TURKISH REPUBLIC
ERCIYES UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE
DEPARTMENT OF COMPUTER ENGINEERING

A NOVEL MODIFIED TEACHING-LEARNING BASED ALGORITHM AND ITS APPLICATIONS

Prepared by
Tareq Meqdam Tareq AL-BASHAQHA

Supervisor
Asst. Prof. Dr. Ahmet Nusret TOPRAK

M.Sc. Thesis

October 2021
KAYSERİ

TURKISH REPUBLIC
ERCIYES UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCINCE
DEPARTMENT OF COMPUTER ENGINEERING

A NOVEL MODIFIED TEACHING-LEARNING BASED ALGORITHM AND ITS APPLICATIONS

(M.Sc. Thesis)

Prepared by
TAREQ MEQDAM TAREQ AL-BASHAQHA

Supervisor
Asst. Prof. Dr. Ahmet Nusret Toprak

October 2021
KAYSERİ

COMPLIANCE WITH SCIENTIFIC ETHICS

I declare that all information in this work were obtained in accordance with academic and ethical rules. All results and material that not been at the essence of this work are also transferred and expressed by giving reference as required by these rules and behavior,

Student

TAREQ MEQDAM TAREQ AL-BASHAQHA

SUITABILITY FOR GUIDE

The MSc thesis entitled "**A Novel Modified Teaching-Learning Based Algorithm and Its Applications**" has been prepared in accordance with Erciyes University Graduate Education and Teaching Institute Thesis Preparation and Writing Guide.

Student	Supervisor
Tareq Meqdam Tareq AL-BASHAQHA	Asst. Prof. Dr. Ahmet Nusret TOPRAK

Chairman of the Department of Computer Engineering

Prof. Dr. Veysel ASLANTAŞ

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation for my advisor Asst. Prof. Dr. Ahmet Nusret TOPRAK. The door to Asst. Prof. Dr TOPRAK office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this study to be my work but steered me in the right direction whenever he thought I needed it.

I would like to thank my friend Ahmed Al-gbure for his constant support and help during this journey, I'm so lucky to have you, also I would like to thank Mohammed Hassan Nabeel AL-MASHAT and Ibraheem Bayan Abdullah, all of them have been there to support me when I collected data for my M.Sc. thesis. A special thanks to my supporting friends Hussien Al-Ezzi, Sabah Al-jurani, Almustafa Al-fawadi and Marwan Khalaf.

To my Parents, there are no words that can describe my gratitude this is wont happened without your support and prayers, A big thank you to my sisters. My parents in law you were my family in the expatriate country I can't express how grateful I am.

The biggest thank you to my little family, my wife thank you for sharing this adventure with me, you were and always be my support system my backbone, my kids I love you and I'm sorry for the times I was busy.

At last, I want to dedicate this thesis to my first supervisor Assist. Prof. Dr Mustafa DANACI, for his support and advice on both research as well as on my career, has been priceless, may his soul rest in peace.

Tareq Meqdam Tareq Al-BASHAQHA

Kayseri, October 2021

A NOVEL MODIFIED TEACHING-LEARNING BASED ALGORITHM AND ITS APPLICATIONS

Tareq Meqdam Tareq AL-BASHAQHA

Erciyes University, Graduate School of Natural and Applied Sciences
M.SC. Thesis, October 2021
Supervisor: Assist. Prof. Ahmet Nusret TOPRAK

ABSTRACT

In the last few decades, a great number of various algorithms based on Human social-based algorithms have been intensively researched and applied to different optimization problems through the study of natural species. The Teaching-Learning Based Optimization (TLBO) is a recently proposed meta-heuristic optimization algorithm that has shown to be effective in a wide range of solving real-life problems. However, it's a high-performance algorithm but the balance between the exploration and exploitation is weak, which is simply dropped in the local solutions.

In this study, we proposed a new modification for the Teaching Learning Based Optimization (TLBO) by using a rearrangement parameter called "Academic Plan" which added to the new solution equation. The mathematical form of this parameter represented as a random number minimized from 1 to 0 with the iterations interval, this mechanism entered to the population term and update agents positions by avoid the local solution and look for a global.

The performance of the proposed algorithm is tested by applying it to 23 numerical benchmark functions of a wide range of types and dimensions, and the results compared with the original TLBO and other swarm-based optimization algorithms. On the other hand, our thesis contributes a new solution to the Symmetric Traveling Salesman Problem (sTSP), which is considered one of the most popular combinatorial optimization problems. We presented a hybrid version of TLBO and the 2-opt algorithm with the hamming distance function. In this section, we applied the new proposed algorithm on the TSP library benchmarks functions in order to test the performance of the new proposed algorithm. Moreover, we compared the results with other algorithms from the literature.

Keywords: Teaching Learning Based Optimization, Travelling Salesman Problems, 2-Opt Algorithm, Hamming Distance Function, Human social-based algorithms.

YENİ BİR DEĞİŞTİRİLMİŞ ÖĞRETME-ÖĞRENME TABANLI ALGORİTMA VE UYGULAMALARI

Tareq Meqdam Tareq AL_BASHAQHA

Erciyes Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans, Ekim 2021
Danışman: Dr. Öğr. Üyesi Ahmet Nusret TOPRAK

ÖZET

Son yıllarda, çok sayıda sosyal tabanlı optimizasyon algoritması geliştirilmiş ve farklı optimizasyon problemlerine uygulanmıştır. Öğretme-Öğrenme Tabanlı Optimizasyon (TLBO) algoritması, çok çeşitli gerçek hayat problemlerinin çözümünde etkili olduğu gösterilen, yakın zamanda önerilen bir meta-sezgisel optimizasyon algoritmasıdır. TLBO yüksek performanslı bir algoritma olmakla beraber arama ve tüketim arasındaki dengesi zayıftır. Bu durum yerel çözümlere takılı kalabilmesine sebep olmaktadır.

Bu çalışmada, algoritmaya eklenen "Akademik Plan" adlı bir yeniden düzenleme parametresi kullanılarak Öğretme Temelli Optimizasyon için yeni bir değişiklik önerilmektedir. Bu mekanizma, algoritmadaki bireylerin pozisyonlarının yerel çözüme takılı kalmadan küresel en iyiyi bulmalarını sağlamaktadır. Önerilen TLBO algoritması, çok çeşitli tür ve boyutlarda yirmi üç test fonsiyonuna uygulanmış ve algoritmanın performansı değerlendirilmiştir. Ayrıca önerilen algoritma, orijinal TLBO ve diğer sürü tabanlı optimizasyon algoritmaları ile karşılaştırılmıştır.

Diğer taraftan, bu çalışmada en popüler kombinasyonel optimizasyon problemlerinden biri olarak kabul edilen Simetrik Gezgin Satıcı Problemine (sTSP) yeni bir çözüm sunulmaktadır. Bu çözümde TLBO algoritmasının değiştirilmiş bir başka versiyonu, Hamming mesafe fonksiyonu ve 2-Opt algoritmasından yararlanılmaktadır. Bu bölümde, önerilen yeni algoritmanın performansını test etmek için algoritma farklı TSP test verilerine uygulanmış ve sonuçlar literatürdeki diğer algoritmalarla karşılaştırılmıştır.

Anahtar Kelimeler: Öğretme-Öğrenme Optimizasyon yöntemi, Gezgin Satıcı Problemleri, 2-Opt Algoritması, Hamming Uzaklık Fonksiyonu, İnsan sosyal tabanlı algoritmalar.

CONTENTS

A NOVEL MODIFIED TEACHING-LEARNING BASED ALGORITHM AND ITS APPLICATIONS

COMPLIANCE WITH SCIENTIFIC ETHICS ii
SUITABILITY FOR GUIDE iii
APPROVAL iv
ACKNOWLEDGEMENTS v
ABSTRACT vi
ÖZET vii
CONTENTS viii
LIST OF TABLES xi
LIST OF FIGURES xiii
INTRODUCTION 1

1. CHAPTER

LITERATURE REVIEW

1.1. Meta-Heuristic Optimization Algorithms 5
1.1.1. Categories Of Meta-Heuristic Algorithms 6
1.1.2. Meta-heuristic Algorithms Employed to Comparison on Solving Benchmark Problems 11
1.1.2.1. Bat Algorithm (BA) 11
1.1.2.2. Firefly Algorithm (FA) 12
1.1.2.3. Particle Swarm Optimization Algorithm (PSO) 13
1.1.2.4. Hybrid Sine Cosine Whale Optimization Algorithm (MVSCWOA) 14
1.1.2.5. Artificial Bee Colony (ABC) 15
1.1.2.6. Modified Whale Optimization Algorithm (MWOA) 15
1.2. Problem Definition 16

1.2.1. Benchmark Problems 16

1.2.1.1 Unimodal Functions (F1-F7) 17

1.2.1.2. Multi-Modal Functions (F8-F13) 19

1.2.1.3. Fixed Dimensional Modal Functions (F14-F23) 22

1.3. Travelling Salesman Problem (TSP) 26

1.3.1. Problem Definition 26

1.3.2. Heuristic and Meta-Heuristic Algorithms Employed to Comparison on Solving symmetric travelling salesman 28

1.3.2.1. Exact Solvers approaches 28

1.3.2.2. Non-exact Solvers approaches 29

1.3.2.3. Well-Known Algorithms for TSP Solving 30

2. CHAPTER

A NOVEL MODIFIED TEACHING-LEARNING-BASED OPTIMIZATION ALGORITHM

2.1. TLBO Algorithm 33

2.1.1. Teacher Phase 33

2.1.2. Learner Phase 34

2.2. Modifying the Teaching Learning Based Optimization (TLBO) 36

2.2.1. The Modification Process 37

2.3. Experimental Results Of The Proposed Modified TLBO Algorithm 43

2.3.1. Results Analysis and Simulation 43

3. CHAPTER

A MODIFIED TEACHING LEARNING BASED OPTIMIZATION ALGORITHM FOR SYMMETRIC TRAVELING SALESMAN PROBLEM

3.1. Teaching Learning Based Optimization Algorithm 48

3.2. 2-Opt Algorithm 48

3.3. A Modified Teaching Learning Based Optimization Algorithm for Symmetric Traveling Salesman Problem 50

3.4. Experimental Results.....51

4. CHAPTER

CONCLUSION AND FUTURE WORK 55

REFERENCES.....58

LIST OF TABLES

Table 1.1. Sphere Function (F1) unimodal benchmark functions used in experiments. Dim: dimension, f min: optimal solution, range: upper and lower bounds. 17
Table 1.2. Schwefel Functions (F2, F3, F4) Unimodal benchmark functions used in experiments. Dim: dimension, f min: optimal solution, range: upper and lower bounds. 18
Table 1.3. Rosenbrock Function (F5) Unimodal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds. 18
Table 1.4. Step Function (F6) Unimodal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds. 19
Table 1.5. Quartic Function (F7) Unimodal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds. 19
Table 1.6. Schwefel Function (F8) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds. 20
Table 1.7. Rastrigin Function (F9) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 21
Table 1.8. Ackley Function (F10) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 21
Table 1.9. Griewank Function (F11) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 22
Table 1.10. Foxholes Function (F14) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 23
Table 1.11. Kowalik Function (F15) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 23

Table 1.12. SixHump Function (F16) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 24
Table 1.13. Branin Function (F17) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 24
Table 1.14. GoldsteinPrice Function (F18) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 25
Table 1.15. Hartman Functions (F19, F20) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 25
Table 1.16. Shekel Functions (F21, F22, F23) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds 26
Table 2.1. Comparison between the TLBO and mTLBO computational time. 39
Table 2.2. Comparison between the 4 cases we studied shows that case 1 got the highest optimal solutions for the tested benchmarks problems.as well as the highest 1st and 2nd close to optimal solutions. 40
Table 2. 3. Shows the average results of 30 time repeated runs on the Unimodal functions benchmark problems used to compare between different meta-heuristic algorithm 45
Table 2. 4. Shows the average results of 30 time repeated runs on the Multimodal functions benchmark problems used to compare between different meta-heuristic algorithm 46
Table 2. 5. Shows the average results of 30 time repeated runs on the Fixed-modal functions benchmark problems used to compare between different meta-heuristic algorithm 47
Table 3.1. Shows the average results of 30 time repeated runs on the TSP-LIB benchmark problems used to compare between different algorithm 54

LIST OF FIGURES

Figure 1.1. Publication structure over the years in Web of Science 7
Figure 1.2. Citation structure over the years, as per Web of Science 8
Figure 1.3. The classification of the nature-based meta-heuristics optimization algorithms 9
Figure 2.1. TLBO Flowchart 36
Figure 2.2. The differences between TLBO and mTLBO search space 38
Figure 2.3. New Modified TLBO Flowchart 42
Figure 3.1. 2-opt main idea 49
Figure 3.2. Hamming distance 50
Figure 3.3. The optimal tour gained by MDTLBO-TSP applied on (bays29) which is a set of 29 cities, from TSPLIB. The minimal tour has length 2020 52
Figure 3.4. The first close to optimal tour gained by MDTLBO-TSP applied on (eil51) which is a set of 51 cities, from TSPLIB. The minimal tour has length 426 53

INTRODUCTION

Real-world optimization problems are complicated and have a high dimensional search space, making them harder in general. The term "optimization" refers to the process of selecting the best possible solution for a given problem from a wide range of appropriate solutions. In most cases, an optimization problem is converted into a multi-dimensional search space. In fact, this means minimizing or maximizing a fitness function that determines the quality of a solution choice, which is generally represented as a vector in the search space. Meta-heuristics are a collection of approximation optimization approaches that deliver acceptable results in a reasonable amount of time and are used to solve difficult and complicated issues in science and engineering [1]. Meta-heuristic optimization algorithms have grown in popularity due to their ease of use and flexibility compared to traditional and exact optimization techniques [2]. The optimization technique determines the best solution variables by minimizing or maximizing the optimal solution of a function or problem [3].

There are two major search techniques used by meta-heuristic optimization algorithms: (1) exploration and (2) exploitation [4]. Exploration is the ability to search in a global search space. This ability is connected to avoiding local optima and overcoming trapping in local optima. On the other hand, exploitation is the ability to explore potential local solutions within the local search space to increase their value locally [5]. An essential part of an effective performance algorithm is considered based on an appropriate balance amongst these two techniques [6]. These properties are used by all population-based algorithms, although with various operators and techniques.

Swarm intelligence algorithms, Physical and Chemistry based algorithms, Plants-based algorithms, and human-based algorithms are all inspirations for one popular classification of meta-heuristics [7]. Swarm intelligence algorithms are a category of meta-heuristics that mimic the behavior of animals in groups when moving or hunting [8]. The main aspect of this group is the sharing of all animals' organism information

via the optimization process. This group's popular methods include the Particle Swarm Optimization (PSO) [9], Artificial Fish Swarm Algorithm (AFSA) [10], Ant Colony Optimization (ACO) [11], Artificial Bee Colony (ABC) [12], and Cuckoo Search (CS) [13] etc.

Another type of optimization algorithm is physics-based methods. This group is inspired by real-world physical and chemical laws and generally represents the communication of search solutions based on managing principles established in physical and chemical techniques [14]. Simulated Annealing (SA) [15] and Big Bang–Big Crunch (BB–BC) [16] are the most widely used algorithms in this category.

Plants-based algorithms are proposed to mimic plant growth, and the optimization problem must be linked to the growth process. In principle, the search space should map to the entire plant growth environment, with each individual representing a virtual branch. Also, supplies should be provided. Except for sunshine, water, CO2, and other elements are believed to be infinite. Because light intensity varies amongst branches, it may be considered a branch's fitness value [17]. The most common algorithms of this category are Artificial Plants Optimization Algorithm [18], Natural Forest Regeneration Algorithm [19], Tree Growth Algorithm [20]. The last category of optimization is human-based algorithms, which are inspired by human activities and community behavior. The Imperialist Competitive Algorithm [21], driven by human socio-political progress, is one of the most often employed algorithms in this category. The Teaching–Learning-Based Optimization Algorithm [22] is another algorithm in this category.

In this thesis, a new modified Teaching-Learning Based Optimization Algorithm is studied. The Teaching-Learning Based Optimization (TLBO) proposed by Rao et al. [22] simulates the human social behavior represented by students' classroom learning actions. According to the algorithm, students gain information not just from a teacher's presentation but also from their mutual communication with other students. The TLBO algorithm employs limited parameters and has proven empirical performance for various optimization problems. It is also straightforward to learn and execute. Its employment has been extended to include function, engineering, and multi-target optimization challenges. On the other hand, TLBO is also considered as an algorithm without an active equation for the searching phase as there is one equation in each

teaching and learning level that supports the exploitation phase. Although it has a fast convergence speed, it can easily fall into the local optimum in solving complex global optimization problems.

In the modified TLBO algorithm, a new equation is added for each teaching and learning phase with random orientation to force the algorithm to avoid the current solution and search for a new, better solution. This way led to the jump from the local solutions and trying to find the global one. As well as it was applied to twenty-three numerical benchmarks and compared its results with other meta-heuristics algorithms like Bat Algorithm (BA) [23], Genetic Algorithm (GA) [24], Practical Swarm Optimization (PSO) [9], Artificial Bee Colony (ABC) [12], Moth-Flame Optimization algorithm (MOF) [25], Whale Optimization Algorithm (WOA) [26], Firefly Algorithm (FA) [27], Hybrid Shelved Frog Lapping algorithm (HSFLA) [28], Multiverse Sine Cosine (MVSCWOA) [29], Modified Whale Optimization Algorithm (MWOA) [30], Hybrid Sine Cosine Ant lion Optimization (HSCALO) [31]. The thesis modification showed an excellent performance in solving continuous optimization problems represented by the numerical benchmarks. Continuous optimization allows variables to take any value within a range, generally real numbers, compared to discrete optimization, where variables can be binary (limited to 0 and 1), integer (only integer values permitted), or more abstract objects taken from sets with finite components.

On the other hand, Discrete optimization requires that some or all of the variables in a model be discrete, whereas continuous optimization allows for any value within a range. One of the most common discrete optimization branches is Combinatorial optimization problems (COPs) [32]. COP is the process of selecting the best solution from a finite set of solutions. In many cases, a comprehensive search is not possible. It performs on optimization problems when the set of feasible solutions is discrete or may be reduced to discrete, and the aim is to identify the optimal answer [33]. One of the most popular COPs is the Traveling Salesman Problem (TSP) [34], which is often used as a benchmark for algorithm testing and is considered one of the well-known NP-hard issues that have received much attention from researchers in the field of computer science and computational modeling. TSP has inspired a lot of social, scientific, and industrial attention in the recent decade. The goal of a salesman is to visit each location just one time and make a complete tour by heading back to the first city for the lowest

cost. Decision variables in TSP are discrete. Several precise approaches have been proposed in the literature, including cutting plane [35], branch and bound [36], Lagrangian dual [37], and so on, which are used to solve TSP.

The recent study proposes a new method for solving TSP by incorporating TLBO with the 2-opt algorithm to get better results. 2-opt is part of the local search algorithm family. Therefore, this study helps to accelerate convergence and find proper solutions.

This thesis is illustrated in four sections; section one contains the literature review, giving brief information about meta-heuristic algorithms. Also, a brief description of the algorithms used to compare their results in solving continuous problems of benchmark problems and a brief review of traveling salesman problem and its variants, heuristic and meta-heuristic algorithms employed to comparison on solving symmetric traveling salesman and well-known algorithms for TSP solving is given in the same section. Section two illustrates the modifying algorithm and other aspects related to it and describes the proposed mechanism and its effect on the TLBO performance. Also, in section three, we presented a modified teaching learning-based optimization algorithm for symmetric traveling salesman problem and employed it on the benchmark problems from TSP-LIB, and results are also presented. Finally, chapter four sets the conclusion and new ideas for the future and next work.

1. CHAPTER

LITERATURE REVIEW

1.1. Meta-Heuristic Optimization Algorithms

In the last decade, the literature on optimization algorithms has increased tremendously. In addition, Swarm Intelligence is also used in nature-inspired optimization algorithms, such as Particle Swarm Optimization (PSO) [9], Ant Colony Optimization (ACO) [11], Artificial Bee Colony Algorithm (ABC) [12], and Cuckoo Search (CS) [13], these algorithms which are based on swarm intelligence, have proven to be very powerful.

Almost all optimization problems are solved using the most well-known algorithms. However, individuals are the key component of bio-inspired algorithms [38], which use biological organisms as their main component. Thus, every individual represents a solution to the problem at hand.

Other bio-inspired algorithms, such as swarm intelligence [38], describe interactive and cooperative behaviors among algorithm agents. Those algorithms represent animal culture through various mechanisms. Some of them follow light like Firefly [39], whales find another way to foraging by creating bubbles [40], and other works in a group form like Particle Swarm [41] or colonies like Ant Colony [42], Bee Colony [43], and Gray Wolf pack [44] formed in a way named surrounded strategy, also other algorithms mimic humans activities, lifestyle and the ability to control and the development of the human ability to face the challenges in day to day life, for example, anarchic society optimization algorithm [45], Imperialist competitive algorithm [21], Political optimizer [46] and Teaching learning based optimization [22]. Ultimately, specific main ideas are illustrated in this study that is often expressed, such as algorithm refers to a step-by-step method for doing calculations or giving instructions.

The specific steps and procedures are determined by the algorithm and the context of interest used to solve particular problems [47]. Heuristics is a problem-solving or self-discovery technique that isn't guaranteed to be ideal, flawless, or rational but is adequate for achieving an approximate solution when the traditional methods fail to find an exact solution [48]. Heuristic approaches can be used to speed up the process of finding a suitable solution when finding an ideal solution is difficult or impractical. Heuristics are behavioral shortcuts that will help you make a decision faster [49]. As well as Meta-heuristic is a higher-level technique or heuristic for finding, generating, or selecting a heuristic which may also provide a necessary better solution to an optimization problem, particularly when there is missing or imperfect knowledge or restricted computation capacity [50]. Meta-heuristics explore a subset of solutions that would otherwise be too massive to be completely addressed or discussed in any other way. Meta-heuristics can be applied to a wide range of problems because they make little predictions about the optimization-specific problem [51]. Problem is a description for an investigation into or demonstration of a finding that begins with the specified conditions. Optimization is the method of selecting the optimal solution from a collection of alternatives. Benchmark functions are used to evaluate the accuracy of the optimization algorithms. Modifying an optimization algorithm is a development operation normally obtained within the algorithm by adding control parameters or changing the mathematical model to make it faster and more efficient [52].

1.1.1. Categories Of Meta-Heuristic Algorithms

Nowadays era, full of computational problems of ever-increasing complexity. There is no single shred of speculation within the scientific community that behavioral dynamics and processes discovered in nature have established the foundations among several meta-heuristic algorithms [53]. The search term is "meta-heuristic," and it will last until the end of 2019. The data spans 30 years, with the quest year spanning 1989 to 2019. Since, from a citation-based standpoint, the word "meta-heuristic" was only coined in the mid-1990s. That can also be verified by looking at the year of the first printing, which was 1994.

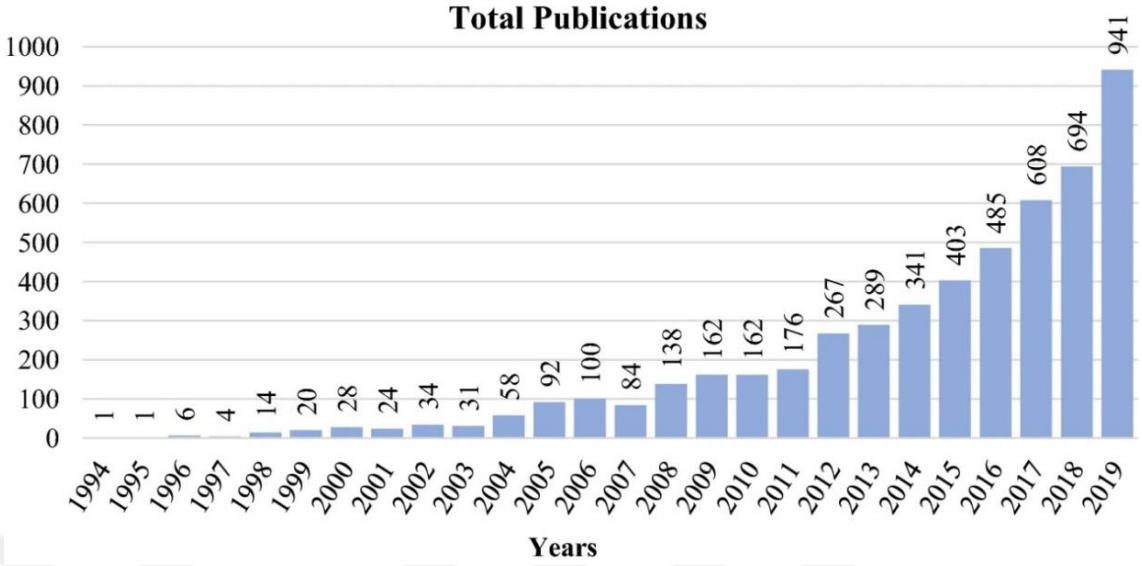


Figure 1.1. Publication structure over the years in Web of Science

Throughout the last three decades, the publishing and citation framework in the meta-heuristic research domain has changed dramatically. As seen in Fig. 1, the meta-heuristic concept was used in a publication at first in 1994. Since there were only 12 publications, the subject didn't Saw the light until 4 years later. Nevertheless, in 1998, the number of publications began to rise. In just 13 years, it achieved a total of 100 publications (2006). Over the years, the rate of growth has been impressive. It can be explained by the fact that 52.84 percent of total publications in the meta-heuristic field were published within only four years (2015–2019). According to the trend, the year 2019 saw the most publications (941); in 2020 and 2021, printed almost 1,430 papers [54].

The citation structure of a study field is another theme to consider. Fig 1.2. shows a line graph of the number of citations obtained by articles in the meta-heuristic domain. There were only 12 citations of work relating to this field, as shown by the number of publications. Since that, the number of citations has risen at an unprecedented rate over time. The largest number of citations (19,506) is issued in 2019, marking the fourth year in a row that the number of citations has exceeded 10,000 [54].

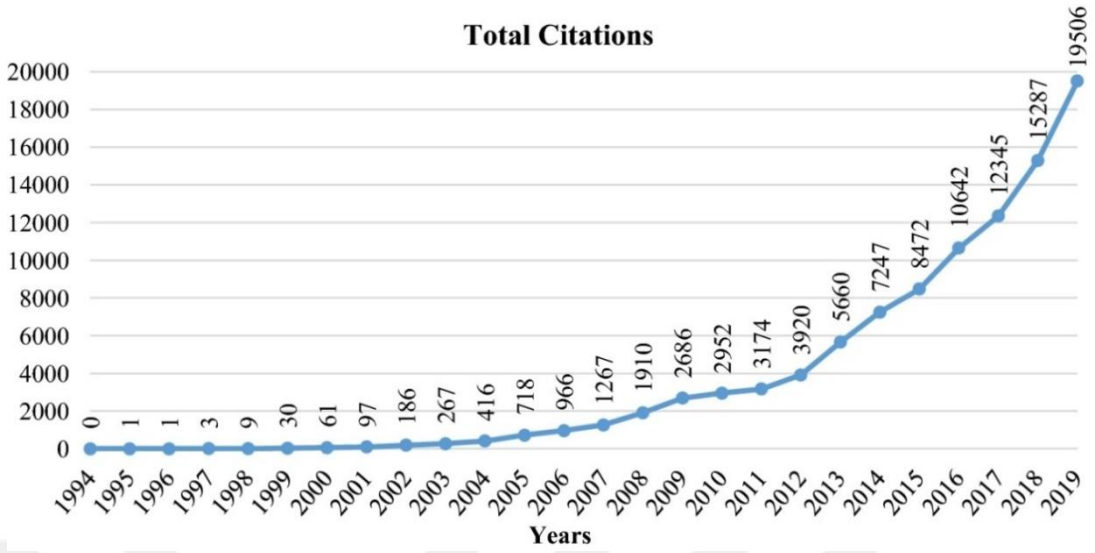


Figure 1.2. Citation structure over the years, as per Web of Science

Almost any meta-heuristics layout in the literature is stated to be motivated by a natural or physical phenomenon. These forms of motivation are caused by biological, chemical, and physical processes, with algorithmic architecture experiences partially or fully trying to imitate the phenomenon mentioned above's behavioral convolutional layer [55]. Most of the so-called meta-heuristic algorithms have also recently become labeled as nature inspired. This is because the algorithmic design ideas were influenced by borrowing some motivational influence from nature. As a result, meta-heuristics are frequently divided into sub-categories based on the type of inspiring source used. Nature-inspired meta-heuristics are a broader category of algorithms created using natural inspiration classified into four main categories [50]:

- Swarm intelligence algorithms,
- Physical and Chemistry based algorithms,
- Human social-based algorithms,
- Evolutionary algorithms.

Swarm intelligence algorithms are a multi-agent mechanism that relies on the emergent actions of multiple agents adhering to a set of basic laws. As a result, certain entities in a swarm may or may not act intelligently. Self-organized activity can occur in such a structure. As a result, it appears that there is some balance in the swarm.

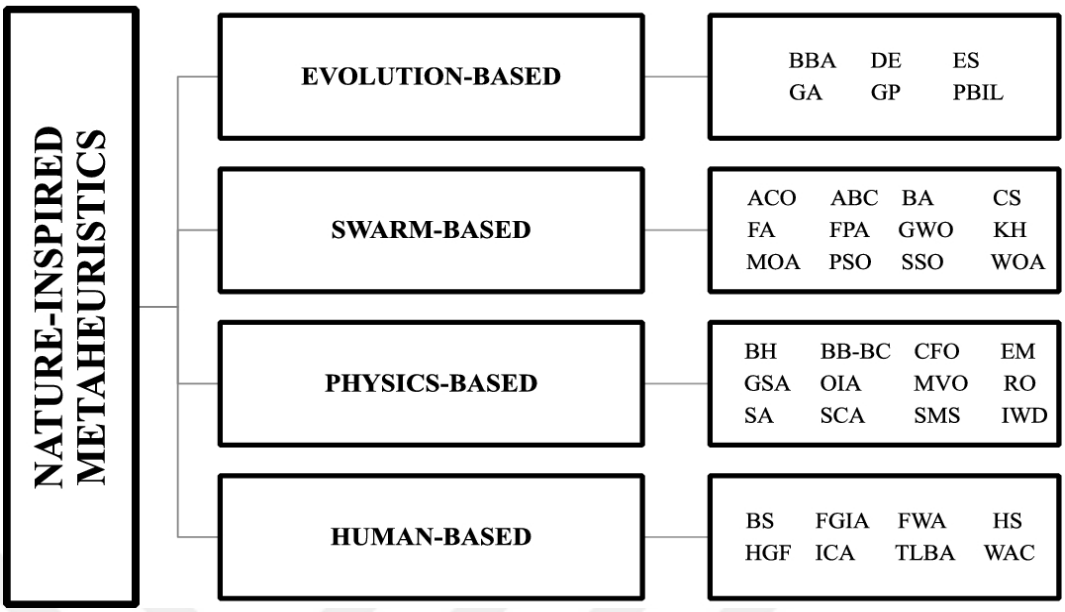


Figure 1.3. The classification of the nature-based meta-heuristics optimization algorithms

Swarm intelligence algorithms include well-known algorithms like Particle Swarm Optimization (PSO) [9], Artificial Fish Swarm Algorithm (AFSA) [10], Ant Colony Optimization (ACO) [11], Artificial Bee Colony (ABC) [12], Cuckoo Search (CS) [13], Bat-Inspired Algorithm (BA) [23], Firefly Algorithm (FA) [27]. Swarm intelligence subcategories are divided into four groups based on an animal's natural actions or the world in which it resides in existence.

- **Water animals:** algorithms that are based on animals that live in water. The most common algorithms in this group are school fish and animals that use echolocation for foraging, such as dolphins and whales.
- **Wild animals:** Meta-heuristics of individuals traveling on the field for discovery, such as Ant Colony, Wolf Optimization Algorithms, and Lion Algorithms, are examples of this subcategory.
- **Animals who can fly:** Particle Swarm (PSO), which mimics birds, Grasshoppers, Fireflies, as well as other flying insects, is a well-known technique for this meta-heuristic emphasis on flying creatures, in which a single species uses flying to move from one location to another.
- **Microorganisms**: they are called by that while bacteria move toward food and viruses that live within cells.

Physical and chemical events are used to classify algorithms in Physical and Chemistry based algorithms category. Since the features of both categories are the same, certain articles grouped them in the same row. As a result, we've grouped all of the categories into a single table. Physical-based algorithms account for more than 15% of all nature-inspired algorithms, while chemistry-based algorithms account for 3%. Familiar physical algorithms such as Simulated Annealing (SA) [15], Big Bang–Big Crunch (BB–BC) [56] reflect physical processes such as gravity, electromagnetism, and water flow. Chemical processes and gas flow, on the other hand, use the Artificial Chemical Reaction Optimization Algorithm (ACROA) [57]. It's worth noting that physical and chemistry algorithms fall into the category of nature-inspired algorithms, and they have little connection to biology-based algorithms.

Almost all papers in the category of the Evolutionary algorithms which are considered as heuristic-based methods solving problems that cannot be solved effectively in polynomial time, such as classic NP-Hard problems which also would take much too long to analyze entirely. When used individually, often used to combinatorial problems; nevertheless, genetic algorithms are frequently employed in conjunction with other algorithms, working as a rapid means to identify a fairly ideal starting point for another algorithm to work off [58]. To understand an evolutionary algorithm, first must understand how natural selection functions. This is where evolutionary algorithms come in. Initialization, selection, genetic operators, and termination are all processes in an EA. To modularize this algorithm category, follow these stages, which each approximately correlate to a different feature of natural selection. Better members will live and multiply in an EA, whereas unworthy ones will die off and not contribute to the gene pool of future generations, just like in natural selection. Unfit members will be eliminated [59]. In the evolutionary computation category, we can mention the following algorithms Genetic Algorithm (GA) [24], Differential Evolution (DE) [60] Evolution Strategy (ES) [61].

Human social-based algorithms this group includes algorithms that are descended from human behavior. Teaching and Learning Optimization Algorithm (TLOA) [22] is an algorithm that represents the potential to learn and teach. Imperialist Competitive Algorithm (ICA) [21], for example, deals with politics. In this section, the most common algorithms are Harmony Search (HS) [62], Taboo Search (TS) [63], Group

Search Optimization (GSO) [64], League Championship Algorithm (LCA) [65], Fireworks Algorithm (FOA) [66], Interior Search Algorithm (ISA) [67], Soccer league competition algorithm (SLCA) [68].

1.1.2. Meta-heuristic Algorithms Employed to Comparison on Solving Benchmark Problems

In this section, we are going to give information about the meta-heuristics that used in this thesis to compare it with our approach such as Firefly Algorithm, Particle Swarm Optimization, Bat Algorithm, Artificial Bee Colony etc. which are used to solve the standard Numerical Benchmark problems.

1.1.2.1. Bat Algorithm (BA)

Bats are incredible creatures. It has a head like Canidae families like wolves and dogs, but its body is made of leather and looks like a bird. Bats are nocturnal creatures, sleeping during the day and hunting at night. The intriguing thing about bats is that they utilize an echolocation mechanism to guide them in the dark. There are around 996 bat spaces, which is a large number that accounts for 20% of all mammals on the planet [23]. Bats have extremely small spaces, such as micro bats with a length of (2.511 cm), and the size ranges from the tine bumblebee bat (1.52 g) to gigantic bats with a wingspan of around (2 m) and weights of up to (1 kg). Behaviors will be mimicked in these algorithm micro bats. [69]. Micro bats are almost like insects. It uses echolocation to hunt, knowing where they are, avoiding obstructions, and finding their roosting holes in the dark. The echolocation method comprises of producing an audible voice and listening for a faction wave. Normally, the volume of a bat's voice varies depending on the species; some utilize short frequency modulation, while others use constant-frequency modulation; signal bandwidth varies depending on the species, while it is frequently enhanced by utilizing additional harmonics [69]. There is the capacity to assess bat algorithm or bat-inspired algorithm below rules simple bats characteristics [69] according to understand echolocation characteristic paradigm. All bats utilize an echolocation system to detect prey location and knowing its location and avoid barriers.

- Bats fly with constant velocity v_i in certain location $x_i$with various frequencies (f_{min}) with change wavelength λ and loudness A_0.

- The loudness will range from a big positive A_0 to a small constant value (f_{min}).

Time delay and three dimensions will be ignored for the sake of simplicity. Also, approximation between frequency [f_{min}, f_{max}] and wavelength [λ_{min}, λ_{min}] will be utilized. Bats' motions are abbreviated by updating current position *xi* and current velocity *vi* to the new position x_i^t and new velocity V_i^t at t time as equation below

$$fi = fmin + fmax - fmin\,\beta \quad (1.1)$$

$$V_i^t = V_i^{t-1} + (x_i^t - x*)fi \quad (1.2)$$

$$x_i^t = x_i^{t-1} + V_i^t \quad (1.3)$$

where β is random value in range of [0.1] drown from uniform distribution and *x** is the current global best solution, *fi* is one of two factors can be used to influence velocity.

Once a solution is chosen from among the existing best solutions for the local search, a new solution for each bat is created locally using a random walk.

$$x_{new} = x_{old} + \epsilon A^t \quad (1.4)$$

where ϵ random number between [-1,1] and $A^t <= A_i^t$.

Since the Bat Algorithm's updates of location and velocity are comparable to those of the Particle Swarm Algorithm, it may be considered a branch of the Particle Swarm Optimization Algorithm.

1.1.2.2. Firefly Algorithm (FA)

The flashing behavior of fireflies is the major motivation for this algorithm. Those insects that flew around at night and searched the area as if they were little lamps that might be found on summer nights. A firefly is a tiny flying insect with a transparent bag at the end of its body that contains a light-emitting material. Females in the same species were attracted to distinct patterns of flashing lights generated by males, which was a fascinating aspect of this insect's behavior.

When it comes to flashing, the most significant factor is brightness [70]. Where a firefly with lower brightness attracts a firefly with higher height, brightness, this behavior may be mathematically modeled using a function. Generally

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} \left(x_j - x_i \right) + \alpha \epsilon_i \quad (1.5)$$

where:

x_i, x_j are position of i_th and j_th fireflies respectively

β_0 is attractiveness coefficient (called beta in MATLAB code implementation)

r is distance

α & ϵ_i third term are randomization (which called alpha and e in MATLAB code)

γ is light absorption coefficient (called gamma) implementation

e is function including mutation operation

where:

$$e = \Delta * unifrnd(-1, +1, v)$$

$$\Delta = 0.05 * (ub - lb)$$

ub, lb are upper bound and lower bound, respectively. The distance between the position of brightness and less bright fireflies can be simulated with the function below generally

$$r_{ij} = \left| x_i - x_j \right| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \quad (1.6)$$

1.1.2.3. Particle Swarm Optimization Algorithm (PSO)

Animals usually live in groups, swarms, or colonies, and these assemblages are the source of power for all creature species. Some Elephant herds, for example, use the

gathering to defend themselves from lions and other predators. Swarms are also used by others for foraging and other purposes [71]. PSO algorithm was inspired by the social swarm behavior of birds, fishes, and insects, with the individuals in these swarms being referred to as particles. The acceleration of each individual and the next position are the main feature in these swarms, these two parameters, where PSO algorithms developed around them, are crucial in these swarms. Other intriguing aspects of this theory are stochastic and deterministic mechanisms, in which each individual tries to arrive at the optimal global position alongside its own best location, and these two stages are deterministically represented. On the other hand, each person has a tendency to migrate free primitiveness, which is stochastic. These two qualities are comparable to exploration and exploitation in other algorithms, which allows algorithms to search space and catch on worldwide while becoming stagnant locally. The functions that match the PSO algorithm's behavior are listed below.:

$$v_i^{t+1} = v_i^t + \alpha\, \varepsilon_1 [g^* - x_i^t] + \beta\, \varepsilon_2 [x_t^* - x_i^t] \qquad (1.7)$$

Here represents the general formulation of velocity for PSO while the next position for each individual calculates via formula below.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \qquad (1.8)$$

where:

g^* is global

x_t^* is individual best location

v_i^t is initial velocity

ε_1 , ε_2 are random vectors in interval [0, 1]

x_i^t is position of individual i in current position.

α , β are acceleration constant where $\alpha \sim \beta \sim 2$.

1.1.2.4. Hybrid Sine Cosine Whale Optimization Algorithm (MVSCWOA)

SCWOA is a hybrid model that balances both by evaluating the WOA's strong search capacity and by considering SCA's good exploits [72] and has already been put into the literature. In this work Alizada [29] has built a three-way hybrid employing the Multi-

Verse Optimizer features to further enhance the current hybrid model. SCWOA, the combination of SCA and WOA already identified in the literature, is the reason for a search and operations algorithm. The study substituted some of the characteristics of MVO [73] with those of SCWOA. The study employed two characteristics. This is the Wormhole Existence Probability (WEP) and Traveling Distance Rate (TDR). WEP is utilized in place of the parameter that sets the SCA updating direction and TDR, that is used to update the inter-element multiplication parameter in WOA, is replaced by a WEP (range from -1 to -2).

1.1.2.5. Artificial Bee Colony (ABC)

The Artificial bee colony is introduced in 2006 by D. Karaboga [12], based on the intelligent behavior of the honeybee swarms. The ABC algorithm includes three types of bees: scouts, employees, and spectators of the artificial bee colony. Scout bees looking for sources of food Random bees with a certain food source and viewing bees are waiting for a choice of food source in the dancing area. Initially, scout bees determine all food supply sites. The nectar of food sources is afterwards used by hired bees and viewers bees. Eventually, continuous intake of nectar will lead to food supplies being exhausted. The bee whose diet is expanded then turns into a scout bee. A feasible solution to the issue is the position of each food source and the amount of nectar from a food source is the quality (fitness) of the related solution. The first half of the colony is the employee in the ABC algorithm [74].

1.1.2.6. Modified Whale Optimization Algorithm (MWOA)

This study proposed a new modified Whale Optimization algorithm (MWOA) by [30], using the elimination mechanism in the Moth Flame Algorithm (MFO) to solve the poor accuracy, slow convergence and getting in the local optimum issues in the original Whale Optimization Algorithm (WOA). This mechanism used to decrease the leaders' population continuously during each iteration process. If the condition satisfied, the population of whale will update according to leader with decreasing mechanism. That will help to jump out from local optimum, as well as increase the probability of obtaining global optimal solution.

1.2. Problem Definition

In this part, we would be explaining the ongoing problems that Numerical Benchmark Problems presented and the combinatorial issues that Traveling Salesman Problem presented.

1.2.1. Benchmark Problems

This section introduces a selected set of common benchmark functions for reliable and efficient testing and validation of optimization algorithms. Ideally, the test functions should be diverse and impartial; however, the literature does not include a standard set of test functions. Therefore, we have used a possible most comprehensive range of testing functions from the literature to compare optimization algorithms.

It is important to check the performance in the relevant list of test functions and to compare it to other algorithms to discover the optimization algorithm efficiency. Many researchers are commonly comparing various algorithms for a wide range of tests, specifically if operational optimization is required for the test. Therefore, we should notice that, if the problem sets are without diverse properties and specialized, the success of algorithms versus other algorithms cannot be measured just by the problems it solves. However, if an algorithm evaluation performs properly or better than other algorithms, the weaknesses must be identified. This allows the types of problems for which an algorithm is designed to solve being characterized [75].

This is reasonable if the tests series is large enough to include a wide range of problems, such as multimodal, unimodal, irregular, regular, not separable, multidimensional problems, and separable. We attempted to employ unimodal, multimodal, and fixed reference aspects in this study. The size of the problem research space is an important factor [76]. In some functions, the global minima are relatively small in comparison to the total search area, or the global minimum is very fairly close to the local minima (Kowalik). Similar to multimodal functions, if the search space cannot be effectively searched by the algorithm and cannot trace changes in direction in functions with a narrow-curved valley (Rosenbrock), it would fail in problems like that.

A function is said to be multi-modal if it has multiple local optimums. Multi-modal features are used to assess algorithms' capacity to avoid local minima. If an algorithm's exploration phase is so inefficient that it cannot search the entire space effectively, the

algorithm will fall in the local minima. The benchmark functions are explained below, which are used for testing optimization algorithms in this study.

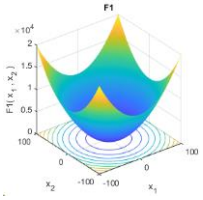
1.2.1.1 Unimodal Functions (F1-F7)

Since unimodal test functions contain a single global optimum, they are appropriate for benchmarking algorithms' local exploitation capabilities. This type of functionality will allow us to concentrate on the convergence rates of the examined algorithms rather than the final results [77].

Sphere Function (F1)

The Sphere function is simple which is used for demonstration purposes. The sphere function is considering continuous, differentiable, separable, scalable, and unimodal. This function is used as a separable component of this test suite while a normally no separable function is used to construct any partly non-separable functions [78] as shown in Table (1.1).

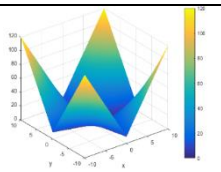
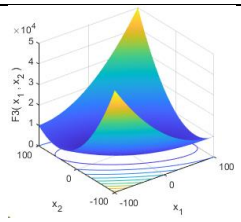
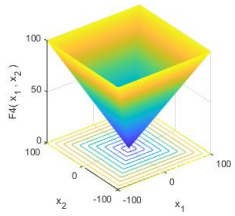
Table 1.1. Sphere Function (F1) unimodal benchmark functions used in experiments. Dim: dimension, f min: optimal solution, range: upper and lower bounds.

Function	Name	Bound	f_{Opt}	Shape
$\sum_{i=1}^{n} x_i^2$	Sphere	[-100,100]	0	

Schwefel Functions (F2, F3, F4)

In this segment, we used three versions of the Schwefel mechanism (Schwefel2.22 (F2), Schwefel1.2 (F3), and Schwefel2.21 (F4) from the unimodal family, as well as Schwefel (F8) from the multimodal family, which we used in the following section. The Schwefel function, whose global minimum value is 0. The function's initialization set is [10, 10] for F2 and [-100,100] for F3 and F4 [79]. Schwefel Functions details illustrated in Table 1.2.

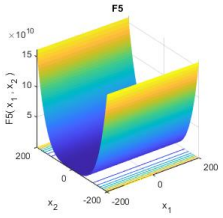
Table 1.2. Schwefel Functions (F2, F3, F4) Unimodal benchmark functions used in experiments. Dim: dimension, f min: optimal solution, range: upper and lower bounds.

Function	Name	Bound	f_{Opt}	Shape
$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	Schwefel2.22	[-10,10]	0	
$\sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	Schwefel1.	[-100,100]	0	
$\max_i \{\lvert x_i \rvert,\ 1 \le i \le n\}$	Schwefel2.21	[-100,100]	0	

Rosenbrock Function (F5)

At its global minimum, the Rosenbrock function value is 0. The function's initialization size is [30, 30]. The global optimum is located inside a long, narrow flat valley with a parabolic shape. Since convergence of the global optimum is complicated, the variables are heavily dependent, and the gradients usually do not point towards the optimum, this problem is often used to measure the efficiency of optimization algorithms [80]. Table 1.3. shows Rosenbrock Function (F5) characteristic.

Table 1.3. Rosenbrock Function (F5) Unimodal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds.

Function	Name	Bound	f_{Opt}	Shape
$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	Rosenbrock	[-30,30]	0	

Step Function (F6)

The Step Function is non-differentiable, discontinuous, separable, scalable, and unimodal as shown in Table (1.4).

Quartic Function (F7)

The quartic function is continuous, separate, differentiable, and scalable. quartic function (F7) details shown in Table (1.5).

Table 1.4. Step Function (F6) Unimodal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds.

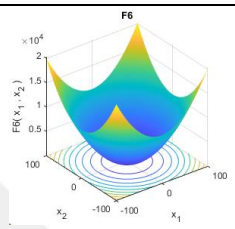
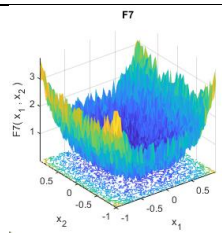
Function	Name	Bound	f_{Opt}	Shape
$\sum_{i=1}^{n}([x_i+0.5])^2$	Step	[-100,100]	0	

Table 1.5. Quartic Function (F7) Unimodal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds.

Function	**Name**	**Bound**	f_{Opt}	**Shape**
$\sum_{i=1}^{n} ix_i^4 + random[0,1)$	Quartic	[-1.28,1.28]	0	

1.2.1.2. Multi-Modal Functions (F8-F13)

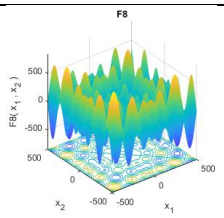
Multimodal test functions feature various local minima, and the number of local optima generally increases exponentially with issue dimension, making them appropriate for testing algorithms' global exploration capabilities. This aspect of multimodal function

can test the algorithm's exploration skills to the test, allowing it to escape from local optima [77].

Schwefel Function (F8)

Schwefel function templates were used. The sulfur equation with a value of 0 (-12095.733, -12095. 733, ... -12095.733). The function's initialization size is [−500, 500]. The Schwefel surface feature consists of several peaks and valleys. The function has a second best minimum, where many search algorithms are stuck further from the global minimum. Moreover, the global minimum is close to the domain boundaries. The Schwefel function's surface contains a large number of peaks and valleys. The function has a second-best minimum that is far from the global minimum, which traps several search algorithms [80]. Furthermore, the global minimum is close to the domain's boundaries as shown in Table (1.6)

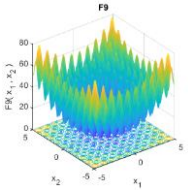
Table 1.6. Schwefel Function (F8) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds.

Function	Name	Bound	f_{Opt}	Shape
$\sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	Schwefel	[-500,500]	-12569.5	

Rastrigin Function (F9)

Table (1.7) display The Rastrigin function, where its value is (0) at its global minimum (0, 0.... 0), is the second function in the multi-model (3.2). The function's initialization set is [-5.12, 5.12]. This function is built on the Sphere function with cosine modulation to generate a large number of local minima. As a result, the feature is multimodal in nature. The minima are distributed in a normal pattern. Seeking optimal solutions to this function is difficult since the optimization algorithm will quickly get stuck in a local optimum on the way to the global optimum [80].

Table 1.7. Rastrigin Function (F9) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bound	f_{Opt}	Shape
$\sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i + 10)]$	Rastrigin	[-5.12,5.12]	0	

Ackley Function (F10)

The Ackley function has a minimum value of 0 (0, 0.... 0). The initialization range for this function is [32, 32]. Ackley has an exponential concept that covers the surface with numerus local minima which shown in Table (1.8).

Griewank Function (F11)

At the global minimum (0, 0.... 0), this problem has a value of zero. The feature range was set to [600,600]. The Griewank feature introduces interdependence within the variable's products to solve the inability of techniques that optimize each variable separately. The optimal solution of the Griewank function is periodically distributed if the number of local optima increases with dimensionality; this function is highly multimodal [81]. Table (1.9) display its feature.

Table 1.8. Ackley Function (F10) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

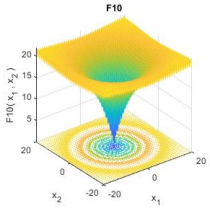
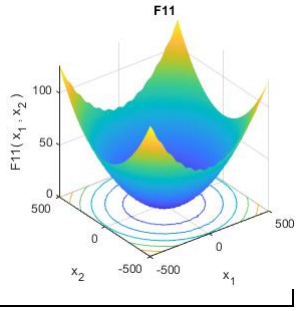
Function	Name	Bound	f_{Opt}	Shape
$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	Ackley	[-32,32]	0	

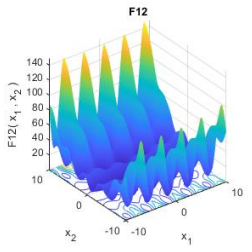
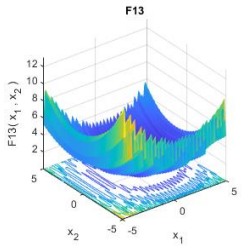
Table 1.9. Griewank Function (F11) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bound	f_{Opt}	Shape
$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Griewank	[-600,600]	0	

Penalized Functions (F12, F13)

The two models Penalized 1 and Penalized 2 were used which are given in Table (1.9).

Table 1.9. Griewank Function (F11) Multi-modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bounds	f_{Opt}	Shape
$\sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $+\frac{\pi}{D}\left\{10\sin^2(3\pi y_i) + \sum_{i=1}^{D-1}(y_i-1)^2[1+\sin^2(3\pi y_{i+1})] + (y_D-1)^2\right\}$ $y_i = 1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m) = \begin{cases} k(x_i-1)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-1)^m, & x_i < -a, \end{cases}$	Penalized 1	[-50,50]	0	
$\sum_{i=1}^{D} u(x_i, 5, 10, 4)$ $+\frac{1}{10}\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + (x_D-1)^2[1+\sin^2(2\pi x_D)]\right\}$ $u(x_i,a,k,m) = \begin{cases} k(x_i-1)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-1)^m, & x_i < -a, \end{cases}$	Penalized 2	[-50,50]	0	

1.2.1.3. Fixed Dimensional Modal Functions (F14-F23)

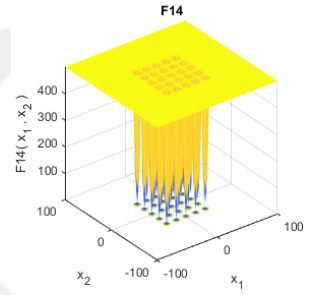
with the same features of multimodal functions. Compared to fixed dimensional modals have multiple local optima, the number of which increases rapidly with the complexity

of the problem size. As a result, if the goal is to evaluate an optimization algorithm's exploration capacity, this type of benchmark problem is quite effective.

Foxholes Function (F14)

In certain articles, foxholes are referred to as the 2-Dimensional DeJong Function. De Jong's function is multimodal, with very steep drops on a mostly flat surface [82].

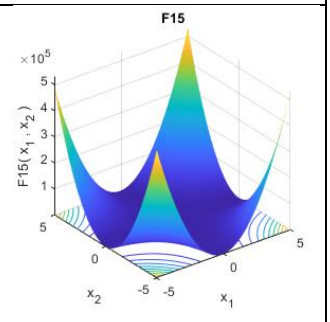
Table 1.10. Foxholes Function (F14) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bounds	f_{Opt}	Shape
$\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	Foxholes	[-65,65]	0.998	

Kowalik Function (F15)

The Kowalik global optimization problem is defined by this class. This is a problem of multimodal minimization as shown in Table (1.11).

Table 1.11. Kowalik Function (F15) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

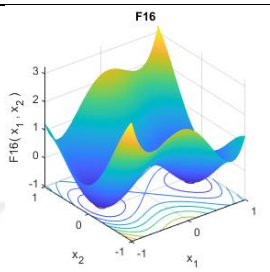
Function	Name	Bounds	f_{Opt}	Shape
$\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	Kowalik	[-5,5]	0.00030 7	

SixHump Function (F16)

The plot on the left illustrates the six-hump Camel function on its recommended input domain, while the plot on the right depicts only a subset of this domain to facilitate

displaying the function's core characteristics. There are six local minima in the function, two of which are global. Table (1.12) displays SixHump Function (F16) feature.

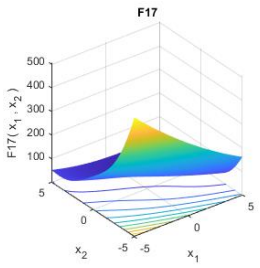
Table 1.12. SixHump Function (F16) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bounds	f_{Opt}	Shape
$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	SHCB	[-5,5]	-1.03163	

Branin Function (F17)

The Branin function shown in Table (1.13), is a two-variable global optimization test function. The function has three equal-sized global optima, with the following proposed values for a = 1, b = 5.1 / (4π2), c = 5 / π, r = 6, s = 10 and t = 1 / (8π).

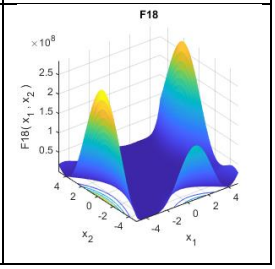
Table 1.13. Branin Function (F17) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bound	f_{Opt}	Shape
$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	Branin	[-5,5]	0.398	

GoldsteinPrice Function (F18)

There are many local minima in the Goldstein-Price function. It's also Continuous, Differentiable, Non-separable, Non-scalable, and Multimodal as shown in Table (1.14).

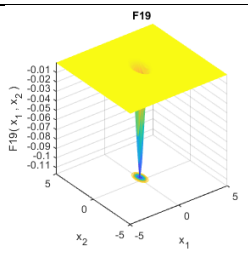
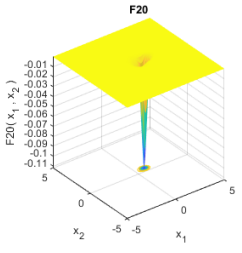
Table 1.14. GoldsteinPrice Function (F18) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bounds	f_{Opt}	Shape
$[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	Goldstein-Price	[-2,2]	3	

Hartman Functions (F19, F20)

Hartmann 3 is a three-dimensional Hartmann function with four local minima. Continuous, differentiable, non-separable, non-scalable, and multimodal Hartman 6 is a 6-dimensional Hartmann function with six local minima. Table (1.15) shows each function (F19, F20).

Table 1.15. Hartman Functions (F19, F20) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

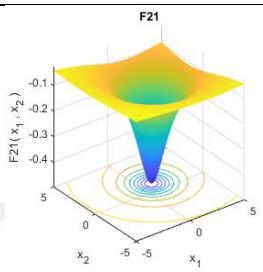
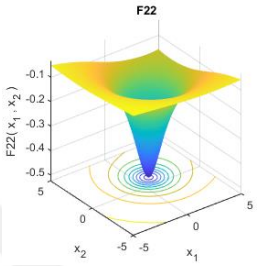
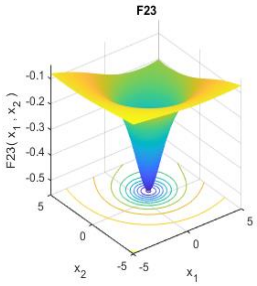
Function	Name	Bounds	f_{Opt}	Shape
$-\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j-p_{ij})^2)$	Hartman3	[1,3]	-3.86	
$-\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2)$	Hartman6	[0,1]	-3.32	

Shekel Functions (F21, F22, F23)

Shekel 5, 7, 10 is Continuous, Differentiable, Non-Separable, Scalable, and Multimodal in all of its 5,7,10 elements. There are several local minima in the Shekel function [83]. The suggested values of m, the β-vector, and the C-matrix are given above; β is an m-

dimensional vector, and C is a 4-by-m-dimensional matrix. All of F21, F22, F23 are shown in Table (1.16).

Table 1.16. Shekel Functions (F21, F22, F23) Fixed dimensional modal benchmark functions used in experiments. Dim: Dimension, f min: optimal solution, Range: Upper and Lower bounds

Function	Name	Bounds	f_{Opt}	Shape
$-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	Shekel5	[0,10]	-10.1532	
$-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	Shekel7	[0,10]	-10.40294	
$-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	Shekel10	[0,10]	-10.53641	

1.3. Travelling Salesman Problem (TSP)

1.3.1. Problem Definition

The TSP is to find the best possible path to visit all the cities and return to the starting point that minimizes the travel cost, given a collection of cities and the cost of travel (or distance) between each possible pair (or travel distance) [84].

Thus, the TSP can be mathematically presented as:

$$\sum_{i=1}^{n} C_{i\pi(i)} \qquad (1.9)$$

The cyclic permutation of integers between 1 and *n* minimizes the sum given the integer $N>=3$ and the matrix $n \times n$ $C = (c_{ij})$, where each c_{ij} is a non-negative integer.

The TSP dates back to the middle of the 18th century when Euler first described the problem. In a German book published by an accomplished traveling salesman in 1932, the word "Traveling Salesman" was first used. RAND Corporation launched TSP in 1948. TSP has become a well-known and influential topic thanks to the organization's prestige. TSP grew in popularity due to its recent linear programming subject and aimed to solve combinatorial problems [84]. The Traveling Salesman Problem has dominated the minds of many scholars over the years.

To begin with, TSP is simple to recognize but extremely difficult to solve. There is no proven solution to a polynomial algorithm. The NP-complete class of problems, of which the TSP is a classic example, is characterized by the lack of any polynomial algorithms. The complexity class of decision problems that can be verified in polynomial time is known as NP (Non-deterministic-polynomial Time). The question of whether or not there is a solution to the dilemma is not addressed here. It is not feasible to solve these problem types in polynomial time because the resolution time in each step is longer than the resolution time in the previous step, making them NP-complete.

On the other hand, TSP can be used to solve a wide range of routing and scheduling issues. Since there is as much knowledge about TSP is already understood, it has become a kind of "testing" problem; new combinatorial optimization methods are often added to the TSP to understand their utility [84]. The following section introduces some of the well-known techniques that are proposed to solve the TSP.

Many researchers have changed the standard TSP model in order to apply it efficiently to a wide range of real-world problems. TSP is classified into three types: symmetric TSP (sTSP), asymmetric TSP (aTSP), and multiple TSP (mTSP). These fundamental TSP variations are as follows: [84].

- sTSP: Let d_{ij} be the Euclidean interval between cities i and j. If $d_{ij} = d_{ji}$, the tour can be tested for the same cost in both directions [80]. The sTSP is the problem of searching for a short, closed tour that tours each city only once [85].

- aTSP: TSP becomes an aTSP if $d_{ij} \neq d_{ji}$ for at least one edge (i,j) [85].
- mTSP: The assumption in the basic model for restricting visits to exactly one for each city can also be omitted; The multi-visit TSP is known as mTSP [86].

1.3.2. Heuristic and Meta-Heuristic Algorithms Employed to Comparison on Solving symmetric travelling salesman

1.3.2.1. Exact Solvers approaches

Exact Solver's main feature is the certainty of finding the optimal solution at the cost of running time and space needs. For example, The Branching and Bounding Approach process indirectly summarizes all applicable solutions by using formulas under which problem integer constraints are reduced [87,88]. In other words, the Branching and Constraining technique splits a problem into a sequence of sub-problems that must be solved. It is a method for solving a series of sub-problems, each of which can have several possible solutions, and the solution chosen for one sub-problem can influence the possible solutions of the subsequent sub-problems. To stop doing a complete estimate on all partial trees, we must first try to find a workable solution.

It is tested, and its value is recorded as an upper bound for optimal. Calculations are performed when the distance approaches the upper limit. If a new, less expensive solution is discovered, its value is used as the new upper bound. This approach is appropriate for ranging from 40 to 60 knots (cities) [87,89].

TSP are often solved by using Branch-and-Bound (BB) algorithms. There are at least three groups independently identified Branch-and-Bound. Dantzig et al. This method applied to the ATSP at first. This important study also presented a number of additional improvements. Land and Doig offered a more general definition in the topic of solving integer programming problems using linear programming. Eventually, little et al. defined and characterized the approach branch and bound in a TSP application [90].

The cutting plane algorithm is another method proposed for the exact solution of the symmetric travelling salesman problem (TSP). Real tours on a typically incomplete road network, which are often non-Hamiltonian, are simply described by an integer linear programming model. For this model, the algorithm generates individual cutting planes

[91]. Where the problem divided into several sub-problems in the branch and bound strategy. It is a system for solving a series of sub problems, which each have various possible solutions, and one sub problem may affect the possible solutions of sub problems where the solution chosen for [92].

Also, The Branch and Cut method proposed as an exact solver for TSP which uses the regular simple algorithm to solve a linear program with no integer constraints. If an optimum solution is reached for a non-integer value for a vector that is supposed to be an integer, a flat intersection algorithm is used [92, 93]. Hong and Miliotis added a primitive version of this concept to the TSP. It was applied to the Linear Order Problem by Grotschel, Junger, and Reinelt [92]. proposed the concept "branch and cut." It was used to decode extremely high TSP cases (up to 2000 cities). As well as Dynamic Programming which presented proposed in Balas (1996) [94], which used to find optimal solutions to problems if the optimal solutions have an optimal substructure, which can be divided into a smaller sub-problem and some cost or profit that is independent of that optimal solution [95].

Brute-force method is one of the best approaches that come to mind while considering solving TSP. The brute-force method involves simply generating all potential tours and calculating their distances. As a result, the shortest tour is the best tour [96].

1.3.2.2. Non-exact Solvers approaches

These solvers have solutions which may not be optimum, but usually quicker. In one way, the same solvers' opposite compensation. non-exact solvers can be classified into:

- Approximation Algorithms
- Heuristic Algorithms

The Approximation Algorithms for the found solution come with a worst-case approximation element. A pure MST-based algorithm, which achieves a factor 2 approximation, and a hybrid MST and Minimum Matching Problem (MMP)-based algorithm, which achieves a factor 3/2 approximation, are the two conventional approaches for solving the TSP. Since they depend on the triangle inequality, both

methods are limited to the MTSP. Due to its prohibitive running time, the PTAS for Euclidean TSP is primarily a theoretical finding.

While Heuristic Algorithms are certain algorithms come with a worst-case approximation factor for the solution that has been discovered. A pure MST based algorithm, which achieves a factor 2 approximation, and a combination MST and Minimum Matching Problem (MMP) based algorithm, which achieves a factor 3/2 approximation, are the two conventional approaches for solving the TSP. Since they both depend on the triangle inequality, both approaches are limited to the MTSP. Because of its prohibitive running time, the PTAS for Euclidean TSP is more a theoretical result [97].

1.3.2.3. Well-Known Algorithms for TSP Solving

- **Christofides' Algorithm**

The Christofides algorithm (named after Nicos Christofides) seeks to solve cases of the traveling salesman problem in which the edge weights fulfill the triangle inequality. Let $G(V, w)$ be a TSP case, i.e., G is a complete graph on the set V of vertices with the weight function w assigning a nonnegative real weight to each edge of G [98].

- **Clarke-Wright Algorithm**

The Clarke-Wright (CW) savings heuristic CW is derived from Clarke and Wright's [1964] more general vehicle routing algorithm [99].

- **Nearest Neighbor**

Since it mimics how a traveling salesman chooses a travel path, this method is a natural technique for the TSP. It chooses a starting point and then adds the closest city to the tour. It then "walks" to that city and continues the process for a new non-selected city before the tour is complete. An edge is inserted between the last chosen city and the starting city to complete the tour [100].

- **k-opt algorithm**

is the most common TSP heuristic method, where k is the number of edge exchanges/re-permutations inside a given neighborhood. Current k-opt techniques, on the other hand, are not adaptive, and the number k is either fixed or explored in a certain sequence. In

fact, the 2-opt and 3-opt algorithms are arguably the most extensively utilized tour improvement methods [101].

- **Insertion Heuristics**

Insertion heuristics are easy to understand, and there are several variations from which to choose. The basic idea behind insertion heuristics is to start with a tour of a subset of all cities, then insert the rest using some kind of heuristic. A triangle or the convex hull is sometimes used as the first sub-tour. As a sub-tour, one can also start with a single tip [102].

- **The Greedy Heuristic**

The greedy heuristic iteratively builds a tour by adding a low-cost edge into a set *T*, as long as the condition to eventually result in a tour is fulfilled. Algorithm describes the structure. Reinelt mentions Frieze's proof that the greedy heuristic is ratio bound by log where the triangular inequality holds (n) [103]. We look at all the arcs coming out of the city (node) and pick the n cheapest arcs to solve TSP using the Greedy Approach. We have an ideal solution if the n cheapest arcs form a Hamiltonian cycle.

- **Hill Climbing (HC)**

Hill climbing is a mathematical optimization strategy that belongs to the local search family in computer science. It's an iterative algorithm that begins with a random solution to a problem and then tries to improve it by updating a single part of the solution incrementally. If the change results in an improved solution, a gradual change is made to the current solution, and the process is repeated until no more changes are possible [104].

- **Simulated Annealing (SA)**

Simulated annealing is a well-known meta-heuristic search technique that was effectively used to solve several problems in combination optimization. It's a hill-climbing algorithm with the additional benefit of being able to escape from local optima in the search space. However, although it produces impressive solutions, it is extremely sluggish as opposed to a straightforward hill climb.

Simulated annealing is a concept borrowed from solid-state annealing, in which we attempt to reduce the system's energy by slowly cooling before the atoms enter a steady

state. Slow cooling causes metal atoms to align to form a normal crystalline structure with high density and low energy. The annealing schedule refers to the starting temperature and the rate at which it is lowered [105].

- **Tabu Search (TS)**

Glover 1977 [106]. Ever since, combinatorial optimization problems have been commonly addressed. Tabu comes from the word 'taboo' which means forbidden or restricted. The focus of the technique is the use of memory during the hunt. Tabu search at the simplest level is similar to immersive development but has additional limitations on which solutions can be visited in the neighborhood of a solution. At the logical level, the constraints of a set of taboo solutions T are implemented. Their decision to examine these depends on the criteria of aspiration is only shifted to where there are valid reasons to do so. The taboo collection is preserved as a mixture of previously attended movements, a set of histories and/or rules which control movements which are true provided the existing solution, its surroundings and history set [107].

- **Ant Colony Optimization (ACO)**

Ant Colony Optimization is a method of meta-heuristic inspired by actual ants' behavior. Dorigo et al. developed its concepts in 1991 [11]. Real ants collaborate to locate food supplies by laying a trail of a chemical substance called 'pheromone' along the nest-to-food path. Depending on the quantity of pheromone on a line, new ants are motivated to take the same path with high probability, resulting in even more pheromones being placed on this path. Shorter food routes have higher pheromone levels. Thus, over time, most ants can use the shortest course. This method of indirect contact is called "stigmergy" [108], using the principle of constructive reinforcement to find the best possible direction, based on previous ants' experience.

- **Genetic Algorithms**

The genetic algorithm is based on the model that John Holland (1975) and his students have proposed and studied [109]. The concept of a GA is to mimic the mechanisms in which the fittest species develop biologically, select naturally, and survive.

Many that are stronger or fitter in genetic properties live to rear their offspring and produce them. Genes with stronger turns will stay in this manner, genes with worse turns are usually gone [110].

2. CHAPTER

A NOVEL MODIFIED TEACHING-LEARNING-BASED OPTIMIZATION ALGORITHM

2.1. TLBO Algorithm

Teaching is an essential aspect where every individual learns many things from different people to develop himself. TLBO algorithm proposed by Rao et al. in 2011 is based on a teacher's impact on students' productivity in a classroom environment [22]. The algorithm has two essential elements: the teacher and the student and two simple learning methods.

At a teaching phase, learners learn from the teacher; during a learning stage, students learn from each other. The TLBO is a population-based approach using a population of solutions to achieve a global solution, like other natural-induced algorithms. In the TLBO, the population is considered as a class of learners or a group of learners. Several variables of the design are identical to the various topics given to the learners; the results of the learners are analogous to the fitness value; as well as the teacher is regarded as the best solution gained up to this point. The TLBO procedure is carried out through the two fundamental procedures of the teacher and learner phases, which are listed below.

2.1.1. Teacher Phase

It is the first phase of the algorithm. Students develop their knowledge with the assistance of the teacher, who is the most experienced person in the class and constantly encourages the students to achieve their goals. Thus, during this phase, a successful teacher is the one who raises his or her learners' education level to his level. However, this is not feasible, and a teacher often can raise a class's mean to a certain limit, based

on the class's capacity. This is a random mechanism that is influenced by a variety of factors. Let M_i be the mean and T_i be the instructor at every iteration i. T_i will attempt to get M_i to its own stage, so T_i will now be designated as M_{new}. The solution is updated based on the difference between the old and new mean as determined by

$$Difference_mean_i = r_i(M_{new} - T_F M_i) \quad (2.1)$$

where M_{new} is the outcome of the best learner in subject j, r_i is the random number between [0,1], and T_F is the teaching factor determining the average number to be updated. Either 1 or 2 is the value of T_F. The T_F value is calculated randomly by the following equation:

$$TF = round\ (1+rand\ (0, 1)) \quad (2.2)$$

According to Eq. (2.3) expression, this difference modifies the current solution.

$$X_{new,i} = x_{old,i} + Difference_mean_i \quad (2.3)$$

$X_{new,i}$ is accepted if it generates the best result.

2.1.2. Learner Phase

Learners develop their knowledge in two ways: through the teacher's feedback and their communications with other learners. When other learners get more experience than the other, they gain new information. Eq. (2.4) express the learning process in this phase:

$$x_{new,i} = \begin{cases} x_{old,i} + ri(x_i - x_j) & if\ f(x_i) > f(x_j) \\ x_{old,i} + ri(x_j - x_i) & otherwise \end{cases} \quad (2.4)$$

where two learners, x_i and x_j, are randomly selected such that $i \neq j$. $X_{new,i}$ is accepted if it generates the best result. As well as $x_{new,i}$ value depends on $f(x_i)$ and $f(x_j)$ which represent the fitness of each learner x_i or x_j as referred before $f(x_i)$ and $f(x_j)$ stand for the learner knowledge and ri is a random number.

(1) **Begin**

(2) Initialize *NP* (number of learners) and *D* (dimension);

(3) Initialize learners and evaluate them;

(4) **while** stopping condition is not met

(5) Choose the best learner as $x_{teacher}$;

(6) Calculate the mean x_{mean} of all learners;

(7) **for** each learner *x*;

(8) // Teacher phase //

(9) T_F = round (1 + rand (0,1));

(10) Update the learner according to Eq. (2.3);

(11) Evaluate the new learner $x_{i,new}$;

(12) Accept $x_{i,new}$ if it is better than the old one $x_{i,old}$;

(13) // Learner phase //

(14) Randomly select another learner x_j which is different from x_i ;

(15) Update the learner according to Eq. (2.4);

(16) Evaluate the new learner $x_{i,new}$;

(17) Accept $x_{i,new}$ if it is better than the old one $x_{i,old}$;

(18) end for

(19) end while

(20) end

Algorithm 1: Pseudo-code of the original TLBO algorithm

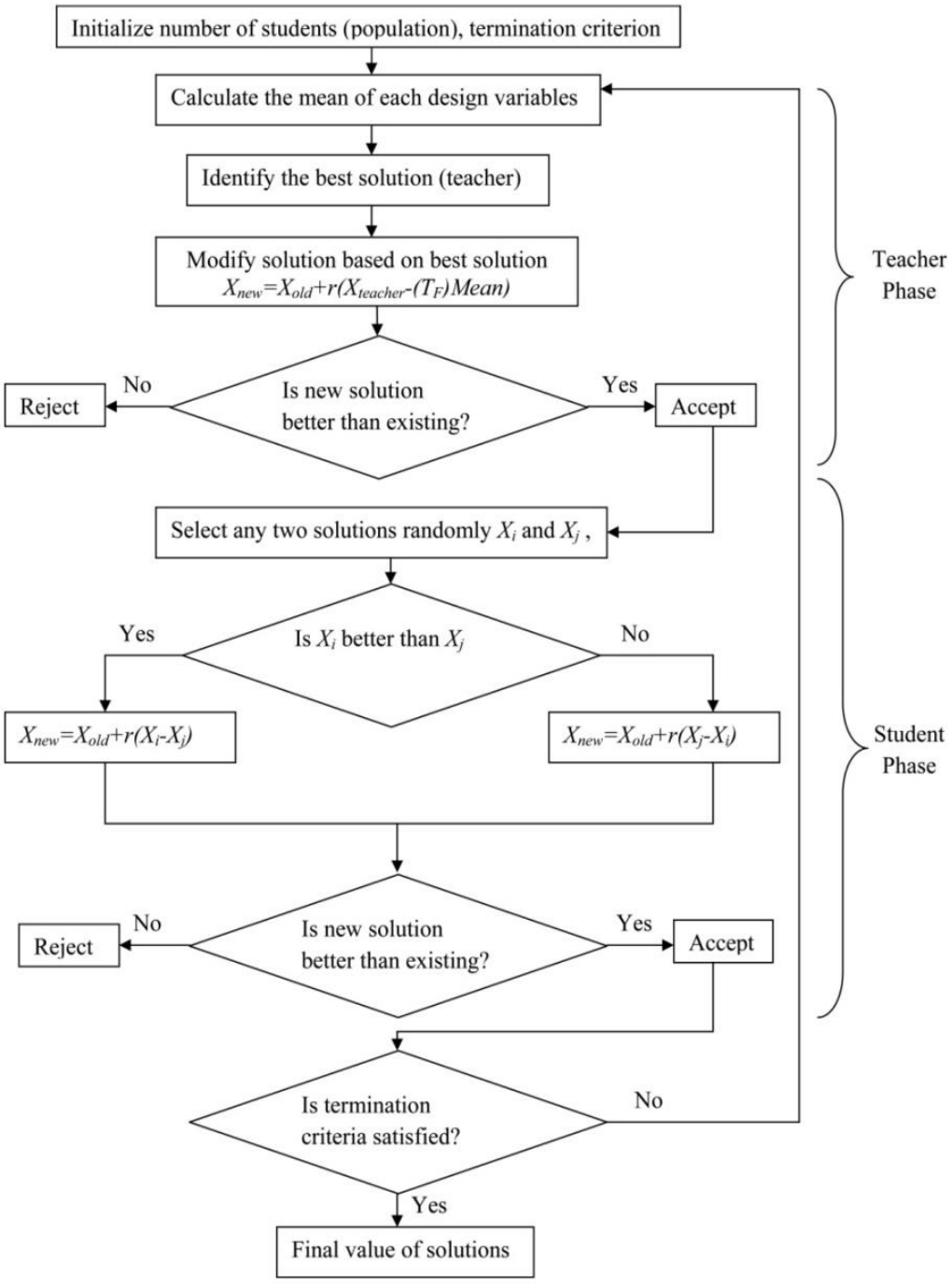


Figure 2.1. TLBO Flowchart

2.2. Modifying the Teaching Learning Based Optimization (TLBO)

Researchers have suggested optimization algorithms to resolve optimization problems in different fields. On some objective functions, optimization algorithms have been effective, but on others, they have failed. Adding modification stages to optimization algorithms and enhancing the optimization process might result in a more appropriate

and acceptable solution. A solution titled a global solution is provided to every optimization problem. Optimization algorithms give a random search-based solution for search space, which is not always a universal solution but is an acceptable solution since it is near an optimal solution. Optimization algorithms name the result the optimal solution. A better optimizing algorithm is hence an optimization algorithm that gives a closer optimum solution than the other algorithm.

In this study, we will propose a new modification on the Teaching Learning Based Optimization Algorithm (TLBO), Which improves the ability of the TLBO to achieve as much as a closer solution to optimal solution.

The TLBO's main idea was indeed to imitate the education process which represented by the Teacher phase and Learner phase as explained in Section 2.1, which we can notice that the (TLBO) is an algorithm with no active search equation where there is one equation that exists in both phases, which supports fast convergence, yet easy to fall into a local optimum to resolve complex global optimization problems. Moreover, as the original TLBO based on producing knowledge between the teacher and learner, and between students, in real life we can notice how it's not easy for a student to pass a certain level in the academic journey, thus comes the idea of our modification by adding a random parameter and a certain condition in both Teacher phase and Learner phase. We have presented this parameter as "Academic Plan".

2.2.1. The Modification Process

Basically, what we have done in both phases, we concentrated on the process of generating the variable solution instead of collecting process of specific solutions convergence to random Teachers; we flip this process by convergence a group of random solutions to a specified group of Teachers, Via ignore the random parameter from the Teacher matrix and multiply the solutions by a random number decreases to 0 with respect to iterations and do the same for the Learner Phase. For example, as shown below in the Teacher phase:

$$Xnew = Academic\ plan. * Xold\ +\ (Xteacher - TF * mean) \qquad (2.6)$$

where “Academic Plan” is a random number decreasing from 1 to 0 multiply by (*ri*), where (*ri*) is a random multiplied by VarSize, as well as VarSize is a result of rand [1-problem Dimension].

For instance, as explained in Chapter 1 Problem Dimension = [10,4,..] for the benchmark problems [sphere, shekl,..].

Furthermore, by applying this equation on the Teacher phase, we change the whole search space, which gives us the opportunity to find a better teacher as well as a better Learner in the Learner phase by adding the same condition as shown below:

$$Xnew = academic_{plan} * Xold + (step) \qquad (2.7)$$

where step is $r_i(x_i - x_j)$ or $r_i(x_j - x_i)$ depends on the fitness of each learner x_i and x_j.

To describe the mechanism of the "Academic Plan" and its role in our proposed algorithm, let us illustrate the search space as shown in Fig 2.2 where it contains three squares, each one representing a search space status.

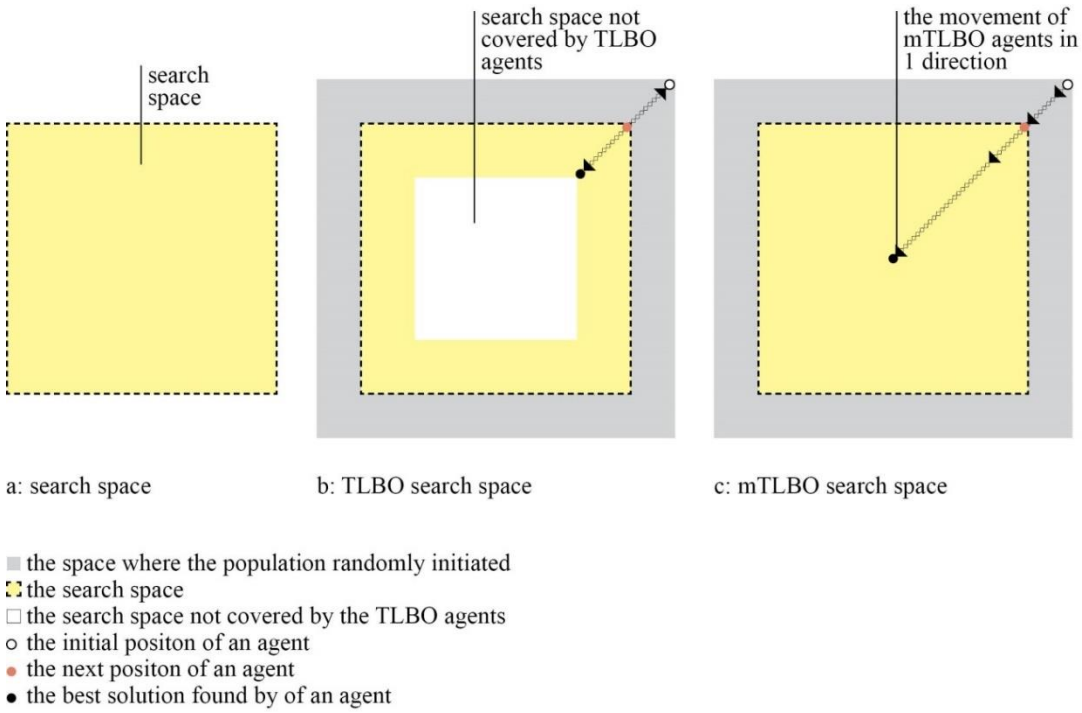


Figure 2.2. The differences between TLBO and mTLBO search space

As displayed in Fig. 2.2, status (a) It is referred to as the yellow region of the search space where a certain agents would be searching for the next position step's best solution and the grey part represents the population's random initial point, as well as the search space of the original TLBO, which shown in Fig. 2.2 status (b) the agents start converge within every iteration seeking the best feasible solution within this yellow part only, where original TLBO don't have the ability to cover this entire search space. leaving the white part out of its cover as the agents travel in opposite directions from the starting position to the final location attained through iterations.

Our proposed modification search where “Academic Plan” allows the algorithm to cover the whole search space looking for the best feasible solution in a one direction movement from the initial point to the last position where its agents obtained the best solution. Our proposed modification's ability to cover the whole search space allowed it to find a better solution in much less time than the original TLBO utilizing the same number of agents and iterations.

In Table 2.1, we compare the computational time of our proposed modification to the original TLBO using the same number of agents and iterations.

Table 2.1. Comparison between the TLBO and mTLBO computational time.

Function	F1 Sphere	F2 Schwefel2.22	F3 Schwefel1.2	F4 Schwefel2.21	F5 Rosenbrock	F6 Step	F7 Quartic
TLBO	1.2358	1.2172	1.2514	1.4451	1.2157	1.2104	1.3007
mTLBO	0.6896	0.7133	0.8606	0.6845	0.6815	0.5954	0.6767

Function	F8 Schwefel	F9 Rastrigin	F10 Ackley	F11 Griewank	F12 Penalized 1	F13 Penalized 2	F14 Foxholes
TLBO	1.1766	1.3801	1.2048	1.2754	1.254	1.2181	1.1703
mTLBO	0.6819	0.6803	0.6433	0.6984	1.0685	1.0338	1.991

Function	F15 Kowalik	F16 SHCB	F17 Branin	F18 Goldstein-Price	F19 Hartman3	F20 Hartman6	F21 Shekel5	F22 Shekel7	F23 Shekel10
TLBO	1.202	1.1719	1.1667	1.1631	1.1724	1.2003	1.1779	1.4349	1.194
mTLBO	0.5992	0.5862	0.5845	0.7022	0.5891	0.616	0.6495	0.6869	0.6924

During our study we have had tried many cases just to certain of our idea of the modification process. Our presented modification labeled under case 1 as explained above.

Table 2.2. Comparison between the 4 cases we studied shows that case 1 got the highest optimal solutions for the tested benchmarks problems.as well as the highest 1st and 2nd close to optimal solutions.

Fn	Opt	Case1	Case 2	Case 3	Case 4
F1 Sphere	0	0	0	2.19E-305	9.30E-165
F2 Schwefel2.22	0	0	1.99E-188	1.03E-154	1.80E-86
F3 Schwefel1.2	0	0	0	3.67E-179	1.33E-108
F4 Schwefel2.21	0	0	3.22E-176	3.36E-129	6.18E-72
F5 Rosenbrock	0	0.067892	0.0035715	0.0064396	3.98E-09
F6 Step	0	0	1.23E-32	0	8.94E-32
F7 Quartic	0	0.0002245	1.14E-05	1.42E-05	4.63E-05
F8 Schwefel	-12569.5	-3716.0755	-3301.5305	-3677.5133	-2282.679
F9 Rastrigin	0	0	0	0.62158	0
F10 Ackley	0	4.44E-15	8.88E-16	4.44E-15	4.44E-15
F11 Griewank	0	0	0	0	0
F12 Penalized 1	0	4.71E-32	4.71E-32	4.81E-32	5.58E-32
F13 Penalized 2	0	1.35E-32	0.021024	0.010987	1.38E-31
F14 Foxholes	0.998	0.998	0.998	0.998	0.998
F15 Kowalik	0.000307	0.0003075	0.0003075	0.0003075	0.00030749
F16 SHCB	-1.03163	-1.0316	-1.0316	-1.0316	-1.0316
F17 Branin	0.398	0.39789	0.39789	0.39789	0.39789
F18 Goldstein-Price	3	3	3	3	3
F19 Hartman3	-3.86	-3.8628	-3.8628	-3.8628	-3.8628
F20 Hartman6	-3.32	-3.322	-3.322	-3.2031	-3.322
F21 Shekel5	-10.1532	-10.1532	-5.0552	-10.1532	-5.0552
F22 Shekel7	-10.40294	-10.4029	-10.4029	-5.0877	-5.0877
F23 Shekel10	-10.53641	-10.5364	-10.5364	-10.5364	-10.5364

Optimal solution | Near to optimal | Next near to the optimal

Case 2 we used the same equation 2.6 but we replaced the “Academic Plan” value to be a random between [1-2]. In case 3 we changed the “Academic Plan” value to be a random number between [1- problem dimension]. Case 4 we replaced the T.F value to be a random number between [1-2]. Table 2.2. shows the results of all the 4 cases by applying each case on the benchmark problems and the results shows that case 1 have the most optimal solution among the 4 cases.

(1) **Begin**

(2) Initialize *NP* (number of learners) and *D* (dimension);

(3) Initialize learners and evaluate them;

(4) **while** stopping condition is not met

(5) Choose the best learner as $x_{teacher}$;

(6) Calculate the mean x_{mean} of all learners;

(7) **for** each learner *x*;

(8) // Teacher phase //

(9) T_F = round(1 + rand(0,1));

(10) If P<0.5

(11) Update the learner according to Eq. (2.6);

(12) Else

(13) Update the learner according to Eq. (2.3);

(15) Accept $x_{i,new}$ if it is better than the old one $x_{i,old}$;

(16) // Learner phase //

(17) Randomly select another learner x_j which is different from x_i ;

(18) If P<0.5

(19) Update the learner according to Eq. (2.6);

(20) Else

(21) Update the learner according to Eq. (2.4);

(22) Evaluate the new learner $x_{i,new}$;

(23) Accept $x_{i,new}$ if it is better than the old one $x_{i,old}$;

(24) end for

(25) end while

(24) end

Algorithm 2: Pseudo-code of the modified TLBO algorithm

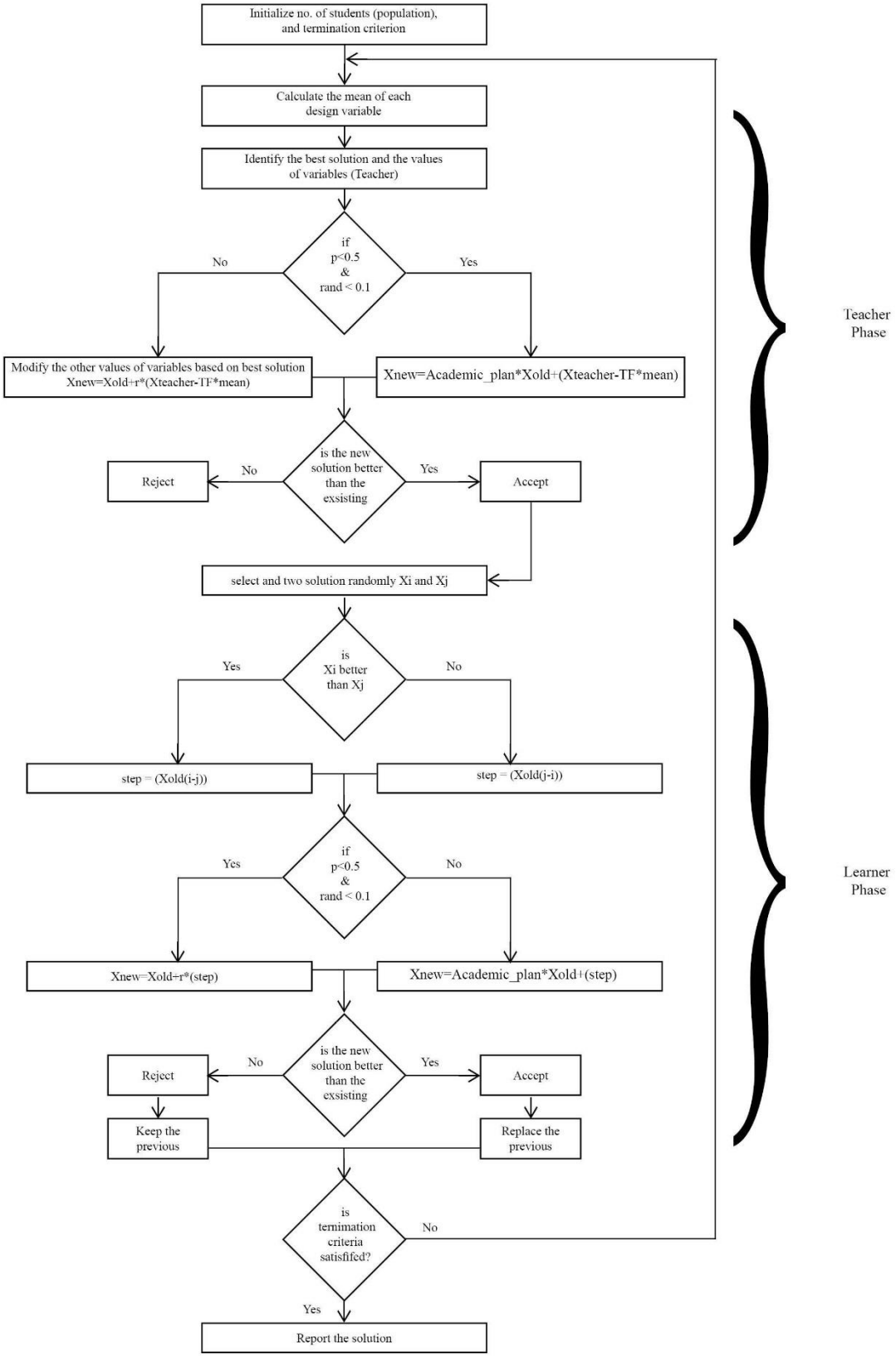


Figure 2.3. New Modified TLBO Flowchart

2.3. Experimental Results of The Proposed Modified TLBO Algorithm

In this section, we will examine unimodal, multimodal, and fixed-model Benchmark functions. We applied our proposed algorithm on seven functions to introduce the unimodal and multimodal also presented by seven functions however, fixed model benchmark functions introduced by nine functions. Thus, testing the algorithm capability to select a solution done using the unimodal benchmarks and testing the algorithm searching skills for the best solution using multimodal benchmarks.

In our case study, we used 13 algorithms to compare their results with our proposed modification. Including the results of the original TLBO, all algorithms used the same control parameters, 1000 iteration, population number were set to 30, and we took the average of 30 independent runs for each benchmark problem.

The algorithms were implemented in MATLAB 2021 programming language, were executed on the Windows 10 operating system. We used an Intel core I7 processor of 7500 GHz with 8 GB of useable RAM computer.

2.3.1. Results Analysis and Simulation

Different benchmark problems are used to test the performance of the meta-heuristic algorithm modifying process, which can be classified into three categories [77]. The functions listed in Chapter 1, Griewank, Rastrigin, Schwefel, Rosenbrock, Schwefel, Quartic, and Step functions are unimodal and high dimensional functions. Schwefel Schwefel, Ackley, Sphere, Penalized, Goldstein-Price, and Penalized are high-dimensional multimodal functions. The Kowalik, Foxholes, Six Hump Camel Back, Branin, Hartman family, and Shekel family functions are multimodal and low dimensional. As previously stated in Chapter 1.

We will show the implementation results of the new modification of TLBO on Benchmarks problems, and we compared the results with several meta-heuristic algorithms taken from the literature. All the experiments in this section were repeated 30 times, and the average best value results were listed. In order to justify the comparison process, we took the absolute optimal value, even the closest value to the

optimal not considered an optimal solution, as well as the difference, were very small such as 0.00001.

Hence, we need to prove the performance of the modification we did on the TLBO; we took 13 algorithms with different categorized as hybrids, modified and original algorithms from the literature randomly throw out the years, as much as there was a significant development on some of the original algorithm we have chosen. Still, it shows a weak performance like the GA, as shown in Table 2.3.

From the results in Table 2.3, we can notice the progress of our modification among all the other algorithms by 11 optimal solutions distributed between the Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21, Step, Rastrigin, Griewank, Foxholes, Kowalik, Goldstein-Price, and Shekel5 functions. This achievement is a result by the balance of exploitation and exploration phases in the algorithm. In general, it was observed that it converges better than the other algorithms. For example, while getting the 1st close to the optimal solution in Ackley, Penalized 1, Penalized 2, and Shekel7. As well as we attempt the 2nd close to the optimal solution in SHCB, Hartman3, Hartman6, and Shekel10.

The MWOA comes after our modification by achieving eight optimal solutions and 1st close to the optimal solution in 7 functions and 4 functions, getting the 2nd close to the optimal solution. None of the algorithms got the optimal solution for the Rosenbrock problem, although our algorithm got the 2nd close to optimal while the original TLBO got the 1st close to the optimal solution. As well as in the Schwefel function where only the ABC got the 1st close to the optimal solution. Hybrid Sheffield Frog Leaping Algorithm (HSFLA) is the only algorithm got the optimal solution in Branin function. The Ackley shows equal performance between our algorithm and the original TLBO, while the Hartman3 function shows equal performance between our algorithm and the original TLBO and ABC. Among the 23 benchmarks, we have used our algorithm is better in 9 functions than TLBO by seven optimal solutions added to the optimal solution the original TLBO already got, which shows the obvious improvement with the TLBO.

Table 2. 3. Shows the average results of 30 time repeated runs on the Unimodal functions benchmark problems used to compare between different meta-heuristic algorithm

Unimodal functions														
Fn	Opti	NM. TLBA	TLBA	ABC	PSO	MFO	BA	GA	WOA	FA	HSFLA	MVSCWOA	MWOA	HSCALO
F1 Sphere	0	0	9.89E-221	7.57E–04	5.68E-01	1.68E+03	1.53E-01	0.118842	1.41E−30	1.72e – 10	8.45E-19	3.60E-65	0	0
F2 Schwefel2.22	0	0	1.13E-109	8.95E–04	2.19E+00	2.58E+01	1.84E-01	0.145224	1.06E−21	6.01e – 07	1.84E-11	1.97E-34	0	0
F3 Schwefel1.2	0	0	9.98E-98	7.01E–04	1.73E+03	2.34E+04	-	0.13902	5.39E−07	1.58e – 10	-	1.64E-55	0	0
F4 Schwefel2.21	0	0	4.17E-92	2.72E+00	3.83E+00	6.89E+01	-	0.157951	0.072581	5.913 – 03	-	1.63E-32	0	0
F5 Rosenbrock	0	0.067892	0.0018189	9.36E–01	1.36E+02	5.36E+06	0.334077	0.714157	27.86558	2.383765	37.007	27.91003	5.4475	11.215807
F6 Step	0	0	1.54E-32	0.00E+00	6.51E-01	1.03E+03	-	0.167918	3.116266	1.9e – 10	-	2.56973	0.000141	0.072589
F7 Quartic	0	0.0002245	0.0001504	9.06E–02	6.56E-01	4.47E+00	0.137483	0.010073	0.001425	1.57e – 04	0.009	0.00033	0.00011	5.91E-05

Optimal Solution

1st close solution to optimal solution

2nd close solution to optimal solution

Table 2. 4. Shows the average results of 30 time repeated runs on the Multimodal functions benchmark problems used to compare between different meta-heuristic algorithm

Multimodal functions														
Fn	Opti	NM. TLBA	TLBA	ABC	PSO	MFO	BA	GA	WOA	FA	HSFLA	MSCWOA	MWOA	HSCALO
F8 Schwefel	-12569.5	-3716.0755	-3558.1442	-12563.673	-3.67E+02	-8.60E+03	-	-2091.64	−5080.76	− 3566.45241	-	-5167.957	-12095.733	-9842.76
F9 Rastrigin	0	0	1.99	4.66E–04	4.23E+01	1.47E+02	1.233748	0.659271	0	7.462188	73.162	0	0	0
F10 Ackley	0	4.44E-15	4.44E-15	7.81E–04	5.81E+00	1.67E+01	0.129359	0.956111	7.4043	8.47e – 07	0.191	8.88E-16	8.88E-16	8.88E-16
F11 Griewank	0	0	0.012316	8.37E–04	2.84E+02	1.90E+01	1.451575	0.487809	0.000289	0.053309	0.008	5.00E-16	0	0
F12 Penalized 1	0	4.71E-32	4.69E-30	6.98E–04	2.29E+00	8.53E+06	-	0.110769	0.339676	1.92e – 12	-	0.191217	1.12E-17	0.000665
F13 Penalized 2	0	1.35E-32	1.84E-32	7.98E–04	1.57E+01	4.64E+01	0.386631	1.29E+01	1.889015	8.21e – 12	2.74E-06	1.934423	1.16E-16	0.010241
F14 Foxholes	0.998	0.998	0.998	9.98E–01	4.30E+02	1.56E+00	4.84	-	2.111973	0.998	0.998	5.718113	0.998	2.125267

Optimal Solution

1st close solution to optimal solution

2nd close solution to optimal solution

Table 2. 5. Shows the average results of 30 time repeated runs on the Fixed-modal functions benchmark problems used to compare between different meta-heuristic algorithm

Fixed-modal functions														
Fn	Opti	NM. TLBA	TLBA	ABC	PSO	MFO	BA	GA	WOA	FA	HSFLA	MSCWOA	MWOA	HSCALO
F15 Kowalik	0.000307	0.0003075	0.0003228	1.18E–03	5.74E-02	1.16E-03	5.15E-03	-	0.000572	3.77e – 04	0.0003	0.000593	0.0003447	0.00035
F16 SHCB	-1.03163	-1.0316	-1.0316	-1.031	-1.02E+00	-1.03E+00	-1.03E	-	−1.03163	− 1.031628	-1.0316	-1.0316	-1.0316	-1.03086
F17 Branin	0.398	0.39789	0.39789	0.3985	3.86E+00	3.98E-01	0.398	-	0.397914	3	0.398	0.397949	0.398048	0.400891
F18 Goldstein-Price	3	3	3	3.00E+00	3.00E+00	3.00E+00	-	-	3	− 3.862782	-	3.000453	3.000103	3.007433
F19 Hartman3	-3.86	-3.8628	-3.8628	-3.862	-3.86E+00	-3.86E+00	-3.84	-	−3.85616	− 3.259273	-2.233	-0.30048	-3.84476	-0.30048
F20 Hartman6	-3.32	-3.322	-3.322	-3.32E+00	-2.60E+00	-3.21E+00	-2.84	-	−2.98105	− 9.316829	-3.28	-3.23337	-3.23572	-3.202093
F21 Shekel5	-10.1532	-10.1532	-10.1532	-10.151	-5.05E+00	-6.47E+00	-4.6	-	−7.04918	– 10.147907	-5.8217	-8.22838	-10.1532	-5.052793
F22 Shekel7	-10.40294	-10.4029	-10.4029	-10.402	-6.39E+00	-8.14E+00	-5.51	-	−8.18178	− 9.398946	-7.1538	-8.34166	-10.4029	-5.132143
F23 Shekel10	-10.53641	-10.5364	-10.5364	-10.535	-4.71E+00	-7.64E+00	-6.1	-	−9.34238	− 10.2809	-6.5341	-8.677407	-10.3561	-5.022336

Optimal Solution

1st close solution to optimal solution

2nd close solution to optimal solution

3. CHAPTER

A MODIFIED TEACHING LEARNING BASED OPTIMIZATION ALGORITHM FOR SYMMETRIC TRAVELING SALESMAN PROBLEM

This section presents a new method for solving symmetric TSP using a modified TLBO algorithm that cooperated with the 2-opt heuristic algorithm. We apply the proposed method to 14 different instances of the traveling salesman problem from TSPLIB and compare the results with other algorithms in the literature.

3.1. Teaching Learning Based Optimization Algorithm

We gave detailed information about the original TLBO in Section 2.1. Because of the similarity between the teacher phase and the learner phase, the only difference was the new position equation, whereas the new position equation was replaced with the hamming distance function. Therefore, the learner phase lost its importance, and for that reason, we ignored it and settled for the teacher phase only.

3.2. 2-Opt Algorithm

The fundamental purpose of TSP is to discover the shortest closed tour with the best route that visits all cities precisely once in a given list. The nearest neighbor, greedy, insertion heuristics, and the Christofides technique are examples to tour creation methods. After a tour is created by using tour construction methods, it can then be improved by utilizing heuristic tour improvement algorithms [111]. Such as 2-Opt [112], 3-Opt [113, 114], k-Opt [112], Lin-Kernighan [114], Tabu-Search [107], Simulated Annealing [104], Genetic Algorithm [110], etc. For this purpose, the 2-Opt technique, a well-known method proposed by Croes [115] as a basic local search algorithm, is used in this work. The 2-opt algorithm joins the two new sub-rounds

formed by removing two edges from the round. This is referred to as a "two-option" strategy. For the round to stay acceptable, there is only one technique to connect two sub rounds. Only as long as the next lap is shorter are the steps repeated. The best path (minimal local) is found by removing and rejoining the trip [112]. The functioning of 2-opt is shown in Fig 3.1.

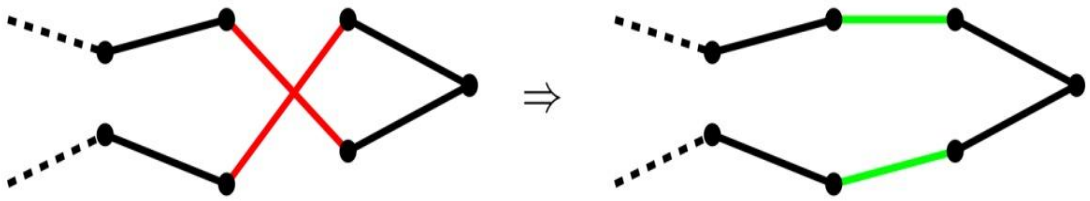


Figure 3.1. 2-opt main idea

Input: A complete graph with distances defined on the edges and a route and its distance
Output: A 2-opt optimal route
1: **repeat**
2: **for** *i* ∈ Nodes eligible to be swapped **do**
3: **for** *j* ∈ Nodes eligible to be swapped such that *j* = *i+1* **do**
4: Apply 2-opt swap to *i* and *j*: create the new route as follows:
5: Take route upto *i* and add in order
6: Take route from *j* to *i* (both including) and add in reverse order
7: Take the route after *k* and add in order
8: Calculate new distance
9: if new distance < distance then
10: Update route to include new ordering
11: **end if**
12: **end for**
13: **end for**
14: **until** no improvement is made

Algorithm 3: Pseudo-code of the 2-OPT algorithm

3.3. A Modified Teaching Learning Based Optimization Algorithm for Symmetric Traveling Salesman Problem

In this study, the Traveling Salesman Problem is solved using a modified Teaching Learning Based Optimization (TLBO) algorithm. Overall, there are two categories of optimization problems: continuous and discrete [112, 116]. TLBO was designed to solve continuous problems, such as mathematical comparison problems, at first. As a result, the TLBO must be adjusted to meet the TSP issue at first. Some portions of the TLBO were changed, and additional strategies were introduced to the algorithm and adapted to discrete problem solutions. Therefore, the proposed algorithm will be referred to as the Modified Discrete TLBO to solve Symmetric TSP (MDTLBO-TSP). In the algorithm, each teacher's position (solution) matches one tour in the TSP problem. Let's assume that a teacher's solution could be (city2 => city50 => city15 => city3...). First, the initial population is randomly created in which each solution is considered a tour that visits every city exactly once and returns to starting city. After the initial population is created, the Teacher Phase starts a comparison between the paths to find the highest fitness among them and choose the best solution as the Teacher.

After we get the best solution (Teacher) in this stage, we get the Teacher to be compared with other solutions and find the difference between the Teacher and the other solution using the Hamming distance function as shown in Fig (3.2) below:

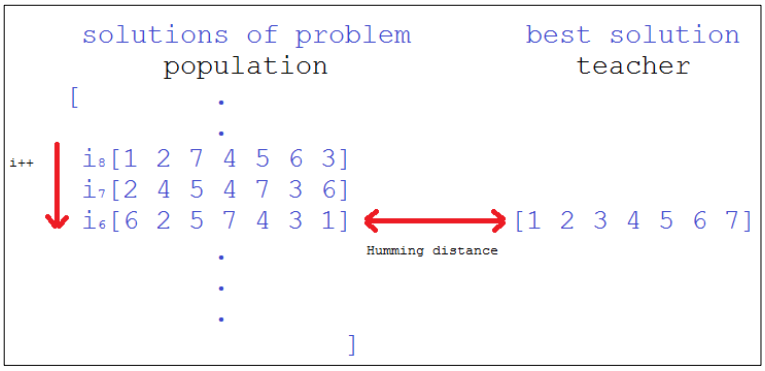


Figure 3.2. Hamming distance

For example, let's assume x_1 is the Teacher and x_2 is one of the other solutions in the presented population:

x_1 : [0,1,2,3,4,5,6,7]

x_2 : [0,1,3,2,5,4,6,7]

Hamming distance between x_1 and x_2 is 4.

When the hamming distance is calculated, the 2-Opt function starts its mission by keeping on swapping edges, as much as the Hamming distance value were, the 2-Opt keep on working until it produces the best fitness proposed for x_2.

The new form of x_2 sends to be compared with its previous form before the 2-OPT; if the new form has better fitness, it is kept. Otherwise, it will be ignored and stick to the original form of x_2. This process continues on the paths within the population until the whole population (matrix) is updated. Then, in the final form of the updated population, each path will be compared with the Teacher; if we find a path that has better fitness than the Teacher, it will then be considered the new Teacher, representing the best-found solution for this specific TSP problem.

3.4. Experimental Results

In this section, we made a comparison. Nine algorithms: Hill Climbing (HC) [117], Simulated Annealing (SA) [117], Tabu Search (TS), Kangaroo Search (KA), Artificial Bee Colony (ABC) [118, 119], Genetic Algorithm (GA) [117, 120], Ant Colony Optimization (ACO) [121], and District Crow Search Algorithm (DCSA) [122] are compared. Each algorithm was run 30 times. The number of populations was adjusted to 50, and the number of iterations to 1000.

Table 5.1 shows the average values of the 30 independent runs for each TSP instance. As can be seen from the Table, the Modified Discrete TLBO (MDTLBO-TSP) provides better results than the other algorithms. In the problems gr17, gr21, gr24, fr26, and bays29, MDTLBO-TSP achieved the optimum results with other algorithms. None of the algorithms could achieve the optimum in the, gr48, eil51, st70, eil76, and pr76 problems, but the other TLBO-TSP got the closest results to the optimum, and TLBO-TSP got the second closest value to the optimum in the swiss42, hk48, and berlin52 issues. Only the ABC (2018) algorithm came close to achieving the best result in Problems swiss42, eil51, berlin52, st70, and eil76.

The ACO algorithm, which was one of the algorithms we compared in Table 3.1., had a 40% success rate by having the best outcomes in the first 6 of 14 tasks. The TLBO-TSP developed generally achieved the best result in 5 out of 14 problems, was closest to the

best result in 6 of them and had 74 percent success with 11 first places in total, and was second best in 3 of them compared to other algorithms. It also had high performance and competitive results when compared to other well-known algorithms. When we look at the order of success in the Table, it is obvious that the (MDTLBO-TSP) proposed in the thesis is first in the ranking because it achieves the best results or results closest to the optimum, DCSA is second, ACO is third, GA is fourth, and SA and HC are last. Fig. 3.3. and Fig. 3.4. display all nodes and edges possible tours to the salesman to visit all the cities and return to first city where the best tour obtained by our proposed algorithm applied on (bays29) TSP benchmark function which its minimal tour length is 2020 and contain of 29 cities in Fig. 3.3. as well as Fig. 3.4. show the first close to optimal tour acquired by MDTLBO-TSP applied on (eil51) TSP benchmark function whereby our proposed algorithm hit 431.3746 while its minimal tour length is 426 and contain of 51 cities.

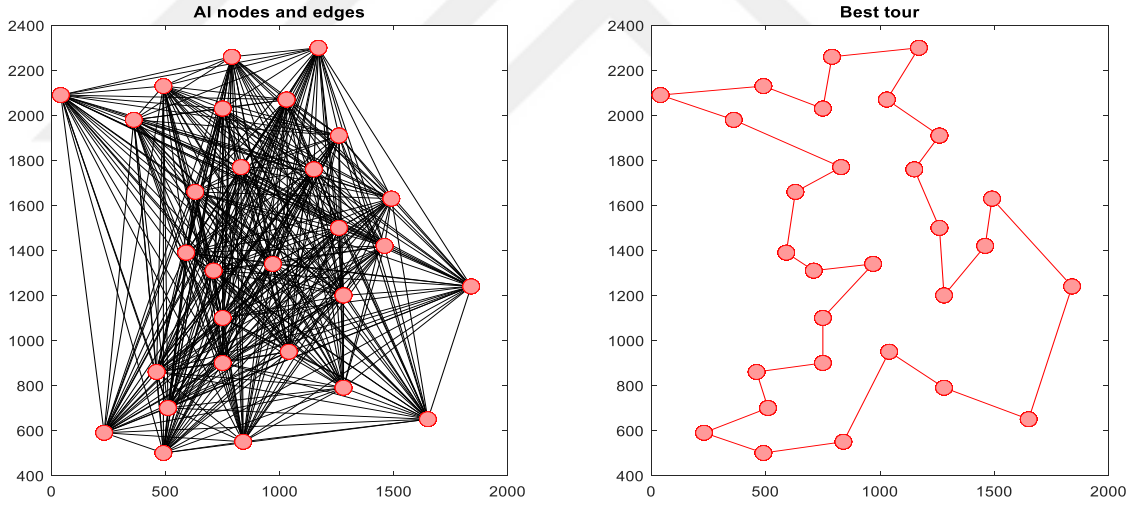


Figure 3. 3. The optimal tour gained by MDTLBO-TSP applied on (bays29) which is a set of 29 cities, from TSPLIB. The minimal tour has length 2020

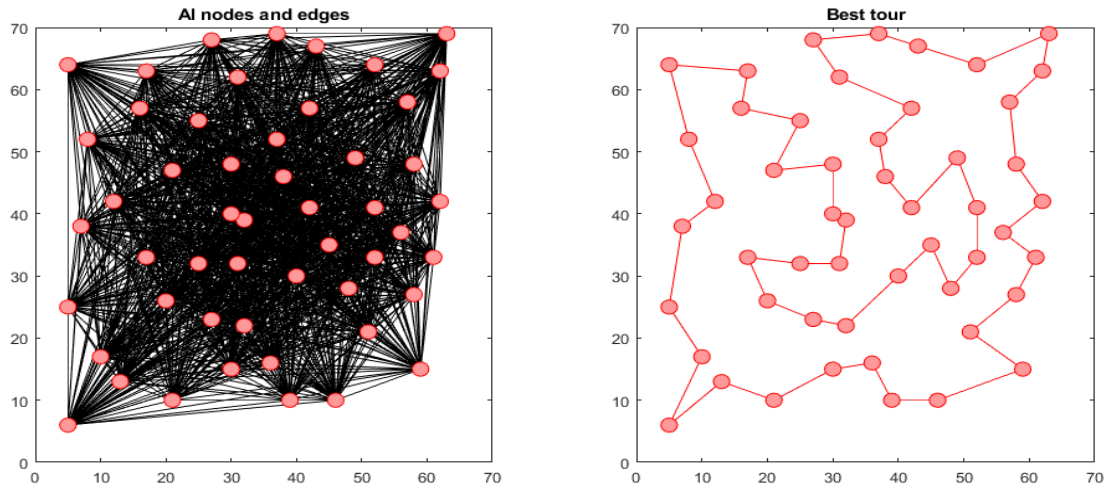


Figure 3. 4. The first close to optimal tour gained by MDTLBO-TSP applied on (eil51) which is a set of 51 cities, from TSPLIB. The minimal tour has length 426

Table 3.1. Shows the average results of 30 time repeated runs on the TSP-LIB benchmark problems used to compare between different algorithm

benchmarks	minimal tour (optimal)	MDTLBO-TSP	DCSA	HC	SA	TS	KA	ABC	ABC2018	GA	ACO
gr17	2085	2085	2085	2354	2114.7	2120	2094.8	2085	2085	2085	2085
gr21	2707	2707	2707	3767	3027.9	2831	2849.6	2707		2726.3	2702
gr24	1272	1272	1275.2	1786	1335.5	1341	1342.7	1281		1291.2	1272
fri26	937	937	937	1395	969.7	988	1004.9	948		955	937
bays29	2020	2020	2020	2187	2062.6	2100	2182.6	2034	2020	2034.5	2020
dantzig42	699	707	707.8	1459	796	748	811.2	744	704	727.6	699
swiss42	1273	1286	1280.2	2444	1483.9	1378	1456.6	1370	1273	1342.2	1297.8
gr48	5046	5057	5115.35	10388	5895.6	5364	5699.2	5595		5183.3	5172.8
hk48	11461	11569	11566.8	23823	15399.9	12287	12621.5	12465		12011.5	11712
eil51	426	431.3746	437.66	596	480.6	457	470.4	459	427.01	443.7	443.7
berlin52	7542	7850.3908	7718.19	10978	9100.6	8338	8841.5	8594	7542	8146.6	7819.1
st70	675	681.5338	691	1880	863.2	849	829.4	830	675	731.3	715.7
eil76	538	564.24	565.39	1367	649.6	604	611.6	622	538	573.5	573.5
pr76	108159	109414.404	110507.1	308382	165690	119510	130888	130729		116416.1	114392.2

refers to optimal sol. | first close to optimal sol. | 2nd close to optimal sol.

4. CHAPTER

CONCLUSION AND FUTURE WORK

In this thesis, we proposed an effective modified meta-heuristic algorithm called, Modified Teaching Learning Based Optimization (MTLBO) to solve numerical optimization problems and Modified Discrete Teaching Learning Based Optimization (MDTLBO-TSP) to symmetric Travelling Salesman Problem that applied to solve benchmark problems from the TSPLIB. The performance of the presented MTLBO algorithm is tested by using twenty-three numerical benchmark functions of various types and dimensions. At the same time, the results were compared with the TLBO and other optimization algorithms.

In the MTLBO algorithm a new parameter called "Academic Plan" is added to the original TLBO algorithm. The main idea of adding the "Academic Plan" parameter provided a strong balance to our proposed algorithm, making it more accurate in solving problems. Although the modification process was simple, the modified algorithm shows promising results and provides an optimal solution for many benchmarks function more than the standard TLBO. To assess the performance of the (MTBLO), we apply it to twenty-three numerical Benchmark functions. The results have been extensively analyzed. These tests show the efficiency of the proposed algorithm in both speed and convergence.

The second part of our study discussed solving the Symmetric Travelling Salesman Problem (sTSP) by using a modified discrete TLBO algorithm based on the 2-OPT heuristic algorithm with the Hamming distance function. The experiments reveal that the proposed method obtained superior results compared with the other algorithms from the literature.

Our study shows that the MTLBO has a great potential to solve real-life problems. Therefore, promising in many fields usage, for the future work to apply our proposed algorithm, we suggest

- Try to enhance MTLBO to binary mode.
- Use it to solve the Asymmetric Traveling salesman problem and multi-visit Traveling salesman problem.
- Enhance our MDTLBO-TSP by using a 3-OPT heuristic algorithm.
- Use it to solve cutting stock, knapsack, loading problems.
- We were able to enhance the TLBO by adding a parameter; we propose to use other mechanisms and ways to evolve the algorithm more effectively.

REFERENCES

1. E.-G. Talbi, 2009. Meta-heuristics: From Design to Implementation, vol. 74. Hoboken, NJ, USA: Wiley.
2. Hussain, K., Mohd Salleh, M. N., Cheng, S., & Shi, Y. 2018. Metaheuristic research: a comprehensive survey. Artificial Intelligence Review.
3. Abualigah, L. 2020. Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Computing and Applications.
4. Abualigah, L., & Diabat, A. 2020. A comprehensive survey of the Grasshopper optimization algorithm: results, variants, and applications. Neural Computing and Applications.
5. Abualigah, L. 2020. Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. Neural Computing and Applications.
6. Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. 2019. Equilibrium optimizer: A novel optimization algorithm. Knowledge-Based Systems.
7. Kallioras, N. A., Lagaros, N. D., & Avtzis, D. N. 2018. Pity beetle algorithm – A new metaheuristic inspired by the behavior of bark beetles. **Advances in Engineering Software, 121**, 147–166.
8. Gandomi, A. H., & Alavi, A. H. 2012. Krill herd: A new bio-inspired optimization algorithm. **Communications in Nonlinear Science and Numerical Simulation, 17**(12), 4831–4845.
9. Sengupta, S., Basak, S., & Peters, R. A. 2019. Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. **Machine Learning and Knowledge Extraction, 1**(1), 157-191.
10. Neshat, M., Sepidnam, G., Sargolzaei, M., & Toosi, A. N. 2014. Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. **Artificial intelligence review, 42(**4), 965-997.
11. Dorigo, M., & Stützle, T. 2019. Ant colony optimization: overview and recent advances. Handbook of metaheuristics, 311-351.

12. Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. 2014. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. **Artificial Intelligence Review, 42**(1), 21-57.

13. Joshi, A. S., Kulkarni, O., Kakandikar, G. M., & Nandedkar, V. M. (2017). Cuckoo search optimization-a review. **Materials Today: Proceedings, 4**(8), 7262-7269.

14. Siddique, N., & Adeli, H. 2017. Nature-Inspired Computing: Physics-and Chemistry-Based Algorithms. Chapman and Hall/CRC.

15. Delahaye, D., Chaimatanan, S., & Mongeau, M. 2019. Simulated annealing: From basics to applications. In Handbook of metaheuristics (pp. 1-35). Springer, Cham.

16. Erol, O. K., & Eksin, I. 2006. A new optimization method: Big Bang–Big Crunch. **Advances in Engineering Software, 37**(2), 106–111.

17. Akyol, S., & Alatas, B. 2017. Plant intelligence-based metaheuristic optimization algorithms. **Artificial Intelligence Review, 47**(4), 417-462.

18. Cui, Z., & Cai, X. 2013. Artificial Plant Optimization Algorithm. Swarm Intelligence and Bio-Inspired Computation, 351–365.

19. Moez, H., Kaveh, A., & Taghizadieh, N. 2016. Natural forest regeneration algorithm: a new meta-heuristic. Iranian Journal of Science and Technology, **Transactions of Civil Engineering, 40**(4), 311-326..

20. Cheraghalipour, A., Hajiaghaei-Keshteli, M., & Paydar, M. M. 2018. Tree Growth Algorithm (TGA): A novel approach for solving optimization problems. **Engineering Applications of Artificial Intelligence, 72**, 393-414.

21. Atashpaz-Gargari, E., & Lucas, C. 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. 2007 IEEE Congress on Evolutionary Computation.

22. Rao, R. V., Savsani, V. J., & Vakharia, D. P. 2011. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. **Computer-Aided Design, 43**(3), 303–315

23. Yang, X. S., & Gandomi, A. H. 2012. Bat algorithm: a novel approach for global engineering optimization. Engineering computations.

24. Mirjalili, S. 2019. Genetic algorithm. In Evolutionary algorithms and neural networks (pp. 43-55). Springer, Cham.

25. Zhao, X., Fang, Y., Liu, L., Li, J., Xu, M. 2020. An improved moth-flame optimization algorithm with orthogonal opposition-based learning and modified position updating mechanism of moths for global optimization problems. **Applied Intelligence, 50**(12), 4434-4458.

26. Mirjalili, S., & Lewis, A. 2016. The Whale Optimization Algorithm. **Advances in Engineering Software, 95**, 51–67.

27. Dey, N., Chaki, J., Moraru, L., Fong, S., & Yang, X. S. 2020. Firefly algorithm and its variants in digital image processing: A comprehensive review. Applications of firefly algorithm and its variants, 1-28.

28. Diallo, M.A., 2020. Meyve sineği ve Kurbağa algoritmaları Temelli Hibrit Yöntemlerin Geliştirilmesi ve Performans Analizi. (Yüksek Lisans Tezi). Erciyes Üniversitesi, Fen Bilimleri Enstitüsü, Kayseri.

29. Danaci M. and Alizada, 2019. "An improvement of hybrid whale optimization algorithm", **Euroasia Journal of Mathematics-Engineering Natural Medical Sciences, 2**, 60-68.

30. Al-Gburi, A.N., 2021. A modified whale optimizations algorithm for bin packing problems. (Master degree thesis). Erciyes University, Graduate School of natural and applied science, Kayseri.

31. Danaci M. and Diallo M.A. 2019. A New Hybrid Fruty Fly Optimization Algorithm for Solving Benchmark Problems, **Euroasia Journal of Mathematics-Engineering Natural Medical Sciences, 2**, 23-27.

32. Hazimeh, H., Mazumder, R., & Radchenko, P. 2021. Grouped Variable Selection with Discrete Optimization: Computational and Statistical Perspectives. arXiv preprint arXiv:2104.07084.

33. Sharma, A. 2020. A Constraint Driven Solution Model for Discrete Domains with a Case Study of Exam Timetabling Problems. arXiv preprint arXiv:2002.03102.

34. Ouaarab, A. 2020. Review of tour generation for solving traveling salesman problems. Nature-Inspired Computation in Navigation and Routing Problems; Yang, X.-S., Zhao, Y.-X., Eds, 85-102.

35. Laporte, G., & Nobert, Y. 1980. A Cutting Planes Algorithm for the m-Salesmen Problem. **Journal of the Operational Research Society, 31**(11), 1017–1023.

36. Poikonen, S., Golden, B., & Wasil, E. A. 2019. A branch-and-bound approach to the traveling salesman problem with a drone. **INFORMS Journal on Computing, 31**(2), 335-346.

37. Pantuza Jr, G., & de Souza, M. C. 2021. Formulations and a Lagrangian relaxation approach for the prize collecting traveling salesman problem. International Transactions in Operational Research.

38. Molina, D., Poyatos, J., Del Ser, J., García, S., Hussain, A., & Herrera, F. 2020. Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. **Cognitive Computation, 12**(5), 897-939.

39. Cuevas, E., Fausto, F., & González, A. 2020. New Advancements in Swarm Algorithms: Operators and Applications. Springer International Publishing.

40. Nasiri, J., & Khiyabani, F. M. 2018. A whale optimization algorithm (WOA) approach for clustering. **Cogent Mathematics & Statistics, 5**(1), 1483565.

41. Misra, R., & Ray, K. S. 2017. Object tracking based on quantum particle swarm optimization. In 2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR) (pp. 1-6). IEEE.

42. Hasan, R. A., Mohammed, M. A., Ţăpuş, N., & Hammood, O. A. (2017, September). A comprehensive study: Ant Colony Optimization (ACO) for facility layout problem. In 2017 16th RoEduNet conference: networking in education and research (RoEduNet) (pp. 1-8). IEEE.

43. Wang, H., Wang, W., Xiao, S., Cui, Z., Xu, M., & Zhou, X. 2020. Improving artificial Bee colony algorithm using a new neighborhood selection mechanism. **Information Sciences, 527**, 227–240.

44. Zhang, S., Zhou, Y., Li, Z., & Pan, W. 2016. Grey wolf optimizer for unmanned combat aerial vehicle path planning. **Advances in Engineering Software, 99**, 121–136.

45. Bozorgi, A., Bozorg-Haddad, O., & Chu, X. 2018. Anarchic society optimization (ASO) algorithm. In Advanced Optimization by Nature-Inspired Algorithms (pp. 31-38). Springer, Singapore.

46. Askari, Q., Younas, I., & Saeed, M. 2020. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. **Knowledge-Based Systems, 195**, 105709.

47. Hoos, H. H. 2011. Automated algorithm configuration and parameter tuning. In Autonomous search (pp. 37-71). Springer, Berlin, Heidelberg.

48. Gunter, D. W. 2021. Is AI in Your Future? AI Considerations for Scholarly Publishers. In Transforming Scholarly Publishing with Blockchain Technologies and AI (pp. 101-117). IGI Global.

49. Holzinger, A., Plass, M., Holzinger, K., Crisan, G. C., Pintea, C. M., & Palade, V. 2017. A glass-box interactive machine learning approach for solving NP-hard problems with the human-in-the-loop. arXiv preprint arXiv:1708.01104.

50. Jahwar, A. F., & Abdulazeez, A. M. 2020. Meta-heuristic algorithms for k-means clustering: A review. **PalArch's Journal of Archaeology of Egypt/Egyptology, 17**(7), 12002-12020.

51. Singh, N., & Singh, S. B. 2017. Hybrid Algorithm of Particle Swarm Optimization and Grey Wolf Optimizer for Improving Convergence Performance. **Journal of Applied Mathematics,** 1–15.

52. Binitha S, Sathya S. 2012, A survey of bio inspired optimization algorithms. **Int J Soft Comput Eng 2**:137–51.

53. Jadhav, A. N., & Gomathi, N. 2017. WGC: Hybridization of exponential grey wolf optimizer with whale optimization for data clustering. Alexandria Engineering Journal.

54. Tawhid, M. A., & Ali, A. F. 2017. A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. **Memetic Computing, 9**(4), 347–359.

55. Tovey, C. A. 2018. Nature-Inspired Heuristics: Overview and Critique. Recent Advances in Optimization and Modeling of Contemporary Problems, 158–192.

56. Jaradat, G. M., & Ayob, M. 2010, December. Big Bang-Big Crunch optimization algorithm to solve the course timetabling problem. In 2010 10th International conference on intelligent systems design and applications (pp. 1448-1452). IEEE.

57. Alatas, B. 2011. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. **Expert Systems with Applications, 38**(10), 13170–13180.

58. Liu, J.; Abbass, H.A.; 2019 Tan, K.C. Evolutionary Community Detection Algorithms. In Evolutionary Computation and Complex Networks; Springer International Publishing: Cham, Switzerland; pp. 77–115.

59. Moriarty, D. E., Schultz, A. C., & Grefenstette, J. J. 1999. Evolutionary algorithms for reinforcement learning. **Journal of Artificial Intelligence Research, 11**, 241-276.

60. Storn R, Price K 1997 Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. **J Glob Optim 11**:341–359.

61. Slowik, A., & Kwasnicka, H. 2020. Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications.

62. Geem, Z. W., Kim, J. H., & Loganathan, G. V. 2001. A new heuristic optimization algorithm: harmony search. **Simulation, 76**(2), 60-68.

63. Fu, Q., Zhou, K., Qi, H., & Jiang, F. 2018. A Modified Tabu Search Algorithm to Solve Vehicle Routing Problem. **J. Comput, 29**, 197-209.

64. Abualigah, L. 2021. Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. **Neural Computing and Applications, 33**(7), 2949-2972..

65. Kashan, A. H. 2009. League championship algorithm: a new algorithm for numerical function optimization. In 2009 international conference of soft computing and pattern recognition (pp. 43-48). IEEE.

66. Arsic, A., Tuba, M., & Jordanski, M. 2016. Fireworks algorithm applied to wireless sensor networks localization problem. In 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 4038-4044). IEEE.

67. Bentouati, B., Chettih, S., & Chaib, L. 2017. Interior search algorithm for optimal power flow with non-smooth cost functions. **Cogent Engineering, 4**(1).

68. Moosavian, N., & Roodsari, B. K. (2014). Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. **Swarm and Evolutionary Computation, 17**, 14-24.

69. Yang, X. S., & He, X. (2013). Bat algorithm: literature review and applications. **International Journal of Bio-inspired computation, 5**(3), 141-149.

70. Yang, X.-S. 2009. Firefly Algorithms for Multimodal Optimization. **Lecture Notes in Computer Science**, 169–178.

71. Kiranyaz, S., Ince, T., & Gabbouj, M. 2014. Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition. Adaptation, Learning, and Optimization.

72. Mirjalili, S. 2016. SCA: A Sine Cosine Algorithm for solving optimization problems. **Knowledge-Based Systems, 96**, 120–133.

73. Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. 2015. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. **Neural Computing and Applications, 27**(2), 495–513.

74. Karaboga, D., & Basturk, B. 2007. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In International fuzzy systems association world congress (pp. 789-798). Springer, Berlin, Heidelberg.

75. Smit, S. K., & Eiben, A. E. 2009. Comparing parameter tuning methods for evolutionary algorithms. In 2009 IEEE congress on evolutionary computation (pp. 399-406). IEEE.

76. Ortiz-Boyer, D., Hervás-Martínez, C., Garcia-Pedrajas, N. 2005. Cixl2: A crossover operator for evolutionary algorithms based on population features. **Journal of Artificial Intelligence Research, 24**, 1-48.

77. Hussain, K., Salleh, M. N. M., Cheng, S., & Naseem, R. 2017. Common benchmark functions for metaheuristic evaluation: A review. JOIV: **International Journal on Informatics Visualization, 1**(4-2), 218-223.

78. Garden, R. W., & Engelbrecht, A. P. 2014. Analysis and classification of optimisation benchmark functions and benchmark suites. In 2014 IEEE Congress on Evolutionary Computation (CEC) (pp. 1641-1649). IEEE.

79. Jamil, M., & Yang, X. S. 2013. A literature survey of benchmark functions for global optimization problems. **International Journal of Mathematical Modelling and Numerical Optimization, 4**(2), 150.

80. Surjanovic S. and Bingham D. 2013. Virtual library of simulation experiments: Test functions and datasets. Retrieved July 2, 2020.

81. Huang, Y., Li, J. P., & Wang, P. 2019. Unusual phenomenon of optimizing the Griewank function with the increase of dimension. **Frontiers of Information Technology & Electronic Engineering, 20**(10), 1344-1360.

82. Bakwad, K. M., Pattnaik, S. S., Sohi, B. S., Devi, S., Panigrahi, B. K., & Gollapudi, S. V. 2010. Multimodal function optimization using synchronous bacterial foraging optimization technique. **IETE Journal of Research, 56**(2), 80-87.

83. Molga, M., & Smutnicki, C. 2005. Test functions for optimization needs. Test functions for optimization needs, 101, 48.

84. Matai, R., Singh, S., & Lal, M. 2010. Traveling Salesman Problem: An overview of Applications, Formulations, and Solution Approaches. Traveling Salesman Problem, Theory and Applications.

85. Maha, Al-Furhud & Ahmed, Zakir. 2020. Genetic Algorithms for the Multiple Travelling Salesman Problem. **International Journal of Advanced Computer Science and Applications. 11**. 553-560.

86. Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. **Omega, 34**(3), 209-219.

87. De Kort, J. B. 1993. A branch and bound algorithm for symmetric 2-peripatetic salesman problems**. European journal of operational research, 70**(2), 229-243.

88. Jiang, Y., Weise, T., Lassig, J., Chiong, R., & Athauda, R. 2014. Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the TSP. 2014 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS).

89. Volgenant, T., & Jonker, R. 1982. A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. **European Journal of Operational Research, 9**(1), 83-89.

90. Chauhan C, Gupta R, Pathak K 2012 Survey of methods of solving TSP along with its implementation using dynamic programming approach. **Int J Compute Appl 52**(4):12–19

91. Fleischmann, B. 1985. A cutting plane procedure for the travelling salesman problem on road networks. **European Journal of Operational Research, 21**(3), 307–317.

92. Lysgaard, J., Letchford, A. N., & Eglese, R. W. 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. **Mathematical Programming, 100**(2), 423-445.

93. Padberg, M., & Rinaldi, G. 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. **SIAM review, 33**(1), 60-100.

94. Gutin, G., & Punnen, A. P. (Eds.). 2006. The traveling salesman problem and its variations (Vol. 12). Springer Science & Business Media.

95. Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. 2018. On the min-cost Traveling Salesman Problem with Drone. Transportation Research Part C: **Emerging Technologies, 86**, 597–621.

96. Saiyed A. R., 2012 The Traveling Salesman Problem, Indiana State University, http://cs.indstate.edu/~zeeshan/aman.pdf , (Date of visit: 5 June 2014).

97. Bandura, A., Adams, N. E., & Bayer, J. 1977. Cognitive processes mediating behavioral change. **Journal of Personality and Social Psychology, 35**, 125–139.

98. Fok, K.-Y., Ganganath, N., Cheng, C.-T., & Tse, C. K. 2016. A 3D printing path optimizer based on Christofides algorithm. 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW).

99. Caccetta, L., Alameen, M., & Abdul-Niby, M. 2013. An improved clarke and wright algorithm to solve the capacitated vehicle routing problem. **Engineering, Technology & Applied Science Research, 3**(2), 413-415.

100. The traveling salesman problem: a case study. 2003. Local Search in Combinatorial Optimization, 215–310.

101. Reinelt, G. 1994. The traveling salesman: Computational solutions for TSP applications. Springer-Verlag.

102. Daoud, A., Balbo, F., Gianessi, P., & Picard, G. 2021. ORNInA: A decentralized, auction-based multi-agent coordination in ODT systems. **AI Communications, 34**(1), 37–53.

103. Laporte, G. 1992. The traveling salesman problem: An overview of exact and approximate algorithms. **European Journal of Operational Research, 59**(2), 231–247.

104. Wang, C., Lin, M., Zhong, Y., & Zhang, H. 2015. Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling. **International Journal of Computing Science and Mathematics, 6**(4), 336-353.

105. Meng, Xianzhe, 2021 Optimization of Cultural and Creative Product Design Based on Simulated Annealing Algorithm, 5538251, 2021.

106. Glover, F. 1977. Heuristics For Integer Programming Using Surrogate Constraints. **Decision Sciences, 8**(1), 156–166.

107. Machado, J. M., Shiyou, Y., Ho, S. L., & Peihong, N. 2001. A common tabu search algorithm for the global optimization of engineering problems.

Computer methods in applied mechanics and engineering, 190(26-27), 3501- 3510.

108. Dorigo, M., Caro, G. D., & Gambardella, L. M. 1999. Ant Algorithms for Discrete Optimization. **Artificial Life, 5**(2), 137–172.

109. Srinivas, M., & Patnaik, L. M. 1994. Genetic algorithms: A survey. **Computer, 27**(6), 17-26.

110. Potvin, J. Y. 1996. Genetic algorithms for the traveling salesman problem. **Annals of Operations Research, 63**(3), 337-370.

111. Fidan Nuriyeva. 2019 A method based on hierarchical clustering for travelling salesman problem. Journal of Modern Technology and Engineering, 2019

112. Rocki, K., & Suda, R. 2012. Accelerating 2-opt and 3-opt local search using GPU in the travelling salesman problem. 2012 International Conference on High Performance Computing & Simulation (HPCS).

113. Zhong, X. 2021. On the approximation ratio of the 3-Opt algorithm for the (1,2)-TSP. **Operations Research Letters, 49**(4), 515–521. t

114. Lin S 1965. "Computer Solutions of the Traveling-Salesman Problem." **Bell System Technology Journal, 44**, 2245–2269.

115. Croes GA, 1958. A Method for Solving Travelling-Salesman Problems. Operations Research, Vol. 6, No. 6, pp. 791-812.

116. Roger Z. Ríos-Mercado O, 2020, optimal Districting and Territory Design, International Series in Operations Research & Management Science book series (ISOR, volume 284)

117. Raja Balachandar, S., & Kannan, K. 2007. Randomized gravitational emulation search algorithm for symmetric traveling salesman problem. **Applied Mathematics and Computation, 192**(2), 413–421.

118. Li, L., Cheng, Y., Tan, L., & Niu, B. 2012. A Discrete Artificial Bee Colony Algorithm for TSP Problem. Lecture Notes in Computer Science, 566–573.

119. Khan, I., & Maiti, M. K. 2018. A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem. Swarm and Evolutionary Computation.

120. Thanh, P. D., Binh, H. T. T., & Lam, B. T. 2013. A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem. 2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR).

121. Saud, S., Kodaz, H., & Babaoğlu, İsmail. 2018. Solving Travelling Salesman Problem by Using Optimization Algorithms. **KnE Social Sciences, 3**(1), 17-32.

122. Akhdir, Zaher, 2020 Development of new versions of crow and bat-based algorithms and evaluation of their performance, Erciyes Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi.

CURRICULUM VITAE

Personal Information

Name: TAREQ MEQDAM TAREQ

Surname: AL-BASHAQHA

Country: Iraq

Education:

Level	Institute Name	Date
High school	Damascus secondary school	2003-2004
University	Technical University	2004-2008

Experience Work

Year	Company	Position
2008-2017	Iraqi media network\ Assabah newspaper	Graphic designer.

Languages

Arabica (Native)

English