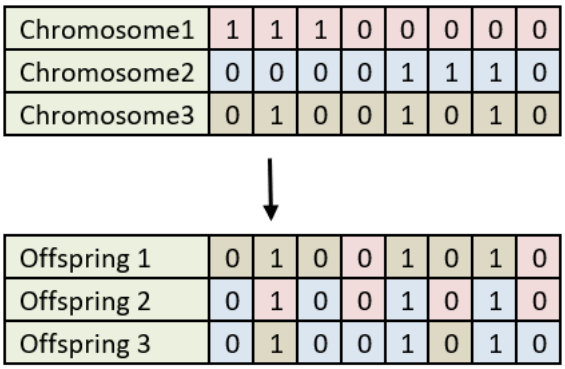

Fig. 3-2 Pseudocode and example to explain TPX deliberation

Numerous real-coded crossover techniques have been developed. These methods involve effectively adjusted real-coded crossover operations that utilize the likelihood function to generate highly diverse sequences, offering potential alternatives to solutions. For instance, the real single-point crossover is analogous to a binary single-point crossover, where two chromosomes are combined, and a real number is assigned for each gene at the crossover point (Herrera et al., 2005). Also, two-point, three-point, and n-point crossovers can also be applied to real-coded representations. In these situations, two genes are crossed, and real numbers are swapped, producing two new offspring. Besides, various crossover techniques are mathematically described in the following sections, including single arithmetic crossover, whole arithmetic crossover, and linear crossover.

Blended Crossover (BX) (Abido, 2006) is considered one of the highly effective crossovers that has shown improvements in various algorithms. If we have a pair of chromosomes with two parameter values, $G1$ as standard of $X1$ and $G2$ as a standard of $X2$, where $G1$ as is smaller than $G2$, the blended

crossover method generates an offspring within a certain range $[G1 - \alpha\,(G2 - G1), G2 + \alpha\,(G2 - G1)]$.

In cases where α is a constant to be determined, the offspring solutions remain within the bounds of the non-variable. This concept is illustrated in Figure 3-3 using a mathematical example, indicating that the number is equal to 2, $G1 = 0.13 < G2 = 0.94$, so calculate the range by $[G1 - \alpha\,(G2 - G1), G2 + \alpha\,(G2 - G1)]$; when α=0.5 the $[-275, 1.345]$, indeed, $G1$ and $G2$ are randomly selected from within the given range. This random selection ensures variability in the offspring and allows for exploration in the solution space. Also, to maintain a balance between exploring and exploiting the search space (Hamid et al., 2011).

Chromosome1	0.88	0.13	0.25	>>>	Offspring1	0.88	-0.275	0.25
Chromosome2	0.64	0.94	0.35		Offspring1	0.64	1.345	0.35

Fig. 3- 3 BX for second Genes by the range calculation

The method could not provide a global solution if applied to the preceding range, as demonstrated in numerous improvement algorithms. Researchers have proposed a novel approach to BX method, which involves computing the parameter using two random numbers, denoted integer as r and the real line as α , within the range (0.0, 1.0). This random number r is then used in the revised blend formula as the incomplete gamma (γ) type function in (equation 3.1) to determine the condition for the BX standard (Deep and Thakur, 2007). The incomplete gamma type function

$$\gamma = (1 + 2\alpha) * r - \alpha \quad (3.1)$$

The offspring solutions $Gene_1$ and $Gene_2$ are determined by the parents according to equations (3.2) and (3.3) (Deep and Thakur, 2007).

$$\text{Gene}_1 = (1 - \gamma) * G_1 + \gamma * G_2 \quad (3.2)$$

$$\text{Gene}_2 = (1 - \gamma) * G_2 + \gamma * G_1 \quad (3.3)$$

To apply a standard crossover operation uniformly across various algorithms, the simulated binary crossover (SBX) is commonly used and preferred. SBX

is specifically designed for real-coded parameters and does not involve a mutation operator. It is an extension of the single-point crossover and can also be utilized with multi-point crossover techniques. This approach centers on the probability distribution of potential offspring (genes) generated from the given parents (genes) as demonstrated in equations (3.4) or (3.5) (Carlos and Azevedo, 2011) and SBX first calculates the number of offspring using formulas (3.6) and (3.7) (Deb and Beyer, 2001). Nonetheless, equations (3.4) and (3.5) operate similarly to the evaluation of equation (3.6) and (3.7), with the subsequent example employs formulas (3.6) and (3.7), and enhance the last two formulas proposed by Azevedo (Carlos and Azevedo, 2011), which are widely employed in practice. To calculate the float number resulting from the crossover, the process begins by selecting a random number $\mu \sim (0, 1)$. Then, α is computed, and the offspring is generated using the calculated α.

$$\text{Gene}_1 = 0.5[(1 + \alpha_i)G_1 + (1 - \alpha_i)G_2] \quad (3.4)$$

$$\text{Gene}_2 = 0.5[(1 - \alpha_i)G_1 + (1 + \alpha_i)G_2] \quad (3.5)$$

$$\text{Gene}_1 = 0.5[(G_1 + G_2) - \alpha_i|G_2 - G_1|] \quad (3.6)$$

$$\text{Gene}_2 = 0.5[(G_1 + G_2) + \alpha_i|G_2 - G_1|] \quad (3.7)$$

The calculation of (α_i) functions in equations (3.8) and (3.9) is dependent on the two-preceding offspring. Eta (η) represents the index of a user-defined distribution, where η is a positive value chosen by the user, indicating the number of parameters selected.

$$\alpha_i = \begin{cases} (2\mu)^{\frac{1}{\eta+1}}, & if\ \mu < 0.5 \\ \left(\frac{1}{2(1-\mu)}\right)^{\frac{1}{\eta+1}}, & otherwise \end{cases} \quad (3.8)$$

Utilize the probability distributions to compute the function of Alpha (α_i).

$$\alpha_i = \begin{cases} 0.5(\eta + 1)\alpha^{\eta}, & if\ \alpha \leq 1 \quad \text{(Contracting Crossover)} \\ 0.5(\eta + 1)\frac{1}{\alpha^{\eta+2}}, & otherwise \quad \text{(Expanding Crossover)} \end{cases} \quad (3.9)$$

When selecting the second gene as a parent 1 and 2 from Figure 3-4 will produce two new offspring genes, we need to find (α_i) if $\mu = 0.4$ from

formula (3.9) and the user chooses two parameters ($G1$ and $G2$), the calculation is executed as follows:

$$\alpha = (2 * 0.4)^{\frac{1}{2+1}} = 0.928$$

$$Gene_1 = 0.5[(0.13 + 0.94) - 0.928|0.94 - 0.13|] = 0.1592$$

$$Gene_2 = 0.5[(0.13 + 0.94) + 0.928 * |0.94 - 0.13|] = \mathbf{0.9108}$$

The second offspring, with a value of 0.9108, falls outside the expected probability distribution range. This discrepancy indicates that occasionally the offspring gene results surpass the intended range due to the probabilistic nature. The issue arises when the updated gene's impact is intended to be larger than the original, but, in this instance, the updated gene is smaller.

Chromosome 1	0.88	0.13	0.25	>>>	Offspring 1	0.88	0.1592	0.25
Chromosome 2	0.64	0.94	0.35		Offspring 2	0.64	0.9108	0.35

Fig. 3- 4 SBX for the second Genes

The third classification of problem techniques includes order-coded crossover methods. This category focuses on the fundamental types of order-coded crossovers. Partially mapped crossover (PMX) (Desjardins et al., 2017) and the cycle crossover operator (CX) (Hussain et al., 2017) are two method examples of this type of operator.

The second parent chromosome determines the number of cycles between two parents. This method is suitable for numerical strings where each component occurs only once, ensuring that each index point in the offspring has a value from one of its parents. According to CX, Figure 3-5 proves the generation of the first offspring planet using the pseudocode when the random cycle contains the numbers (2, 5, 7, 6, 11). Also, the pseudocode outlines a process for generating two offspring, Offspring1 and Offspring2, based on certain conditions involving two genomes, Genome1 and Genome2, within a cycle. It essentially decides which genome to select for each offspring based on whether the respective genome is part of the cycle or not. This selection

process helps generate diverse offspring by considering the presence or absence of each genome in the cycle.

```
1   //To generate offspring 1
2   if Gene at C1 is in the cycle
3       then select Gene at C1
4   if Gene at C1 is not in the cycle
5       then select Gene at C2
6   //To generate offspring 2
7   if Gene at C2 is in the cycle
8       then select Gene at C2
9   if Gene at C2 is not in the cycle
10      then select Gene at C1
```

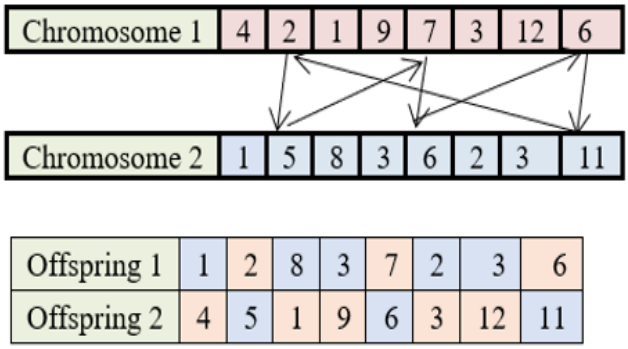

Fig. 3-5 CX operator progressive

3.2.2. Lagrangian Problem Crossover Operator

Population-based algorithms have used binary and real code numbers crossover operators. However, Crossover standards generally have strengths and weaknesses. The main purpose of validating the crossover standards is to introduce and showcase a unique crossover technique. This section is devoted to providing a novel generated category to help enhance and generate novel algorithms. Crossover operators assume varying levels of responsibility in achieving global convergence rapidly. The recommended technique is based on LDF (Ouattara and Aswani, 2018) for gene crossings. Thus, several reasons are involved in generating a novel LPX operator. Similarly, the Lagrange dual problem is crucial because, under certain conditions (like convexity), it provides a lower bound on the optimal value of the primal (original) optimization problem. This relationship is established through the Lagrange duality theory. The LDF is a key concept in optimization theory that plays a fundamental role in establishing relationships between the primal and dual optimization problems, shedding light on the optimal solutions and

providing insights into the nature of the original constrained optimization problem.

In cases where input values are constrained which can be equality or inequality conditions that the solution must adhere to, the Lagrange multiplier technique can be employed to determine the maximum or minimum of a multivariable function (Lin et al., 2010). Inspired by the Lagrange multiplier and LDF approaches, LPX endeavors to generate offspring that significantly deviate from each parent, setting it apart from other conventional operators. The primary objective of LPX is to identify regions where the contour lines of the multivariable function closely align with the input space. Besides, the Lagrange multiplier technique is employed to convert the constrained population-based optimization problem into an unconstrained one, allowing for the optimal solution to be obtained as a reference point in the crossover standard. Additionally, optimization with the Lagrangian method explores the application of Lagrange multiplier methods to achieve both local and global convergence in constrained minimization or maximization problems.

The utilization of the Lagrange multiplier method is observed in identifying local maxima and minima of a function while considering equality constraints or requirements. The relationship between the function's gradient and the gradients of the constraints naturally formulates the global problem, known as the Lagrangian Function. Points in proximity to these slopes may play a role in generating new genes within the specific chromosome.

As indicated in the preceding discussion, Figure 3-6 depicts an objective function denoted as $f(x, y)$, which needs to be optimized while being subject to the constraint $g(x, y) = c$. The Gradient $\nabla f(x, y)$ acts like a compass at each position (x, y), guiding the way for the function f to ascend most effectively. As long as the point keeps moving in this direction, f will continue to climb along the steepest path. The gradient function calculated at

a specific location (x, y) provides a vector that stands perpendicular to the contour line traversing through that point. As the exploring point ascends along the gradient vector's peak, it must always stay on the constraint curve $g(x, y) = c$. In simpler terms, the solution can only move in directions that are tangents to this constraint curve. These tangent values remain consistent throughout the constraint curve $g(x, y) = c$ since they are perpendicular to the gradient of the constraint function g.

The Gradient Vector serves as the watchful guide to the optimizer's journey on the surface of f while following the constraint curve $g(x, y) = c$. It ensures that the solution point keeps ascending in f even when it ventures along a direction indicated by the non-trivial component of the Gradient $\nabla f(x, y)$. However, if the gradient only flows in a direction perpendicular to the Gradient $\nabla f(x, y)$, the solution can be moved orthogonally to the gradient only once. In this scenario, the solution has reached a local maximum, where both gradients of f and g point in the same general direction.

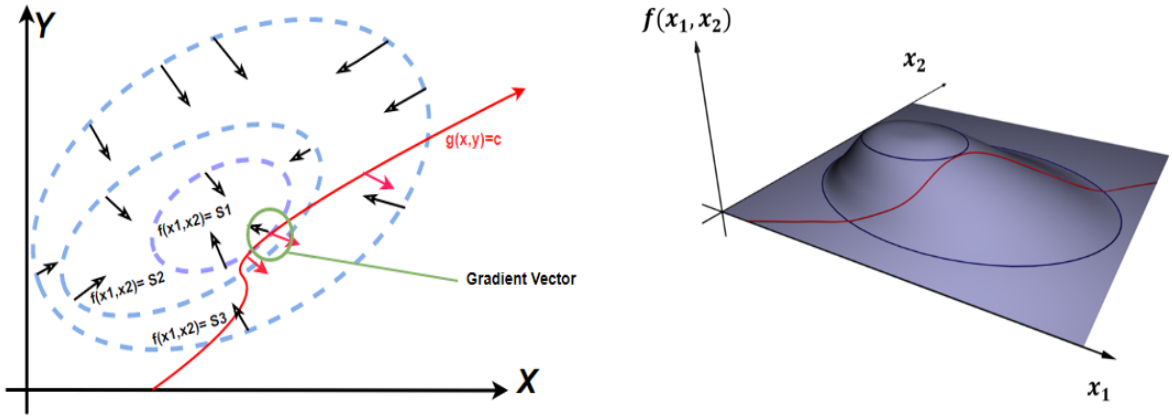

Fig. 3-6 The Lagrange multiplier shows the contour lines of the tangent function when gradient vectors are parallel.

Furthermore, in both diagrams in Figure 3-6, the constraint $g(x, y) = c$ is represented by a red curve, while the blue curves represent the characteristics of $f(x_i, y_i)$. As $S_1 > S_2$, the point where the red constraint tangentially contacts a blue curve corresponds to the maximum $f(x_1, y_2)$, which can be seen as being tangential to S_1 in the lateral constraint. On uppermost, the

observation reveals that the assumption of line graphs being tangent does not affect the magnitude of any of these gradient vectors. We may retrieve the other vector by multiplying one by a constant when two vectors have the same orientation. The Lagrange multiplier is based on the assumption that the points of local minimum and maximum along the constraint occur when the constraint is tangential to the contours, as represented by $S_1,\ S_2,\ S_3$.

In the Figure 3-6, finding ourselves amidst a contour line, where the pursuit of computing $\mathcal{L}$ at a particular point takes centre stage. It is a realm where the function $\nabla f(x,y)$ intertwines with $\lambda \nabla g(x,y)$, the enigmatic Lagrange multiplier λ adding a touch of intrigue. Herein lies the embodiment of a celebrated technique, the widely employed Lagrange multiplier approach, skilfully presented and scrutinized by (Ito and Kunisch, 2008), guiding us in unveiling the profound equation (12) that gracefully addresses the challenges of constrained optimization.

$\mathcal{L}(x,y) = f(x,y) - c$

$g(x,y) = c$ If c is a constant

$\mathcal{L}(x,y) = f(x,y) - g(x,y)$

By exploring these intricate points, a profound revelation emerges, showcasing the significance of the Lagrange multiplier λ in the improved form of equation (3.10) through maximization or minimization.

$$\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda\, g(x,y) \tag{3.10}$$

Using gradient vectors, we can determine the optimum point by computing many examples. For instance, in the realm of physical routing, the Lagrange multiplier proves to be invaluable. By selecting the smallest point, it aids in discovering the shortest and most efficient physical path. However, when it comes to unearthing multiple global solutions, the Lagrange dual function (LDF) emerges as a useful tool. The LDF theorem's effectiveness hinges on

the utilization of real equation samples. Employing a novel crossover operator, this theorem can efficiently identify numerous local points. In this process, each station contributes to generating offspring genes from parent chromosomes, enabling a fruitful exploration of potential solutions. LDF theory implies the development of an alternative to the Conic Duality theory. The Lagrangian Duality Problem theory exhibits a significant influence in optimizing general nonlinear constraints (Mahmudov, 2011). Through the application of the Lagrange dual function (LDF) theorem, a remarkable outcome is achieved as an offspring emerges from the stationary point in the equation (3.11).

$$Offspring = \mathcal{L}(x_{1,}, x_{2,}\, \alpha) = f(x_1, x_2) - \alpha\, g(x_1, x_2) = f(x_1, x_2) - \sum_{i=1}^{n=2} \alpha\, g_i\,(x_1, x_2) \qquad (3.11)$$

For Offspring 1: $f(x_1, x_2) = (x_1 - x_2)^2 + (x_2 - 1)^2$

Subject to $g_1(x_1, x_2) = x_1 + 2x_2 - 1$

$g_2(x_1, x_2) = 2x_1 + x_2 - 1$

For Offspring 2: $f(x_2, x_1) = (x_2 - x_1)^2 + (x_1 - 1)^2$

Subject to $g_1(x_2, x_1) = x_2 + 2x_1 - 1$

$g_2(x_2, x_1) = 2x_2 + x_1 - 1$

Subsequently, it is possible to generate Offspring One ($O_{ţ1}$) and Offspring Two ($O_{ţ2}$) by including equation (3.11) at the stationary point when (ţ) is initialized step selection, when the Lagrange multiplier generates a random value (α) within the designated range based on the population-based generation crossover rate, it leads to the development of equations (3.12) and (3.13).

$$O_{ţ1} = (x_{ţ1} - x_{ţ2})^2 + (x_{ţ2} - 1)^2 - \left(\alpha\,(x_{ţ1} + 2x_{ţ2} - 1) + \alpha\,(2x_{ţ1} + x_{ţ2} - 1)\right) \qquad (3.12)$$

$$O_{ţ2} = (x_{ţ2} - x_{ţ1})^2 + (x_{ţ1} - 1)^2 - \left(\alpha\,(x_{ţ2} + 2x_{ţ1} - 1) + \alpha\,(2x_{ţ2} + x_{ţ1} - 1)\right) \qquad (3.13)$$

The proposed standard (LPX) shares similarities with the real-coded crossover. To create a modified sample crossover, we introduce a novel approach of inserting a sub-sequence from one of the genes into the parent. The sub-sequence is chosen to retain the initial order, encompassing as many point states as practical. This modified sample crossover is depicted in Figure 3-7, presenting a unique and effective technique to explore new solutions and optimize gene combinations in the optimization process. When the stationary multiplier is specified arbitrarily as ($\alpha = 0.2$), x_1 is shown as Gene two G_1 on chromosome one and x_2 is reported as Gene two G_2 on chromosome two. The comparison findings are enhanced heuristically and statistically in the next section.

Chromosome 1	0.88	0.13	0.25	>>>	Offspring 1	0.88	0.4177	0.25
Chromosome 2	0.64	0.94	0.35		Offspring 2	0.64	1.171	0.35

Fig. 3-7 Create two new offspring depending on LPX

3.3. Lagrange Elementary for Optimization

The Leo is the Lagrange elementary optimization algorithm which mimics the activities of a swarm of immune systems, drawing inspiration from a group of people during imitation. Generally, the Lagrange multiplier technique is employed to minimize a multivariate function, as discussed earlier. The input may consist of any number of dimensions for the function $f(x, y, \lambda)$, often taking the form of another multivariate function $g(x, y)$ set equal to a constant (c). By utilizing gradients (g) as a two-variable function, this approach presents an effective method to find the optimal solution for reaching the top of the cliff side depicted in Figure 3-8.

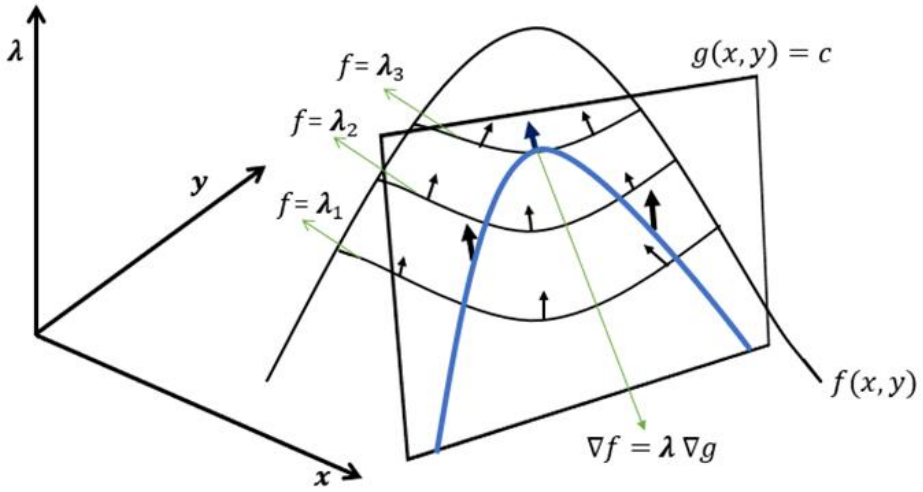

Fig. 3-8 Given that the solution can't ascend significantly higher than the point where the restriction g=c crosses the top, the objective is to climb as high on the top as possible using the Lagrange theorem.

Equation (3.14) introduces the Lagrange function F, where the numbers $\lambda_0, \lambda_1, \dots, \lambda_m$ are referred to as Lagrange multipliers. When $g_i(y)$ represents a regular value of the map g = $g = (g_1, g_2, \dots, g_m)$, the statement assumes a more elegant form. This approach proves particularly advantageous for the first and second theorems of Lagrange multipliers since it often facilitates the solution of associated conditions without the need for explicit formulas expressing point set accumulation in terms of $(n - m)$ independent variables. Generally, the necessary conditions lead to the formation of a private blockchain of relations or a system of $(n + m)$ equations in $(n + m)$ variables when utilizing the Lagrange function.

$$F(\lambda, x) = f(x) + \sum_{i=0}^{m} \lambda i\big(gi(y) - gi(x)\big) \qquad (3.14)$$

When $\frac{\partial F}{\partial x_j}(x^*, \lambda^*) = 0 \qquad \forall_j \in \{1, \dots, n\}$

And $\frac{\partial F}{\partial x_i}(x^*, \lambda^*) = 0 \qquad \forall_i \in \{1, \dots, m\}$

It may often summarize these criteria; it aims to look for constants x_0 $, y_0$ and λ_0 that fulfill $g(x_0, y_0) = c$. Depending on the requirements, equation (3.15) illustrates the tangency conditions.

$$f(x_0, y_0) = \lambda_0 \nabla g(x_0, y_0) \tag{3.15}$$

This can be broken into its components as equations (3.16) and (3.17):

$$f_x(x_0, y_0) = \lambda_0 \nabla g_x(x_0, y_0) \tag{3.16}$$

$$f_y(x_0, y_0) = \lambda_0 \nabla g_y(x_0, y_0) \tag{3.17}$$

Lagrange equation (3.18) stands as a wholly independent function, separate from the broken tangency conditions. This equation takes all the same inputs as functions f and g, along with the introduction of a new variable, the "new kid" in the action, now treated as a variable rather than a constant when $c = 0$. The function, linked to the concept of Lagrange multipliers, serves as a powerful tool to establish conditions for finding conditional maxima or minima of functions with multiple variables, or more generally, of functionals. Its primary objective is to identify local minima (or maxima) within the specified problem domain.

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c) \tag{3.18}$$

3.4. Algorithm Deterministic Process

The deterministic process of Leo encompasses the stages of initialization, evaluation, selection, crossover, and mutation, working together to drive the search for optimal solutions. Understanding this process provides a foundation for exploring and implementing Leo's in various domains, empowering researchers and practitioners to tackle challenging optimization problems with confidence.

3.4.1. Leo Comprehensive Definition

At the core of this method lies the inspiration drawn from parents' endeavors to select an appropriate and compatible group of individuals from numerous candidate groups, with a specific focus on identifying Alb_{serum} in human blood. Moreover, the process of choosing the most effective immune system

(IgG) from multiple positive systems is believed to converge towards optimality. Each genome that explores new groups of parents with a high Q_{Alb} offers a potential solution within this algorithm, providing a path to uncover latent solutions and optimize the search process. The algorithm commences with the random initialization of an Alb_{serum} population within the search space, denoted as $\mathrm{X}_{\mathrm{t,i}}(i = 1, 2, \ldots N)$; where t represents the initialization iteration selection step, and each genomic position signifies a newly discovered Q_{Alb} solution. By employing a population-based approach, this algorithm aims to identify the highest-quality parents from a stochastic space, thus beginning with the essential step of selecting individuals from the initialized population.

In the initial section of the textual Leo algorithm pseudocode, depicted in Figure 3-9, all Leo parameter settings and symbols have been defined. If the termination condition is not satisfied, a specific parameter will be utilized to randomly select a percentage of individuals from the total population N. Initially, The algorithm standard and time computation complexity impose a limitation on the population size, confining it within the range of 30 to 80 individuals. The Genetic Algorithm (GA) dictates that the best individuals must be selected even before being subjected to new operators to generate improved genomes. This step is of utmost importance, as it involves dividing the main population into two equal sub-populations after sorting the entire population in descending order, ensuring optimal selection of individuals for further processing.

Next, the primary Half Group (ɦg) is further divided into two random subgroups: the First Group (fg) and Second Group (ʂg). Subsequently, individuals will be selected from these half-group populations based on the fitness function evaluation equation (3.19), which is derived from equation (2.1), to determine the most optimal $\mathrm{Q}_{\mathrm{Alb}}$ from $\mathrm{Alb}_{\mathrm{serum}}$. The selection

process involves choosing the highest fitness obtained from the fitness function evaluation of individuals in the First Group (fg) and Second Group (sg) to complete the selection of individuals for the identified Half Group (hg) at the next time step, denoted as ($X_{t,i+1}$). This strategic selection of individuals ensures that the algorithm progressively hones in on the most promising solutions as it iterates through the optimization process.

$$X_{t,i+1\ for\ hg} = \frac{X_{t,i}}{X_{t,i+1}} \tag{3.19}$$

By allowing individuals to be selected from the initial section of the sub-population, the issue of converging towards local optima is effectively mitigated (Kleinberg et al., 2018). The last two stages involve enhancing individuals by enabling gene interactions within groups. This necessitates collaboration among genes and the individuals chosen for parent selection, thereby facilitating the creation of appropriate subsequent steps. The optimal weight of metacognition typically influences how a genome explores problems through mutations.

Indeed, during the process of crossover, individual genomes collaborate and collectively influence their behavior. This collaborative effort involves sharing genetic information and combining traits from different individuals, leading to the creation of new genomes with potentially improved characteristics. Once the genomes are accepted and evaluated based on Alb_{serum} from human blood to estimate Q_{Alb}, they may possess an effective ratio of the human immune system.

Consequently, there can be advantages in increasing the IgG rate through vaccinations, which involves providing support and fostering collaboration with others. Furthermore, as mentioned in (Saravanan et al., 2022), genomes can mutually influence their occurrence and partake in gene group interactions within the blood serum. They might also seek support when

vaccine doses impact the genome. As previously mentioned, the Leo algorithm is founded on GA, where GA operators simulate gene inheritance to create new individuals in each generation. In the subsequent stages, these operators are utilized to alter the structure of individuals. The primary genetic operators employed are selection, crossover, and mutation, which are commonly used in genetic algorithms. In Leo, genes function as selection operators, with a particular focus on group-based selection, contributing to the algorithm’s distinctive approach in optimizing solutions. Subsequently, this case delved into an explanation of how the Leo algorithm functions, incorporating the utilization of crossover and mutation operators.

In the initial segment of the pseudocode illustrated in Figure 3-9, commence by initializing the generation: randomly creating an initial population. Subsequently, proceed to identify parameters, crossover rate, and mutation rate by recognizing genomes for all individuals, taking into account the albumin quotient to bolster the overall immunity system. The subsequent phase centers on the selection groups of genomes: randomly selecting a percentage based on a specified parameter, evaluating individual fitness using Equation 3.19, and sorting individuals (parents) in decreasing order, as elaborated in previous sections. Select ɦɠ as N/2, dividing N into two equal parts of populations. The computational process continues by incrementing k by 1 until the iteration's conclusion. The transition then advances to the Lagrangian Problem Crossover LPX stage, involving the implementation of equations 3.12 and 3.13. This includes swapping the first genome to derive new individuals' fitness functions. Following this, the Gaussian Mutation process is executed, iterating until the fitness of the fittest individual in the population attains a sufficiently high level. Finally, the best solution is selected from the created individuals.

Algorithm: $Leo\ (N, X, \mu, ţ, ĥĝ, fĝ, ŝĝ, k)$
$N \leftarrow 0$ (The original 'initial' random population)
$X \leftarrow 0$ (The number of Parents in the novel population)
$\mu \leftarrow 0.5$ (The percentage of Parents chosen from N)
$ţ \leftarrow 0$ (Iteration Step Initialize selection)
$ĥĝ$: half-group-population chosen from N in the end of iteration
$fĝ$: first-group- population
$ŝĝ$: second- group- population
k: calculate the number of newly created parents (Individuals)by counter

```
//1- Initialize generation
Randomly create a population N: X_ţi (i = 1, 2, ..., N)
//2- Identify parameters, crossover rate and mutation rate
Identify genomes X for all Q_Alb [albumin quotient]
to get high IgG [best immunity system]
//3- Selection Groups of genomes
do {
        Use μ parameter to randomly choose a percentage
        Evaluate the fitness of individuals in X by Equation 3.19
        According to the fitness function
        sort decreasing the individuals (Parents) in N
        Select ĥĝ: N/2 [Divide N to two equal parts of populations]
        // generate two groups from ĥĝ
        fĝ: Parent has upper fitness from first group
        ŝĝ: parent has upper fitness from second group
        determine the best fitness evaluation in first group by Equation 3.19
        //compare the highest fitness with high and low populations
          while k <= X
                if fĝ include parents
                      if     X_ţi from ĥĝ highest fitness <= ĥĝ fĝ: highest fitness
                        collect X_ţi in the fĝ
                      else
                        collect X_ţi in the ŝĝ
                      end if
                 else
                     collect X_ţi in the ŝĝ
                end if
                k ← k+1;
        end while
//4- Lagrangian Problem Crossover LPX
Required: Implement equations 3.12 and 3.13
swapping first genome to find new individuals fitness function use
depend on creating new individuals from X_ţi parents and crossover offspring
//5- Mutation [Gaussian Mutation]
function Gauss
ţ← ţ+1;
} while fitness of fittest individual in ĥĝ is not high enough
  [Select the best solution from the ĥĝ];
```

Fig. 3-9 the proposed pseudocode for Leo Algorithm

As a population-based method, individuals in Leo algorithm generate diverse offspring by modifying suggested self-adaptive systems at each stage of

development. The crossover parameter plays a vital role in enhancing global search capability and increasing variance in the differential vector. On the other hand, a rounding procedure reduces the second component of the weighted difference vector to the nearest integer value. After the selection process, crossover, and mutation are employed to generate updated genomes based on the selected parents. For further clarity, Figure 3-10 presents a flowchart illustrating the step-by-step process of the Leo algorithm. This flowchart visually explains the various stages involved in optimizing solutions and achieving convergence.

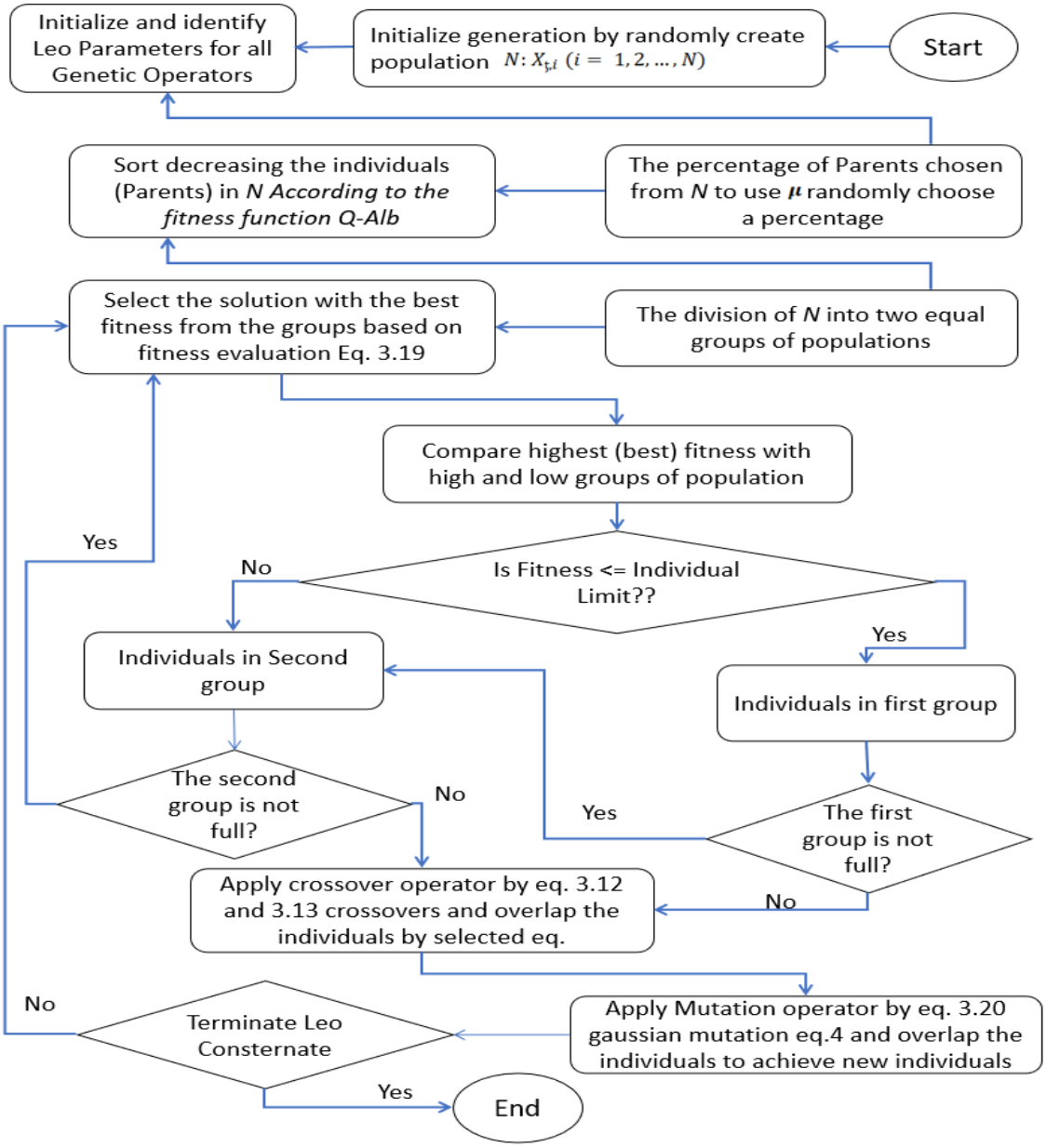

Fig. 3-10 Leo algorithm flowchart process

3.4.2. Leo Crossover Process

In the Leo algorithm, the GA-based crossover operator plays a crucial role in this stage. By employing the crossover operator, the exchange of genes between genomes is facilitated, leading to the development of an optimal immune system after vaccination. As a consequence, the genome becomes a complete set of genes distinct from the original genomes or genes of Alb_{serum}. This results in a significant impact on the overall genome of both individuals, and the newly produced individuals exhibit variations from Alb_{serum}.

As Adam optimization pointed out (Kingma and Ba, 2014); stochastic gradient-based optimization plays a vital role in numerous fields of science and engineering. Various gradient theorems can be applied to ascertain the differentiability of a function using the Lagrange method. Gradient descent proves to be a quite effective optimization technique, particularly when its parameters are appropriately tuned or lagged. Consequently, Leo is utilized with the LPX standard to generate comprehensive individual structures by exploring alternative genes. The algorithm initiates with a crossover rate of 0.6 multiplied by half of the population size. As previously mentioned, the LDF theorem draws inspiration from real-world equation examples within the LPX standard. Moreover, a novel crossover operator is proposed, capable of identifying multiple local points. The LDF theory represents a significant advancement in replacing the Conic Duality hypothesis, as highlighted earlier. Equations (3.12) and (3.13) serve as the key tools to discover novel structural individuals($O_{t,i}$).

The Lagrangian dual function and the Leo problem optimization exhibit deterministic properties, confirming their well-defined characteristics and behaviors. In this algorithm, the values are sampled from a uniform distribution on the interval [0.2, 0.3]. During the creation of new individuals,

it is feasible to swap the first old gene for the second new gene. This exchange determines the updated fitness function, which is then evaluated using equation (2.1) to calculate the fitness. The Leo crossover is constructed using pseudocode, as illustrated in Figure 3-11, showcasing the step-by-step implementation of the crossover operation within the algorithm.

$X_{ţ,k}$, $X_{ţ,K+1}$: are the two given Genes;
$O_{ţ,k}$, $O_{ţ,k+1}$: are the two new Offspring;
$Y_{ţ,k}$, $Y_{ţ,k+1}$: are the two highest fitness Genes
α : is a random value between (0.2,0.3);
k: is counter;
i: is additive by one gene number
n: number of parents (individuals)

Leo_Crossover ($X_{ţ,k}$, $X_{ţ,K+i}$, $Y_{ţ,k}$, $Y_{ţ,k+i}$){
Required: Calculate α value
While k smaller than n
Calculate the first offspring $O_{ţ,k}$ from equation 3.12;
Calculate the second offspring $O_{ţ,k+i}$ from equation 3.13;
Calculate the new first gene: $Y_{ţ,k} = \frac{O_{ţ,K+i}}{X_{ţ,k}}$;
Calculate the new second gene: $Y_{ţ,k+i} = \frac{O_{ţ,k}}{X_{ţ,K+i}}$;
$k + i + 1$;
End while
} the best individuals found during the evaluation

Fig. 3-11 Leo Crossover Process Pseudocode

3.4.3. Leo Mutation Process

Although scientists can provide explanations for the immunological ambiguities resulting from genetic mutations, the precise nature of these ambiguities remains unidentified. Additionally, mutations can have random effects that influence behavioral changes in individuals. The most fundamental form of mutation involves the modification of one or more genes. As mentioned, metacognition can introduce a stochastic influence on the overall behavior of genes. As a result, individuals have the ability to adjust their behavior in specific directions by modulating the levels of immunity-related activities in their genes, guided by the mutation rate. The mutation operator, inherent in the GA-based algorithm, is effectively

employed in the Leo algorithm to demonstrate this progressive adaptation. Through the mutation operator, individuals can explore and navigate the search space, allowing for diversity and exploration in the optimization process.

The primary objective of EA mutation is to introduce diversity into the entire population sample. Mutation operators are employed to steer clear of local minima or local maxima by ensuring that the population of genomes does not become overly similar to each other. By introducing variations through mutations, individuals can be adapted to different situations or carry genes that were not originally present in the initial population. There are several methods to mutate individual representations, such as uniform mutation, replacement mutation, scramble mutation, inversion mutation, dynamic mutation, boundary mutation, and so on. These diverse mutation techniques help maintain genetic diversity and aid in exploring the solution space more effectively, leading to better optimization outcomes in evolutionary algorithms. Leo operates Gaussian Mutation (Bell, 2022) as the standard for EA. Self-adaptation enables a GA to modify its algorithm during problem-solving (Smith, 2008). The Gaussian mutation operator has proven to be the most effective and popular choice for self-adaptation in GA. By assigning a random value between (-1, 1) to sigma (σ) and taking a random sample for (j_i), equation (3.20) is employed to develop Leo mutation operator, denoted as $M_{t,i}$. In Leo algorithm, the percentage mutation for individuals in the sample $X_{t,i}\,(j_i\,)$ is set to 0.3.

$$M_{t,i} = X_{t,i}\,(j_i\,) + \sigma * \ randn\big(size\,(j_i)\big) \tag{3.20}$$

CHAPTER FOUR

4. Results and Discussion

This chapter presents the implementation results, followed by direct analysis and discussion, aiming to enhance readability. Consequently, the results section is relatively extensive, allowing for a more straightforward mechanism by incorporating results analysis into each corresponding section. The main sections in this chapter focus on determining the results for validating the LPX and Leo algorithms. In this manner, the chapter offers substantiation and showcases the efficacy of the suggested standard and algorithm.

The initial section focuses on gauging the exploitation level and convergence of standards in population algorithms. To assess these standard operators and the performance of population-based algorithms, various benchmark test functions are available. For this study, three specific test functions have been chosen to analyze the newly introduced operator. In this sub-section, three unimodal test functions, namely TF1, TF3, and TF7, are selected from the classical benchmark tests. The first part of this section involves a heuristic evaluation, where the LPX, BX, and SBX operators are compared. Moving on to the second part, the standard operator is compared with BX, SBX, and Qubit-X operators, all tested using LPB as a population-based algorithm to analyze exploitation and convergence results. Finally, the last part involves the statistical evaluation of all results using a Wilcoxon Rank Sum test.

The second part of the study aims to validate the proposed algorithm's functionality and assess its effectiveness. The execution of the Leo algorithm results on Apple silicon Macs is enabled and supported by MATLAB R2019b. The generation of real applications is contingent on the utilization of this particular version. To achieve this, a set of widely recognized benchmark functions from existing literature are employed. Additionally, Leo algorithm's

results are compared with five other well-known algorithms from the literature in 19 classical benchmark tests (Hussain et al., n.d.). Among these algorithms, one is a popular approach, such as DA, PSO, or GA, while the remaining two are novel methods, namely FDO and LPB. Besides, the results obtained from the proposed algorithm are compared to those achieved by Leo algorithm specifically on the CEC-2019 test functions (Brest et al., 2019) such as DA, WOA, SSA, FDO, LPB, and FOX. Subsequently, a statistical analysis is performed on these results using the Wilcoxon rank-sum test to determine the significance of the outcomes by IBM SPSS Statistics Version 26. Finally, it is worth noting that Leo functions have demonstrated successful applications in solving real-world problems.

4.1. Results and Discussion of Lagrangian Problem Crossover

LPX is a novel crossover standard being proposed, which is evaluated by comparing it to the previous standard methods as described in the methodology chapter. The evaluation primarily centers around estimating the time required to discover the best optimal solution. It is important to note that the results are influenced by the value of a random number to determine the most suitable solution. Additionally, the study demonstrates the processing time for dynamic cost minimization. To accomplish this, gradient vectors have been employed to guide this model in computing numerous examples and identifying the optimal point. The significance of this methodology lies in its applicability to physical routing, where it streamlines the process of identifying the smallest point to reveal the shortest physical path.

4.1.1. Heuristic Evaluation Results

Through mathematical comparisons, this assessment sought to pinpoint significant usability and performance issues associated with the standard operators. The tests were conducted over a series of 100 stochastic generations (genes) involving various pairs of parents (chromosomes) and

crossover rate values (α) set at 0.3, 0.5, and 0.7. The variation in random values plays a crucial role in assisting newly proposed algorithms to efficiently select the optimal range of values. In the previous chapter, various methods of standards have been provided, but the focus of the testing lies on the BX and SBX crossovers. Therefore, the outcomes of the two standard heuristics are contrasted with those obtained using the LPX method. Additionally, the performance can be assessed by examining the statistical values generated. A higher averaged value signifies a more favorable outcome. Regarding the crossovers, the outcomes of mathematical calculations performed on parent chromosomes are utilized to generate genes in the offspring chromosomes (Malik and Wadhwa, 2014). The arithmetic operations are determined by applying equations (3.2) and (3.3) for BX, equations (3.4) and (3.5) for SBX, and equations (3.12) and (3.13) for LPX.

The test is conducted by choosing three unimodal functions from a collection of conventional benchmark functions. The purpose is to assess the appropriateness of each gene on a chromosome during the evaluation process. The benchmarks consist of three test functions: TF1, TF3, and TF7, which are listed in Table (4-5). Achieving the global optimum requires an algorithm to steer clear of local optimal solutions, and these selected sample test functions can aid in devising an effective exploration strategy. The individual results are statistically analysed by summing up the generations on the parent chromosome. After conducting the evaluation on the selected test functions, the next step involves calculating summation (Sum), the average (Mean) and standard deviation (STD) for each standard depend on 100 genes. The comparison between the different standards is based on the alpha value (α), which is a randomly generated value during the algorithm's execution.

The Lagrangian functions are specifically designed to explore novel points in intricate applications by searching around the most prominent local optimum.

As previously discussed, the Lagrange multiplier method in mathematics is a mathematical technique utilized to ascertain the local maximum or minimum value of an action while adhering to equality constraints. In the context of LPX, Lagrangian functions are purposely crafted to explore new points in complex applications, focusing on searching for the most prominent local optimum. As mentioned earlier, in mathematics, the Lagrange multiplier method is a powerful technique used to determine the local maximum or minimum value of an action while satisfying equality constraints. LPX outperforms BX and SBX for all values of alpha (α). Whereas BX and SBX may have been reasonable choices in previous generations, at present, LPX emerges as a better choice. Additionally, the results suggest that TF7 demonstrates strong convergence and exploitation characteristics across all alpha values. Moreover, LPX holds potential for ranking social classes and conducting stochastic analysis of metaheuristic algorithms as focused on Table 4-1.

Table 4-1 The performance result test for selected crossover standards with LPX

	Standards		**BX**			**SBX**			**LPX**	
α	Test Functions	Sum	Mean	STD	Sum	Mean	STD	Sum	Mean	STD
0.2	TF1	42.36	0.42	0.30	31.37	0.31	0.32	**1737.56**	**17.38**	**17.09**
	TF3	60.00	0.60	0.66	60.00	0.60	0.66	**3197.01**	**31.97**	**31.27**
	TF7	779.24	7.79	10.78	487.58	4.88	9.52	**1937510.53**	**19375.11**	**33631.08**
0.5	TF1	30.00	0.30	0.30	38.58	0.39	0.30	**2776.00**	**27.76**	**26.17**
	TF3	60.00	0.60	0.66	60.00	0.60	0.66	**5273.89**	**52.74**	**50.06**
	TF7	461.88	4.62	9.41	661.72	6.62	10.21	**4348187.50**	**43481.88**	**64658.48**
0.7	TF1	35.49	0.35	0.31	46.82	0.47	0.30	**3648.64**	**36.49**	**34.78**
	TF3	60.00	0.60	0.66	60.00	0.60	0.66	**7019.17**	**70.19**	**67.65**
	TF7	579.02	5.79	9.88	941.45	9.41	11.81	**7300002.73**	**73000.03**	**109185.64**

Despite that, all the standards of crossovers were considered reasonable in terms of the invincible algorithms of evolution. Nevertheless, we presented two instances exemplifying the maximum and minimum genes on parent chromosomes for TF1, while considering three different alpha values. The performance of BX and SBX in TF1 was straightforward, but LPX displayed

slightly better outcomes, as the average and standard deviation outputs increase, it indicates enhanced performance in algorithms and real application problems. Moreover, LPX demonstrated both the maximum and minimum gene expressions that exceeded the performance of both BX and SBX methods. Besides, the convergence process in LPX revealed significantly higher inter-generational relationships than the other two crossover standards for all alpha values in TF3. Additionally, it successfully illustrated the relationships between generations for both maximum and minimum gene values based on TF3. In the case of TF7 results for all alpha values, LPX demonstrated oddly high performance compared to BX and SBX, mainly due to its ability to define and obtain maximum and minimum genes effectively. The evaluation and highlighting of the chromosome comparative results for TF7 are depicted, showcasing the divergent maximum and minimum genes. Figures 4-1 to 4-9 illustrate the frequency of subsequent generations originating from the parent genes on the Y-axis. This representation is essential because it takes into account the presence of distinct maximum and minimum genes on each chromosome, which are used to assess and compare the results. The heuristic evaluation confirms that LPX frequencies rapidly reach their maximum for these three test functions. Therefore, these solutions according to LPX support obtaining the best optima in both stochastic and bio-inspired optimizations.

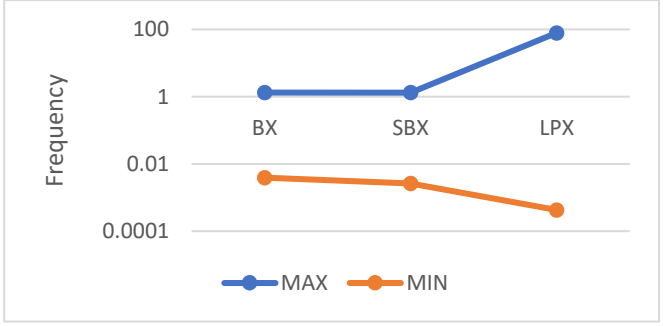

Fig. 4-1 Parents' Generation for TF1(α=0.2)

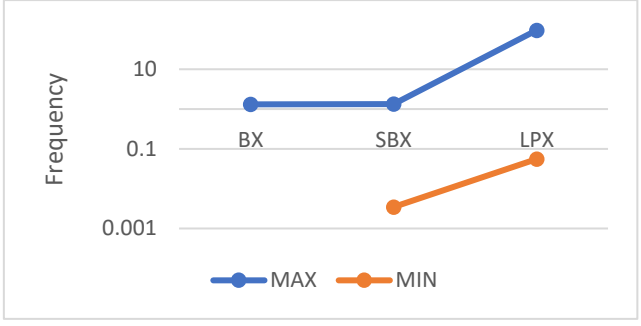

Fig. 4- 2 Parents' Generation for TF1(α=0.5)

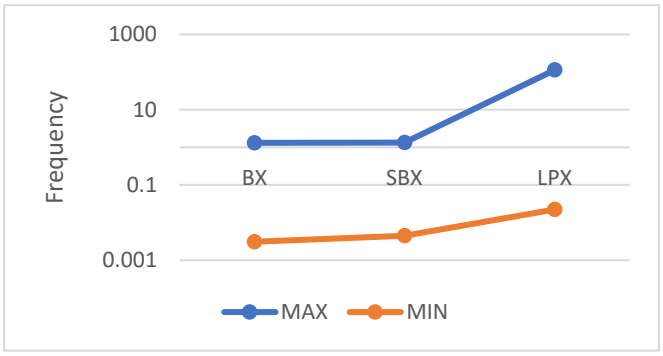

Fig. 4-3 Parents' Generation for TF1(α=0.7)

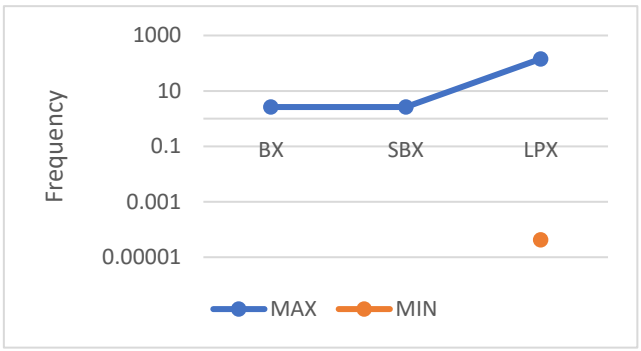

Fig. 4-4 Parents' Generation for TF3(α=0.2)

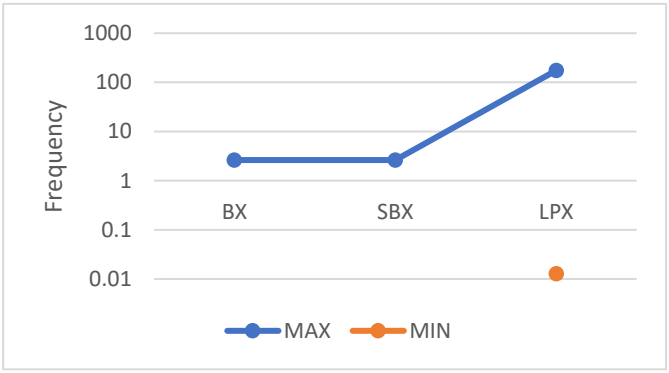

Fig. 4-5 Parents' Generation for TF3 (α=0.5)

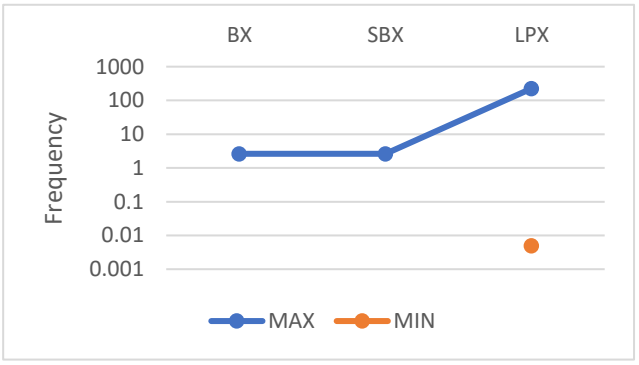

Fig. 4-6 Parents' Generation for TF3(α=0.7)

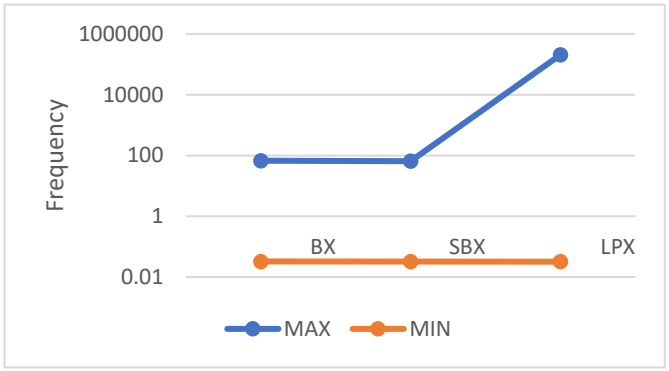

Fig. 4-7 Parents' Generation for TF7 (α=0.2)

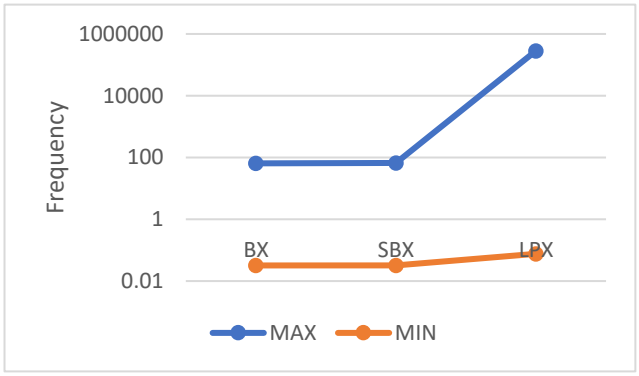

Fig. 4-8 Parents' Generation for TF7 (α=0.5)

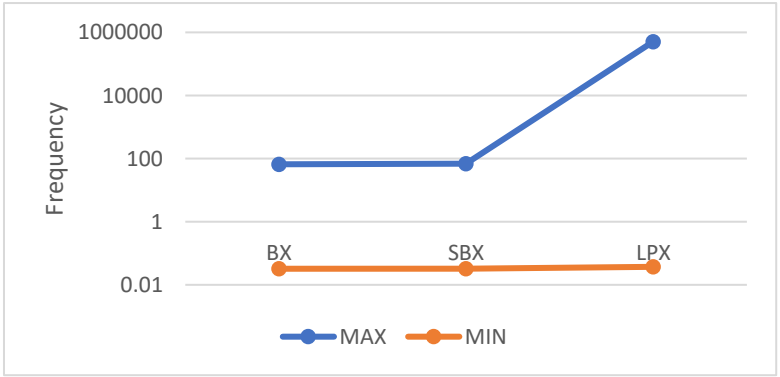

Fig. 4-9 Parents' Generation for TF7 (α=0.7)

4.1.2. Exploitation and Convergence Evaluation Results

The prowess of LPX has undergone rigorous evaluation through a captivating array of experiments. These inventive designs delve into the intricacies of LPX, scrutinizing its qualitative and quantitative traits with keen interest. The qualitative analysis beautifully showcases LPX's exceptional exploitation capabilities and convergence behavior in problem-solving, drawing insights from exploration and average fitness values. To accomplish this objective, LPB, the population algorithm mentioned earlier, is carefully chosen. Alongside, a quantitative analysis is conducted to compare LPX against other standard crossovers like BX, SBX, and Qubit-X. We have handpicked three classical test functions, presented in Table (4-5), to gauge LPX effectiveness. LPX's performance evaluation involves two crucial aspects: its capacity to break free from local optima and its convergence speed, gauged by summing the elapsed time to reach the optimal fitness point. This study demonstrates the influence and efficacy of the random value factor. To establish this, three distinct random values are selected for each test function, illuminating the significance of the stochastic element in LPX's outcomes.

The experiment involved 500 iterations, testing each standard with different random values using the LPB algorithm. LPB was executed 30 times with 80 search agents in each run. The evaluation included computing SUM, STD, and processing time. Remarkably, LPX showcased exceptional performance, consistently ranking first or second in SUM and STD depend on 30 rounds across almost all three test functions. The remarkable convergence speed and outstanding results are boldly highlighted in Table (4-2). In particular, when comparing the LPX standard with others on unimodal test functions, it was observed that LPX exhibited a higher rate of exploitation and convergence, specifically in TF7 for all random value approaches, and in TF1, LPX high time computation for all random values rather than compared other crossover standards. Also, LPX

Table 4- 2 The crossover operator's comparison results of classical test functions

TFs	α	LPX			SBX			BX			Qubit-X		
		Mean	STD	Time (s.)	Mean	STD	Time (s.)	Mean	STD	Time (s.)	Mean	STD	Time (s.)
TF1	0.2	0.0635	0.0184	**141.740**	**0.01751**	**0.0236**	161.423	0.04428	0.0446	150.384	0.1758	0.0926	144.474
	0.5	0.0680	0.0281	**149.798**	**0.04161**	**0.0270**	162.160	0.04178	0.0323	157.700	0.1411	0.0510	151.992
	0.7	0.0596	0.0279	**151.112**	**0.02959**	**0.0172**	188.501	0.04150	0.0294	163.809	0.1425	0.1045	157.042
TF3	0.2	43.5652	**24.8093**	**159.289**	83.37500	59.0221	178.260	**41.60497**	28.4041	169.970	120.7210	73.2963	164.986
	0.5	**40.4260**	**26.2073**	165.144	78.18210	52.1304	**161.057**	52.58699	37.7840	169.130	175.6268	119.6147	165.608
	0.7	66.7197	58.8220	164.711	85.92191	71.4473	180.216	**50.67240**	**49.2447**	167.669	81.3198	52.6167	**164.450**
TF7	0.2	**0.0048**	**0.0031**	**143.005**	0.01351	0.0188	153.884	0.00624	0.0045	156.457	0.0076	0.0043	160.718
	0.5	**0.0049**	**0.0027**	152.029	0.00770	0.0066	164.322	0.00616	0.0029	162.973	0.0094	0.0051	**147.530**
	0.7	**0.0052**	**0.0033**	157.893	0.00773	0.0056	164.504	0.00709	0.0042	163.520	0.0123	0.0064	**155.061**

exhibited marginally faster execution times in comparison to the calculations for TF3 with the random value approaches 0.2. Furthermore, during TF3, when the random value approaches 0.5, LPX demonstrated the optimal convergence and exploitation rates with second rank after for time commutation after SBX.

4.1.3. Statistical Evaluation Results

Stochastic optimization algorithms allow for the use of non-parametric statistical tests like Wilcoxon signed-rank sum and analysis of variance (ANOVA) to assess their overall performance. Another non-parametric test, the Wilcoxon rank-sum test (also known as Mann-Whitney U test), compares two independent groups or samples (Fay and Proschan, 2010). As it makes no assumptions about the data distribution, it is very useful when comparing optimization techniques. A considerable difference between the ranks of data from two different algorithms is also assessed. Optimization performance evaluates whether one approach consistently outperforms the other. Additionally, it offers an accurate technique for comparing algorithms or standards based on their rank-based performance metrics, without assuming anything about the distribution's underlying properties.

When using a standard to address optimization problems, it is essential to assess its statistical applicability. Table (4-3) presents a comparison using the Wilcoxon signed-rank sum test between LPX and SBX, as well as LPX and BX standards. The statistical tests conduct on LPX results show significant results, rejecting the null hypothesis. All p-values obtained for these three test functions, which are tested with random values (α), are smaller than 0.05.

The statistical analysis presented in Table (4-4) is conducted using the Wilcoxon rank-sum test to examine the results. The objective is to determine the significance of the crossover operators when compared to LPX . Based on the results, Qubit-X demonstrated statistically significant outcomes for all three test functions, except for TF2 (α=0.7), where the significance was not observed. As a result, the null hypothesis was rejected for TF2 (α=0.7).

Furthermore, SBX demonstrated statistically significant results for all test functions and values, except for TF3 and TF7 at α=0.7. On the other hand, LPX showed statistical significance compared to the BX standard for TF1 across all alpha (α) values, indicating a (p-value) lower than 0.05.

Table 4-3 The Wilcoxon rank-sum test (p-value) between crossovers operator for random generations

TFs	α	**Standards**	
		LPX vs SBX	LPX vs BX
TF1	0.2	3.6746E-16	5.3124E-16
	0.5	4.7409E-17	4.0951E-17
	0.7	4.7409E-17	3.8618E-17
TF3	0.2	8.5768E-16	8.5768E-16
	0.5	9.5355E-17	9.5355E-17
	0.7	8.2482E-17	8.2482E-17
TF7	0.2	1.6983E-16	2.6103E-16
	0.5	4.0951E-17	3.2378E-17
	0.7	3.2378E-17	2.0802E-17

Table 4-4 The Wilcoxon rank-sum test (p-value) between standards by the LPB algorithm

TFs	α	**Standards**		
		LPX vs SBX	LPX vs BX	LPX vs Qubit-X
TF1	0.2	0.000031	0.002415	0.000005
	0.5	0.000241	0.001965	0.000002
	0.7	0.00042	0.015658	0.000031
TF3	0.2	0.002765	**0.517048**	0.000002
	0.5	0.006836	**0.318491**	0.000002
	0.7	**0.328571**	**0.393334**	**0.271155**
TF7	0.2	0.000716	**0.298944**	0.009271
	0.5	0.044919	**0.085896**	0.000664
	0.7	**0.071903**	**0.057096**	0.000058

4.2. Results and Discussion of Single-Objective Lagrange Elementary for Optimization

The performance evaluation of the single-objective Leo algorithm involves using various standard benchmark test functions from existing literature. Furthermore, we compare our results with five other well-known algorithms mentioned in the introduction. It is worth mentioning that the results of 19 classical benchmark test functions are obtained from a prior study, while we conduct the CEC-C06 tests. To determine the statistical significance of the test outcomes, we use the Wilcoxon rank-sum test for comparison. Furthermore, four measurement metrics are utilized for additional analysis and observation.

4.2.1. Classical Benchmark Test Functions

The Leo algorithm underwent 30 tests, with each test utilizing 80 search agents. In each test, the algorithm searched for the most efficient optimum solution within 500 iterations, and subsequently, the Mean and STD values were calculated from 30 round tests. Detailed information about the parameter sets for DA, PSO, GA, FDO, and LPB can be found in Tables (9 and 10) of this work. In the context of the general convex learning problem, our results are comparable to the best-known bound.

To evaluate the effectiveness of the Leo algorithm, we selected three sets of test functions, which are grouped into three distinct categories: unimodal, multimodal, and composite test functions (Arora et al., 2020). Each of these test functions evaluates different aspects of algorithm effectiveness and benchmarking specific characteristics. For instance, unimodal benchmark functions are used to assess the level of exploitation and convergence of algorithms or standard operator effect on the algorithm, as they have a single optimal solution. On the other hand, multimodal benchmark functions include

multiple optimal solutions, enabling the evaluation of the ability of algorithm to avoid local optima and explore the search space.

Indeed, the test functions encompass a variety of optimal solutions, including the global optimum and several individual optimal solutions. This characteristic resembles the scenarios often encountered in multimodal optimization problems, much like the scenarios encountered in multimodal optimization problems. To obtain a globally optimal solution, an algorithm must effectively avoid local optima. Composite benchmark functions, in particular, are constructed as a combination of blended, rotated, shifted, and biased versions of other test functions. These composite benchmarks contain a significant number of local optima and exhibit various shapes in different regions of the search landscape. Examples of such benchmark functions can be found in Tables (4-5, 4-6, and 4-7) (Hussain et al., n.d.).

Table 4-5 Unimodal benchmark functions (Hussain et al., n.d.)

Functions	Dimension	Range	Shift position	f_{min}
$\boldsymbol{TF1}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$	10	[-100, 100]	[-30, -30, … -30]	0
$\boldsymbol{TF2}(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[-10,10]	[-3, -3, … -3]	0
$\boldsymbol{TF3}(\boldsymbol{x}) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	10	[-100, 100]	[-30, -30, … -30]	0
$\boldsymbol{TF4}(\boldsymbol{x}) = \max_i \{\lvert x \rvert, 1 \le i \le n\}$	10	[-100, 100]	[-30, -30, … -30]	0
$\boldsymbol{TF5}(\boldsymbol{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_1^2)^2 + (x_i - 1)^2]$	10	[-30,30]	[-15, -15, … -15]	0
$\boldsymbol{TF6}(\boldsymbol{x}) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	10	[-100, 100]	[-750, … -750]	0
$\boldsymbol{TF7}(\boldsymbol{x}) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1]$	10	[-1.28,1.28]	[-0.25, …-0.25]	0

Table 4-6 Multimodal benchmark functions (10 dimensional) (Hussain et al., n.d.)

Functions	Range	Shift position	f_{min}
$TF8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]	[-300, … -300]	-418.9829* n When n equals to dimensions
$TF9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12,5.12]	[-2, -2, …-2]	0
$TF10(x) = -20exp\left(-0.2\sqrt{\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$	[-32, 32]		0
$TF11(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]	[-400, … -400]	0
$TF12(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\ \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + .\sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $. \quad u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases} \quad y_i = 1 + \frac{x+1}{4}$	[-50,50]	[-30, 30, … 30]	0
$TF13(x) = 0.1\{\sin^2(3\pi x1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$.	[-50,50]	[-100, … -100]	0

Table 4-7 Composite benchmark functions (Hussain et al., n.d.)

Functions	Dimension	Range	f_{min}
TF14 (CF1) $f1, f2, f3 \dots f10 =$ Sphere function $\delta1, \delta2, \delta3 \dots \delta10$ [1,1,1, ….1] $\lambda1, \lambda2, \lambda3 \dots \lambda10 = \left[\frac{5}{100}, \frac{5}{100,}, \frac{5}{100}, \dots \frac{5}{100}\right]$	10	[-5, 5]	0
TF15 (CF2) $f1, f2, f3 \dots f10$ Griewank's function $\delta1, \delta2, \delta3 \dots \delta10$, [1,1,1, ….1] $\lambda1, \lambda2, \lambda3 \dots \lambda10 = \left[\frac{5}{100}, \frac{5}{100,}, \frac{5}{100}, \dots \frac{5}{100}\right]$	10	[-5, 5]	0
TF16 (CF3) $f1, f2, f3 \dots f10$ Griewank's function $\delta1, \delta2, \delta3 \dots \delta10$, [1,1,1, ….1] $\lambda1, \lambda2, \lambda3 \dots \lambda10 =$ [1,1,1, ….1]	10	[-5, 5]	0
TF17 (CF4) $f1, f2 =$ Ackley's function, $f3, f4 =$ Rastrigin's function, $f5, f6 =$ Weierstrass functio, $f7, f8 =$ Griewank's function, $f9, f10 =$ Sphere function, $\delta1, \delta2, \delta3 \dots \delta10 =$ [1,1,1, ….1] $\lambda1, \lambda2, \lambda3 \dots =$ $\left[\frac{5}{32}, \frac{5}{32,}, 1,1, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}\right]$	10	[-5, 5]	0
TF18 (CF5) $f1, f2 =$ Rastrigin's function, $f3, f4 =$ Weierstrass function, $f5, f6 =$ Griewank's function, $f7, f8 =$ Ackley's function $f9, f10 =$ Sphere function, $\delta1, \delta2, \delta3 \dots \delta10 =$ [1,1,1, ….1] $\lambda1, \lambda2, \lambda3 \dots \lambda10 =$ $\left[\frac{1}{5}, \frac{1}{5,}, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}\right]$	10	[-5, 5]	0
TF19 (CF6) $f1, f2 =$ Rastrigin's function, $f3, f4 =$ Weierstrass function, $f5, f6 =$ Griewank's function, $f7, f8 =$ y's function, $f9, f10$Sphere function, $\delta1, \delta2, \delta3 \dots \delta10$ [0.1,0.2,0.3, 0.4,0.5,0.6,0.7,0.8,0.9,1], $\lambda1, \lambda2, \lambda3 \dots \lambda10 =$ $\left[0.1 * \frac{1}{5}, 0.2 * \frac{1}{5}, 0.3 * \frac{5}{0.5}, 0.4 * \frac{5}{0.5}, 0.5 * \frac{5}{100}, 0.6 * \frac{5}{100}, 0.7 * \frac{5}{32}, 0.8 * \frac{5}{32}, 0.9 * \frac{5}{100}, 1 * \frac{5}{100}\right]$	10	[-5, 5]	0

The Leo algorithm is executed following the fundamental steps as indicated earlier. The obtained results are then compared to those of three well-known alternative algorithms, namely DA, PSO, and GA. The results of DA, PSO, and GA can be found in Table (4-8), reported in the papers by (Abdullah and Ahmed, 2019; Mirjalili, 2016; Rahman and Rashid, 2021). Thus, in the first unimodal function, the DA algorithm demonstrated optimal performance, while the PSO algorithm showed superior efficiency in the sixth test function. However, upon analyzing the results of tests TF2, TF3, TF4, TF5, and TF7, it becomes evident that the Leo algorithm consistently outperforms the other algorithms in terms of exploitation capacity and achieving optimal results. Furthermore, when compared to the FDO and LPB algorithms, whose results are reported in papers by (Abdullah and Ahmed, 2019 and Rahman and Rashid, 2021) respectively, the Leo algorithm consistently achieved superior outcomes. However, it should be noted that for TF1 and TF6, the Leo algorithm exhibited even better levels of exploitation and convergence. Leo is the only algorithm to do better than the others in TF12 and also outperforms them in TF11 and TF13, earning the second spot. Leo obtained the second rank in TF11 and TF13 when compared to all other methods in Table (4-9) except for these two functions. In multimodal functions, the Leo algorithm achieves the second rank, demonstrating an improved outcome in TF12 compared to the other algorithms in TF11 and TF13. Additionally, in Table (4-9), Leo performs better than every other algorithm, except for TF9 and TF10, where it attains the runner-up position. Leo also outperforms DA, PSO, and GA in composite test functions for each test function. Furthermore, Leo algorithm consistently outperforms the LPB algorithms in all test functions. In comparison to FDO, Leo achieves the second rank with better performance in all test functions, as shown in Tables (4-8 and 4-9).

Table 4-8 Comparing the results of Leo with DA, PSO, and GA algorithms on classical test functions

TF	**Leo**		**DA**		**PSO**		**GA**	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
TF1	2.69874E-09	7.49992E-09	**2.85E-18**	**7.16E-18**	4.20E-18	1.31E-18	748.5972	324.9262
TF2	**3.7305E-06**	**3.95635E-06**	1.49E-05	3.76E-05	0.003154	0.009811	5.971358	1.533102
TF3	**5.31468E-09**	**2.07901E-08**	1.29E-06	2.10E-06	0.001891	0.003311	1949.003	994.2733
TF4	**3.60286E-05**	**3.22842E-05**	0.000988	0.002776	0.001748	0.002515	21.16304	2.605406
TF5	**10.60296667**	**13.93285916**	7.600558	6.786473	63.45331	80.12726	133307.1	85007.62
TF6	4.31581E-10	5.51803E-10	4.17E-16	1.32E-15	**4.36E-17**	**1.38E-16**	563.8889	229.6997
TF7	**0.001449721**	**0.002690575**	0.010293	0.010293	0.005973	0.003583	0.166872	0.072571
TF8	-2989.147333	202.684514	-2857.58	383.6466	-7.10E+11	1.2E+12	**-3407.25**	**164.478**
TF9	37.07867	12.2775166	16.01883	9.479113	**10.44724**	**7.879807**	25.51886	6.66936
TF10	4.8836E-05	2.89869E-05	**0.23103**	**0.487053**	0.280137	0.601817	9.498785	1.271393
TF11	**2.7393E-08**	**5.51514E-08**	0.193354	0.073495	0.083463	0.035067	7.719959	3.62607
TF12	1.87667E-08	2.89749E-08	0.031101	0.098349	**8.57E-11**	**2.71E-10**	1858.502	5820.215
TF13	**8.90491E-09**	**1.88063E-08**	0.002197	0.004633	0.002197	0.004633	68047.23	87736.76
TF14	**6.9979**	**5.833242622**	103.742	91.24364	150	135.4006	130.0991	21.32037
TF15	**0.001673093**	**0.003539145**	193.0171	80.6332	188.1951	157.2834	116.0554	19.19351
TF16	**-0.622100333**	**0.396782974**	458.2962	165.3724	263.0948	187.1352	383.9184	36.60532
TF17	**1.788405333**	**2.237631581**	596.6629	171.0631	466.5429	180.9493	503.0485	35.79406
TF18	**3.590623333**	**0.711917144**	229.9515	184.6095	136.1759	160.0187	118.438	51.00183
TF19	**-2.670808**	**1.185307969**	679.588	199.4014	741.6341	206.7296	544.1018	13.30161

Table 4-9 Comparing the results of Leo with FDO and LPB algorithms on classical test functions

TF	Leo		FDO		LPB	
	Mean	STD	Mean	STD	Mean	STD
TF1	2.69874E-09	7.49992E-09	**7.47E-21**	**7.26E-19**	0.001877545	0.002093616
TF2	**3.7305E-06**	**3.95635E-06**	9.39E-06	6.91E-06	0.005238111	0.003652512
TF3	**5.31468E-09**	**2.07901E-08**	8.55E-07	4.40E-06	36.4748883	29.22415523
TF4	**3.60286E-05**	**3.22842E-05**	6.69E-04	0.0024887	0.393866	0.135818
TF5	**10.60296667**	**13.93285916**	23.501	59.7883701	16.76919	22.19251
TF6	4.31581E-10	5.51803E-10	**1.42E-18**	**4.75E-18**	0.00203173	0.0027832
TF7	**0.001449721**	**0.002690575**	0.544401	0.3151575	0.004975	0.002965
TF8	-2989.147333	202.684514	-2285.207	206684.91	**-3747.65**	**189.0206**
TF9	37.07867	12.2775166	14.56544	5.202232	**0.001567**	**0.001842**
TF10	4.8836E-05	2.89869E-05	**4.00E-15**	**6.38E-16**	0.017933	0.013532
TF11	**2.7393E-08**	**5.51514E-08**	0.568776	0.1042672	0.066355	0.030973
TF12	**1.87667E-08**	**2.89749E-08**	19.83835	26.374228	2.79E-05	3.84E-05
TF13	**8.90491E-09**	**1.88063E-08**	10.2783	7.42028	0.000309	0.000512
TF14	6.9979	5.833242622	**3.79E-07**	6.32E-07	0.998004	**1.26E-11**
TF15	0.001673093	0.003539145	**0.001502**	**0.0012431**	0.002358	0.003757
TF16	-0.622100333	0.396782974	0.006375	0.0105688	**-1.03163**	**2.46E-06**
TF17	1.788405333	2.237631581	23.82013	0.2149425	**0.397888**	**3.16E-06**
TF18	3.590623333	0.711917144	222.9682	**9.96E-06**	**3.000142**	0.000283
TF19	-2.670808	1.185307969	22.7801	0.0103584	**-3.86278**	**9.61E-07**

4.2.2. CEC-C06 2019 Benchmark Test Functions

In real-world scenarios, there are situations where obtaining an accurate solution is more crucial than quick results. Moreover, refining an algorithm and running it multiple times is possible for many individuals. Clients seek the most effective algorithm that suits their specific needs, regardless of the time it takes. As part of this modern benchmark collection, ten test functions were introduced at the CEC-2019 conference (Bacanin et al., 2022; Brest et al., 2019). These test functions have been evaluated using Leo algorithm to assess their performance. The test functions known as "The 100-Digit Challenge" are intended for use in annual optimization competitions, as shown in Table (4-10). These functions, introduced in CEC-2019, have become popular and are considered cutting-edge benchmarks for evaluating the performance of various algorithms in addressing real-world problems. In this evaluation, we have selected several highly competitive and widely

employed algorithms, including DA, WOA, SSA, FDO, LPB, and FOX. These algorithms were chosen due to their extensive citations in the literature background to determine exceptional performance on benchmark test functions and practical applications. The creators of these algorithms facilitate their accessibility to enhance algorithm credibility and research replicability.

Table 4-10 CEC-2019 benchmarks "the 100-digit challenge" (Brest et al., 2019)

No.	Functions	Dimension	Range	f_{min}
1	STORN'S CHEBYSHEV POLYNOMIAL FITTING PROBLEM	9	[-8192, 8192]	1
2	INVERSE HILBERT MATRIX PROBLEM	16	[-16384, 16384]	1
3	LENNARD-JONES MINIMUM ENERGY CLUSTER	18	[-4,4]	1
4	RASTRIGIN'S FUNCTION	10	[-100, 100]	1
5	GRIEWANGK'S FUNCTION	10	[-100, 100]	1
6	WEIERSTRASS FUNCTION	10	[-100, 100]	1
7	MODIFIED SCHWEFEL'S FUNCTION	10	[-100, 100]	1
8	EXPANDED SCHAFFER'S F6 FUNCTION	10	[-100, 100]	1
9	HAPPY CAT FUNCTION	10	[-100, 100]	1
10	ACKLEY FUNCTION	10	[-100, 100]	1

CEC01 to CEC03 functions have varying dimensions, while the remaining functions share a common range between [-100, 100]. Thus, CEC01 to CEC03 functions are unaffected by shift and rotation, while CEC04 to CEC10 functions undergo such transformations. This ensures that all test procedures remain scalable and flexible. The parameter set for the CEC benchmark allows algorithms to run 30 times with 30 or 80 agents according to basic of algorithms and conduct 500 iterations for landscape search. In Table (4-11), Leo outperforms widely cited algorithms in the literature, except for the CEC04 test function. Leo achieves comparable results to WOA in benchmarks like CEC02 and CEC05. However, as indicated in Table (4-12), Leo surpasses FOX, a recent algorithm. Markedly, Leo performs exceptionally well in just CEC02 but for CEC06, Leo better performance rather than FDO and LPB.

Table 4-11 Comparing the results of Leo with DA, WOA, and SSA algorithms on CEC-2019 test functions

CEC	Leo		DA		WOA		SSA	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
CEC01	**7294147266**	**5767198154**	5.43E+10	6.69E+10	4.11E+10	5.42E+10	6.05E+09	4.75E+09
CEC02	17.47763	0.098108754	78.0368	87.7888	**17.3495**	**0.0045**	18.3434	0.0005
CEC03	**12.70311**	**0.000889537**	13.7026	0.0007	13.7024	0	13.7025	0.0003
CEC04	69.86527333	23.78089555	344.3561	414.0982	394.6754	248.5627	**41.6936**	**22.2191**
CEC05	2.760246667	0.432754261	2.5572	0.3245	2.7342	0.2917	**2.2084**	**0.1064**
CEC06	**3.01982**	**0.755956506**	9.8955	1.6404	10.7085	1.0325	6.0798	1.4873
CEC07	**195.5583033**	**236.5351502**	578.9531	329.3983	490.6843	194.8318	410.3964	290.5562
CEC08	**5.062283333**	**0.459751941**	6.8734	0.5015	6.909	0.4269	6.3723	0.5862
CEC09	**3.26147**	**0.744492954**	6.0467	2.871	5.9371	1.6566	3.6704	0.2362
CEC10	**20.01238667**	**0.028550895**	21.2604	0.1715	21.2761	0.1111	21.04	0.078

Table 4-12 Comparing the results of Leo with FDO, LPB, and FOX algorithms on CEC-2019 test functions

CEC	Leo		FDO		LPB		FOX	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
CEC01	7294147266	5767198154	**4585.27**	**20707.627**	7494381364	8138223463	2.58E+04	22624.86
CEC02	**17.47763**	**0.098108754**	**4**	**3.22414E-09**	17.63898	0.31898	18.3442	0.000529
CEC03	12.70311	0.000889537	13.7024	1.649E-11	**12.7024**	**0**	13.7025	0.000449
CEC04	69.86527333	23.78089555	**34.0837**	**16.528865**	77.90824	29.88519	1.06E+03	501.8163
CEC05	2.760246667	0.432754261	2.13924	0.085751	**1.18822**	**0.10945**	6.295	1.27819
CEC06	**3.01982**	**0.755956506**	12.1332	0.600237	3.73895	0.82305	5.0325	1.285264
CEC07	195.5583033	236.5351502	**120.4858**	**13.59369**	145.28775	177.8949	456.3214	189.4313
CEC08	5.062283333	0.459751941	6.1021	0.756997	**4.88769**	**0.67942**	5.6778	0.52774
CEC09	3.26147	0.744492954	**2**	**1.5916E-10**	2.89429	0.23138	3.7959	0.339462
CEC10	20.01238667	0.028550895	**2.7182**	**8.8817E-16**	20.00179	0.00233	20.9878	0.005376

4.2.3. Statistical Tests and Scalability Analysis

As mentioned earlier, statistical tests for comparing multiple result groups use both parametric and non-parametric approaches. The two-sample T-test necessitates certain assumptions, while the Wilcoxon Rank-Sum Test serves as a non-parametric alternative Tables (4-13 and 4-14) present the statistical analysis results in these distributions, where the samples consist of 30 rounds of solution for Leo algorithm which are noted in appendix tables (Table 8-1 to 8-6).

Researchers should be confronted a noteworthy challenge when they applied a statistical model randomly to determine significance values (p-values), lacking prior studies to select a specific model for evaluating performance outcomes. The study employed rigorous statistical testing to illuminate substantial performance variations between pairs of algorithms, emphasizing the pivotal role of statistical significance in comparative analysis. Furthermore, it offers valuable insights into the suitability of algorithms for diverse optimization challenges, empowering professionals with information for informed decision-making. This is achieved through the identification of algorithm pairs with favourable statistical distributions, facilitating practical algorithm selection. The presumptions concerning near equality or symmetry are upheld, but the fulfilment of requirements regarding variance spread and normalcy is not entirely satisfactory.

The research shows that DA results with Leo are statistically significant compared to PSO and GA. Additionally, DA was already evaluated against PSO, GA, FDO, and LPB algorithms in this research work. To further assess Leo's performance, statistical comparisons between Leo and DA, FDO, and LPB algorithms are presented in Table (4-13).

In all statistical tests conducted on unimodal, multimodal, and composite test functions, Leo algorithm's results are deemed significant, leading to the rejection of the null hypothesis, except for TF6 and TF12 when compared to DA, where the p-values are greater than 0.05. Additionally, Leo consistently outperforms the FDO and LPB algorithms, except for TF5 and TF15, where the results do not reject the null hypothesis. Furthermore, there are no significant differences between Leo and FDO in TF17.

Table 4-13 P-value by the Wilcoxon rank-sum test overall runs for classical benchmark test functions.

TF	Leo VS DA	Leo VS LPB	Leo VS FDO
TF1	0.000031	0.000031	0.000002
TF2	0.000002	0.000002	0.047162
TF3	0.000002	0.000002	0.002585
TF4	0.000031	0.000002	0.000002
TF5	0.000148	**0.781264**	**0.557743**
TF6	**0.057096**	0.000002	0.000002
TF7	0.000002	0.000097	0.000002
TF8	0.031603	0.000002	0.000016
TF9	0.000002	0.000002	0.000002
TF10	0.000002	0.000002	0.000002
TF11	0.000002	0.000002	0.000002
TF12	**0.328571**	0.000002	0.000002
TF13	0.517048	0.000002	0.000002
TF14	0.000013	0.000013	0.002929
TF15	0.000359	**0.012453**	**0.781264**
TF16	0.000001	0.000002	0.000115
TF17	0.000001	0.000002	**0.120288**
TF18	0.00015	0.000393	0.00015
TF19	0.000002	0.000002	0.000004

Interestingly, in all composite test functions, the null hypothesis is not rejected for DA, FDO, and LPB algorithms across all 30 individual tests. Table (4-14) showcases the statistical comparisons of Leo with DA, SSA, WOA, FDO, and FOX algorithms, employing the Wilcoxon rank-sum test. The results for Leo are highly significant, as the p-values are significantly less than 0.05, leading to the rejection of the null hypothesis when compared to DA and FOX. Additionally, DA results for Leo are statistically significant

when compared to SSA, WOA, and FDO, except for CEC01 with SSA, CEC05 with WOA, and CEC07 with FDO.

Table 4-14 P-value by the Wilcoxon rank-sum test overall runs for CEC-2019 test functions

TF	Leo VS DA	Leo VS SSA	Leo VS WOA	Leo VS FDO	Leo VS FOX
CEC01	0.000012	**0.360039**	0.038723	0.000002	0.000002
CEC02	0.000002	0.000002	0.000002	0.000002	0.000002
CEC03	0.000001	0.000001	0.000001	0.000001	0.000001
CEC04	0.000012	0.000125	0.000002	0.000005	0.000002
CEC05	0.033264	0.000026	**0.688359**	0.000004	0.000002
CEC06	0.000002	0.000002	0.000002	0.000002	0.000002
CEC07	0.000359	0.011079	0.000115	**0.171376**	0.000148
CEC08	0.000002	0.000002	0.000002	0.000008	0.000082
CEC09	0.000003	0.002765	0.000002	0.000002	0.001593
CEC10	0.000002	0.000002	0.000002	0.000002	0.000002

4.2.4. Quantitative Measurement Metrics

In this subsection, the proposed measurement metrics are used to observe and analyze Leo algorithm's performance in more detail. This experiment evaluates convergence and assesses how well Leo algorithm tackles real-world problems. The first metric measures convergence, drawing an analogy to how vaccinations enhance immune responses within the body landscape. For each experiment, benchmark functions are selected carefully, covering unimodal, multimodal, and composite benchmark functions, respectively. Figure 4-10 illustrates the agents' rapid exploration of the entire region, gradually converging towards optimality, with the offspring finding the most optimal positions. This pattern is observed in each experiment for (FT2), (TF10), and (TF17). The experiment records the agents' performance from the beginning to the end of the test, showing their progress over time.

In Figure 4-11, the second experiment for each of (FT2), (TF9), and (TF17) demonstrates Leo starting with a high fitness value and gradually reducing it until reaching the desired optimum. In multimodal tests, functions undergo rapid development within a few iterations. The third test metric, displayed in

Figure 4-12 for each of (FT2), (TF9), and (TF17), shows the average fitness value of all Leo agents decreasing significantly throughout the iterations. The results indicate that the algorithm not only improves the global most effective agent but also enhances the optimal solution for all agents. The fourth measure records the convergence of the global most optimal agent throughout each iteration, showing that it becomes more accurate as the number of iterations increases. Figure 4-13, observed in each experiment of (FT2), (TF8), and (TF17), illustrates a significant shift due to the algorithm's focus on local search and exploitation. This emphasizes the algorithm's ability to fine-tune and improve the overall performance over time.

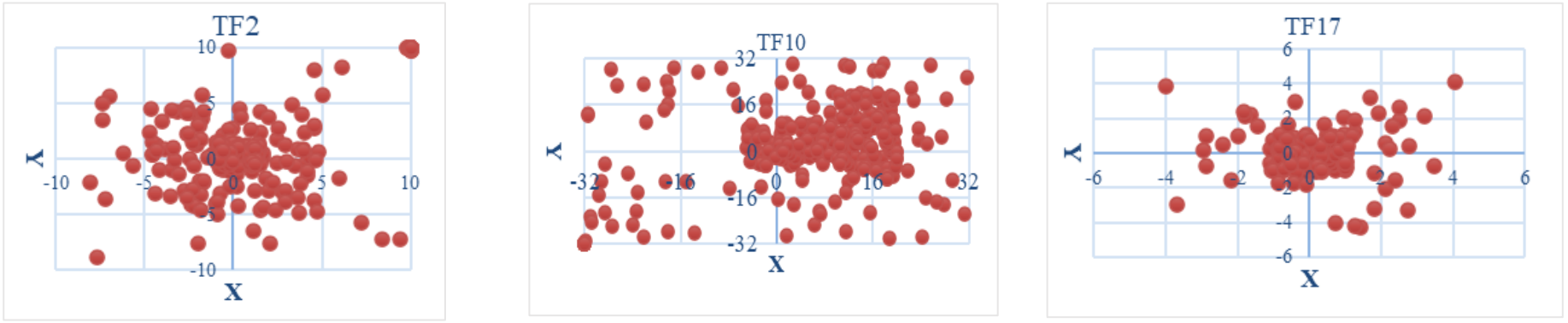

Fig. 4-10 Search history of the Leo algorithms on unimodal, multimodal, and composite test functions

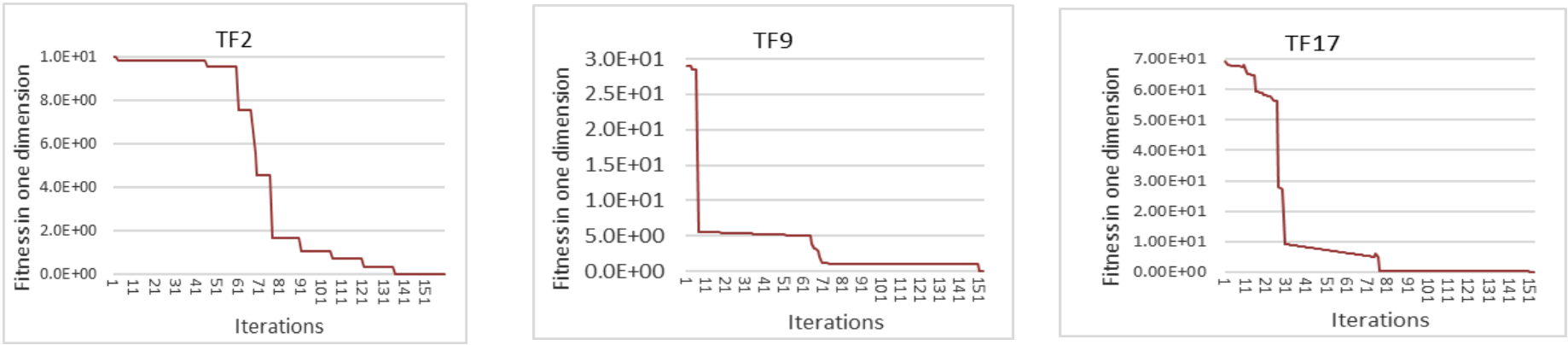

Fig. 4-11 The trajectory of Leo's search agents on unimodal, multimodal, and composite test functions

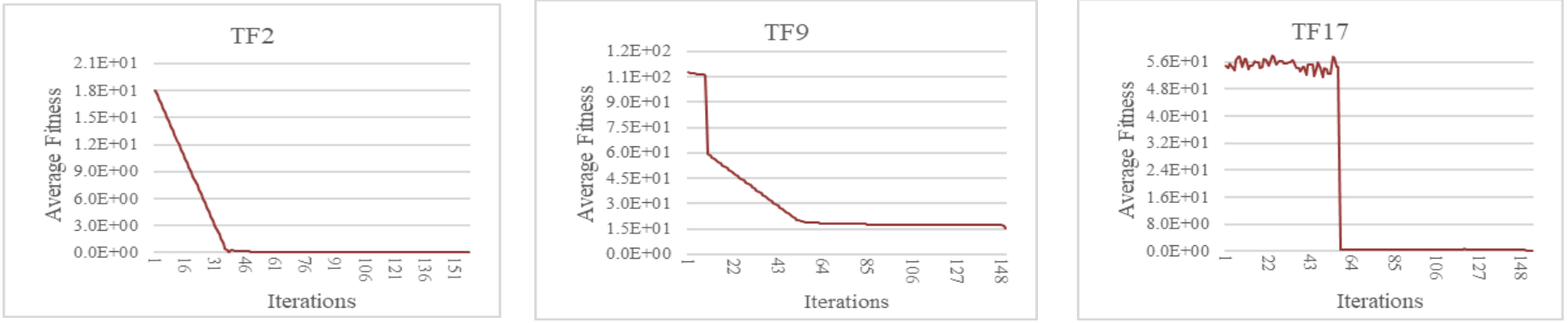

Fig. 4-12 The average fitness of Leo's search agents on unimodal, multimodal, and composite test function

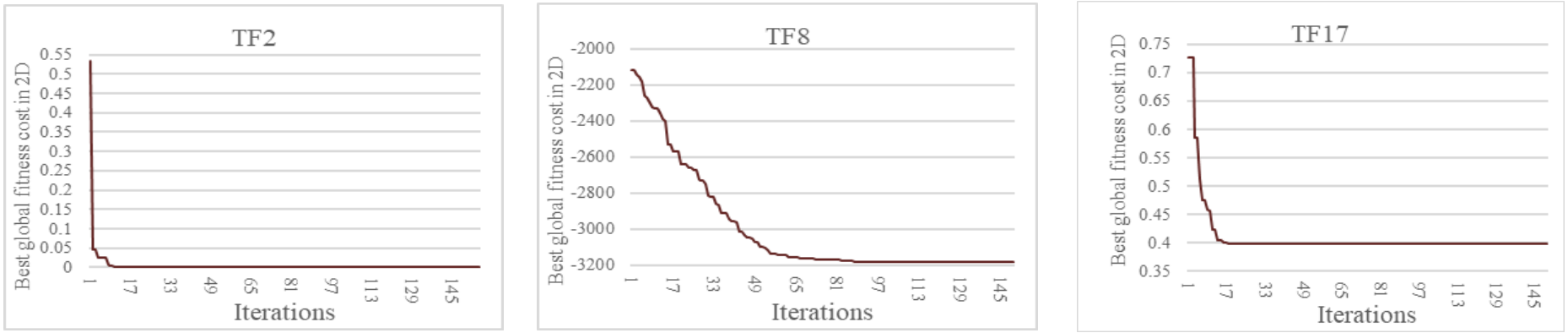

Fig. 4-13 Convergence curve of Leo algorithms on unimodal, multi-modal, and composite test function

4.2.5. Real-World Application

As with any other metaheuristic algorithm, Leo is capable of addressing application-specific challenges in the real world. In this section, Leo is utilized in two distinct applications, offering customized enhancements that cater to their unique real-world contexts.

4.2.5.1. The Pathological IgG Fraction in the Nervous System

The determination method used is not influenced by variables that could potentially impact individuals, such as sex, blood-brain barrier condition, extraction volume of cerebrospinal fluid (CSF), and the specific protein measurement technique. This approach ensures optimal evaluation of pathogenic IgG values in CSF and outperforms other methods described in the existing literature, particularly in statistical analysis and biochemical considerations (Link and Huang, 2006). The core goal of this problem is to find the best solution that effectively evaluates pathological IgG values in CSF, considering the variations in the nervous system. Taking both statistical and functional aspects into account, the frequency of the regression line passing through the source, represented by equation (4.1), is considered reasonable and has been improved through the aggregation of statistical regression lines (LEFVERT and LINK, 1985; Su and Chiu, 1986). Most inquiries concentrate on establishing a connection between serum and fluid albumin concentrations. However, in this real-world application, a link between serum albumin levels and IgG levels in cerebrospinal fluid is demonstrated. To pinpoint the optimal correlation, Leo algorithm is employed. This method helps identify the most suitable point, ensuring a comprehensive examination of the correlation between serum albumin and IgG concentrations in CSF.

Equation (4.2) can be utilized to calculate the locally generated concentration of pathological IgG $(IgGp)$ in CSF by evaluating the patient's unique albumin ratio and IgG quotient. Moreover, $STD_{(x,y)}$ represents the standard deviation of the (y) values from the regression line within the range of (-0.001, +0.001). These two variables, along with the regression line, provide the confidence interval of the IgG quotient (y) for a given albumin quotient (x).

$$IgGp = IgG(CSF) - (0.43\, Alb(Serum) - Alb(CSF) + 0.001) * IgG(Serum) \quad (4.1)$$

To prove that: $IgGp = X_i$ So, IgG (IgGp) = Y (X_i) then:

$$Y(X_i) = \sum_{i=1}^{n} (0.41 + 0.0014\, X_i) \quad (4.2)$$

The stained gel strip reveals a cutting sequence with twelve fractions, which are then employed to calculate the number of search agents. Stippled regression lines depict the level of each proportion. Neutralization is represented by the two antibody actions, which are displayed for statistical-level results (Vandvik and Norrby, 1973). Considering the constraints of equation (4.2), the problem is addressed using Leo algorithm. The outcomes are depicted in Figure 4-14, demonstrating both the global average fitness and the average fitness value for each iteration. To optimize the problem, twelve search agents were employed over 150 iterations. The results indicate that the optimal solution was attained at iteration 61, with a value of 5.088.

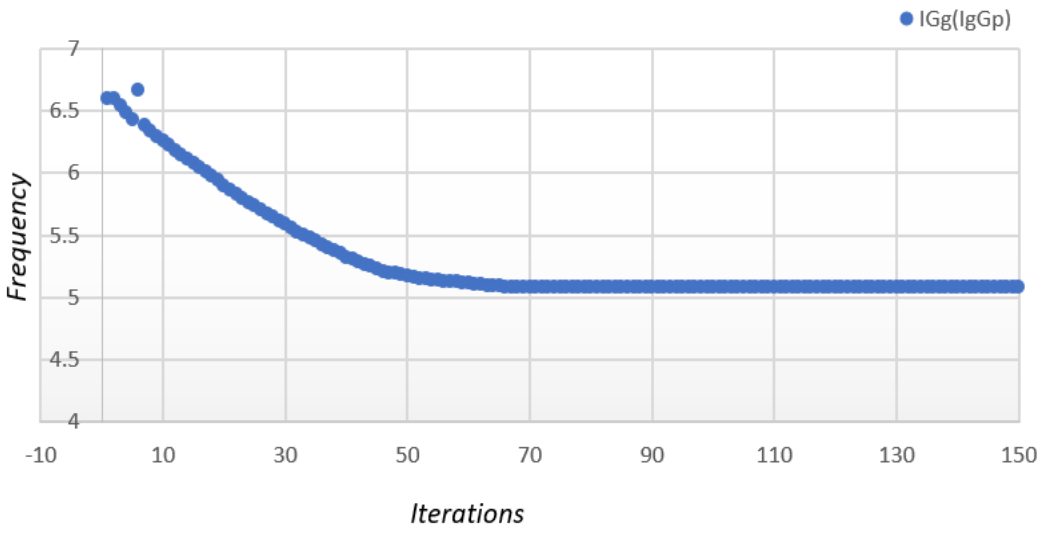

Fig. 4-14 Global best with average fitness results from for150 Iteration with 12 search agents (dimensions) in (IgGp) fraction in the nervous system

4.2.5.2. Integrated Cyber-Physical-Attack for Manufacturing System

Computational analysis can be a valuable asset for the security community, as it aids in comprehending risks and analyzing attacker behavior in cyber-physical attacks. This understanding allows for actual responses to adversarial behaviors at various stages of an attack. Despite its potential benefits, there is a notable gap in research concerning the evaluation and assessment of defensive systems, particularly from a security standpoint. Therefore, it becomes imperative to change a suitable theoretical model to identify optimal solutions and selecting the global one.

One promising approach to address this challenge is by utilizing or integrating cyber-physical attacks on manufacturing systems (CPAMS). Such integration can enhance flexibility and responsiveness while maintaining product quality to meet clients' demands. By doing so, manufacturers can strengthen their security measures and ensure a robust defence against potential threats. Through computational analysis and the implementation of an effective theoretical model, the security community can better protect cyber-physical systems and respond proactively to evolve cyber threats in manufacturing

environments (Tran et al., 2019). The presented model is an object-oriented Petri net-based formal model of a cyber-physical-attack manufacturing system to enhance system integrity during dynamic simulation (Yu et al., 2017). The verification and validation of this system can be achieved by optimizing Leo system using mathematical techniques and supportive tools, including Petri nets.

Petri nets (Bordbar et al., 2000) have evolved into captivating artistic masterpieces for distributed systems, providing mesmerizing visualizations and profound evaluations for a diverse range of complex applications. These applications span from orchestrating communication networks and refining healthcare systems to harnessing the power of artificial intelligence and optimizing manufacturing engineering systems, creating an imaginative canvas of possibilities. Petri nets also serve as computational mathematical tools, facilitating the simulation and analysis of dynamic systems. These nets form a directed graph model, where arcs (F) connect two sets of nodes: locations (P) and transitions (T). Tokens (or 'marks') are represented by dots inside the spots. Figure 4-15 illustrates this configuration, particularly when $R1$ is explicitly defined in the net interpretation between transitions (T) and locations (P).

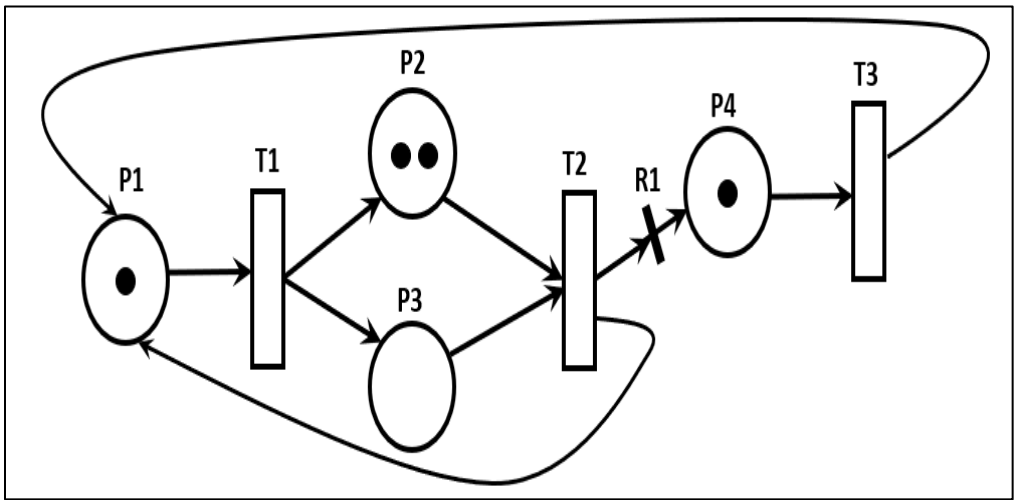

Fig. 4-15 The network station is represented by a stochastic Petri net

Syntax: A Petri net is a net of the form $PN = (N, M, W)$, which extends the elementary net so that:

Given a net $N = (P, T, F)$, a configuration is a set C so that $C \subseteq P$.

$M: P \rightarrow Z$ is a place multiset, where is a countable set, covers the concept of *configuration*, and is normally described concerning Petri net diagrams as a *marking*.

$W: F \rightarrow Z$ is an arc multiset. The count (or weight) for each arc indicates the multiplicity of arcs that can be calculated.

Its transition relation can be described as a pair of $|P|$by $|T|$ matrices:

F^-, defined by $\forall s, t: F^- [p, t] = F(p, t)$

F^+, defined by $\forall s, t: F^+ [p, t] = F(t, p)$

When the pre-set of a transition t is the set of its input places: ${}^*t = \{p \in P) | F(p, t) > 0\}$; its posset is the set of its output places: $t^* = \{p \in P) | F(p, t) > 0\}$;. Definitions of pre- and post-sets of places are analogous.

Following the syntax of Petri nets, system evaluation entails examining numerous instances of CPAMS. In this context, various nodes, including machines, robots, sensors, and AGVs, are susceptible to becoming infected by malicious software and subsequently transmitting it to other vulnerable nodes. As time progresses, these vulnerable nodes transform into infected nodes. Once the malicious software is eliminated, the infected nodes can transition back to being recovered nodes. The susceptible nodes, infectious nodes, and recovered nodes are symbolized, respectively, as S, I, and R (Singh et al., 2018). When considering the attributes of Logistic, new nodes are categorized as susceptible nodes, and they exhibit a growth rate denoted by p.

To combat harmful software bifurcation in CPAMS, a novel hybrid bifurcation law control strategy was introduced by (Zhou et al., 2018). This

strategy, defined by equation (4.3), is designed to alleviate the adverse effects of bifurcations and maintain the integrity of CPAMS. In the equation, K_1 denotes the variable parameter, while K_2 represents the feedback parameter.

$$N\big(I(p,t)\big) = k_1 F\big(I(p,t)\big) + k_2(I(p,t) + I(p,t)^3) \tag{4.3}$$

To evaluate the probabilistic complex system, the fitness function $F(I(p,t)$ is derived from the Jacobian matrix at the equilibrium point (Jia, 2007). This process yields simulation results and the optimal point that regulates the probabilistic occurrence of harmful nodes, updated by the susceptible node. Consequently, the probabilistic equation (4.4) serves as a method to identify the node point where it equals zero.

$$F((I(p,t)) = \sum_{i=1}^{n} X^3 + A\sum_{i=1}^{n} X^2 + B\sum_{i=1}^{n} X + C \tag{4.4}$$

When,

$$A = 0.0283\left(1 + \frac{1}{d} - k_2\right)$$

$$B = \frac{0.0283 - 1.0283k_2}{d}$$

$$C = \frac{0.0013\,k_1 - 0.0283k_2}{d}$$

In this realistic scenario, let's consider a set of parameters: the number of nodes (d) ranges from 15 to 36, k_1 varies from 0 to 1, and k_2 varies from 0.1 to 0.5. To identify the optimal method for updating infection nodes, the Leo algorithm is utilized. The algorithm produces valuable outcomes, including the average fitness value and the global average fitness for each iteration. These results offer valuable insights into the effectiveness of different updating strategies for infection nodes in the given complex system. In this training, a total of 300 iterations were conducted with 10 search agents. The findings indicate that the most successful result was achieved during iteration

209 of the globally optimized solution, which yielded a fitness value of 0.072028. Figure 4-16 visually represents the progression of the process.

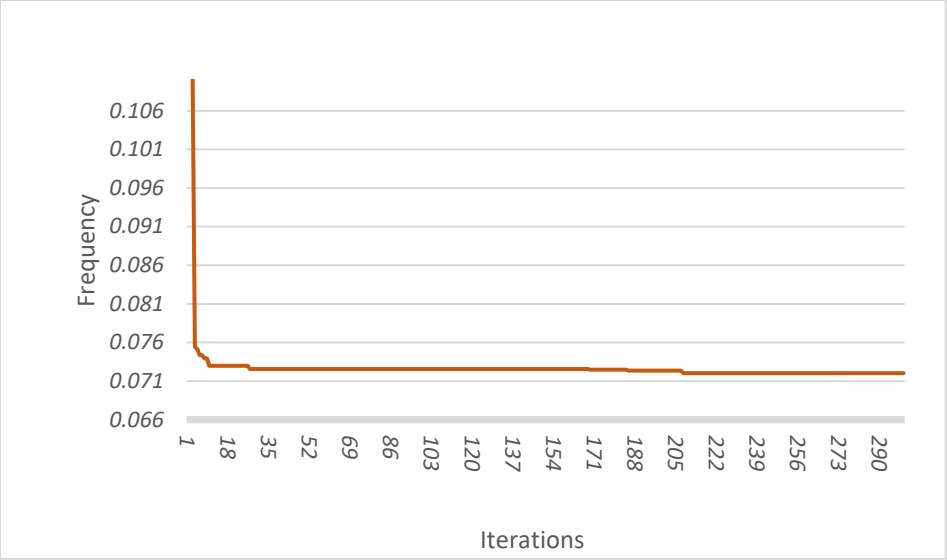

Fig. 4-16 Fitness results in Leo process for 300 Iteration with 10 search agents depend on the Jacobian matrix for cyber-physical-attack in the manufacturing system

CHAPTER FIVE

5. Conclusions, Future Works and Limitations

In this chapter, we provide a comprehensive summary of the proposed technique and demonstrate its main findings. Furthermore, we offer valuable recommendations for potential modifications and approaches to enhance the effectiveness of the technique in the future. These suggestions aim to further improve the proposed approaches' effectiveness and efficiency.

5.1. Conclusions

Optimization has been proposed for solving complex dependent problems. These multiple optimization algorithms draw inspiration from the search agent's reproductive swarming process, which involves the birth of new search agents that explore optimal individuals and sites. Thus, metaheuristic population-based algorithms such as auto-self optimization were powerful optimization techniques widely used to solve complex problems across various domains. These algorithms were designed to mimic natural systems such as evolution, swarm intelligence, and social interactions to find near-optimal or even optimal solutions. As a result, metaheuristic population-based algorithms offer a powerful and flexible approach to solve optimization problems in diverse domains. Their ability to perform global exploration, efficiency, and scalability make them attractive tools for tackling real-world challenges.

Exploration refers to the process of searching the solution space broadly to discover new and diverse regions that may contain better solutions. The primary goal of exploration is to ensure that the algorithm explores a wide range of possibilities, avoiding premature convergence to suboptimal solutions. During exploration, the algorithm may focus on moving away from the current solutions and exploring uncharted regions of the search space. This helps in identifying novel solutions and understanding the overall

landscape of the problem such as algorithms based on GA and PSO. Exploitation involves intensively searching the local neighbourhood of promising solutions to find the best possible solutions in the vicinity. The main objective of exploitation is to refine and improve the quality of solutions by focusing on the most promising areas of the search space. Exploitation is often characterized by a focus on the best-known solutions, exploiting their local structure and making incremental improvements. This helps in refining the search around regions where good solutions are likely to be found.

Thus, striking the right balance ensures that the algorithm is capable of both exploring diverse regions of the solution space and exploiting the local structure to refine solutions. This balance is often controlled through algorithm parameters, such as mutation rates, crossover probabilities, or particle velocities, depending on the specific optimization algorithm being used. Achieving an optimal balance is essential for effectively solving complex optimization problems.

Nevertheless, users should be aware of their non-deterministic nature, the need for parameter tuning, and the possibility of varying convergence speeds based on the problem at hand. Optimization processes can be improved by suggesting, understanding, and selecting the most suitable algorithm tailored to a specific problem. By carefully considering the problem's characteristics and requirements, one can make informed decisions regarding the choice of algorithm aiming at maximizing the chances of obtaining global solutions in a timely manner.

In this dissertation, our goal was achieved through two pivotal works in the field of metaheuristic optimization. First part, the involvement included the creation of a standardized variant of the crossover operator, known as LPX. As well, a comprehensive overview of conventional types of crossover operators was undertaken. This novel operator presented a fresh perspective

on the amalgamation of genetic information and solution optimization. The integration of encoding techniques and standard operators, with a special emphasis on crossover operators, has been recognized as a pivotal factor in bolstering the capabilities of metaheuristic optimization and influencing its overall performance and results.

In the second part, a groundbreaking algorithm titled "Leo" has been introduced. This innovative algorithm offered an effective and novel approach to tackle optimization challenges, demonstrating promising outcomes across various applications. These outcomes collectively have propelled optimization science and provided invaluable tools for solving complex problems. Furthermore, Leo was rigorously tested and validated through two real-life applications, showcasing its practical applicability and robustness.

In the introductory phase, the primary objective was to aid researchers in identifying an efficient crossover operator that could lead to selecting a global solution for the problem under investigation. A significant proportion of these standards were notable for their computational simplicity, resulting in faster calculations, as demonstrated through heuristic, exploitation, and convergence evaluations of the selected methods. Additionally, these crossover operators facilitated the generation of a diverse set of offspring by combining attributes from two-parent solutions. Thus, this study led to the enhancement of novel mathematical evolutionary algorithms' performance through the implementation of an improved standard option for the crossover operator.

Typically, the standard crossover operators were classified into three types based on their mathematical definitions: binary-coded crossover, real-coded crossover (floating point), and order-coded problem crossover. With the introduction of LPX as a novel mathematical approach to crossover standards, its effectiveness on algorithms was examined and compared to other existing standards to assess its efficiency. Likewise, the selection of these random

values has played a crucial role in determining the appropriate range for generating newly developed population-based populations. Besides, the study performed a heuristic evaluation to assess the technique's proficiency in generating parent chromosomes, comparing it with BX and SBX methods. Through a comparison involving three test functions from classical benchmark testing functions, LPX exhibited the most performance in terms of exploitation rate and convergence fitness for the chosen random values. LPX is built upon the stationary Lagrange multiplier (λ), derived from the LDF theorem, which proved to be a superior standard compared to BX and SBX in the obtained results.

The first part of excremental evaluation in this study was concluded by evaluating the performance of LPX in conjunction with LPB algorithm. For the assessment, the performance of LPX was compared with that of SBX, BX, and Qubit-X, where all these algorithms were employed. The majority of test functions showed reasonable convergence during the exploitation evaluation, considering the chosen random values. LPX for random value (0.2) achieved a Mean value of 0.0048 with a STD of 0.0031 in TF7. Also, the corresponding execution time was 143.005 seconds. To compare, the mean and standard deviation for the other algorithm were Mean and STD, with an execution time of. LPX is better comparable based metrics then other crossover standards. The statistical findings for LPX, when compared to the other standards, provided substantial evidence, confirming that LPX achieved the optimal balance between effectiveness and exploitation. However, it should be noted that while LPX showed promising results, a comprehensive evaluation of the proposed standard with a few other population-based algorithms would be necessary to fully establish its superiority.

The second part of the research centred on the primary objective of LEO, which aimed to achieve precise immunizations by leveraging the albumin

quotient of human blood. This novel algorithm was conceptualized by incorporating the transfer of genetic chromosomes and drew inspiration from genetic algorithms. Moreover, the study delved into explaining the two fundamental stages of metaheuristic algorithms: exploitation and exploration. Both of these factors were instrumental in influencing the effectiveness of LEO. The exploration phase was meticulously crafted to emulate the immune system's improvement through effective vaccinations. During the process, these parameters efficiently divide the population into multiple groups. The Leo algorithm comprises crossover and mutation parameters that function separately from the exploitation of individuals. The process established a population distribution that depended on the level of success. Leo demonstrated auto-adaptivity by integrating these crossover and mutation techniques, particularly through the implementation of Lagrange orientation and Lagrange multiplier stationary point navigation. These mechanisms allowed Leo to dynamically adjust and optimize its performance as it progressed through the optimization process.

Leo's initialization, exploration, and exploitation stages all employ a randomization approach. To evaluate Leo's performance, the study employed 19 classical single-objective benchmark testing functions, which were categorized into three subgroups: unimodal, multimodal, and composite test functions. Leo was separately compared to other algorithms in each subgroup, where the first subgroup included DA, PSO, and GA. Remarkably, Leo's efficiency and performance were consistently close to those of the comparison algorithms in each subgroup of test functions. In the comparison with FDO and LPB, Leo emerged as the superior choice in most cases, except for the composite benchmark functions sub-group. Dependently, Leo demonstrated commendable performance and effectiveness across various types of benchmark functions.

Moreover, Leo was tested on 10 current CEC-C06 benchmarks and consistently outperformed its competitors in the majority of scenarios when compared to two well-known algorithms (PSO and GA), three modern algorithms (DA, WOA, and SSA), and three recent algorithms (FDO, LPB, and FOX). However, it should be noted that the results did not perfectly align with all recent algorithms, as well as the other algorithms utilizing Leo approach. This suggests that while Leo showed significant promise and superiority in many cases, further evaluations and refinements may be needed to achieve consistent success across all scenarios and with all algorithms. The Wilcoxon rank-sum test was used to determine the statistical significance of the results. Additionally, Leo was put to practical use in two newly proposed real-life applications to assess and validate its performance and suitability in tackling real-world scenarios.

In sum, during the evaluation of various test functions and real-world applications, it was noticed that the algorithm's performance is largely affected by the number of search agents used. The algorithm relies heavily on the Lagrange stationary point during gene crossover, which is a vital part of its search mechanism. Due to this characteristic, the algorithm is appropriately named Lagrange Elementary for Optimization. It was observed that using a small number of search agents (below seven, as the median for all benchmarks) significantly decreases the algorithm's accuracy. Conversely, increasing the number of search agents improves the accuracy of the algorithm but comes with higher gene costs and more frequent offspring updates. The study results indicate that the proposed approach surpasses the performance of most algorithms in the field. Nevertheless, it is important to acknowledge that Leo faces challenges when dealing with some problem in inspired optimization and may not identify the best optimal solution for all

specific problems. Despite this limitation, Leo has remained focused, aligning with achieved results and optimizing methods across various domains.

5.2. Recommendations for Future Works

Future studies should pursue several of the following research trajectories:

- The first suggestion includes an assessment of LPX's performance through comparisons with established crossover techniques, covering binary, real-coded, and order-coded problem methods. Expanding the evaluation, LPX will undertake testing on a diverse set of test functions, which will include two-dimensional functions among others. The effectiveness of LPX will be demonstrated through functional tests involving multimodal and composite test functions. Furthermore, researchers have improved a novel evolutionary metaheuristic algorithm, designed to address both single-objective and multi-objective optimization scenarios by operating on populations.
- In the future, Leo's major areas of focus will be twofold. First, the researchers aim to modify, implement, and test Leo for multi-objective and binary objective optimization tasks, expanding its capabilities to handle a broader range of optimization jobs. Secondly, researchers will explore incorporating evolutionary operators into Leo to enhance its performance and search abilities. Additionally, they will investigate combining Leo with other algorithms to create more powerful optimization approaches that leverage various techniques' strengths. This hybridization could lead to improved optimization solutions and increased applicability across different problem domains. These explorations hold great potential to enhance and advance the capabilities of Leo as an optimization method, leading to significant improvements in its performance and applicability.

- A novel Lagrange mutation standard will be introduced to replace the current Leo parameters, aiming to enhance the optimization process significantly. This innovative approach will enable the application of the proposed technique to diverse problem domains, facilitating result comparisons with other heuristic techniques. Moreover, there is potential for identifying and incorporating new parameters into the genetic algorithm, leading to further improvements in the optimization performance.
- The last one; however, it is challenging. Embarking on the development of a novel technique anchored in reinforcement learning or deep learning to identify non-dominated solutions represents a captivating and demanding trajectory in the realm of future research. This avenue also possesses the capability to introduce an innovative classification methodology. The advancements and refinements within this proposed approach hold substantial promise in broadening the scope of the optimization domain, thereby opening up new avenues for adeptly and efficiently tackling intricate problems.

5.3. Limitations

Every research endeavor encounters specific and general limitations, both during the identification of problem gaps and the subsequent efforts to address these gaps, particularly in the realm of solving independent optimization problems. Consequently, this study inevitably faced certain limitations, which are outlined below:

- The primary limitation of stochastic methods lies in their inherent challenge to deliver high accuracy, often falling short of precision, although typically maintaining proximity to the actual solution.

- A notable drawback is the requirement for extensive tuning, as these methods may not perform optimally without meticulous parameter adjustments.
- Additionally, the lack of assured convergence poses a significant concern, making it unpredictable and, at times, necessitating careful consideration and monitoring during the optimization process.

References

Abbas, A.K., 2020. Basic immunology : functions and disorders of the immune system, Sixth edition. ed. Elsevier, Philadelphia, PA.

Abdullah, J.M., Ahmed, T., 2019. Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process. IEEE Access 7, 43473–43486. https://doi.org/10.1109/ACCESS.2019.2907012

Abido, M.A., 2006. Multiobjective evolutionary algorithms for electric power dispatch problem. IEEE transactions on evolutionary computation 10, 315–329.

Ackora-Prah, J., Gyamerah, S.A., Andam, P.S., 2014. A heuristic crossover for portfolio selection.

Ahmed, Z.H., 2010. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. Int J Biom Bioinformatics 3, 96.

Andersson, M., Alvarez-Cermeñio, J., Bernardi, G., Cogato, I., Fredman, P., Frederiksen, J., Fredrikson, S., Gallo, P., Grimaldi, L.M., Grønning, M., Keir, G., Lamers, K., Link, H., Magalhães, A., Massaro, A.R., Öhman, S., Reiber, H., Rönnbäck, L., Schluep, M., Schuller, E., Sindic, C.J.M., Thompson, E.J., Trojano, M., Wurster, U., 1994. Cerebrospinal fluid in the diagnosis of multiple sclerosis: A consensus report. J Neurol Neurosurg Psychiatry 57, 897–902. https://doi.org/10.1136/jnnp.57.8.897

Arcuri, A., Briand, L., 2011. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering, in: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11. Association for Computing Machinery, New York, NY, USA, pp. 1–10. https://doi.org/10.1145/1985793.1985795

Arora, K., Kumar, A., Kamboj, V.K., Prashar, D., Jha, S., Shrestha, B., Joshi, G.P., 2020. Optimization Methodologies and Testing on Standard Benchmark Functions of Load Frequency Control for Interconnected Multi Area Power System in Smart Grids. Mathematics 8, 980. https://doi.org/10.3390/math8060980

Bacanin, N., Zivkovic, M., Al-Turjman, F., Venkatachalam, K., Trojovsky, P., Strumberger, I., Bezdan, T., 2022. Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application. Sci Rep 12, 6302. https://doi.org/10.1038/s41598-022-09744-2

Bäck, T., Fogel, D.B., Michalewicz, Z., 2018. Evolutionary computation 1: Basic algorithms and operators. CRC press.

Bao, T.N., Huynh, Q.-T., Nguyen, X.-T., Nguyen, G.N., Le, D.-N., 2020. A novel particle swarm optimization approach to support decision-making in the multi-round of an auction by game theory. International Journal of Computational Intelligence Systems 13, 1447–1463.

Belciug, S., Gorunescu, F., 2016. A hybrid genetic algorithm-queuing multi-compartment model for optimizing inpatient bed occupancy and associated costs. Artif Intell Med 68, 59–69.

Bell, O., 2022. Applications of Gaussian Mutation for Self Adaptation in Evolutionary Genetic Algorithms.

Beloglazov, A., Buyya, R., 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput 24, 1397–1420.

Beyer, H.-G., Deb, K., 2001. On self-adaptive features in real-parameter evolutionary algorithms. IEEE Transactions on evolutionary computation 5, 250–270.

Bordbar, B., Giacomini, L., Holding, D.J., 2000. UML and Petri Nets for design and analysis of distributed systems. https://doi.org/10.1109/CCA.2000.897497

Bosch, W., 2007. Discrete crossover analysis, in: Dynamic Planet: Monitoring and Understanding a Dynamic Planet with Geodetic and Oceanographic Tools IAG Symposium Cairns, Australia 22–26 August, 2005. Springer, pp. 131–136.

Boussaïd, I., Lepagnot, J., Siarry, P., 2013a. A survey on optimization metaheuristics. Inf Sci (N Y) 237, 82–117.

Boussaïd, I., Lepagnot, J., Siarry, P., 2013b. A survey on optimization metaheuristics. Inf Sci (N Y) 237, 82–117.

Brest, J., Maučec, M.S., Bošković, B., 2019. The 100-Digit Challenge: Algorithm jDE100, in: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 19–26. https://doi.org/10.1109/CEC.2019.8789904

Carlos, P., Azevedo, R.B., 2011. "Geração de Diversidade na Otimização Dinâmica Multiobjetivo Evolucionária por Paisagens de Não-Dominância."

Chaparro, B.M., Thuillier, S., Menezes, L.F., Manach, P.-Y., Fernandes, J. V, 2008. Material parameters identification: Gradient-based, genetic and hybrid optimization algorithms. Comput Mater Sci 44, 339–346.

Chu, S.-C., Tsai, P.-W., Pan, J.-S., 2006. Cat swarm optimization, in: PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence Guilin, China, August 7-11, 2006 Proceedings 9. Springer, pp. 854–858.

Civicioglu, P., 2012. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput Geosci 46, 229–247. https://doi.org/https://doi.org/10.1016/j.cageo.2011.12.011

Cook, S.A., Mitchell, D.G., 1997. Finding hard instances of the satisfiability problem: A survey. Satisfiability Problem: Theory and Applications 35, 1–17.

Copeland, B.J., 2000. The modern history of computing.

Deb, K., Beyer, H.-G., 2001. Self-adaptive genetic algorithms with simulated binary crossover. Evol Comput 9, 197–221.

Deep, K., Thakur, M., 2007. A new crossover operator for real coded genetic algorithms. Appl Math Comput 188, 895–911.

Desjardins, B., Falcon, R., Abielmona, R., Petriu, E., 2017. Planning robust sensor relocation trajectories for a mobile robot with evolutionary multi-objective optimization. Computational Intelligence in Wireless Sensor Networks: Recent Advances and Future Challenges 179–210.

Dey, N., 2017. Advancements in applied metaheuristic computing. IGI global.

Dey, S., Bhattacharyya, S., Maulik, U., 2017. Efficient quantum inspired meta-heuristics for multi-level true colour image thresholding. Appl Soft Comput 56, 472–513.

Dey, S., Saha, I., Bhattacharyya, S., Maulik, U., 2014. Multi-level thresholding using quantum inspired meta-heuristics. Knowl Based Syst 67, 373–400.

Dhal, K.G., Ray, S., Das, A., Das, S., 2019. A Survey on Nature-Inspired Optimization Algorithms and Their Application in Image Enhancement Domain. Archives of Computational Methods in Engineering 26, 1607–1638. https://doi.org/10.1007/s11831-018-9289-9

Eli Benjamini, Geoffrey Sunshine, Richard Coico, 2000. Immunology : A Short Course, 4th edition. ed. Wiley-Liss, New York.

Ewens, W.J., Lessard, S., 2015. On the interpretation and relevance of the fundamental theorem of natural selection. Theor Popul Biol 104, 59–67.

Fang, E., Liu, X., Li, M., Zhang, Z., Song, L., Zhu, B., Wu, X., Liu, J., Zhao, D., Li, Y., 2022. Advances in COVID-19 mRNA vaccine development. Signal Transduct Target Ther 7, 94. https://doi.org/10.1038/s41392-022-00950-y

Fay, M.P., Proschan, M.A., 2010. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. Stat Surv 4, 1.
Fister Jr, I., Yang, X.-S., Fister, I., Brest, J., Fister, D., 2013. A brief review of nature-inspired algorithms for optimization. arXiv preprint arXiv:1307.4186.
Fogel, D.B., 1994. An introduction to simulated evolutionary optimization. IEEE Trans Neural Netw 5, 3–14. https://doi.org/10.1109/72.265956
Gain, A., Dey, P., 2020. Adaptive Position–Based Crossover in the Genetic Algorithm for Data Clustering. Recent Advances in Hybrid Metaheuristics for Data Clustering 39–59.
García-Martínez, C., Lozano, M., Herrera, F., Molina, D., Sánchez, A.M., 2008. Global and local real-coded genetic algorithms based on parent-centric crossover operators. Eur J Oper Res 185, 1088–1113.
Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001. A new heuristic optimization algorithm: harmony search. Simulation 76, 60–68.
Gill, P.E., Murray, W., Saunders, M.A., Tomlin, J.A., Wright, M.H., 2008. George B. Dantzig and systems optimization. Discrete Optimization 5, 151–158. https://doi.org/https://doi.org/10.1016/j.disopt.2007.01.002
Gopalakrishna, A.K., Ozcelebi, T., Lukkien, J.J., Liotta, A., 2019. Runtime evaluation of cognitive systems for non-deterministic multiple output classification problems. Future Generation Computer Systems 100, 1005–1016.
Gutierrez, J.C.T., Adamatti, D.S., Bravo, J.M., 2019. A new stopping criterion for multi-objective evolutionary algorithms: application in the calibration of a hydrologic model. Comput Geosci 23, 1219–1235.
Haldurai, L., Madhubala, T., Rajalakshmi, R., 2016. A study on genetic algorithm and its applications. Int. J. Comput. Sci. Eng 4, 139–143.
Hamid, Z.A., Musirin, I., Othman, M.M., Rahim, N.A., 2011. Efficient power scheduling via stability index based tracing technique and blended crossover continuous ant colony optimization. Aust J Basic Appl Sci 5, 1335–1347.
Hassanat, A.B.A., Alkafaween, E., 2017. On enhancing genetic algorithms using new crossovers. International Journal of Computer Applications in Technology 55, 202–212.
Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future generation computer systems 97, 849–872.
Henderson, D., Jacobson, S.H., Johnson, A.W., 2003. The theory and practice of simulated annealing. Handbook of metaheuristics 287–319.
Herrera, F., Lozano, M., Sánchez, A.M., 2005. Hybrid crossover operators for real-coded genetic algorithms: an experimental study. Soft comput 9, 280–298.
Herrera, F., Lozano, M., Verdegay, J.L., 1997. Fuzzy connectives based crossover operators to model genetic algorithms population diversity. Fuzzy Sets Syst 92, 21–30.
Hilding, F.G., Ward, K., 2005. Automated operator selection on genetic algorithms, in: Knowledge-Based Intelligent Information and Engineering Systems: 9th International Conference, KES 2005, Melbourne, Australia, September 14-16, 2005, Proceedings, Part IV 9. Springer, pp. 903–909.
Hoos, H.H., Stützle, T., 2004. Stochastic local search: Foundations and applications. Elsevier.
Hussain, A., Muhammad, Y.S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., Gani, S., 2017. Genetic algorithm for traveling salesman problem with modified cycle crossover operator. Comput Intell Neurosci 2017.

Hussain, K., Najib, M., Salleh, M., Cheng, S., Naseem, R., n.d. Common Benchmark Functions for Metaheuristic Evaluation: A Review.

Iqbal, M., Azam, M., Naeem, M., Khwaja, A.S., Anpalagan, A., 2014. Optimization classification, algorithms and tools for renewable energy: A review. Renewable and sustainable energy reviews 39, 640–654.

Ito, K., Kunisch, K., 2008. Lagrange multiplier approach to variational problems and applications. SIAM.

Jalali, M.R., Afshar, A., MARINO, M., 2005. Ant Colony Optimization Algorithm (ACO); A new heuristic approach for engineering optimization 2.

Jeong, Y.-S., Shin, K.S., Jeong, M.K., 2015. An evolutionary algorithm with the partial sequential forward floating search mutation for large-scale feature selection problems. Journal of the Operational Research Society 66, 529–538. https://doi.org/10.1057/jors.2013.72

Jia, Q., 2007. Hyperchaos generated from the Lorenz chaotic system and its control. Phys Lett A 366, 217–222. https://doi.org/https://doi.org/10.1016/j.physleta.2007.02.024

Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N., 2014. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42, 21–57.

Katayama, K., Sakamoto, H., Narihisa, H., 2000. The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem. Math Comput Model 31, 197–203.

Kaya, Y., Uyar, M., 2011. A novel crossover operator for genetic algorithms: ring crossover. arXiv preprint arXiv:1105.0355.

Kennedy, J., 2006. Swarm Intelligence, in: Zomaya, A.Y. (Ed.), Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies. Springer US, Boston, MA, pp. 187–219. https://doi.org/10.1007/0-387-27705-6_6

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks. pp. 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kita, H., Ono, I., Kobayashi, S., 1999. Theoretical analysis of the unimodal normal distribution crossover for real-coded genetic algorithms. Transactions of the Society of Instrument and Control Engineers 35, 1333–1339.

Kleinberg, B., Li, Y., Yuan, Y., 2018. An Alternative View: When Does SGD Escape Local Minima?, in: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, pp. 2698–2707.

Kora, P., Yadlapalli, P., 2017. Crossover operators in genetic algorithms: A review. Int J Comput Appl 162.

Kuo, H.C., Lin, C.H., 2013. Cultural evolution algorithm for global optimizations and its applications. Journal of applied research and technology 11, 510–522.

Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S., 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artif Intell Rev 13, 129–170.

LEFVERT, A.K., LINK, H., 1985. Igg Production Within the Central Nervous System – a Critical Review of Proposed Formulae, in: PEETERS, H. (Ed.), Protides of the Biological Fluids. Elsevier, pp. 199–202. https://doi.org/https://doi.org/10.1016/B978-0-08-031739-7.50051-3

Li, S., Chen, H., Wang, M., Heidari, A.A., Mirjalili, S., 2020. Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems 111, 300–323.
Liang, Y.-C., Juarez, J., 2016. A novel metaheuristic for continuous optimization problems: Virus optimization algorithm. Engineering Optimization 48, 73–93. https://doi.org/10.1080/0305215X.2014.994868
Lin, Z., Chen, M., Ma, Y., 2010. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. arXiv preprint arXiv:1009.5055.
Ling, S.-H., Leung, F.H.F., 2007. An improved genetic algorithm with average-bound crossover and wavelet mutation operations. Soft comput 11, 7–31.
Link, H., Huang, Y.-M., 2006. Oligoclonal bands in multiple sclerosis cerebrospinal fluid: An update on methodology and clinical usefulness. J Neuroimmunol 180, 17–28. https://doi.org/https://doi.org/10.1016/j.jneuroim.2006.07.006
Lundstrom, K., 2020. The Current Status of COVID-19 Vaccines. Front Genome Ed 2. https://doi.org/10.3389/fgeed.2020.579297
Lyakhov, A.O., Oganov, A.R., Stokes, H.T., Zhu, Q., 2013. New developments in evolutionary structure prediction algorithm USPEX. Comput Phys Commun 184, 1172–1182.
Mahmudov, E.N., 2011. Approximation and optimization of discrete and differential inclusions. Elsevier.
Malik, S., Wadhwa, S., 2014. Preventing premature convergence in genetic algorithm using DGCA and elitist technique. International Journal of Advanced Research in Computer Science and Software Engineering 4.
Matrajt, L., Eaton, J., Leung, T., Brown, E.R., 2021. Vaccine optimization for COVID-19: Who to vaccinate first? Sci Adv 7, eabf1374.
McGinley, B., Maher, J., O'Riordan, C., Morgan, F., 2011. Maintaining healthy population diversity using adaptive crossover, mutation, and selection. IEEE Transactions on Evolutionary Computation 15, 692–714.
Melanie, M., 1999. 0−262−63185−7 (PB) 1. Genetics-Computer simulation.2. Genetics-Mathematical models.
Meraihi, Y., Ramdane-Cherif, A., Acheli, D., Mahseur, M., 2020. Dragonfly algorithm: a comprehensive review and applications. Neural Comput Appl 32, 16625–16646. https://doi.org/10.1007/s00521-020-04866-y
Mirjalili, S., 2016. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27, 1053–1073. https://doi.org/10.1007/s00521-015-1920-1
Mirjalili, S., 2015. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl Based Syst 89, 228–249.
Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M., 2017. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in engineering software 114, 163–191.
Mirjalili, S., Lewis, A., 2016. The Whale Optimization Algorithm. Advances in Engineering Software 95, 51–67. https://doi.org/https://doi.org/10.1016/j.advengsoft.2016.01.008
Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Advances in engineering software 69, 46–61.
Mohammed, H., Rashid, T., 2023. FOX: a FOX-inspired optimization algorithm. Applied Intelligence 53, 1030–1050. https://doi.org/10.1007/s10489-022-03533-0
Naidu, D.S., 2002. Optimal control systems. CRC press.

Nakrani, S., Tovey, C., 2004. On honey bees and dynamic server allocation in internet hosting centers. Adaptive behavior 12, 223–240.
Ndeffo Mbah, M.L., Liu, J., Bauch, C.T., Tekel, Y.I., Medlock, J., Meyers, L.A., Galvani, A.P., 2012. The Impact of Imitation on Vaccination Behavior in Social Contact Networks. PLoS Comput Biol 8, e1002469-.
Nicholl, D.S.T., 2023. An introduction to genetic engineering. Cambridge University Press.
Nuwarda, R.F., Ramzan, I., Weekes, L., Kayser, V., 2022. Vaccine Hesitancy: Contemporary Issues and Historical Background. Vaccines (Basel) 10, 1595. https://doi.org/10.3390/vaccines10101595
Ono, I., 1997. A Real-coded Gemetic Algorithm for Function Optimization Using Unimodal Normal Disribution Crossover, in: Proc. 7th Int. Conf. on Genetic Algorithms. pp. 246–253.
Osaba, E., Diaz, F., Onieva, E., Carballedo, R., Perallos, A., 2014. AMCPA: A population metaheuristic with adaptive crossover probability and multi-crossover mechanism for solving combinatorial optimization problems. International Journal of Artificial Intelligence 12, 1–23.
Osuna-Enciso, V., Cuevas, E., Castañeda, B.M., 2022. A diversity metric for population-based metaheuristic algorithms. Inf Sci (N Y) 586, 192–208.
Ouattara, A., Aswani, A., 2018. Duality approach to bilevel programs with a convex lower level, in: 2018 Annual American Control Conference (ACC). IEEE, pp. 1388–1395.
Patel, R., Collins, D., Bullock, S., Swaminathan, R., Blake, G.M., Fogelman, I., 2001. The effect of season and vitamin D supplementation on bone mineral density in healthy women: a double-masked crossover study. Osteoporosis International 12, 319–325.
Pongcharoen, P., Stewardson, D.J., Hicks, C., Braiden, P.M., 2001. Applying designed experiments to optimize the performance of genetic algorithms used for scheduling complex products in the capital goods industry. J Appl Stat 28, 441–455.
Puljić, K., Manger, R., 2013. Comparison of eight evolutionary crossover operators for the vehicle routing problem. Mathematical Communications 18, 359–375.
Qin, A.K., Suganthan, P.N., 2005. Self-adaptive differential evolution algorithm for numerical optimization, in: 2005 IEEE Congress on Evolutionary Computation. pp. 1785-1791 Vol. 2. https://doi.org/10.1109/CEC.2005.1554904
Rahman, C.M., Rashid, T.A., 2021. A new evolutionary algorithm: Learner performance based behavior algorithm. Egyptian Informatics Journal 22, 213–223. https://doi.org/10.1016/j.eij.2020.08.003
Reiber, H., 2003. Proteins in cerebrospinal fluid and blood: Barriers, CSF flow rate and source-related dynamics. Restor Neurol Neurosci 21, 79–96.
Saravanan, K.A., Panigrahi, M., Kumar, H., Rajawat, D., Nayak, S.S., Bhushan, B., Dutt, T., 2022. Role of genomics in combating COVID-19 pandemic. Gene 823, 146387. https://doi.org/https://doi.org/10.1016/j.gene.2022.146387
Singh, J., Kumar, D., Hammouch, Z., Atangana, A., 2018. A fractional epidemiological model for computer viruses pertaining to a new fractional derivative. Appl Math Comput 316, 504–515. https://doi.org/https://doi.org/10.1016/j.amc.2017.08.048
Sivanandam, S.N., Deepa, S.N., Sivanandam, S.N., Deepa, S.N., 2008. Genetic algorithms. Springer.
Smith, J.E., 2008. Self-Adaptation in Evolutionary Algorithms for Combinatorial Optimisation, in: Cotta, C., Sevaux, M., Sörensen, K. (Eds.), Adaptive and Multilevel Metaheuristics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 31–57. https://doi.org/10.1007/978-3-540-79438-7_2

Su, C., Chiu, H., 1986. [Measurements of IgG and albumin in CSF and serum in various neurological diseases]. Zhonghua Min Guo Wei Sheng Wu Ji Mian Yi Xue Za Zhi 19, 250–257.

Takahashi, M., Kita, H., 2001. A crossover operator using independent component analysis for real-coded genetic algorithms, in: Proceedings of the 2001 Congress on Evolutionary Computation (Ieee Cat. No. 01th8546). IEEE, pp. 643–649.

Tawhid, M.A., Ali, A.F., 2016. Simplex particle swarm optimization with arithmetical crossover for solving global optimization problems. Opsearch 53, 705–740.

Thapatsuwan, P., Chainate, W., Pongcharoen, P., 2006. Investigation of genetic algorithm parameters and comparison of heuristic arrangements for container packing problem. Curr Appl Sci Technol 6, 274–284.

Tran, N.-H., Park, H.-S., Nguyen, Q.-V., Hoang, T.-D., 2019. Development of a Smart Cyber-Physical Manufacturing System in the Industry 4.0 Context. Applied Sciences 9, 3325. https://doi.org/10.3390/app9163325

Tu, J., Chen, H., Wang, M., Gandomi, A.H., 2021. The colony predation algorithm. J Bionic Eng 18, 674–710.

Tuan, H.D., Apkarian, P., Nakashima, Y., 2000. A new Lagrangian dual global optimization algorithm for solving bilinear matrix inequalities. International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal 10, 561–578.

Umbarkar, A.J., Sheth, P.D., 2015. Crossover operators in genetic algorithms: a review. ICTACT journal on soft computing 6.

Vandvik, B., Norrby, E., 1973. Oligoclonal IgG Antibody Response in the Central Nervous System to Different Measles Virus Antigens in Subacute Sclerosing Panencephalitis. Proceedings of the National Academy of Sciences 70, 1060–1063. https://doi.org/10.1073/pnas.70.4.1060

Wang, G.-G., 2018. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memet Comput 10, 151–164.

Wong, K.P., Dong, Z.Y., 2005. Differential evolution, an alternative approach to evolutionary algorithm, in: Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems. IEEE, pp. 73–83.

Xu, Y., Liu, Xin, Cao, X., Huang, C., Liu, E., Qian, S., Liu, Xingchen, Wu, Y., Dong, F., Qiu, C.-W., Qiu, J., Hua, K., Su, W., Wu, J., Xu, H., Han, Y., Fu, C., Yin, Z., Liu, M., Roepman, R., Dietmann, S., Virta, M., Kengara, F., Zhang, Z., Zhang, Lifu, Zhao, T., Dai, J., Yang, J., Lan, L., Luo, M., Liu, Z., An, T., Zhang, B., He, X., Cong, S., Liu, Xiaohong, Zhang, W., Lewis, J.P., Tiedje, J.M., Wang, Q., An, Z., Wang, Fei, Zhang, Libo, Huang, T., Lu, C., Cai, Z., Wang, Fang, Zhang, J., 2021. Artificial intelligence: A powerful paradigm for scientific research. The Innovation 2, 100179. https://doi.org/https://doi.org/10.1016/j.xinn.2021.100179

Yang, X.-S., 2018. Social algorithms. arXiv preprint arXiv:1805.05855.

Yang, X.-S., 2010. A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010) 65–74.

Yang, X.-S., Deb, S., 2009. Cuckoo Search via Lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). pp. 210–214. https://doi.org/10.1109/NABIC.2009.5393690

Yang, Y., Chen, H., Heidari, A.A., Gandomi, A.H., 2021. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. Expert Syst Appl 177, 114864.

Yu, Z., Ouyang, J., Li, S., Peng, X., 2017. Formal modeling and control of cyber-physical manufacturing systems. Advances in Mechanical Engineering 9. https://doi.org/10.1177/1687814017725472

Zamani, H., Nadimi-Shahraki, M.H., Gandomi, A.H., 2021. QANA: Quantum-based avian navigation optimizer algorithm. Eng Appl Artif Intell 104, 104314.
Zhang, Xin, Zhang, Q., Zhang, Xiu, 2017. Nonuniform antenna array design by parallelizing three-parent crossover genetic algorithm. EURASIP J Wirel Commun Netw 2017, 1–7.
Zhou, W., Huang, C., Xiao, M., Cao, J., 2018. Hybrid tactics for bifurcation control in a fractional-order delayed predator–prey model. Physica A: Statistical Mechanics and its Applications 515. https://doi.org/10.1016/j.physa.2018.09.185
Zhou, Y., Hao, J.-K., Duval, B., 2020. Frequent pattern-based search: A case study on the quadratic assignment problem. IEEE Trans Syst Man Cybern Syst 52, 1503–1515.

6. APPENDIX

Table 6-1Thirty turns result of the Leo Algorithm for solving the classical benchmark TF1 to TF5

Turns	TF1	TF2	TF3	TF4	TF5
T1	5.7150E-11	5.6762E-07	1.2188E-09	1.0351E-04	7.3772E+00
T2	9.7092E-09	1.4933E-06	1.8392E-09	3.3539E-05	8.5613E+00
T3	1.5077E-11	8.2411E-06	3.9375E-10	1.4748E-05	8.1545E+00
T4	1.4520E-08	2.0330E-05	6.6221E-09	8.9398E-05	1.0881E+01
T5	2.4333E-09	4.5991E-06	3.1757E-09	3.6133E-05	1.1694E+01
T6	3.1336E-09	7.2257E-07	3.5319E-10	3.3947E-05	1.1959E+01
T7	3.8117E-08	2.8734E-06	5.6831E-10	1.3090E-05	7.4091E+00
T8	1.0295E-09	2.9747E-07	5.6088E-09	1.9830E-06	7.5175E+00
T9	2.0270E-09	9.1651E-06	1.1493E-07	3.7754E-05	5.8598E+00
T10	4.2913E-11	4.0526E-06	2.3359E-10	1.5411E-05	7.7933E+00
T11	1.1978E-09	1.2771E-06	1.5672E-10	3.7263E-05	4.8039E+00
T12	8.7826E-13	1.8492E-06	4.2791E-10	2.9099E-05	1.1516E+01
T13	1.8536E-09	3.3741E-06	1.8434E-09	3.5766E-05	8.4861E+00
T14	1.2309E-10	3.9459E-06	1.6274E-09	2.4712E-06	9.5808E+00
T15	8.7594E-11	4.5005E-07	5.0069E-09	1.0353E-05	7.6523E+00
T16	1.1294E-10	4.4229E-06	4.6875E-10	4.0288E-05	3.7435E+00
T17	2.3213E-10	2.9973E-06	4.4602E-09	4.7540E-05	8.0218E+00
T18	8.4904E-13	1.8652E-06	1.9485E-11	2.7245E-06	8.2870E+01
T19	5.7187E-10	6.6887E-06	3.4550E-10	8.7197E-05	5.0199E+00
T20	1.4445E-10	3.7356E-06	1.8066E-09	1.1329E-04	7.6728E+00
T21	1.9079E-12	2.4884E-07	4.8771E-11	9.7449E-06	6.5145E+00
T22	1.4302E-10	7.2899E-06	9.7532E-12	3.0302E-05	7.6977E+00
T23	2.8635E-10	4.1894E-06	3.3301E-10	1.0475E-05	3.4204E+00
T24	5.6676E-11	1.8531E-06	1.2847E-10	1.7099E-06	5.6097E+00
T25	7.1417E-10	3.1320E-06	6.3531E-13	8.5247E-06	1.0548E+01
T26	7.7583E-10	3.5953E-06	4.7430E-09	2.2244E-05	1.7122E+01
T27	2.7639e-11	4.8630E-08	6.9433E-12	2.7588E-05	9.1110E+00
T28	8.2552E-12	5.3771E-06	7.0212E-12	9.4228E-05	8.8853E+00
T29	8.5767E-10	2.8846E-06	1.5666E-09	1.8485E-05	7.8466E+00
T30	9.5326E-12	3.4788E-07	1.4899E-09	7.2053E-05	4.7603E+00
STD	**7.49992E-09**	**3.956E-06**	**2.079E-08**	**3.228E-05**	**13.932859**
Average	**2.6987E-09**	**3.7305E-06**	**5.3147E-09**	**3.6029E-05**	**1.0603E+01**

Table 6-2 Thirty turns result of the Leo Algorithm for solving the classical benchmark TF6 to TF10

Turns	TF6	TF7	TF8	TF9	TF10
T1	7.0149E-11	2.5099E-03	-2.9856E+03	5.6712E+01	3.4618E-05
T2	7.4016E-10	3.8098E-04	-2.5117E+03	2.5869E+01	8.6764E-05
T3	6.8509E-10	3.0393E-03	-2.7092E+03	4.7758E+01	6.6370E-05
T4	7.8929E-12	3.6169E-04	-2.8671E+03	4.5768E+01	8.1759E-05
T5	4.3749E-11	1.0235E-03	-2.5301E+03	4.2783E+01	7.0249E-05
T6	1.5629E-11	6.6818E-03	-3.2803E+03	4.9748E+01	9.9201E-05
T7	2.4195E-12	8.0361E-04	-3.0251E+03	3.0844E+01	5.5008E-05
T8	8.4324E-12	8.2193E-05	-2.7289E+03	4.5768E+01	3.4954E-06
T9	1.6519E-12	5.6315E-04	-3.2408E+03	4.7758E+01	7.9909E-05
T10	5.9208E-10	7.6974E-05	-3.2422E+03	4.2783E+01	3.4072E-05
T11	8.5238E-11	4.6642E-05	-2.9856E+03	5.1738E+01	6.4573E-06
T12	9.5523E-10	1.1847E-04	-3.1633E+03	1.1940E+01	7.4044E-05
T13	9.0493E-12	2.0231E-05	-2.9460E+03	3.6813E+01	5.1958E-05
T14	4.7900E-10	1.1363E-03	-3.0448E+03	1.7909E+01	6.2033E-06
T15	2.4611E-09	1.9565E-04	-3.0843E+03	3.5818E+01	2.7682E-05
T16	3.2176E-11	2.4935E-04	-3.0236E+03	3.7808E+01	5.7349E-05
T17	6.4177E-10	8.8483E-05	-2.8277E+03	4.6763E+01	7.2002E-05
T18	3.3274E-12	5.6389E-04	-3.0843E+03	2.6864E+01	8.4044E-05
T19	1.5748E-09	3.0174E-04	-3.0236E+03	3.0844E+01	7.8349E-05
T20	5.2737E-10	7.5564E-05	-3.0647E+03	6.4672E+01	5.2670E-05
T21	1.6897E-10	1.1689E-02	-2.9066E+03	3.1839E+01	5.6411E-06
T22	2.7896E-12	5.0307E-04	-3.1634E+03	2.3879E+01	6.4216E-06
T23	3.1894E-10	2.5549E-04	-2.8078E+03	2.7859E+01	2.7110E-05
T24	8.7873E-10	8.3372E-04	-2.9856E+03	3.2834E+01	3.0071E-05
T25	6.8917E-12	9.1768E-04	-2.8474E+03	2.1889E+01	3.5876E-05
T26	5.4885E-10	5.9488E-04	-2.9659E+03	5.1738E+01	1.5349E-05
T27	4.4977E-11	3.5254E-04	-2.9066E+03	3.3829E+01	8.8663E-05
T28	8.9078E-10	8.2515E-03	-3.2013E+03	3.7808E+01	4.8216E-05
T29	5.2499E-10	1.0392E-03	-3.2818E+03	3.0844E+01	5.2336E-05
T30	6.2519E-10	7.3514E-04	-3.2393E+03	2.2884E+01	3.3192E-05
STD	**5.51803E-10**	**0.002690575**	**202.684514**	**12.2775166**	**2.89869E-05**
Average	**4.3158E-10**	**1.4497E-03**	**-2.9891E+03**	**3.7079E+01**	**4.8836E-05**

Table 6-3 Thirty turns result of the Leo Algorithm for solving the classical benchmark TF11 to TF15

Turns	TF11	TF12	TF13	TF14	TF15
T1	5.2619E-09	8.8402E-10	9.1906E-08	4.9505E+00	1.2267E-03
T2	2.6104E-09	5.9989E-08	1.5827E-10	9.8039E+00	1.2274E-03
T3	3.6737E-09	4.0766E-08	5.8812E-10	9.9800E-01	9.4100E-04
T4	2.5042E-07	1.1926E-10	1.9637E-09	1.2671E+01	7.0377E-04
T5	1.4561E-07	4.1079E-08	9.4619e-11	9.9800E-01	1.2266E-03
T6	1.5060E-09	7.8765E-09	2.5188E-09	9.9800E-01	7.1367E-04
T7	8.7729E-08	2.5148E-11	4.0983E-09	1.0763E+01	7.0209E-04
T8	4.2372E-08	4.9800E-08	8.5378E-09	2.9821E+00	2.0363E-02
T9	4.2149E-11	3.2272E-08	5.8833E-12	5.9288E+00	1.2328E-03
T10	2.3184E-09	2.3220E-08	2.7538E-09	1.0763E+01	7.4800E-04
T11	2.8083E-10	1.9841E-09	1.2700E-08	9.9800E-01	1.2584E-03
T12	1.2006E-09	1.4427E-10	6.9274E-10	4.9505E+00	1.2267E-03
T13	4.6796E-10	1.5565E-08	2.9044E-08	5.9288E+00	5.7737E-04
T14	7.0717E-10	1.1461E-10	4.1487E-08	1.2671E+01	1.2232E-03
T15	6.2201E-10	8.0409E-08	6.0606E-10	4.9505E+00	1.2268E-03
T16	1.6439E-08	1.3261E-10	2.3379E-08	9.9800E-01	1.2260E-03
T17	3.7406E-09	2.4453E-08	6.2936E-11	3.9683E+00	1.2310E-03
T18	9.0209E-08	2.1404E-08	4.1740E-11	1.9920E+00	8.6937E-04
T19	2.9759E-10	1.2371E-08	8.0147E-09	1.0763E+01	1.2276E-03
T20	3.2030E-10	1.1869E-09	1.5614E-08	2.9821E+00	1.2324E-03
T21	7.5053E-09	1.2559E-07	2.5654E-10	2.1988E+01	9.8483E-04
T22	5.4198E-09	3.1577E-10	2.4013E-12	1.0763E+01	1.2341E-03
T23	8.1218E-08	3.7855E-10	2.3334E-09	5.9288E+00	1.4480E-03
T24	3.8170E-08	4.5392E-11	1.6102E-10	2.0154E+01	7.5459E-04
T25	4.5173E-10	8.4744E-09	2.9168E-10	1.3619E+01	7.4609E-04
T26	7.6004E-11	9.0137E-10	1.7740E-11	1.9920E+00	6.6188E-04
T27	6.3383E-11	1.0420E-09	2.8680E-09	1.5504E+01	6.8985E-04
T28	2.5184E-08	4.2723E-09	7.2103E-09	2.9821E+00	1.2537E-03
T29	2.4614E-13	9.3154E-10	8.5324E-10	9.9800E-01	1.2315E-03
T30	7.8738E-09	7.2539E-09	7.5359E-11	4.9505E+00	8.0438E-04
STD	**5.51514E-08**	**2.89749E-08**	**1.88063E-08**	**5.833242622**	**0.003539145**
Average	**2.7393E-08**	**1.8767E-08**	**8.9049E-09**	**6.9979E+00**	**1.6731E-03**

Table 6-4 Thirty turns result of the Leo Algorithm for solving the classical benchmark TF16 to TF19

Turns	TF16	TF17	TF18	TF19
T1	-2.1546E-01	3.9789E-01	3.0000E+00	-3.7807E+00
T2	-1.0316E+00	4.9398E-01	3.9871E+00	-2.9525E+00
T3	-2.1546E-01	4.9398E-01	5.0172E+00	-3.0832E+00
T4	-1.0316E+00	3.9789E-01	5.0172E+00	-8.3916E-01
T5	-1.0316E+00	3.9789E-01	3.0000E+00	-1.8881E+00
T6	-2.1546E-01	2.4153E+00	3.9871E+00	-1.3441E+00
T7	-2.1546E-01	4.9398E-01	3.0000E+00	-1.3781E+00
T8	-2.1546E-01	3.9789E-01	3.0000E+00	-1.4012E+00
T9	-1.0316E+00	2.7054E+00	3.0000E+00	-1.0008E+00
T10	-4.1618E-01	2.7054E+00	3.0000E+00	-1.0008E+00
T11	-1.0316E+00	5.8444E+00	3.0000E+00	-9.9928E-01
T12	-2.1546E-01	5.8444E+00	3.0000E+00	-1.0008E+00
T13	-1.0316E+00	2.7054E+00	3.9871E+00	-2.4215E+00
T14	-2.1546E-01	3.9789E-01	3.9871E+00	-3.8160E+00
T15	-2.1546E-01	3.9789E-01	3.9871E+00	-3.8628E+00
T16	-1.0316E+00	2.4153E+00	3.9871E+00	-3.8305E+00
T17	-2.1546E-01	3.9789E-01	3.9871E+00	-3.6584E+00
T18	-1.0316E+00	2.7054E+00	5.0172E+00	-1.0924E+00
T19	-2.1546E-01	3.9789E-01	5.0172E+00	-2.0298E+00
T20	-1.0316E+00	3.9789E-01	3.9871E+00	-2.0298E+00
T21	-2.1546E-01	4.9398E-01	3.0000E+00	-3.0014E+00
T22	-2.1546E-01	4.9398E-01	3.9871E+00	-3.5176E+00
T23	-4.1618E-01	3.9789E-01	3.0000E+00	-3.5707E+00
T24	-1.0316E+00	3.9789E-01	3.0000E+00	-3.6177E+00
T25	-1.0316E+00	7.7827E+00	3.0000E+00	-3.7767E+00
T26	-1.0000E+00	7.7827E+00	3.1532E+00	-3.8160E+00
T27	-7.6566E-01	3.9789E-01	3.1532E+00	-3.8425E+00
T28	-1.0316E+00	3.9789E-01	3.1532E+00	-3.8628E+00
T29	-1.5610E-01	2.7054E+00	3.1532E+00	-3.8624E+00
T30	-9.4417E-01	3.9789E-01	3.1532E+00	-3.8465E+00
STD	**0.39678297**	**2.2376316**	**0.7119171**	**1.18530797**
Average	**-6.2210E-01**	**1.7884E+00**	**3.5906E+00**	**-2.6708E+00**

Table 6-5 Thirty turns result of the Leo Algorithm for CECC06 2019 benchmark from CEC01 to CEC05

Turns	CEC01	CEC02	CEC03	CEC04	CEC05
T1	5815932303	17.378	12.7039	39.1798	2.7937
T2	5712129381	17.3807	12.7024	49.8577	2.6028
T3	6216690442	17.6698	12.7024	39.1013	3.4027
T4	1585174688	17.4542	12.7049	78.8027	2.4969
T5	9220344881	17.6344	12.7024	83.2009	3.64
T6	8418457844	17.5348	12.7039	52.0287	2.993
T7	1567417025	17.667	12.7024	53.1781	3.066
T8	1688209977	17.5919	12.7024	61.6233	3.0998
T9	27536483350	17.6247	12.7038	44.9834	2.6969
T10	6322536308	17.4901	12.7024	85.0411	3.2708
T11	17143729506	17.3681	12.7047	103.1185	2.8597
T12	9307644320	17.3956	12.7039	64.3646	2.4969
T13	2545929820	17.3448	12.7024	62.0859	3.2391
T14	9927465777	17.4605	12.7024	130.5873	2.729
T15	4846304503	17.3664	12.7024	69.881	2.0278
T16	4908828949	17.4189	12.7039	122.7165	3.0347
T17	9708582167	17.41	12.7024	86.7732	2.8715
T18	1694437894	17.4241	12.7044	80.3519	2.3543
T19	15092830730	17.406	12.7024	40.8155	2.8656
T20	6056256418	17.5058	12.7037	80.2602	2.5656
T21	9461370318	17.4199	12.7049	76.094	2.4162
T22	7274760836	17.4107	12.7035	42.2611	2.7223
T23	2738732176	17.4562	12.7026	71.4022	1.8685
T24	2203478766	17.4308	12.7027	67.0105	2.6082
T25	8324911133	17.5325	12.7039	76.7896	2.3647
T26	1470849386	17.4333	12.7024	93.1453	2.9336
T27	8868821921	17.4984	12.7026	42.4095	3.4587
T28	3427675957	17.6728	12.7024	78.893	2.9821
T29	3257616112	17.4298	12.7024	38.9709	1.9253
T30	16480815108	17.5187	12.7024	81.0305	2.421
STD	**7294147266**	**17.47763**	**12.70311**	**69.86527333**	**2.760246667**
Average	**5767198154**	**0.098108754**	**0.000889537**	**23.78089555**	**0.432754261**

Table 6-6 Thirty turns result of the Leo Algorithm for CECC06 2019 benchmark from CEC06 to CEC10

Turns	CEC06	CEC07	CEC08	CEC09	CEC10
T1	2.4444	111.9963	4.6443	3.2709	19.9997
T2	2.729	6.279	5.2963	2.8535	20.0002
T3	3.7425	341.5468	5.2276	2.7229	19.9999
T4	3.2169	90.4789	4.4763	2.766	20.0761
T5	1.9316	514.6286	4.7009	2.7147	20.0001
T6	2.8572	138.608	4.939	2.9986	20.0214
T7	2.8216	48.3782	5.8893	2.8272	20.0002
T8	2.7715	466.8038	5.2182	2.7412	20.0001
T9	2.289	-215.6031	5.7596	2.6432	20.0683
T10	3.532	607.4944	4.9719	2.9111	20
T11	1.8078	86.8596	5.6254	3.1509	20
T12	4.8188	1059.1107	5.1002	2.9682	20
T13	3.1563	177.642	5.578	2.6983	20.0002
T14	3.3149	99.4334	6.1724	2.7382	20.0001
T15	3.129	256.9525	5.2011	3.0207	20.0005
T16	2.5807	187.3906	4.3387	2.9384	20.0004
T17	3.4561	60.3966	5.0964	2.6718	20.0001
T18	3.3106	-13.6686	4.9131	2.9292	20.0001
T19	4.0572	181.4768	4.7398	2.9802	20.0002
T20	3.5707	182.4722	5.1698	2.632	20
T21	3.8623	90.2608	5.3156	4.41	20
T22	4.3316	214.1007	4.684	3.6393	20.0002
T23	2.2843	97.8762	5.2246	3.6572	20.0992
T24	2.3059	55.0337	4.8591	3.5732	20.0002
T25	2.486	465.8648	4.7907	4.392	20.0195
T26	2.5048	183.0413	4.5701	3.4268	20.0002
T27	1.6545	107.7778	4.6682	4.1321	20.0843
T28	2.6987	-20.8859	4.6043	6.0846	20.0001
T29	3.9246	121.9088	4.4443	3.3914	20.0001
T30	3.0041	163.0942	5.6493	3.9603	20.0002
STD	**3.01982**	**195.5583033**	**5.062283333**	**3.26147**	**20.01238667**
Average	**0.755956506**	**236.5351502**	**0.459751941**	**0.744492954**	**0.028550895**

7. Publications

The initial crossover standard operator, LPX, has been published in the journal "Systems" by MDPI publisher. The paper titled "A New Lagrangian Problem Crossover: A Systematic Review and Meta-Analysis of Crossover Standards" has been published in a journal with an impact factor of 2.895 and is indexed by reputable databases like Clarivate Analytics and Scopus, among others. As for the second proposed algorithm, Leo, it is currently under consideration at (Expert Systems with Applications Journal-IF:8.5) for review and has not been published yet.

زانکۆی پۆلیتەکنیکی هەولێر
ERBIL POLYTECHNIC UNIVERSITY

ئەلگۆریتمی باشکردنی سەرەتایی لاگرانج لەسەر بنەمای کارپێکەرێکی نوێی کرۆس ئۆڤەر

تێزێکی دکتۆرایە

پێشکەشی ئەنجومەنی کۆلێژی تەکنیکی ئەندازیاری هەولێر کراوە لە زانکۆی پۆلیتەکنیکی هەولێر وەکو بەشێك لە پێداویستیەکانی بەدەست هێنانی پلەی دکتۆرای فەلسەفە لە ئەندازیاری سیستەمی زانیاری

لەلایەن

ئاسۆ محمد علاءالدین محمد

ماستەر لەپرۆگرامسازی و تەکنەلۆجیای ئینتەرنێت - زانکۆی شەفێلد ٢٠١٢

بەکالۆریۆس **لە** ئامارو کۆمپیوتەر- زانکۆی سلێمانی ٢٠١٠

بەسەرپەرشتیاری

پ.د. طارق احمد رشید

کانونی دووەم ٢٠٢٣

پوختە

شێوازی وردبینی پەرەسەندن کێشەکانی باشکردن (ئۆپتیمایزەیشن) چارەسەر دەکات! بەڵام دەتوانرێت کاریگەری وقەبارەدانانەکەی بخرێتە بەرتەحەدا لەگەڵ زیادبوونی ئاڵۆزی کێشەکان. ئەلگۆریتمەکانی میتاهیوریستی پەرەسەندن لەسەر بنەمای دانیشتوان زۆرپشت بە کارپێکەرەکان (ستاندەرەکان - ئۆپەراتۆرەکان) دەبەستن کە ئەدای گشتی خۆیان دیاری دەکەن. ئەم ئۆپەراتۆرانە کاریگەری لەسەر گەڕان و ئیستغلالکردن بەرز دەکەنەوە، کە ڕۆڵیکی زۆر گرنگ دەگێڕن بۆ گەڕان و باشکردنی کاریگەر. توێژینەوەکە کارپێکەری کرۆس ئۆڤەربەناوی کرۆس ئۆڤەری کێشەی لاگرانجی (LPX - Lagrangian Problem Crossover) دەناسێنێت، بۆ بەرزکردنەوەی ئەدای ئەلگۆریتمەکانی پەرەسەندن لە ڕووبەڕووبوونەوەی کێشە نوێیەکانی باشترکردن. سەرەڕای ئەوەش، باشکردنی سەرەتایی لاگرانج (Leo - Lagrange Elementary Optimization) دەخاتە ڕوو وەک ئەلگۆریتمێکی تاکە ئامانجیی (Single-Objective)، کە LPX ڕۆڵێکی بەرچاو دەگێڕێت.

کارپێکەری کرۆس ئۆڤەر لە ئەلگۆریتمەکانی بنەمای دانیشتووواندا زۆر گرنگە بۆ هەڵبژاردنی چارەسەری گونجاو لە پرۆسەکانی باشترکردندا. کارایەکەی پاشەکەوتی کات دەکات وهەڵەکان کەم دەکاتەوە و تێچوونەکان لە بەرنامە ئەندازیارییەکاندا کەم دەکاتەوە. قۆناغی سەرەتایی توێژینەوەکە تێڕوانینێکی گشتی لە شێوازەکانی کرۆس ئۆڤەری ئێستا دەخاتە ڕوو کە پشتپێدەبەستریت لە کارەکانی ئەندازیاری و نوێنەرایەتیکردنی کێشەکاندا. جگە لەوەش، پێشکەشکردنی LPX، تەکنیکێکی تێکەڵاوی تازە و داهێنەرانەیە کە وەردەگیرێت لە بنەماکانی کارکردی دووانەی لاگرانجی (Dual Function LDF – Lagrangian). هەڵسەنگاندنە تاقیکارییەکان LPX لەگەڵ ستانداردەکانی تری وەکو کرۆس ئۆڤەری دووانەیی هاوشێوەکراو(SBX) ، کرۆسۆڤەری تێکەڵکراو(BX) ، و کرۆس ئۆڤەری کوبیت-کرۆسۆڤەر (Qubit-X) لە کرۆس ئۆڤەرە کۆدکراوەکانی ڕاستەقینەدا بەراورد دەکەن. ئەنجامەکان ئاماژە بەوە دەکەن کە LPX بەگشتی لە شێوازەکانی تر باشترە و ئەدای بەراوردکاری لە حاڵەتەکانی دیکەدا نیشان دەدات. بە تایبەتی، لە TF7بۆ LPX ئەدای باشتر و کاتی حیسابکردنی کورتتر لە سەرانسەری هەر سێ بەهای هەڕەمەکیدا نیشان دەدات بە بەراورد ستانداردەکانی تر بۆ بەهای ناوەندی لە (α=0.2) دا دەکاتە 0.0048، لادانێکی ستاندارد لە (α=0.2) دا دەکاتە 0.0031، وەهەژمارکردنی کاتی پێویست بۆ (α=0.2)دەکاتە 143.005 یەکە. شیکاری ئاماری گرنگی و متمانەپێکراوی LPX بە بەراورد لەگەڵ ستانداردەکانی تری کرۆس ئۆڤەر پشتڕاستکراوەتەوە.

لە قۆناغی دووەمی توێژینەوەکەدا شێوازێکی پەرەسەندنی ڕۆمانسی بە ناوی لیۆ دەناسێنرێت. لیۆ پشتی بە هاوکێشەی پرۆسەی وردی کوتان بەستووە کە بەشە ئەلبومینەکانی خوێنی مرۆڤ بەکاردەهێنێت. لیۆ ڕێبازێکی خۆگونجاندن بەکاردەهێنێت بۆ پەرەپێدانی هۆکارە زیرەکەکان لە ڕێگەی بڕینی جینەکانەوە لەسەر بنەمای بەهاکانی کارکردی هاوکێشە دیاریکراوەکان و سیستەمی نوێی ستانداردی کرۆسئۆڤەر کە لە بەشی یەکەم باسی لێوە کراوە. وردی و وردبینی ئەلگۆریتمەکە بە شێوەیەکی بەرفراوان لە ڕێگەی تاقیکردنەوەی تونەوە لەسەر ئەرکە جۆراوجۆرەکانی پێوەرەکان پشتڕاستکراوەتەوە، لەوانەش هەردوو پێوەرەکانی تەقلیدی وCECC06 2019 . ئەدای لیۆ لە بەرامبەر ئەلگۆریتمەکانی ناسراوی وەک Dragonfly، Genetic Algorithm، Practical Swarm Optimization و ئەوانی تر لە سەرانسەری چەندین ئەرکدا پێوانە دەکرێت. بەراوردێکی گشتگیر کاریگەری و کارایی لیۆ لە چارەسەرکردنی کێشەکانی باشکردن لە بەرامبەر ئەم ئەلگۆریتمانەی کە دامەزراون هەڵدەسەنگێنێت. لە باشکردنی ئەرکەکانی تاقیکردنەوەی فرە شێوازی(TF8-TF13) ، بە تایبەتی TF11، ئەلگۆریزمە پێشنیارکراوەکە لە ئەلگۆریتمەکانی تر باشتر بوو، بە تێکڕای بەهای TF11 کە (2.7393E-08). جێی ئاماژەیە، لە سەرانسەری ئەرکەکانی تاقیکردنەوەی پێکهاتەدا(TF14-TF19) ، شێوازی پێشنیارکراو بەردەوام ئەدای بەرزی بەراورد بە ئەلگۆریتمەکانی بنەڕەت نیشاندا. شیکاری ئاماری پشتگیری لە ئەنجامەکانی توێژینەوەکە دەکات و هەروەها دوو بەکارهێنانەکانی ئەپلیکەیشنی نوێ نمایش دەکرێن بۆدۆزینەوەی کێشە نوێیەکان. سەقامگیری ئەلگۆریتمی لیۆ بە بەکارهێنانی پێوەرە ستانداردەکان بۆ گەڕان و ئیستغلالکردن پشتڕاستکراوەتەوە.

زانكۆی پۆلیتەكنیكی هەولێر
ERBIL POLYTECHNIC UNIVERSITY

الخوارزمية التحسين الابتدائية لاغرانج تعتمد على مشغل جديد لكروس-اوفر

رسالة

مقدمة الى مجلس الكلية التقنية الهندسية-اربيل فى الجامعة التقنية-أربيل كجزء من متطلبات نيل درجة الدكتوراه في اختصاص هندسة نظم معلومات .

من قبل

ئاسو محمد علاءالدين محمد

ماجستير أنظمة برمجيات و تكنولوجيا الانترنت - جامعة شفيلد ٢٠١٢

بكالوريوس في الأحصاء والكومبيوتر - جامعة السليمانية ٢٠١٠

باشراف

أ.د. طارق احمد رشيد

ديسمبر ٢٠٢٣

المختصر

طريقة التطور الخوارزميات تحل مشاكل التحسين، ومع ذلك يمكن أن تتعرض فعاليتها وقابلية التوسع للتحدي مع زيادة تعقيد المشكلة. تعتمد خوارزميات التطورية المعتمدة على السكان بشكل كبير على المشغلين الذين يحددون أدائهم العام. يعمل هؤلاء المشغلون على تعزيز الاستكشاف والاستغلال، وهو أمر بالغ الأهمية للبحث والتحسين الفعاليات. يقدم البحث مشغل التقاطع (Lagrangian Problem Crossover -LPX)، لتعزيز أداء الخوارزميات التطورية في معالجة مشاكل التحسين الجديدة. بالإضافة إلى ذلك، فإنه يقدم خوارزمية (Lagrange Elementary Optimization - Leo)، وهي خوارزمية ذات هدف واحد حيث يلعب LPX دورًا مهمًا.

يعد مشغل التقاطع في الخوارزميات المعتمدة على السكان أمرًا بالغ الأهمية لاختيار الحلول المناسبة في عمليات التحسين. تعمل كفاءتها على توفير الوقت وتقليل الأخطاء وتقليل التكاليف في التطبيقات الهندسية. تقدم المرحلة الأولية من الدراسة لمحة عامة عن أساليب التقاطع الحالية المستخدمة في العمليات الهندسية وتمثيل المشكلة. مع ذلك فإن تقديم LPX عبارة عن تقنية هجينة جديدة ومبتكرة تستمد الإلهام من مبادئ وظيفة لاغراض المزدوجة (Lagrangian Dual Function - LDF). تقارن التقييمات التجريبية LPX مع المعايير الأخرى مثل (SBX – Simulated Binary Crossover)، و (BX – Blend Crossover)، و (Qubit-X – Qubit-Crossover) في عمليات الانتقال المشفرة الحقيقية. تشير النتائج إلى أن LPX يتفوق عمومًا على الطروق الأخرى ويظهر أداءً مشابهة للحالات المتبقية. على وجه التحديد في TF7، يُظهر LPX أداءً فائقًا ووقتًا حسابيًا أقصرعبر القيم العشوائية الثلاثة مقارنةً بالمتوسط (α=0.2) عند 0.0048، والانحراف المعياري (α=0.2) عند 0.0031، وحساب الوقت (α=0.2) عند 143.005 وحدة. يتحقق التحليل الإحصائي من أهمية وموثقة LPX مقارنة بمعايير التقاطع الأخرى.

في المرحلة الثانية من هذا الدراسة، تم تقديم طريقة تطورية جديدة تسمى خوارزمية Leo. هذه الخوارزمية مستوحى من عملية التطعيم الدقيقة التي تستخدم في زلال الدم البشري. يستخدم Leo أسلوب التكيف الذاتي، حيث يقوم بتطوير عوامل ذكية من خلال التقاطع الجيني بناءً على قيم وظيفية. يتم التحقق من دقة الخوارزمية وإحكامها على نطاق واسع من خلال اختبارات صارمة على وظائف قياس الأداء المتنوعة، بما في ذلك المعايير التقليدية ومعايير CECC06 2019 . يتم قياس أداء Leo مقابل خوارزميات معروفة مثل Dragonfly، Genetic Algorithm، Practical Swarm Optimization، وغيرها عبر وظائف متعددة. تقوم المقارنة الشاملة بتقييم فعالية Leo وكفاءته في حل مشكلات التحسين مقابل هذه الخوارزميات المعمول بها. في تحسين وظائف الاختبار متعدد

الوسائط (TF8-TF13)، وخاصة TF11، تفوق النهج المقترح على الخوارزميات الأخرى، بمتوسط (E-082.7393). والجدير بالذكر أنه عبر وظائف الاختبار المركبة (TF14-TF19)، أظهرت الطريقة المقترحة أداءً عاليا باستمرار مقارنة بالخوارزميات الأساسية. ويدعم التحليل الإحصائي استنتاجات البحث، كما يتم عرض تطبيقات العالم الحقيقي لهذه خوارزمية. يتم تأكيد استقرار هذه خوارزمية باستخدام المعايير القياسية للاستكشاف والاستغلال.