MARCH 2021 M.Sc. in Industrial Engineering ŞEYMA YILKICI YÜZÜGÜLDÜ

REPUBLIC OF TURKEY

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

A HEALTH-BASED METAHEURISTIC ALGORITHM: CHOLESTEROL ALGORITHM

M.Sc. THESIS

IN

INDUSTRIAL ENGINEERING

BY

ŞEYMA YILKICI YÜZÜGÜLDÜ

MARCH 2021

A HEALTH-BASED METAHEURISTIC ALGORITHM: CHOLESTEROL ALGORITHM

M.Sc. Thesis

in

Industrial Engineering

Gaziantep University

Supervisor

Prof. Dr. Serap ULUSAM SEÇKİNER

by

Şeyma YILKICI YÜZÜGÜLDÜ

March 2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Şeyma YILKICI YÜZÜGÜLDÜ

ABSTRACT

A HEALTH-BASED METAHEURISTIC ALGORITHM: CHOLESTEROL ALGORITHM

YILKICI YUZUGULDU, Şeyma
M.Sc. in Industrial Engineering
Supervisor: Prof. Dr. Serap ULUSAM SECKINER
March 2021
64 pages

Optimization is as old as the history of the universe. It can be seen anywhere from the farthest point of the universe to the atomic particle. Optimization can sometimes be seen in an engineering design and sometimes in an ant's search for food. The main purpose of all is to find the best possible result under certain conditions. The universe, nature, animals solve their problems easily by instinct. For this reason, people try to solve complex problems by imitating the universe and nature. They developed metaheuristics to solve complex optimization problems. There are many algorithms in the literature that are inspired by animals and nature. However, the number of algorithms inspired by the human body such as artificial neural networks and artificial immune systems are very few. In this thesis, a new health-based metaheuristic algorithm has been developed, inspired by the cholesterol metabolism in the human body. The cholesterol algorithm is modeled using the cholesterol levels in the human body. This thesis focuses on the performance of the cholesterol algorithm on unconstrained continuous optimization problems. By keeping the structure of the cholesterol algorithm simple, a faster and more flexible algorithm has been developed. For the performance analysis of the cholesterol algorithm, 23 comparison tests were used and the test results were compared with 11 different algorithms.

Key Words: Optimization, Metaheuristic Algorithms, Cholesterol Algorithm, Health-Based Algorithms, Unconstrained Continuous Problems

ÖZET

SAĞLIK TABANLI BİR METASEZGİSEL ALGORİTMA: KOLESTEROL ALGORİTMASI

YILKICI YÜZÜGÜLDÜ, Şeyma
Yüksek Lisans Tezi, Endüstri Mühendisliği
Danışman: Prof. Dr. Serap ULUSAM SEÇKİNER
Mart 2021
64 sayfa

Optimizasyon evrenin tarihi kadar eskidir. Evrenin en uç noktasından atom parçacığına kadar her yerde görülebilir. Optimizasyon bazen bir mühendislik tasarımında bazen de bir karıncanın yiyecek arayışında görülebilir. Hepsinin temel amacı belli koşullar altında en iyi mümkün sonucu bulmaktır. Evren, doğa, hayvanlar problemlerini içgüdüsel olarak kolaylıkla çözer. Bu sebeple insanoğlu evreni doğayı taklit ederek karmaşık problemlerini çözmeye çalışır. Karmaşık optimizasyon problemlerini çözmek için metasezgisel yöntemler geliştirmiştir. Literatürde doğadan hayvanlardan esinlenilerek oluşturulan birçok algoritma mevcuttur. Fakat yapay sinir ağları, yapay bağışıklık sistemi gibi insan vücudundan esinlenilen algoritma sayısı çok azdır. Bu tezde insan vücudundaki kolesterol metabolizmasından esinlenilerek sağlık tabanlı yeni bir metasezgisel algoritma geliştirilmiştir. Kolesterol algoritması insan vücudundaki kolesterol seviyelerini kullanılarak modellenmiştir. Bu tezde, kolesterol algoritmasının kısıtsız sürekli optimizasyon problemleri üzerindeki performansına odaklanılmıştır. Kolesterol algoritmasının yapısı basit tutularak, daha hızlı ve esnek bir algoritma geliştirilmiştir. Kolesterol algoritmasının performans analizi için 23 karşılaştırma testi kullanılmıştır ve test sonuçları 11 farklı algoritmayla karşılaştırılmıştır.

Anahtar Kelimeler: Optimizasyon, Metasezgisel Algoritmalar, Kolesterol Algoritması, Sağlık Tabanlı Algoritmalar, Kısıtsız Sürekli Problemler

"Dedicated to my family"

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Dr. Serap ULUSAM SECKINER for her guidance and support throughout the study. I am thankful for her encouragement and motivation.

I would like to express my husband and gratitude to my family, especially my brother Recep YILKICI, for their support, always best wishes.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

GA	Genetic Algorithm
SA	Simulated Annealing
TS	Tabu Search
ACO	Ant Colony Optimization
GP	Genetic Programming
AIS	Artificial Immune System
PSO	Particle Swarm Optimization
DE	Differential Evolution
HA	Harmony Search
BF	Bacterial Foraging
ABC	Artificial Bee Colony
WC	Weed Colonization
BBO	Biogeography- Based Optimization
CS	Cuckoo Search
GSA	A Gravitational Search Algorithm
FFA	Firefly Algorithms
BA	Bat-Inspired Algorithm
TLBO	Teaching Learning Based Optimization
WCA	Water Cycle Algorithm
FPA	Flower Pollination Algorithm
LA	Lion's Algorithm
GWO	Grey Wolf Optimizer
CA	Cholesterol Algorithm
MFO	Moth Flame Optimization
WOA	The Whale Optimization Algorithm

VCS	Virus Colony Search
SSA	Salp Swarm Algorithm
HHO	Harris Hawks Optimization
GBMO	Gases Brownian Motion Optimization
HTSO	Heat Transfer Search Optimization
WSA	Weighted Superposition Attraction
EFO	Electromagnetic Field Optimization
FA	Forest Algorithm
HBA	Hurricane Based Algorithm
BHA	Black Hole Algorithm
ES	Evolutionary Strategy
PFA	Paddy Field Algorithm
ANN	Artificial Neural Network
CSA	Crow Search Algorithm
TSA	Tree Seed Algorithm
MSE	Mean Square Error
RMSE	Root Mean Square Error

CHAPTER I

INTRODUCTION

Optimization is everywhere in our life. We use the shortest path on the way home from school. We want to buy the best car or house in our limited budget. A company wants to maximize its profit while minimizing its costs. An engineer designs the most suitable system or product by finding a balance between customer expectations and current resources. As is seen, optimization is a process that tries to find the best option among alternatives under certain situations. Before finding the best option among alternatives, the model of the problem should be created. Then, the problem type and the optimization methods to solve this problem should be determined according to the model.

While modeling the optimization problem, components of the model should be determined. Components of the model are objective function, decision variables, and constraints[1]. The desired result is translated into a mathematical expression as an objective function. The objective function's value shows the quality of the solution (Equation 1.1). The variables that affect the result are also decision variables (Equation 1.3). Constraints are restrictions on the values of decision variables (Equation 1.2) [1].

The generic form of a mathematical model is that:

$$minimize\ or\ maximize\ f_i(x) \qquad (i= 1,2,\dots I) \qquad (1.1)$$

$$g_j(x)\ <,=,>\ 0 \qquad (j= 1,2,\dots J) \qquad (1.2)$$

$$x = x_n \qquad (n= 1,2,\dots N) \qquad (1.3)$$

The types of optimization problems are determined according to the properties of components of the model. For example, single-multi objective by the number of objectives, constrained-unconstrained by the number of constraints, linear-nonlinear by decision variables' forms, discrete-continuous-mixed by type of decision variables, and static-dynamic by the nature of design variables [1] and robust, stochastic and fuzzy by data uncertainty [2] (Figure 1.1).

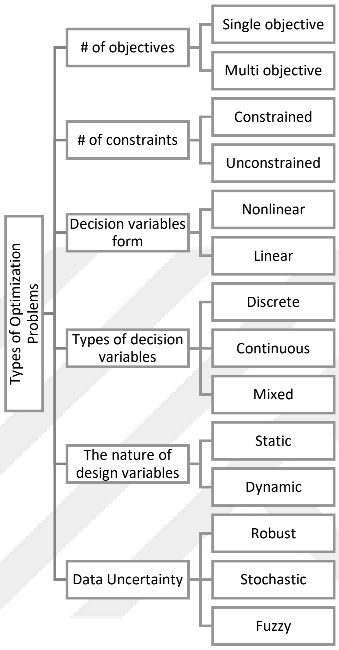

Figure 1.1 Types of optimization problems

Many different optimization algorithms have been developed for solving these problem types in literature. Some of them are that:

- **Deterministic Algorithms:** In problem-solving, they are algorithms that give the same solution under the same conditions every time. All components of the model are certain. Deterministic algorithms are available for unimodal functions that have only a global optimum point because these algorithms use value and differential information of function [4, 7, 8]. Ex. Hill-climbing, Levenberg-Marquardt Algorithm.
    - **Gradient-Based Algorithms:** The algorithm uses derivative information. Gradient-based methods converge faster to exact solutions [4]. Ex.Conjugate gradient-based methods.

- **Gradient-Free Algorithms**: Algorithms don't use any derivative information [4]. Ex. Hooke-Jeeves pattern search.
- **Linear Programming:** The objective function and constraints of the problem are all linear [4].
- **Nonlinear Programming:** The objective function or constraints of the problem are nonlinear [4].

❖ **Stochastic Algorithms:** Stochastic algorithms use randomness. So the algorithm will usually found a different solution every time. Their convergence speed is slow, but they are better than deterministic algorithms about avoiding the solutions being trapped at local optima [4, 7, 8].

- **Fuzzy/Possibilistic Optimization Methods:** Fuzzy/Possibilistic Methods solve problems that objective function and constraints are not clear [3].

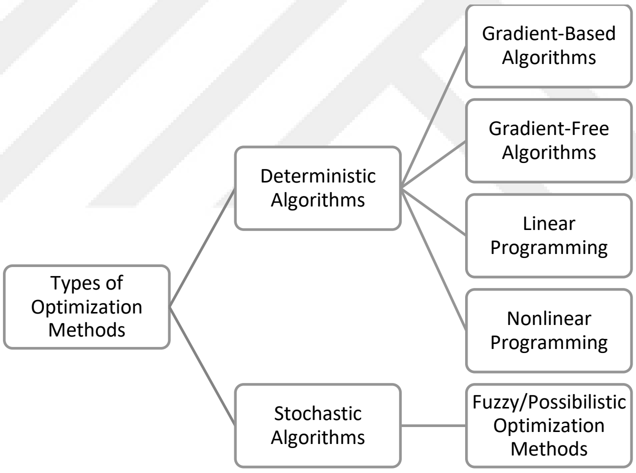

Figure 1.2 Types of optimization methods

Today's problems are complex, dynamic, to solve hard, and time-consuming. Sometimes they can have more objectives and decision variables. Classical optimization methods like the Simplex method, Newton's method are inadequate for solving these problems. These methods use gradient information. So, they converge faster and guarantee the exact solution. While classical methods solve these problems, they follow step by step procedure for finding solutions. So these methods are rigid and slow for solving these complex problems. Therefore, the metaheuristics that converge the solutions intuitively are developed for solving such problems.

Metaheuristics produce admissible solutions by trial-and-error in a reasonably practical time. They aim to find the global best solution with the minimum time and effort. Metaheuristics are approximate methods. They don't guarantee the exact solution, but solutions are approximate to the exact solutions. Metaheuristics have helpful procedures for finding solutions. So they are faster and more flexible than classical methods [7, 8, 10].

Many metaheuristics are inspired by nature or animals. Since nature or animals solve their intrinsically and unconsciously their problems. Thanks to this feature, people have learned to solve their hard and complex problems by mimicking them. Nature or animals can solve their problems with very simple rules with little or no knowledge of their living space [4].

Nature or animals have been the best teacher for people. For example, finding the shortest way for traveling sales problems has been learned from ants. Best to survive principles of evolution are used for many optimization problems like job-scheduling, assignment problems, etc.

In literature, there are many metaheuristics algorithms inspired by behaviors of animal and nature events. However, there are very few algorithms inspired by the human body like artificial neural networks, artificial immune systems. In this thesis, a new health-inspired algorithm has been developed by inspired cholesterol that is essential for cell membranes and hormones in the human body [53]. In this thesis, Cholesterol Algorithm focused on solving unconstrained continuous optimization problems. It has a very simple structure. Therefore, it is faster and more flexible.

The rest of this thesis is organized as follows:

Chapter II gives brief information about metaheuristics in terms of being fundamental. Also developing process of metaheuristics has been presented by reviewing metaheuristics algorithms in the literature.

Chapter III contains brief information about cholesterol and introduces Cholesterol Algorithm in detail. The developing process of the Cholesterol Algorithm is described.

In Chapter IV, a performance analysis of CA is performed. It is tested in 23 benchmark test functions and their results are compared with 11 algorithms.

Finally, in Chapter V, the conclusion and suggestions for further research are given.

CHAPTER II

METAHEURISTICS

2.1 Definition of Metaheuristics

Metaheuristics are a general-purpose heuristic method toward promising regions of the search space containing high-quality solutions. They aim to find the global best solution with the minimum time and effort. Heuristics solve a specific problem. Thus, they are problem-dependent. Metaheuristics are general-purpose algorithms that can be applied to solve any optimization problem. Thus, they are problem-independent.

Metaheuristics are popular among scientists [5, 6]. The background of metaheuristics is based on simple ideas. So the complex problems are solved easier than classical methods thanks to algorithm simplicity and ease of implementation. They don't need to use inclination of data. Also, they can be applied in different areas.

2.2 Classification of Metaheuristics

Many classification criteria may be used for metaheuristics. Some of them [7-9] are the search path they follow, the inspired area, the use of memory, the kind of neighborhood exploration used, or the number of current solutions carried from one iteration to the next.

The search path they follow

- ***Trajectory-based algorithms:*** The algorithm uses a single agent or one solution with multiple iterations [7-9]. Ex. Hill-Climbing, SA, TS, etc.

- ***Population-based algorithms:*** The algorithm uses multiple agents which interact. The principal ideas of the population-based heuristic algorithms are self-adaptation, cooperation, and competition. These concepts help an algorithm to find a global optimum. Population-based algorithms can be classified as swarm-based and evolutionary algorithms [7-9]. For example, PSO, ACO, GA.
- ***Constructive algorithms:*** The algorithm constructs solutions by adding one element at a time to a partial solution. Constructive metaheuristics are adaptations of greedy algorithms that add the best possible element at each iteration. To improve the quality of the final solutions, most constructive metaheuristics include a local search phase after the construction phase. For example, GRASP [10].

The inspired area

- ***Nature Inspired Algorithms:*** Nature-inspired algorithms are metaheuristic algorithms that mimic the specific phenomena or behaviors of nature for the different optimization problems [7-9]. Nature-inspired algorithms are derived as directly and indirectly from nature.
    - ***Physical Processes and Laws Based***: Such as SA, GSA, GBMO, HTSO, EFO, WSA.
    - ***Natural Events Based:*** FA, HBA, WCA, BHA.
- ***Bio-Inspired Algorithms:*** Bio-inspired algorithms are metaheuristics algorithms that are simulated or inspired by the biological behaviors of animals or the biological activities of an organism to find the optimal solution to a given problem [7-9].
    - ***Evolutionary algorithms:*** Evolution algorithms like GA, GP, DE, estimation of distribution algorithms, coevolutionary algorithms, cultural algorithms, and Scatter Search and Path Relinking. They are population-based algorithms and perform with best-to-survive criteria.
    - ***Swarm-based algorithms:*** Swarm-based algorithms are inspired by the collective behavior of a group of social insect colonies and other

animals. Swarm-based algorithms have interrelated agents or communities of individuals. Relations among agents are finding food, building or expanding the nest, effective division of labor among the agents, feeding offspring, and responding to external factors.

- ***Ecology:*** WC, BBO.

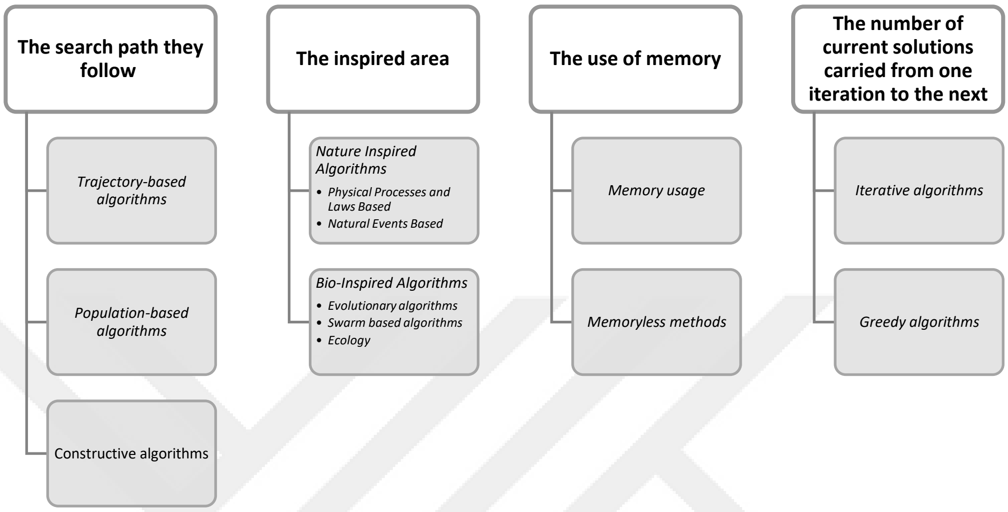

Figure 2.1 Classification of metaheuristics

The use of memory:

- ***Memoryless methods:*** Some metaheuristic algorithms do not use memory explicitly. Their current best or current state is recorded and all the search history may be ignored [7-9]. For instance, local search, GRASP, and SA, GA, PSO.

- ***Memory usage:*** Some metaheuristics use memory/history explicitly [7-9]. For instance, short and long-term memories in TS.

The number of current solutions carried from one iteration to the next:

- ***Iterative algorithms:*** Iterative algorithms start a population of solutions and change it at each iteration by using some search operators [7-9].

- ***Greedy algorithms:*** Greedy algorithms start from an empty solution. At each iteration, a decision variable of the problem is tried to determine until a complete solution is obtained [7-9].

In Table 2.1, the popular metaheuristics algorithms were given in chronologic order.

Table 2.1 Metaheuristic algorithms

Name of Algorithm	Developed By	Date	Inspired By
GA[11]	Holland	1975	the principles of Charles Darwin Theory of survival of the bests
SA [12]	Kirkpatrick et al.	1983	the annealing technique used by the metallurgists
TS [13]	Glover	1986	the human memory
ACO [14]	Dorigo et al.	1991	the ants' social behaviors in finding the shortest paths.
GP [15]	Koza	1992	an extension to GA
AIS [16]	Dasgupta	1993	the human immune system
PSO [17]	Kennedy and Eberhart	1995	the social behavior of bird flocking or fish schooling
DE [18]	Storn and Price	1997	similar to GA
HA [19]	Geem et al.	2001	music improvisation process
BF [20]	Passino	2002	the social foraging behavior of E. Coli bacteria in the human intestines.
ABC[21]	Karaboga	2005	foraging behavior of a honeybee swarm
WC [22]	A.R. Mehrabian, C. Lucas	2006	colonizing weeds
BBO [23]	Simon	2008	the equilibrium theory of island biogeography
CS [24]	Yang	2009	the brood parasitic behavior of some cuckoo species
GSA [25]	Esmat Rashedi et. al.	2009	on the law of gravity and mass interactions
FA [26]	Yang	2009	the flashing light patterns of tropic fireflies
BAT [27]	Yang	2010	the echolocation behavior of microbats
TLBO [28]	Rao et.al	2011	the influence of the teachers on the students in a classical school learning process
WCA [29]	Eskandar et. al	2012	the observation of the water cycle process
FPA [30]	Yang	2012	the pollination process of flowers
LA [31]	B.R.Rajakumar	2012	Lion's social behavior
GWO [32]	Mirjalili et.al	2014	the leadership hierarchy and hunting mechanism of grey wolves in nature

Table 2.1 Metaheuristic algorithms (continued)

Name of Algorithm	Developed By	Date	Inspired By
MFO [33]	Mirjalili	2015	the navigation method of moths in nature called transverse orientation
WOA [34]	Mirjalili et.al	2015	the social behavior of humpback whales
SCA [35]	Mirjalili et.al	2015	a mathematical based algorithm
MVO [36]	Mirjalili et.al	2015	three concepts in cosmology: white hole, black hole, and wormhole
AAA [37]	Uymaz et.al	2015	by the living behaviors of microalgae in nature
VCS [38]	Li et.al	2015	diffusion and infection strategies for the host cells adopted by a virus to survive and propagate in the cell environment
JAYA[39]	Rao	2015	move towards the best solution and avoid the worst solution
TSA[40]	Kiran	2015	the relation between trees and their seeds
CSA[[41]	Askarzadeh	2016	the behaviors of crows
SSA [42]	Mirjalili et.al	2017	the swarming behavior of salps when navigating and foraging in oceans
HHO [43]	Heidari et.al	2019	the hunting behavior of Harris' Hawks

2.3 Developing Process Of Metaheuristics

Each metaheuristics algorithm has its particular process. The parameters, methods, memory, the type of solution set they use and problems they solved are different from each other because the animals and cases which they mimic are different. It is very difficult to put metaheuristics in a common standard pattern. However, a general procedure used in the development of a metaheuristics algorithm can be done as Figure 2.1.

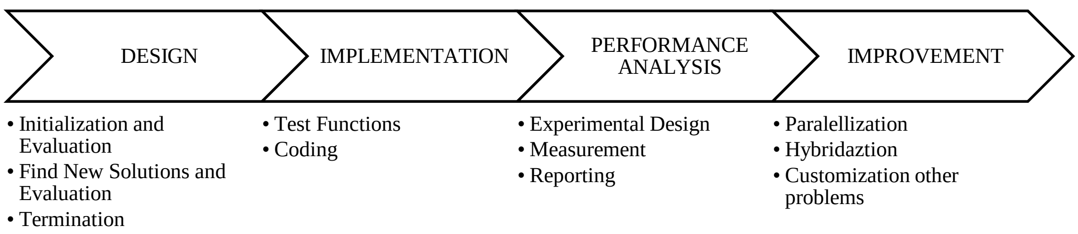

Figure 2.2 Developing process of metaheuristics

2.3.1 Design

Researchers who develop metaheuristics need to adapt the behavior of animals or nature events they inspired to the algorithms that solve the optimization problems. They design a model by observing and analyzing the behavior of animals or natural events. Then, this model is converted into a mathematical module. In the mathematical module, representation of solutions and evaluation of these solutions in the optimization problem iteratively until the best solution is found, parameters used in the algorithm, how to do a local and global search, and when the algorithm will terminate are determined.

2.3.1.1 Initialization and Evaluation

In the design of metaheuristics, firstly, the imitated metaphor should be adapted to the problem.

The particle used in the metaphor represents the solution set of the problem. The quality of these particles is associated with the problem's objective function. Representation of solution sets and function values of some algorithms are given in Table 2.2. As seen in Table 2.2, candidate solutions in the solution set can be chromosomes in GA, ants in ACO, bees in ABC, birds in PSO. The suitability or quality of these solutions is to the function values of these equal solutions.

Table 2.2 Representation of solution sets and parameters of some algorithms

Name of Algorithm	Representation Of Solution	Function Value	Parameters
GA[11]	chromosome	-	population size, max. generation number, cross over probability, mutation probability, length of the chromosome, chromosome encoding
PSO[17]	Single solution: particle Population: swarm	-	number of particles, the dimension of particles, range of particles, Vmax, inertia weight, the maximum number of iterations
DE[18]	individuals	-	the population size, the constant of differentiation, the crossover control parameter

Table 2.2 Representation of solution sets and parameters of some algorithms (continued)

Name of Algorithm	Representation Of Solution	Function Value	Parameters
ACO[14]	ants	-	number of ants ,iterations , pheromone evaporation rate, amount of reinforcement
ABC[21]	candidate solutions: bees exploring/exploiting food resources solutions: the food resources	the nectar amount of each resource	number of food sources which is equal to the number of employed or onlooker bees (SN), the value of limit, the maximum cycle number (MCN)
CS[24]	solution:each egg in a nest a new solution: cuckoo egg	quality/fitness of the host nest	the number of host nests (or the population size n) and the probability pa
FFA[26]	firefly	the brightness of a firefly	the attractiveness, the randomness, the population size
BAT[27]	bats	amount of frequencies	pulse rates, the loudness, number of iterations
GWO[32]	grey wolf	fitness of search agents	a, A, and C coefficient vectors
MVO[36]	each solution is a universe and each variable in the solution is an object in that universe.	an inflation rate	d is the number of variables and n is the number of universes (candidate solutions)
JAYA[39]	candidate solution	objective function	-
HHO[43]	hawks	the fitness values of hawks	the initial energy, jump strength J, the population size, and maximum number of iterations T
WOA[34]	whale	fitness of search agents	a, A, and C coefficient vectors
TSA[40]	trees and seeds	tree location	search tendency

Parameters are important for some algorithms because they affect the efficiency and effectiveness of the algorithm. They allow larger flexibility and robustness for the search. Determining the correct parameters affects the success of the algorithm, and there is no guideline to determine which parameters should be used. The optimal values for the parameters depend on the problem and search time. Strategies for parameter tuning are the off-line parameter initialization and the online parameter tuning strategy. In off-line parameter initialization, the values of parameters are determined at the beginning of the algorithm and are fixed during the algorithm. In the online approach, the parameters are controlled and updated dynamically or adaptively during the algorithm [44].

Most metaheuristics are based on stochastic randomization. They use random distribution to identify initial solutions and update solution sets (to find new solution sets). Some algorithms use different randomization techniques to determine the initial solution set. They can be uniform distributions and/or Gaussian distributions, Monte Carlo simulations [44].

2.3.1.2 Find New Solutions and Evaluation

New solutions are generated by improving the previous/existing solutions (neighborhood solutions) or by searching different regions in the solution space to discover new places [45].

Exploration is the ability to search over a larger search space more efficiently. Because of exploration, the algorithm avoids the solutions being trapped at local optima. Exploration aims to decrease the convergence rate of the algorithm. In exploitation, algorithms search around the current best solutions and select the best candidates or solutions. Exploitation aims to increase the speed of convergence [45].

A metaheuristic will be successful on a given optimization problem if it can provide a balance between exploration and exploitation. There is no common guideline for this balance. So each algorithm has different degrees of exploitation and exploration [45].

The use of local information of the search and neighborhood is used for exploitation. Also, randomization can be used for the neighborhood around the current best. The rule used to select the new current solution is search strategy. A common search strategy is the steepest descent or steepest ascent strategy. Diverse solutions are generated to explore the search space on a global scale. Randomization is used for exploration. It enables an algorithm to have the ability to jump out of any local optimum to explore the search globally [45].

GA[11] uses operators like selection, crossover, and mutation to find new solutions. Storn and Price proposed the process of DE in 1997 [18].

In PSO [17], the fitness function is evaluated for each particle and each particle's position(x_k^i) and velocity(v_k^i) are updated in each iteration. The updating equations for positions x_{k+1}^{i} and velocities v_{k+1}^{i} are these:

$$x_{k+1}^{i} = x_k^i + v_{k+1}^i \tag{2.1}$$

$$v_{k+1}^i = v_k^i + c_1 r_1 (p_k^i - x_k^i) + c_2 r_2 (p_k^i - x_k^i) \tag{2.2}$$

In ACO [46], ants find the shortest paths between the ants' nest and the food sources by tracing pheromone trails Then, ants choose the path to follow by a probabilistic decision biased by the amount of pheromone.

In CS [24], when generating new solutions $x^{(t+1)}$ for a cuckoo i, a Lévy flight is performed with Equation 2.3.

$$x_{i}^{t+1} = x_i^t + \alpha \oplus \ L\acute{e}vy(\lambda) \tag{2.3}$$

where $\alpha > 0$ is the step size that should be related to the scales of the problem of interests [24].

The process of GWO, WOA, MVO were presented by Mirjalili et. al.[32,34,36]. Heidari et.al proposed the process of HHO in 2019 [43]. Yang presented the process of FFA in 2009 [26]. Rao proposed the process of JAYA in 2014 [39].

2.3.1.3 Termination

Termination criteria can be the maximum number of iterations, CPU time, or that continues to search until an allowable tolerance between the last two results [44].

2.3.2 Implementation

2.3.2.1 Test Functions

Most of the metaheuristics are firstly tested in benchmark tests that are accepted in the literature. These test functions have different features; such as unimodal and multimodal, regular and irregular, separable and non-separable, etc. Unimodal functions are functions that have only one local minimum. These types of functions are used to test the speed at which algorithms converge to the optimum value[47,48].

Multi-modal functions are functions that have more than one local minimum and are more difficult to solve than single-mode functions. These types of functions are used to test algorithms' ability to get rid of local minima. A separable function can be expressed as functions with one variable. There is a dependency between variables in non-separable functions. Therefore, the solution of non-separable functions is more difficult than separable functions. Another feature of functions is the number of variables. As the size of the function increases, its solution becomes more difficult [47,48].

Researchers chose a different set of functions according to their metaheuristics' features. Table 2.3 shows the number of test functions used in metaheuristics to be used in this thesis.

Table 2.3 Number of used test functions of metaheuristics

Name of Algorithm	Number of Used Test Functions
CS [24]	10 benchmark tests
FFA [26]	10 benchmark tests
WOA [34]	29 optimization problems and 6 structural design problems
GWO [32]	29 test functions, 3 engineering design problems, and the field of optical engineering as a real application
MVO [36]	19 test problems and 5 engineering problems
JAYA [39]	24 constrained and 30 unconstrained benchmark functions
HHO [43]	29 benchmark problems and 6 engineering problems
TSA [40]	24 benchmark functions and multilevel thresholding

DE was developed to solve the Chebyshev polynomial fitting problem [18]. Hussain et. al presented a review about benchmark tests used in metaheuristics [47]. Jamil and Yang compiled a set of 175 benchmark functions for unconstrained optimization problems [48].

2.3.2.2 Coding

Many frameworks are developed in different program languages for metaheuristics. Some of them are ParadisIO coded in C++, ECJ coded in Java, HeuristicLab coded in C#, PiPy and Evolopy [49,50] coded in Python. Parejo et.al presented a detailed survey about frameworks developed for metaheuristics [51].

2.3.3 Performance Analysis

2.3.3.1 Experimental Design

In this step, the aims of the experiments and performance indicators have to be defined. The aims of the experiments can be search time, quality of solutions, robustness in terms of the samples, solving large-scale problems, parallel scalability in terms of the number of processors, easiness of implementation, easiness to combine with other algorithms, flexibility to solve other problems or optimization models, innovation using nature-inspired paradigms, automatic tuning of parameters [44]. Performance indicators can be solution quality, computational effort, robustness, development cost, simplicity, ease of use, flexibility, and maintainability may be used. Also, the metaheuristics to be compared must be determined [44].

2.3.3.2 Measurement

Measures to compute are selected. These measures may be mean, median, minimum, maximum, standard deviation, success rate, MSE, and so on. After different experiments, statistical analysis is applied to the obtained results [44].

Table 2.4 shows measures, the number of runs, compared metaheuristics, the method of statistical analysis used in the algorithm to be used in this thesis.

Table 2.4 Information about performance analysis of some algorithms

Name Of Algorithm	Measures	The Number of Runs	Compared Metaheuristics	The Method Of Statistical Analysis
CS [24]	the number of function evaluations in the form of mean ± the standard deviation	100	PSO and GA	-
FFA [26]	the number of function evaluations in the form of mean ± the standard deviation	100	PSO and GA	-
WOA [34]	the standard deviation and the mean of the results	30	PSO, GSA, DE, FEP	-
GWO [32]	the standard deviation and the mean of the results	30	PSO,GSA,DE, EP,ES	-

Table 2.4 Information about performance analysis of some algorithms (continued)

Name Of Algorithm	Measures	The Number of Runs	Compared Metaheuristics	The Method Of Statistical Analysis
MVO [36]	the standard deviation and the mean of the results	30	GWO, PSO, GA, GSA	Wilcoxon's nonparametric
JAYA [39]	the mean and the best and worst results and standard deviations	100	HM, ASCHEA, SMES, GA, PSO, DE, ABC, BBO, TLBO, HTS	Friedman rank test and Holm-Sidak test
HHO [43]	the best, worst, standard deviation, and the average of the results	30	GA, BBO, DE, PSO, CS, TLBO, BAT, FPA, FA, GWO, and MFO	Wilcoxon's nonparametric
TSA [40]	the standard deviation and the mean of the results	30	BA, FA, HS, ABC, PSO	-

2.3.3.3 Reporting

The results are presented comprehensively, and an analysis is carried out of the defined goals [44].

2.3.4 Improvement

Metaheuristics algorithms firstly focus on solving a particular problem type. For example, while ACO initially focused on solving the TSP problem, many other algorithms focused on solving benchmark tests, which are continuous functions found in the literature. Then, they focused on solving other problem types such as multiobjective, discrete, constrained, etc.

Hybridization is combinations of metaheuristics with other metaheuristics, deterministic algorithms, artificial intelligence, or machine learning techniques. They have provided very powerful search algorithms [44].

Parallelization is used to enhance the effectiveness of metaheuristics thanks to the development of technology in designing processors, networks, and data storage [44].

Bat algorithm's variants and applications were presented by Yang [52]. Joshi et.al presented a review about CS's applications [53]. Areas of applications for GA, DE, ACO, ABC, AIS are presented by Binitha et. al [54]. Fister et al. [55] presented a survey on FFA's variants and applications. Karaboga et al. [56] presented a comprehensive survey about ABC's variants and applications.

CHAPTER III

CHOLESTEROL ALGORITHM

3.1 About Cholesterol

Cholesterol is a waxy, fat-like substance that's found in all the cells in the human body [57]. It is in the form of white crystals that are tasteless, odorless, melting at 1500C [58]. Cholesterol usually melts in organic solvents (such as ether, chloroform, petroleum ether, carbon-sulfur, acetone) [58]. It oxidizes when it contacts air and light [58].

Cholesterol is important for the body. It is used for cell membranes, hormones, vitamin D, and bile acids [59]. The body produces all the cholesterol it needs. Especially, cholesterol production occurs in the liver and the intestines. Also, the body takes cholesterol from foods like animal products [57, 60].

Cholesterol is essential for all cell membranes because cholesterol affects membrane fluidity and maintains membrane integrity. Cholesterol is used for intracellular transport, cell signaling, and nerve conduction. Cholesterol is a precursor molecule for the synthesis of vitamin D and all steroid hormones, including the adrenal, cortisol and aldosterone, and sex hormones. Bile acids are derivatives of cholesterol synthesized by liver cells. While the body uses cholesterol to produce cells and certain hormones, it uses triglycerides as a source of energy [59].

Cholesterol is transported via lipoproteins in the body. Lipoproteins are a combination of lipid and protein. Lipoproteins are chylomicrons, VLDL, LDL, HDL [61].

Chylomicrons carry lipids from the intestine to muscle and other tissues for energy or fat production. The synthesis of chylomicrons takes place in the small intestine during active fat absorption. They contain mainly triglyceride. The size of the chylomicron particle depends on its triglyceride content [61].

VLDL is very-low-density lipoprotein. VLDL is bad cholesterol because it causes the accumulation of plaque in arteries. VLDL is produced by the liver. VLDL mainly carries triglycerides throughout the body. The size of the VLDL particle is related directly to the triglyceride content and inversely to the phospholipid and protein. The triglycerides are used by cells for energy. Extra triglycerides are stored in lipid cells and released at a later time when needed for energy [61].

LDL is low-density lipoprotein. Its compositions are 50% cholesteryl esters, 30% phospholipids, 10% unesterified cholesterol, and 10% triglycerides[61]. They transfer cholesterol to cells. LDL molecules provide cholesterol for membrane biosynthesis or storage within the cell. It is bad cholesterol. If the levels of LDL cholesterol are too high in the blood, LDL can combine with other substances in the blood. Then, the high LDL level causes plaque in arteries. This plaque can be caused narrow arteries, limit the blood flow, and raise the risk of blood clots. If a blood clot blocks an artery in the heart or brain, it can cause a heart attack or stroke. The main difference between VLDL and LDL is that they have different amounts of cholesterol, protein, and triglycerides that make up each lipoprotein. VLDL contains more triglycerides. LDL contains more cholesterol [61].

HDL stands for high-density lipoprotein. Its main components are phospholipids (42-51%), cholesterol (32%), and triglycerides (10%) [61]. It is good cholesterol because it carries cholesterol from other parts of the body back to the liver. HDL can remove cholesterol from the arterial wall through the liver. The liver removes the cholesterol from the body. HDL functions are reverse cholesterol transport, antioxidant, and anti-inflammatory activities [62, 63].

Factors affecting the level of cholesterol in the body are age, heredity, weight, race, and diet [57]. To measure cholesterol levels, a simple blood test called a lipid panel is used. It is used to determine levels of total cholesterol, LDL cholesterol, HDL cholesterol, and triglycerides [60,64]. Total cholesterol is calculated by using Equation 3.1.

$$Total\ Cholesterol = HDL + LDL + Triglycerides/5 \qquad (3.1)$$

Cholesterol levels in a human body [60] are as in Table 3.1. Cholesterol levels that should be in a healthy person are that HDL is 60 and above, LDL is less than 100, Triglycerides are less than 150, Total Cholesterol is less than 200. Otherwise, cardiovascular diseases are seen [57-64].

Table 3.1 Cholesterol levels in a human body

	Low	**Good**	**High**
HDL	Men: Less than 40 Women: Less than 50	60 or higher	n/a
LDL	n/a	Less than 100	Greater than 160
Triglycerides	n/a	Less than 150	Greater than 200
Total Cholesterol	n/a	Less than 200	240 or higher

3.2 Cholesterol Algorithm

3.2.1 Design of Cholesterol Algorithm

Cholesterol algorithm has been developed inspired by cholesterol metabolism which is important for the human body. Most of the cholesterol is produced in the liver and distributed to other tissues. Besides, cholesterol enters the body with the foods. The transport of cholesterol in the body is done through lipoproteins. Cholesterol is transported from the liver to the body with LDL, excess cholesterol in the body cells is carried to the liver with HDL.

In the cholesterol algorithm, each optimization problem is considered as a human body. Transport of nutrients and oxygen between tissues in the human body occurs through vessels. In an optimization problem, the quality of each solution candidates is considered as cholesterol status in a each vessel. After the function values of the solution candidates are calculated, the quality of solution candidate or cholesterol levels in the vessel is obtained by using the worst and best function values.

Cholesterol levels of each solution are calculated by using Equation 3.2-3.5. Function values are transformed cholesterol levels through these equations. Cholesterol levels in a person should be within certain ranges. So the level of HDL, good cholesterol, should be above **40**, and the level of LDL, bad cholesterol, should be below **160**. These levels and the good-bad cholesterol relationship are two of the issues that the CA algorithm is inspired by cholesterol metabolism. The CA algorithm uses ranges to determine the quality of the solution and decides whether solutions are good or bad relative to these ranges.

The cholesterol ranges to be used in the CA algorithm are **[$HDL_{LowerBound}$, 2*$HDL_{LowerBound}$]** for the range of HDL, **[0, $LDL_{UpperBound}$]** for the range of LDL, **[0, $Triglycerides_{UpperBound}$]** for the range of triglycerides. In this thesis, it is preferred to use the real cholesterol levels in a human body in Table 3.1. In other words, **40** for $HDL_{LowerBound}$, **160** for $LDL_{UpperBound}$, **200** for $Trigliycerides_{UpperBound}$ were used. So the cholesterol ranges of the CA algorithm are **[40, 80]** for the range of HDL, **[0, 160]** for the range of LDL, and **[0, 200]** for the range of Triglycerides. It is more important to use only these ranges for determining levels of HDL, LDL, and Triglicerides in the proposed CA algorithm. In other words, according to the results obtained from the observations, levels of HDL, LDL, and Triglicerides in the proposed CA algorithm can be determined at any level as long as under ranges. A range determination is necessary as it constitutes the working principle of the CA algorithm.

$$HDL = HDL_{LowerBound} + HDL_{LowerBound} * (f_{Current} - f_{Worst}) / (f_{Worst} - f_{Best}) \tag{3.2}$$

$$LDL = LDL_{UpperBound} * (f_{Best} - f_{Current}) / (f_{Worst} - f_{Best}) \tag{3.3}$$

$$Triglycerides = Triglycerides_{UpperBound} * random(0,1) \tag{3.4}$$

$$Total\ Cholesterol = HDL + LDL + Triglycerides/5 \tag{3.5}$$

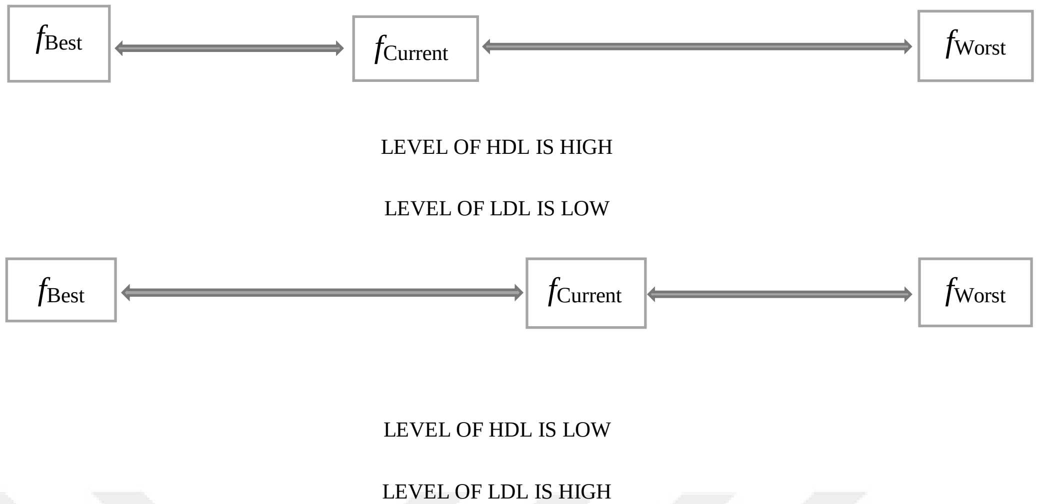

Figure 3.1 Relationship of function values and levels of HDL and LDL

In Figure 3.1, the relationship between the distance of the current solution from the best-worst solutions and the HDL-LDL levels has been shown. That is, the closer the current solution is to the best known solution, the higher the HDL value and the lower the LDL value.

Another of the issues that the CA algorithm is inspired by cholesterol metabolism is that levels of lipoprotein affect the blood flow through the vessel. If the level of LDL is high in the vessel, there is a blockage in the vessel because there is excess fat in the vessel. The amount of HDL reduces the amount of fat in the vessel. If we adapt this situation to an optimization problem, the amount of HDL increases when the solution gets better, and the amount of LDL increases when it gets worse. When levels of HDL and LDL in the vessel, in the solution path, go beyond the predetermined ranges, the vessel is blocked. Vascular occlusion shows that there are no good solutions in the search area of solution space, so this solution path is blocked. These solutions are forgotten because optimization problems are focused on maximization or minimization. The search continues with the neighboring solutions of the best solution found instead of them. The exploitation process in the optimization algorithms is performed in this way in the cholesterol algorithm. As long as cholesterol levels continue to be within specified limits, blood flow in the vessel continues. This situation shows that the searched solution space is suitable for exploration and that the search will continue in this solution space until occlusion occurs.

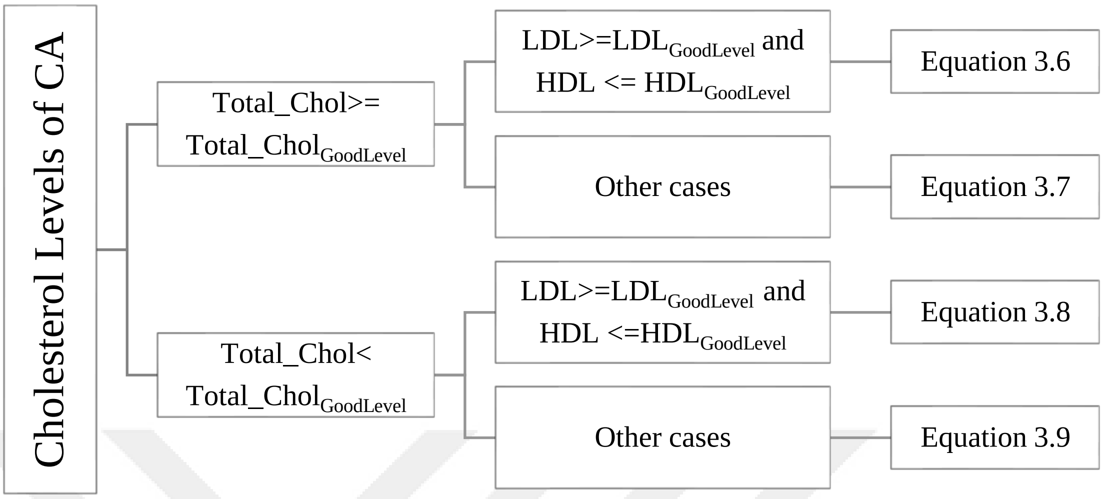

Figure 3.2 Cholesterol levels of CA

$$x_{new} = x_{best} - x_{best} * \text{Trig of } x_{best} * \text{random}(-1,1) \tag{3.6}$$

$$x_{new} = x_{current} - (x_{current} * \text{LDL of } x_{current} / \text{HDL of } x_{current} - \text{Trig of } x_{current}) \tag{3.7}$$

$$x_{new} = x_{best} + x_{best} * \text{Trig of } x_{best} * \text{random}(-1,1) \tag{3.8}$$

$$x_{new} = x_{current} + (x_{current} * \text{LDL of } x_{current} / \text{HDL of } x_{current} - \text{Trig of } x_{current}) \tag{3.9}$$

New solutions are obtained based on the cholesterol levels of current solutions or best solutions. According to Figure 3.2, if the amount of LDL exceeds $LDL_{GoodLevel}$ and the amount of HDL falls below $HDL_{GoodLevel}$, this solution is bad and is forgotten due to the blockage of this solution (vessel). The neighboring solutions of the best solution are focused on improving the best solution. If the cholesterol level of the solution is within the desired range, this solution is good and the search continues in the area where the solution is located. $Total\_Chol_{GoodLevel}$, $LDL_{GoodLevel}$, and $HDL_{GoodLevel}$ must be in the range of Total_Chol, LDL, and HDL, respectively. In this thesis, values in the ‘Good’ Column of Table 3.1 were used for these values.

$Total\_Chol_{GoodLevel}$, $LDL_{GoodLevel}$, and $HDL_{GoodLevel}$ were taken as 200, 100, and 60, respectively.

In Figure 3.2, Total Cholesterol, LDL, and HDL are used in the decision-making of cholesterol levels in the CA algorithm. Since total cholesterol consists of the sum of LDL, HDL, and triglycerides (Equation 3.1), the amount of triglycerides indirectly affects total cholesterol. Therefore, it was not considered necessary to use triglycerides in decision-making.

CA algorithm is focused on solving minimization problems. In order to solve maximization problems, if the objective function of the problem is *f(x)* and constraints of it are $g(x) \leq 0$, they are taken as $-f(x)$, $-g(x) \geq 0$ respectively.

The process of the cholesterol algorithm is as follows:

Step 1: The initial solution set is randomly generated in uniform distribution(Equation 3.10) and the function values are calculated (Equation 3.11).

$$x = lowerbound + (upperbound - lowerbound) * Random \qquad (3.10)$$

$$Function\,Value = f(x) \qquad (3.11)$$

Step 2: Solution set has best and worst values according to function values. Cholesterol levels are calculated according to these values, which shows the quality of the solution. The equations for calculating cholesterol levels of the problem are Equation 3.2-3.5.

Step 3: New sets of solutions are calculated based on cholesterol levels according to Figure 3.2. Function values of these solution sets are calculated.

Step 4: Step 2 proceeds until the specified stopping criterion is terminated. Step 2 and Step 3 are repeated.

Set the termination criteria and vessel number
Generate initial solution set in each vessel randomly and calculate function values of initial solution set
For(termination criteria)
    Determine local_best, global_best, local_worst
    local_best = min(Function_value)
    local_worst = max(Function_value)
    if global_best > local_best:
    global_best = local_best
    best_solution = current_solution[local_best]
    For(vessels)
        Calculate cholesterol levels of each vessel
        LDL= $160 * (f_{Best} - f_{Current}) / (f_{Worst} - f_{Best})$
        HDL=$40 + 40 * (f_{Current} - f_{Worst}) / (f_{Worst} - f_{Best})$
        Triglicerides=$200 * random(0,1) /5$
        Total_Chol=HDL+LDL+Triglicerides
        Update solution set
        if Total_Chol>=200
            if LDL>=100 and HDL<=60:
            $x_{new} = x_{best} + x_{best} * Trig\ of\ x_{best}$*random(-1,1)*
            else:
            $$x_{new} = x_{current} - (x_{current} * LDL\ of\ x_{current} / HDL\ of\ x_{current} - Trig\ of\ x_{current})$$
        else:
            if LDL>=100 and HDL<=60:
            $x_{new} = x_{best} + x_{best} * Trig\ of\ x_{best}$*random(-1,1)*
            else:
            $$x_{new} = x_{current} + (x_{current} * LDL\ of\ x_{current} / HDL\ of\ x_{current} - Trig\ of\ x_{current})$$

        Calculate function values of new solution set
    End for
End For

Figure 3.3 CA pseudocode

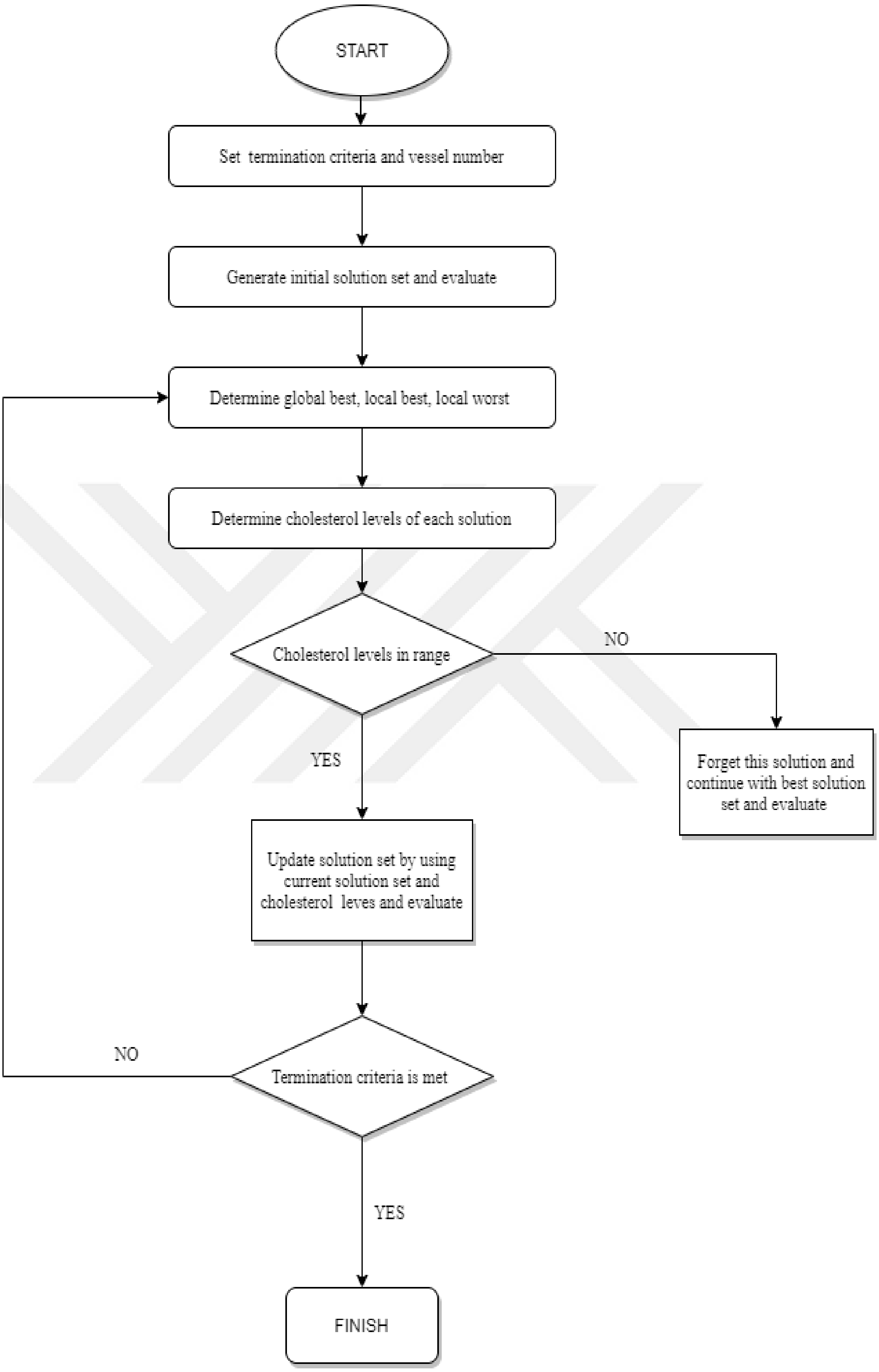

Figure 3.4 Flow chart of CA

3.2.2 Implementation of Cholesterol Algorithm

3.2.2.1 Coding

All tests and performance analyzes were performed using a computer with Intel Core i5 10th Gen, 1.60GHz processor, 8GB Ram, Microsoft Windows 10 operating system. The Cholesterol Algorithm was coded in Python, the most popular programming language of recent times. Cholesterol algorithm and 11 other algorithms were run in 23 benchmark tests using Evolopy Framework [42, 43].

3.2.2.2 Test Functions

Table 3.2 Unimodal test functions

	Name	**Function**	**Search Range**	**Dimension**	**Feature**	**Global Optimum**
1	Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	[-100,100]	30	US	0
2	Schwefel2.22	$f(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	[-10,10]	30	UN	0
3	Schwefel1.2	$f(x)=\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	[-100,100]	30	UN	0
4	Schwefel2.21	$f(x)=max_i\{\lvert x_i\rvert, 1 \leq i \leq n\}$	[-100,100]	30	UN	0
5	Rosenbrock	$f(x)=\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30,30]	30	UN	0
6	Step	$f(x)=\sum_{i=1}^{n} ([x_i + 0.5])^2$	[-100,100]	30	US	0
7	Quartic	$f(x)=\sum_{i=1}^{n} ix_i^4 + random[0,1)$	[-1.28,1.28]	30	US	0
8	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	[-4.5,4.5]	5	UN	0

Table 3.3 Multimodal test functions

	Name	Function	Search Range	Dimension	Feature	Global Optimum
9	Rastrigin	f$(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	[-5.12,5.12]	30	MS	0
10	Ackley	f$(x)=\sum_{i=1}^{n}-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	[-32,32]	30	MN	0
11	Griewank	f$(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600,600]	30	MN	0
12	Penalized	$f(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1}]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10{,}100{,}4)$ $y_i=1+\frac{x_i+1}{4}u(x_i,a,k,m)=f(x)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0-a & <x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	[-50,50]	30	MN	0
13	Penalized 2	$f(x)=0.1\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5{,}100{,}4)$	[-50,50]	30	MN	0
14	Booth	$f(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	[-10,10]	2	MS	0
15	Kowalik	$f(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2}{b_i^2+b_ix_3+x_4}\right]^2$	[-5,5]	4	MN	0.00030748610
16	Six Hump Camel Back	$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	[-5,5]	2	MN	-1.01316
17	Branin	$f(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	[-5,10]x[0,15]	2	MS	0.398
18	GoldStein-Price	$f(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	[-2,2]	2	MN	3
19	Hartman 3	$f(x)=-\sum_{i=1}^{4}c_iexp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2$	[1,3]	3	MN	-3.86
20	Hartman 6	$f(x)=-\sum_{i=1}^{4}c_iexp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2$	[0,1]	6	MN	-3.32

Table 3.3 Multimodal test functions (continued)

	Name	Function	Search Range	Dimension	Feature	Global Optimum
21	Shekel 5	$f(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	[0,10]	4	MN	-10.1532
22	Shekel 7	$f(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	[0,10]	4	MN	-10.4028
23	Shekel 10	$f(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	[0,10]	4	MN	-10.5363

3.2.3 Performance Analysis of Cholesterol Algorithm

In this thesis, before analyzing the performance of the algorithm, all tests worked under equal conditions as follows:

- Compared algorithms had been applied on the same problems, 23 benchmark tests.
- The properties of the applied problems were determined as the same in all tests.
- Compared algorithms were run 500 iteration numbers as stopping criteria.
- The numbers of algorithms were taken 30 runs independently of each other.
- Initial candidate solutions were started randomly between the lower and upper limits specified for each function.
- Algorithms' specific parameters were kept independent from problems.

Compared algorithms contained in the Evolopy Framework were used. They are PSO, FFA, GWO, WOA, MVO, CS, HHO, JAYA, DE, and GA. Also, TSA was adapted to Evolopy Framework and used as the compared algorithm.

The parameters of each algorithm are as follows:

- In GA, crossover probability is 0.8, mutation probability is 0.05, elitism ratio is 0.5.
- In FFA, α is 0.5 as randomness, the minimum value of β (the variation of attractiveness) is 0.20, the absorption coefficient is 1.
- In the CS algorithm, the discovery rate of alien eggs/solutions is .0.25.

- In MWO, the wormhole existence probability(WEP) is between 0.2 and 1.
- Parameters of PSO are v_{max} is 6, w_{Max} is 0.9, w_{Min} is 0.2, c_1 and c_2 are 2.
- For TSA, the search tendency is 0.01.

The quality of the solutions found by CA and other compared algorithms to evaluate;

- Statistical results such as the best solution, mean, and standard deviation values achieved by algorithms in all studies were used.
- Mean Square Error (MSE) and Root Mean Square Error (RMSE) had been calculated by using the differences between the best-known solutions and the best solutions found. These values were calculated by Equation 3.12 and 3.13. Then, these values were used for general performance evaluation. Also, the convergence curve and box-plot graphics of the CA were shown with algorithms that had the lowest two values according to the MSE calculations of the algorithms.

$$MSE = \frac{\sum_{f=1}^{n}(evaluated(f) - predicted(f))^2}{n} \tag{3.12}$$

$$RMSE = \sqrt{\frac{\sum_{f=1}^{n}(evaluated(f) - predicted(f))^2}{n}} \tag{3.13}$$

  $evaluated\ (f)$ is the optimum value calculated when the algorithm reaches the stopping criterion. $predicted\ (f)$ is the global optimum value expected from the algorithm and n is the function number.

- Wilcoxon signed-rank test was used to statistically compare the quality of the solutions obtained from all independent studies given.
- The run time from the beginning of the algorithms to reaching the stopping criterion was calculated to evaluate the speed at which the algorithms reach the solution.

CHAPTER IV

RESULTS AND DISCUSSION

In Table 4.1 and 4.2, statistical results such as the best solution, mean, and standard deviation values achieved by algorithms for each function with 30 independent runs are shown. The mean values for the function are indicated in bold colors.

If the data in Table 4.1 is examined in detail on function-based, CA is the best algorithm in F1, F2, F3, and F4 unimodal functions. Unimodal functions are used to test the exploitation capability of the algorithm. The CA algorithm gave the best results in these four functions, and the exploitation capability of the CA algorithm can be mentioned. The algorithms that give the best results in F5, F6, F8 unimodal functions are HHO, PSO, and DE, respectively.

Multimodal functions have many local minima, they are used to test the exploration capabilities of algorithms. If the data in Table 4.2 is examined in detail on function-based, CA and HHO gave the best results for F9, F10, and F16 multimodal functions. CA, WOA, and HHO had the best result in F11 function. HHO is the best algorithm in F12, F13, and F15 functions. For F14 function, the best algorithms are DE,PSO, and JAYA. For F16, F17, F18, and F19 functions, most algorithms' means are the same, but standard deviations are different. Therefore, algorithms with minimum standard deviation are DE, PSO, and TSA algorithms in F16 function, DE and PSO algorithms in F17 function, CS algorithm in F18 function, DE and CS algorithm in F19 function. For F20 function, the best algorithms are DE,CS and TSA algorithms. TSA has the best result for the F21 function.For F22 and F23 multimodal functions, the best result is in the CS algorithm.

CA algorithm gave approximate results to known correct answers in most multimodal functions, but CA algorithm did not have better results than other algorithms in other multimodal functions except F9, F10, F11, and F16. The exploration capability of the CA algorithm is not as good as the exploitation capability of it. So, this capability needs to be improved a little more.

The success of the algorithms varies according to the dimension and characteristics of the problem and the parameters it uses. So, the conditions under which each algorithm is successful are different from each other. In order to compare the success of algorithms, each algorithm must be run under same conditions such as termination criteria, population number, and dimension and characterictics of problems. So, in this thesis, the dimension of problem was taken as 30 for multi-dimensional functions. Population number of all algorithms was taken as 50 and termination criteria was taken as 500 iteration number. Therefore, the performance of some algorithms was worse than other algorithms for some problems under these conditions.

Table 4.1 The best solution, mean and std. dev. values of unimodal functions

		CA	GWO	WOA	HHO	DE	FFA	CS	PSO	MVO	JAYA	GA	TSA
F1	**best**	6.00E-112	3.52E-38	8.52E-95	1.12E-85	1.11E-03	3.02E-03	1.06E+01	1.63E-07	2.99E-01	1.82E-07	2.32E+01	3.69E+02
	mean	**2.13E-109**	7.91E-37	8.58E-85	1.42E-69	4.44E-03	6.40E-03	2.14E+01	8.60E-06	7.43E-01	1.12E-05	4.87E+01	5.92E+02
	std	3.44E-109	1.13E-36	3.60E-84	7.75E-69	2.19E-03	2.14E-03	5.98E+00	9.55E-06	2.25E-01	1.73E-05	1.55E+01	1.46E+02
F2	**best**	2.46E-57	1.28E-22	3.28E-60	1.38E-43	2.54E-02	1.83E-01	8.65E+00	5.04E-04	3.94E-01	1.44E-05	1.16E+00	6.94E+00
	mean	**1.78E-55**	6.99E-22	8.65E-54	5.55E-37	4.77E-02	9.59E+00	1.35E+01	4.00E+00	7.46E-01	8.14E-05	2.06E+00	1.13E+01
	std	2.38E-55	5.11E-22	2.32E-53	2.69E-36	1.55E-02	1.17E+01	3.32E+00	6.75E+00	7.83E-01	4.73E-05	4.35E-01	2.51E+00
F3	**best**	3.46E-111	1.47E-11	1.15E+04	5.80E-73	1.90E+04	3.10E+02	2.63E+03	1.81E+01	3.73E+01	5.86E+03	4.28E+03	2.60E+04
	mean	**2.12E-108**	2.71E-07	3.32E+04	2.67E-48	2.85E+04	1.32E+03	3.48E+03	4.21E+01	9.49E+01	1.84E+04	1.10E+04	3.57E+04
	std	5.02E-108	1.06E-06	1.13E+04	1.46E-47	4.57E+03	9.03E+02	7.26E+02	1.51E+01	4.73E+01	7.98E+03	2.88E+03	5.10E+03
F4	**best**	5.76E-56	1.75E-09	3.87E-06	8.85E-44	8.35E+00	9.03E-02	9.23E+00	5.09E-01	4.78E-01	6.84E+00	9.38E+00	5.16E+01
	mean	**5.57E-55**	7.26E-08	3.52E+01	7.03E-36	1.65E+01	2.18E-01	1.36E+01	8.47E-01	1.32E+00	1.67E+01	1.33E+01	6.31E+01
	std	4.41E-55	7.90E-08	3.12E+01	2.69E-35	4.90E+00	7.97E-02	1.62E+00	2.05E-01	5.50E-01	6.31E+00	2.05E+00	4.59E+00
F5	**best**	2.70E+01	2.53E+01	2.68E+01	3.54E-05	2.97E+01	2.28E+01	5.40E+02	1.32E+01	3.19E+01	2.57E+01	5.88E+02	2.80E+05
	mean	2.85E+01	2.66E+01	2.76E+01	**6.58E-03**	4.68E+01	5.09E+02	1.04E+03	5.98E+01	4.33E+02	7.18E+01	1.58E+03	9.07E+05
	std	5.46E-01	7.69E-01	3.94E-01	7.62E-03	2.86E+01	1.26E+03	4.66E+02	5.49E+01	6.80E+02	9.14E+01	9.57E+02	3.33E+05
F6	**best**	2.62E+00	3.10E-05	1.98E-02	1.24E-07	1.02E-03	2.98E-03	9.37E+00	5.28E-07	2.91E-01	3.00E+00	2.13E+01	3.23E+02
	mean	4.02E+00	4.33E-01	8.24E-02	4.27E-05	5.21E-03	6.81E-03	2.35E+01	**2.32E-05**	7.40E-01	3.77E+00	5.11E+01	5.79E+02
	std	9.15E-01	2.85E-01	7.16E-02	5.26E-05	1.97E-03	2.37E-03	7.23E+00	8.33E-05	2.06E-01	5.36E-01	2.69E+01	1.29E+02
F7	**best**	1.07E-05	3.31E-04	4.12E-05	1.09E-06	3.51E-02	1.38E-01	3.75E-02	3.49E-02	8.97E-03	6.62E-03	7.36E-02	3.50E-01
	mean	1.29E-04	1.63E-03	2.10E-03	**6.88E-05**	4.88E-02	2.63E-01	7.98E-02	1.25E+00	2.26E-02	4.12E-02	1.63E-01	6.64E-01
	std	1.04E-04	9.60E-04	1.75E-03	6.51E-05	9.04E-03	7.15E-02	2.38E-02	2.31E+00	8.06E-03	3.82E-02	5.03E-02	2.27E-01
F8	**best**	5.82E-08	1.08E-08	4.90E-17	0.00E+00	0.00E+00	4.35E-11	5.07E-23	0.00E+00	5.21E-10	0.00E+00	7.88E-03	5.44E-21
	mean	1.44E-01	1.12E-07	4.84E-10	1.58E-12	**0.00E+00**	5.68E-10	1.50E-17	**0.00E+00**	7.62E-02	1.62E-03	4.59E-01	2.41E-16
	std	2.66E-01	9.99E-08	2.27E-09	3.45E-12	0.00E+00	5.04E-10	3.33E-17	0.00E+00	2.33E-01	6.63E-03	3.00E-01	1.22E-15

Table 4.2 The best solution, mean and std. dev. values of multimodal functions

		CA	**GWO**	**WOA**	**HHO**	**DE**	**FFA**	**CS**	**PSO**	**MVO**	**JAYA**	**GA**	**TSA**
F9	**best**	0.00E+00	1.14E-13	0.00E+00	0.00E+00	1.68E+02	3.88E+01	8.34E+01	4.39E+01	3.94E+01	2.82E+01	8.20E+00	2.17E+02
	mean	**0.00E+00**	8.14E+00	5.68E-15	**0.00E+00**	1.88E+02	7.69E+01	1.09E+02	1.02E+02	1.14E+02	8.86E+01	1.14E+01	2.44E+02
	std	0.00E+00	8.16E+00	1.73E-14	0.00E+00	6.98E+00	2.20E+01	1.38E+01	3.42E+01	2.93E+01	4.04E+01	2.16E+00	1.37E+01
F10	**best**	4.44E-16	3.24E-14	4.44E-16	4.44E-16	1.57E-02	1.66E-02	4.57E+00	6.08E-04	3.29E-01	4.17E-04	2.50E+00	7.64E+00
	mean	**4.44E-16**	3.82E-14	3.76E-15	**4.44E-16**	2.72E-02	6.90E-01	5.88E+00	2.32E-03	1.96E+00	1.34E-03	2.99E+00	8.90E+00
	std	1.50E-31	3.30E-15	1.85E-15	1.50E-31	8.17E-03	6.50E-01	7.06E-01	1.71E-03	3.34E+00	1.15E-03	2.83E-01	9.69E-01
F11	**best**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.92E-03	8.51E-03	1.07E+00	5.63E-08	6.01E-01	1.84E-06	1.22E+00	4.57E+00
	mean	**0.00E+00**	2.05E-03	**0.00E+00**	**0.00E+00**	4.79E-02	2.00E-02	1.20E+00	8.13E-03	7.77E-01	5.66E-02	1.41E+00	6.36E+00
	std	0.00E+00	4.77E-03	0.00E+00	0.00E+00	8.79E-02	8.21E-03	6.53E-02	7.65E-03	8.07E-02	1.08E-01	1.19E-01	1.12E+00
F12	**best**	3.53E-01	2.86E-06	1.91E-03	1.03E-09	1.61E-02	3.94E-05	3.01E+00	1.69E-08	6.80E-01	7.02E-01	2.58E-01	4.64E+04
	mean	5.69E-01	7.60E-02	1.93E-02	**1.89E-06**	8.93E-02	3.64E-01	3.83E+00	3.80E-02	1.94E+00	1.17E+00	6.02E-01	5.53E+05
	std	1.41E-01	6.13E-02	3.91E-02	2.33E-06	7.51E-02	2.64E-01	5.59E-01	7.93E-02	8.88E-01	4.11E-01	2.68E-01	4.54E+05
F13	**best**	1.76E+00	6.00E-05	2.83E-02	4.06E-08	4.77E-03	3.64E-04	4.31E+00	2.75E-07	4.72E-02	1.67E+00	1.64E+00	9.27E+04
	mean	2.57E+00	3.85E-01	1.46E-01	**2.60E-05**	2.53E-02	5.27E-03	9.01E+00	5.15E-03	1.09E-01	3.54E+02	2.87E+00	2.80E+06
	std	3.76E-01	2.13E-01	9.86E-02	3.90E-05	1.96E-02	6.82E-03	2.67E+00	5.56E-03	5.32E-02	1.90E+03	9.31E-01	1.74E+06
F14	**best**	2.48E-08	5.11E-09	1.10E-06	9.65E-11	0.00E+00	4.99E-10	1.24E-26	0.00E+00	1.12E-08	0.00E+00	4.91E-05	0.00E+00
	mean	5.42E-06	2.40E-07	1.98E-04	2.66E-07	**0.00E+00**	6.35E-09	1.74E-22	**0.00E+00**	5.58E-07	**0.00E+00**	4.88E-02	1.75E-29
	std	7.15E-06	1.84E-07	2.06E-04	2.98E-07	0.00E+00	7.23E-09	5.86E-22	0.00E+00	5.37E-07	0.00E+00	5.30E-02	5.51E-29
F15	**best**	3.08E-04	3.07E-04	3.08E-04	3.08E-04	3.07E-04	7.06E-04	3.35E-04	3.09E-04	5.13E-04	3.22E-04	7.89E-04	4.81E-04
	mean	3.48E-04	2.32E-03	7.95E-04	**3.10E-04**	3.69E-04	1.08E-03	5.13E-04	4.77E-03	4.05E-03	1.08E-03	6.19E-03	6.96E-04
	std	7.19E-05	6.12E-03	5.21E-04	2.43E-06	2.32E-04	2.39E-04	1.08E-04	7.94E-03	7.42E-03	5.08E-04	8.33E-03	8.31E-05
F16	**best**	0.00E+00	1.14E-13	0.00E+00	0.00E+00	1.68E+02	3.88E+01	8.34E+01	4.39E+01	3.94E+01	2.82E+01	8.20E+00	2.17E+02
	mean	**0.00E+00**	8.14E+00	5.68E-15	**0.00E+00**	1.88E+02	7.69E+01	1.09E+02	1.02E+02	1.14E+02	8.86E+01	1.14E+01	2.44E+02
	std	0.00E+00	8.16E+00	1.73E-14	0.00E+00	6.98E+00	2.20E+01	1.38E+01	3.42E+01	2.93E+01	4.04E+01	2.16E+00	1.37E+01

Table 4.2 The best solution, mean and std. dev. values of multimodal functions (continued)

		CA	GWO	WOA	HHO	DE	FFA	CS	PSO	MVO	JAYA	GA	TSA
F17	**best**	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00
	mean	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	**-1.03E+00**	-1.03E+00	-1.03E+00	**-1.03E+00**	-1.03E+00	-1.03E+00	-1.03E+00	**-1.03E+00**
	std	2.08E-07	1.57E-08	2.13E-10	2.76E-10	6.78E-16	3.66E-09	1.59E-13	6.78E-16	2.52E-07	2.17E-05	2.89E-03	6.78E-16
F18	**best**	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
	mean	3.98E-01	3.98E-01	3.98E-01	3.98E-01	**3.98E-01**	3.98E-01	3.98E-01	**3.98E-01**	3.98E-01	3.99E-01	4.02E-01	3.98E-01
	std	2.08E-05	1.48E-06	1.61E-06	3.48E-07	1.13E-16	3.91E-09	3.20E-09	1.13E-16	3.36E-07	4.59E-03	6.70E-03	5.61E-16
F19	**best**	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	mean	3.00E+00	3.00E+00	3.00E+00	5.70E+00	3.00E+00	3.00E+00	**3.00E+00**	3.00E+00	3.00E+00	3.04E+00	7.77E+00	**3.00E+00**
	std	2.63E-04	1.23E-05	4.73E-05	8.24E+00	4.85E-15	3.36E-08	4.52E-16	1.95E-15	2.84E-06	1.64E-01	1.08E+01	4.52E-16
F20	**best**	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.85E+00	-3.86E+00	-3.86E+00
	mean	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	**-3.86E+00**	-3.86E+00	**-3.86E+00**	-3.86E+00	-3.86E+00	-3.72E+00	-3.86E+00	**-3.86E+00**
	std	2.71E-04	2.50E-03	2.70E-03	1.36E-05	1.36E-15	8.34E-10	1.36E-15	2.00E-03	8.12E-07	1.04E-01	2.16E-04	1.36E-15
F21	**best**	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-2.94E+00	-3.32E+00	-3.32E+00
	mean	-3.27E+00	-3.24E+00	-3.27E+00	-3.26E+00	-3.24E+00	-3.27E+00	-3.32E+00	-3.25E+00	-3.27E+00	-2.16E+00	-3.27E+00	**-3.32E+00**
	std	7.74E-02	6.66E-02	7.40E-02	6.06E-02	5.54E-02	6.21E-02	7.69E-07	8.65E-02	5.99E-02	4.74E-01	5.92E-02	2.00E-08
F22	**best**	-1.01E+01	-1.01E+01	-1.01E+01	-1.01E+01	-1.01E+01	-1.01E+01	-1.01E+01	-1.01E+01	-1.01E+01	-3.57E+00	-1.01E+01	-1.01E+01
	mean	-9.43E+00	-9.43E+00	-9.60E+00	-5.21E+00	-9.94E+00	-7.67E+00	**-1.01E+01**	-6.59E+00	-6.99E+00	-1.58E+00	-5.99E+00	-1.01E+01
	std	1.75E+00	1.74E+00	1.54E+00	9.24E-01	9.17E-01	2.68E+00	1.03E-07	2.84E+00	2.62E+00	8.11E-01	3.63E+00	1.41E-03
F23	**best**	-1.02E+01	-1.02E+01	-1.02E+01	-5.06E+00	-1.02E+01	-1.02E+01	-1.02E+01	-1.02E+01	-1.02E+01	-4.20E+00	-1.02E+01	-1.02E+01
	mean	-6.64E+00	-9.66E+00	-7.98E+00	-5.06E+00	-9.42E+00	-7.67E+00	**-1.02E+01**	-7.74E+00	-6.14E+00	-1.82E+00	-4.40E+00	-1.02E+01
	std	3.70E+00	1.54E+00	3.00E+00	5.46E-05	1.99E+00	3.21E+00	1.00E-06	3.11E+00	2.83E+00	9.50E-01	3.01E+00	2.95E-06

Table 4.3 MSE and RMSE values of algorithms

	MSE	RMSE
HHO	6.70	2.59
CA	38.93	6.24
PSO	872.26	29.53
MVO	2.87E+04	169.27
FFA	1.88E+05	433.49
GWO	3.87E+05	622.12
CS	6.05E+05	777.66
JAYA	1.76E+07	4197.76
GA	5.78E+06	2.40E+03
DE	3.62E+07	6015.39
WOA	5.34E+07	7307.14
TSA	5.30E+11	7.28E+05

Table 4.3 shows the MSE and RMSE values of algorithms. Values of MSE and RMSE show how far the results of the algorithms are from the true value. HHO algorithm has the minimum value of MSE and RMSE. CA has the second minimum value.

Table 4.4 shows the results of the Wilcoxon signed-rank test. For problem-based pairwise comparisons of test algorithms, all independent studies are. It was carried out using the globally optimum values.

The hypothesis thesis is as follows:

- H_0 (null hypothesis): 'There is no significant difference between the results of the two algorithms.
- H_1 (alternative hypothesis) is the opposite.

In the tests, the significance level was used as $\alpha = 0.05$. The p-value is the estimated probability of rejecting the H_0 hypothesis when the hypothesis is true. A small p-value indicates strong evidence against the null hypothesis. The '+' value refers to the sums of rank values in the results, where the first algorithm is better than the second. The '–' refers to the sums of rank values in the results, where the second algorithm is better. '=' indicates that there is no statistically significant difference between the success of the two algorithms in solving the problem does. T represents the smallest of these sums.

The last lines of Table 4.4 show three statistical meanings in binary comparison show the total numbers for the status (marked with '+', '=' or '-').

Table 4.4 shows that the CA algorithm was better than other algorithms in special problems such as F1, F2, F3, and F4 unimodal functions, but it could not have a noticeable success compared to other algorithms in F6 and F8 unimodal functions. CA algorithm was not as good as other algorithms in multimodal functions except F9, F10, F11, and F15. When Table 4.2 was examined in detail, the best and mean values of the CA algorithm were the same as the other best algorithms for these multimodal functions. The standard deviation values of the CA algorithm were higher than other algorithms.

Table 4.4 Results of Wilcoxon Signed-Rank Test

F	CA - GWO			CA - WOA			CA - HHO			CA - DE		
	p	T	W	p	T	W	p	T	W	p	T	W
F1	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F2	1.83E-06	0	+	3.88E-01	190	+	1.83E-06	0	+	1.83E-06	0	+
F3	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F4	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F5	2.02E-06	1	-	1.18E-05	19	-	1.83E-06	0	-	1.83E-06	0	+
F6	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-
F7	1.83E-06	0	+	4.08E-06	8	+	6.23E-03	99	-	1.83E-06	0	+
F8	4.97E-06	10	-	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-
F9	1.81E-06	0	+	1.49E-01	0	=	1.00E+00	0	=	1.83E-06	0	+
F10	8.30E-07	0	+	1.96E-06	0	+	1.00E+00	0	=	1.83E-06	0	+
F11	5.91E-02	0	=	1.00E+00	0	=	1.00E+00	0	=	1.83E-06	0	+
F12	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-
F13	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-
F14	2.26E-05	26	-	2.24E-06	2	+	5.08E-05	35	-	1.83E-06	0	-
F15	2.02E-01	170	=	1.88E-05	24	+	5.32E-02	138	=	3.73E-04	59	-
F16	2.97E-05	29	-	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-
F17	5.54E-05	36	-	4.26E-05	33	-	5.48E-06	11	-	1.83E-06	0	-
F18	1.83E-06	0	-	2.02E-06	1	-	2.86E-03	87	-	1.83E-06	0	-
F19	5.51E-01	203	=	2.26E-05	26	+	4.97E-06	10	-	1.83E-06	0	-
F20	8.53E-01	223	=	7.73E-01	218	=	5.79E-01	205	=	4.11E-01	192	=
F21	4.72E-01	197	=	4.83E-02	136	+	4.97E-06	10	+	2.26E-05	26	-
F22	4.53E-03	94	-	2.10E-01	171	=	1.21E-02	110	+	1.20E-04	45	-
F23	3.19E-04	57	-	1.28E-02	111	-	7.69E-02	146	=	1.83E-06	0	-
+/=/-	7/5/11			10/4/9			6/6/11			9/1/13		

Table 4.4 Results of Wilcoxon Signed-Rank Test (continued)

F	CA - FFA			CA - CS			CA - PSO			CA - MVO		
	p	T	W	p	T	W	p	T	W	p	T	W
F1	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F2	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F3	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F4	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F5	6.39E-04	66	+	1.83E-06	0	+	5.85E-02	140	=	1.83E-06	0	+
F6	1.83E-06	0	-	1.83E-06	0	+	1.83E-06	0	-	1.83E-06	0	-
F7	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F8	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-	2.18E-03	83	-
F9	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F10	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F11	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F12	2.67E-03	86	-	1.83E-06	0	+	1.83E-06	0	-	2.48E-06	3	+
F13	1.83E-06	0	-	1.83E-06	0	+	1.83E-06	0	-	1.83E-06	0	-
F14	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-	5.93E-04	65	-
F15	1.83E-06	0	+	4.08E-06	8	+	1.83E-06	0	+	1.83E-06	0	+
F16	3.69E-06	7	-	1.83E-06	0	-	1.83E-06	0	-	5.24E-01	201	=
F17	2.02E-06	1	-	1.83E-06	0	-	1.83E-06	0	-	3.69E-06	7	-
F18	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-	1.83E-06	0	-
F19	1.83E-06	0	-	1.83E-06	0	-	3.73E-04	59	-	1.83E-06	0	-
F20	2.41E-01	175	=	1.83E-06	0	-	9.34E-01	228	=	9.17E-02	150	=
F21	3.76E-01	189	=	1.83E-06	0	-	1.80E-02	117	+	1.80E-02	122	+
F22	2.01E-02	119	-	1.83E-06	0	-	4.60E-02	135	-	7.89E-01	219	=
F23	8.07E-06	15	-	1.83E-06	0	-	4.71E-04	62	-	9.55E-03	106	-
+/=/-	10/2/11			13/0/10			10/2/11			12/3/8		

Table 4.4 Results of Wilcoxon Signed-Rank Test (continued)

F	CA - JAYA			CA - GA			CA - TSA		
	p	T	W	p	T	W	p	T	W
F1	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F2	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F3	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F4	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F5	1.15E-03	74	+	1.83E-06	0	+	1.83E-06	0	+
F6	3.24E-01	184	=	1.83E-06	0	+	1.83E-06	0	+
F7	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F8	2.67E-03	86	-	6.89E-04	67	+	1.83E-06	0	-
F9	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F10	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F11	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F12	2.24E-06	2	+	9.02E-01	226	=	1.83E-06	0	+
F13	3.34E-06	6	+	2.02E-01	170	=	1.83E-06	0	+
F14	1.83E-06	0	-	1.83E-06	0	+	1.83E-06	0	-
F15	1.83E-06	0	+	1.83E-06	0	+	1.83E-06	0	+
F16	3.03E-06	5	+	1.83E-06	0	+	1.83E-06	0	-
F17	6.81E-01	212	=	2.02E-06	1	+	1.83E-06	0	-
F18	1.90E-02	118	+	2.02E-06	1	+	1.83E-06	0	-
F19	1.83E-06	0	+	7.69E-02	146	=	1.83E-06	0	-
F20	1.83E-06	0	+	2.58E-01	177	=	1.83E-06	0	-
F21	1.83E-06	0	+	1.43E-05	21	+	1.42E-04	47	-
F22	8.07E-06	15	+	7.50E-03	102	+	1.83E-06	0	-
F23	8.88E-06	16	+	6.07E-01	207	=	1.83E-06	0	-
+/=/-	19/2/2			18/5/0			13/0/10		

Table 4.5 The average run time of the algorithms in seconds

	CA	GWO	WOA	HHO	DE	FFA	CS	PSO	MVO	JAYA	GA	TSA
F1	**0.67**	16.43	17.36	1.47	3.64	18.02	12.84	16.05	16.07	2.6	4.39	7.23
F2	**1.21**	16.52	14.07	2.19	2.63	17.66	13.5	15.96	16.4	3.42	4.33	6.11
F3	8.03	19.64	18.03	7.14	**5.83**	21.24	19.51	18.78	19.49	7.31	6.02	10.14
F4	**0.62**	16.25	14.2	1.19	2.24	17.55	13.04	15.45	16.37	3.05	4.09	5.97
F5	**1.08**	16.57	14.48	1.51	2.5	17.71	13.28	15.74	16.45	3.29	4.37	6.06
F6	**0.77**	16.31	14.4	1.28	3.21	17.7	13.21	15.64	16.72	3.16	4.25	6.22
F7	**1.45**	16.32	14.75	1.72	3.07	17.67	13.72	15.67	17.09	3.24	4.48	6.34
F8	0.63	1.53	1.33	1.55	1.08	18.04	1.82	1.25	1.42	**0.32**	3.31	0.91
F9	**0.89**	15.79	15.85	1.59	2.36	17.7	12.71	16.14	16.94	3.06	4.09	5.71
F10	**1.48**	16.34	17.77	1.96	2.73	18.36	13.59	16.41	18.52	3.56	4.52	6.34
F11	**2.00**	16.53	17.69	2.3	2.88	18.66	13.85	16.52	18.29	3.74	4.66	6.30
F12	**2.96**	16.95	18.67	2.99	3.42	19.15	14.79	17.15	18.88	4.35	5.05	7.52
F13	**2.73**	17.96	18.07	2.91	3.38	19.38	14.72	17.4	18.36	4.29	5.00	6.63
F14	0.58	1.72	1.52	1.45	1.34	19.09	2.05	1.24	1.8	**0.4**	3.24	0.80
F15	1.70	3.36	3.46	2.28	1.59	20.49	3.21	3.2	3.07	**1.06**	3.96	1.80
F16	0.66	1.5	1.54	1.3	0.98	21.08	1.51	1.41	1.45	**0.36**	3.19	0.93
F17	0.66	1.53	1.59	1.37	1.02	16.74	1.52	1.51	1.51	**0.38**	3.27	0.92
F18	1.01	1.71	1.78	1.66	1.16	16.82	1.88	1.67	1.72	**0.55**	3.34	1.04
F19	2.90	3.49	3.77	3.68	2.14	17.73	4.02	3.61	3.4	**1.84**	4.20	2.17
F20	2.90	5.32	5.75	3.66	2.33	17.81	5.3	5.11	5.11	**2.1**	4.23	2.78
F21	8.27	7.67	8.82	9.53	**5.14**	20.72	9.83	7.58	8.1	5.35	7.35	5.23
F22	11.80	9.38	9.7	12.51	**6.79**	22.39	13.35	9.46	10.4	7.51	8.85	6.93
F23	16.78	12.23	12.49	17.13	**9.38**	27.47	18.58	12.23	13.69	9.98	10.79	9.76
sum	71.77	251.05	247.09	84.37	70.84	439.18	231.83	245.18	261.25	74.92	110.99	113.85
ave	3.12	10.92	10.74	3.67	3.08	19.09	10.08	10.66	11.36	3.26	4.83	4.95

Table 4.5 shows the average run time of the algorithms in seconds. DE has the best run time, CA is the second-best run time. JAYA is third. Especially, in F1, F2, F4, F5, F6, F7, F9, F10, F11, F12, and F13 functions, the CA algorithm has faster than other algorithms.

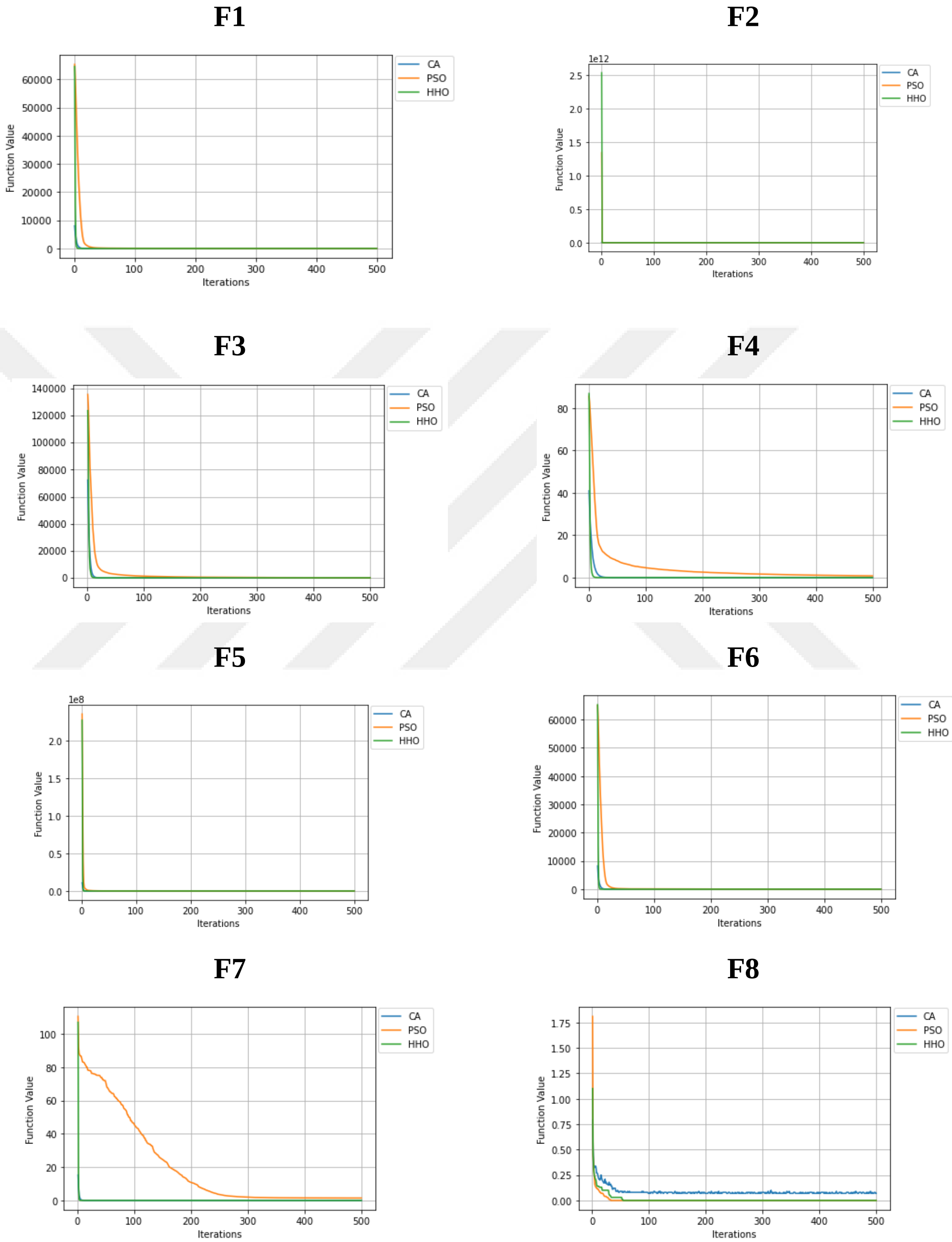

Figure 4.1 Convergence curve of CA, PSO, and HHO

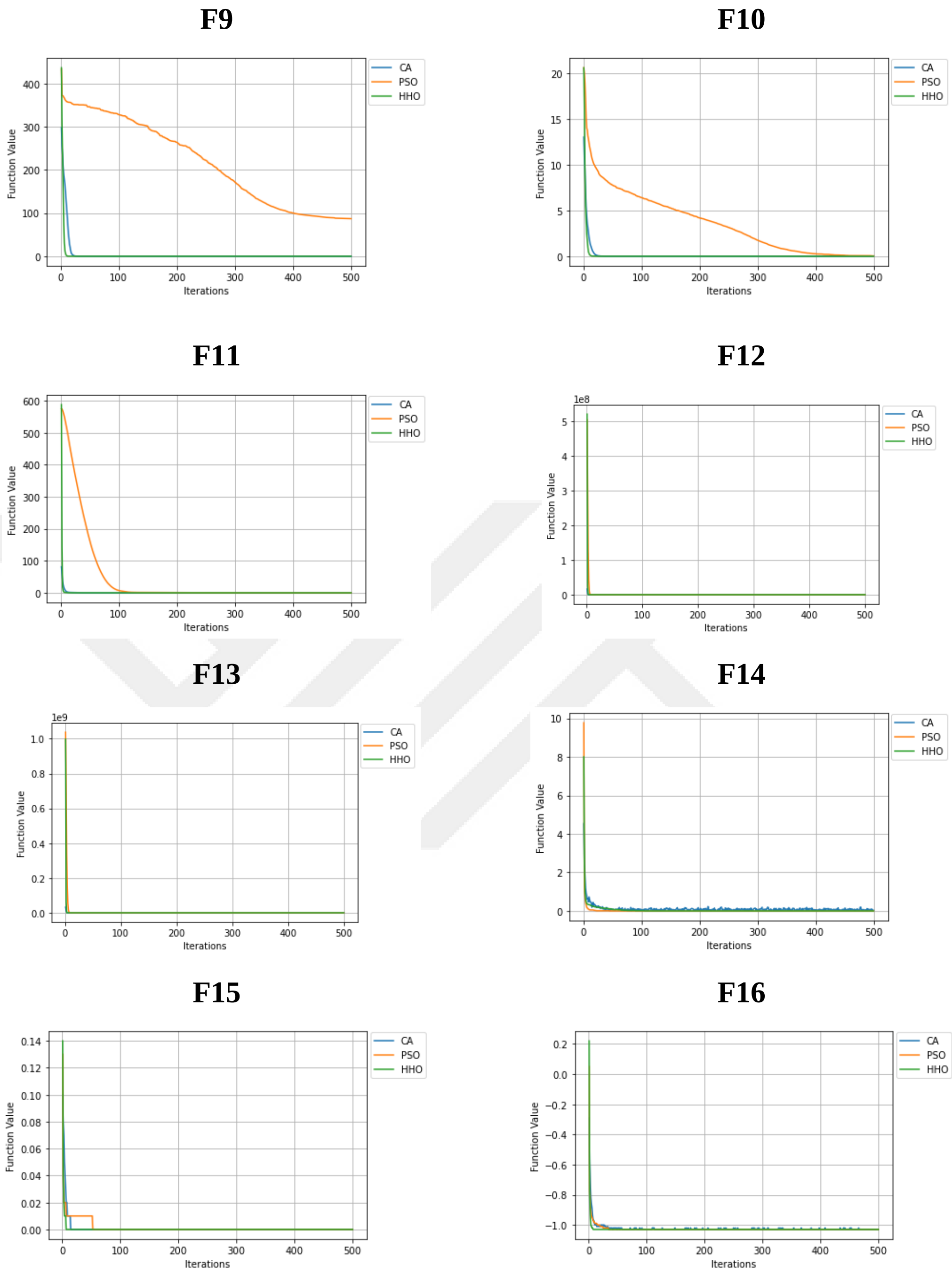

Figure 4.1 Convergence curve of CA, PSO, and HHO (continued)

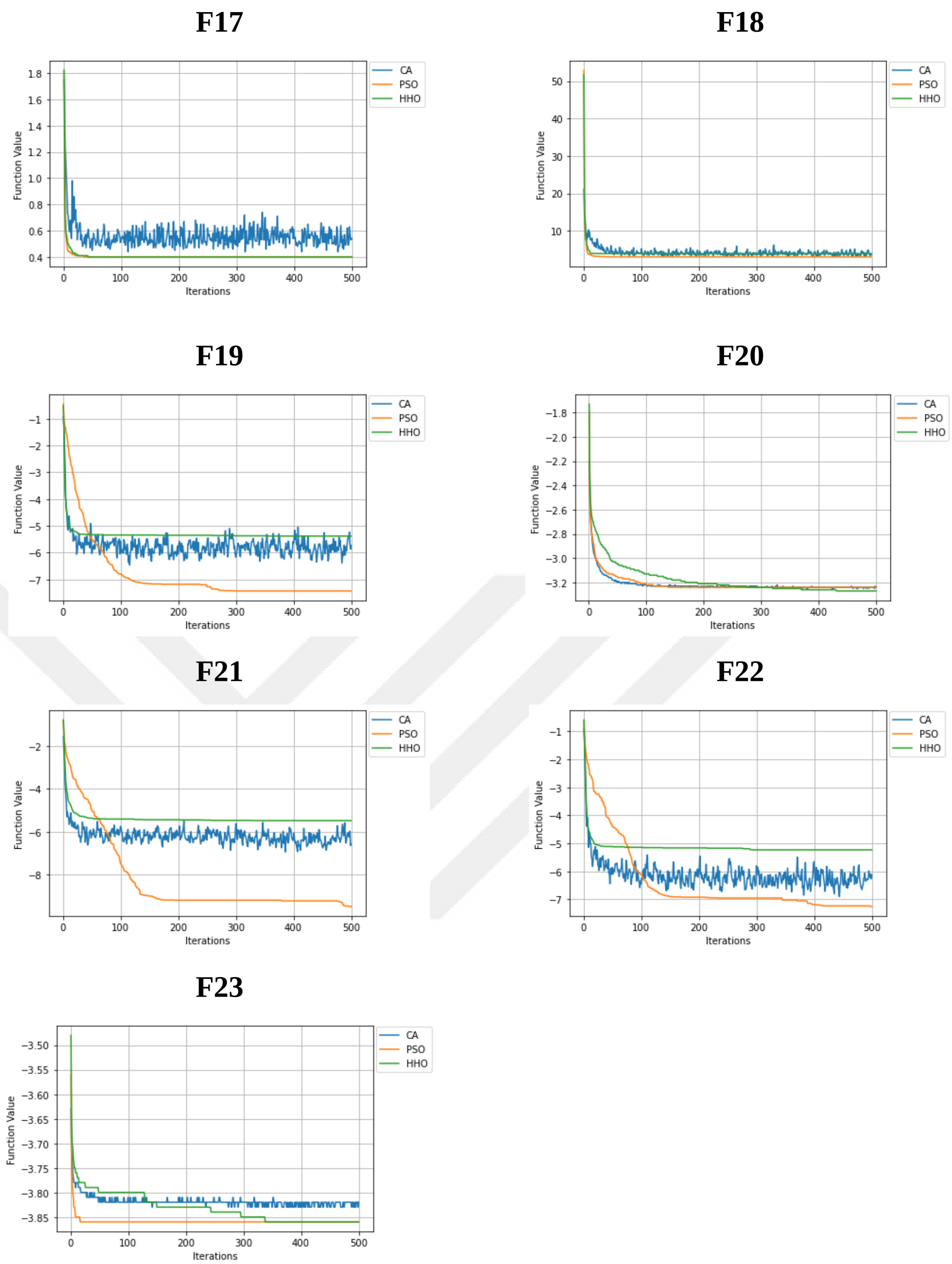

Figure 4.1 Convergence curve of CA, PSO, and HHO (continued)

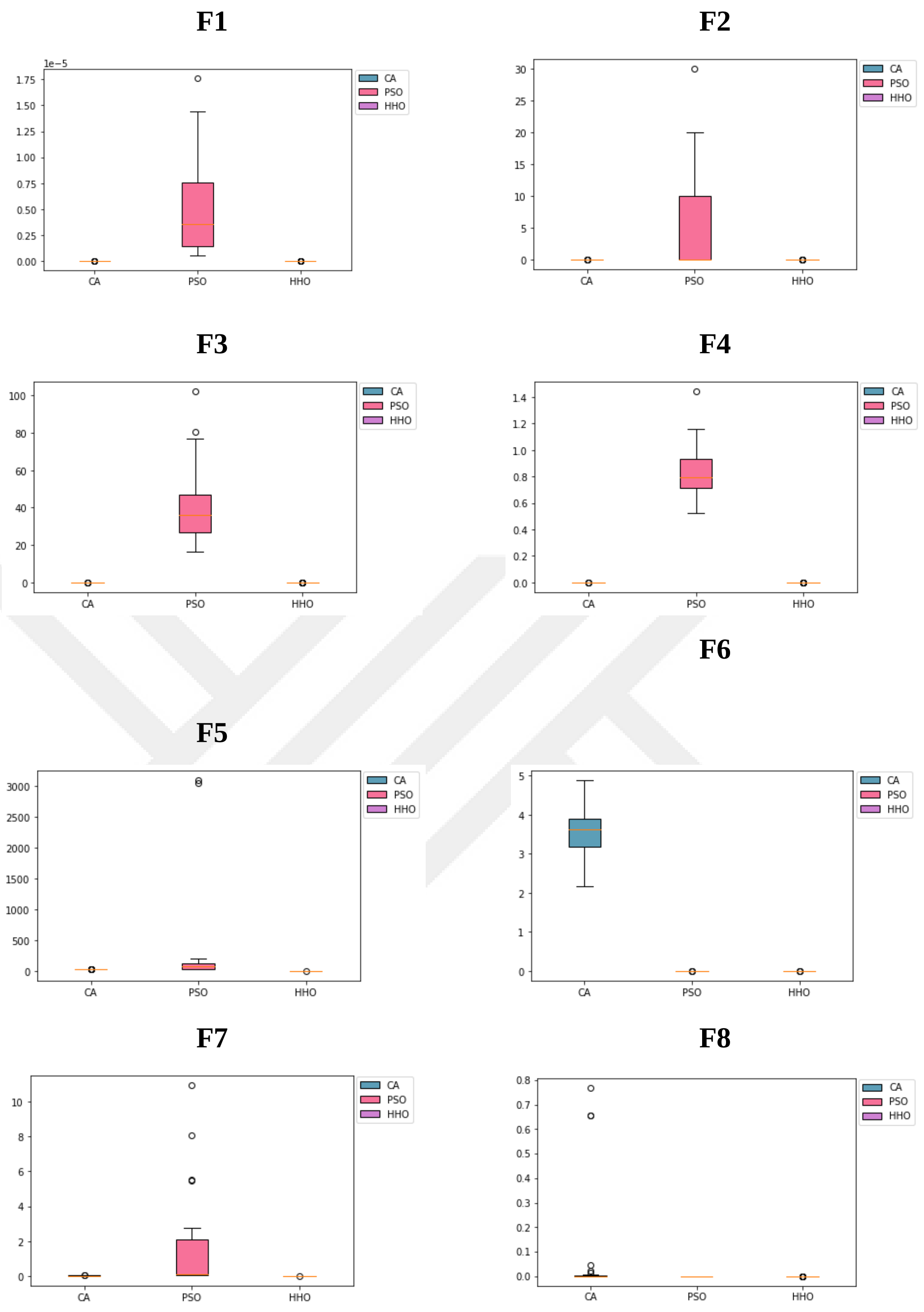

Figure 4.2 Box plots of CA, PSO, and HHO

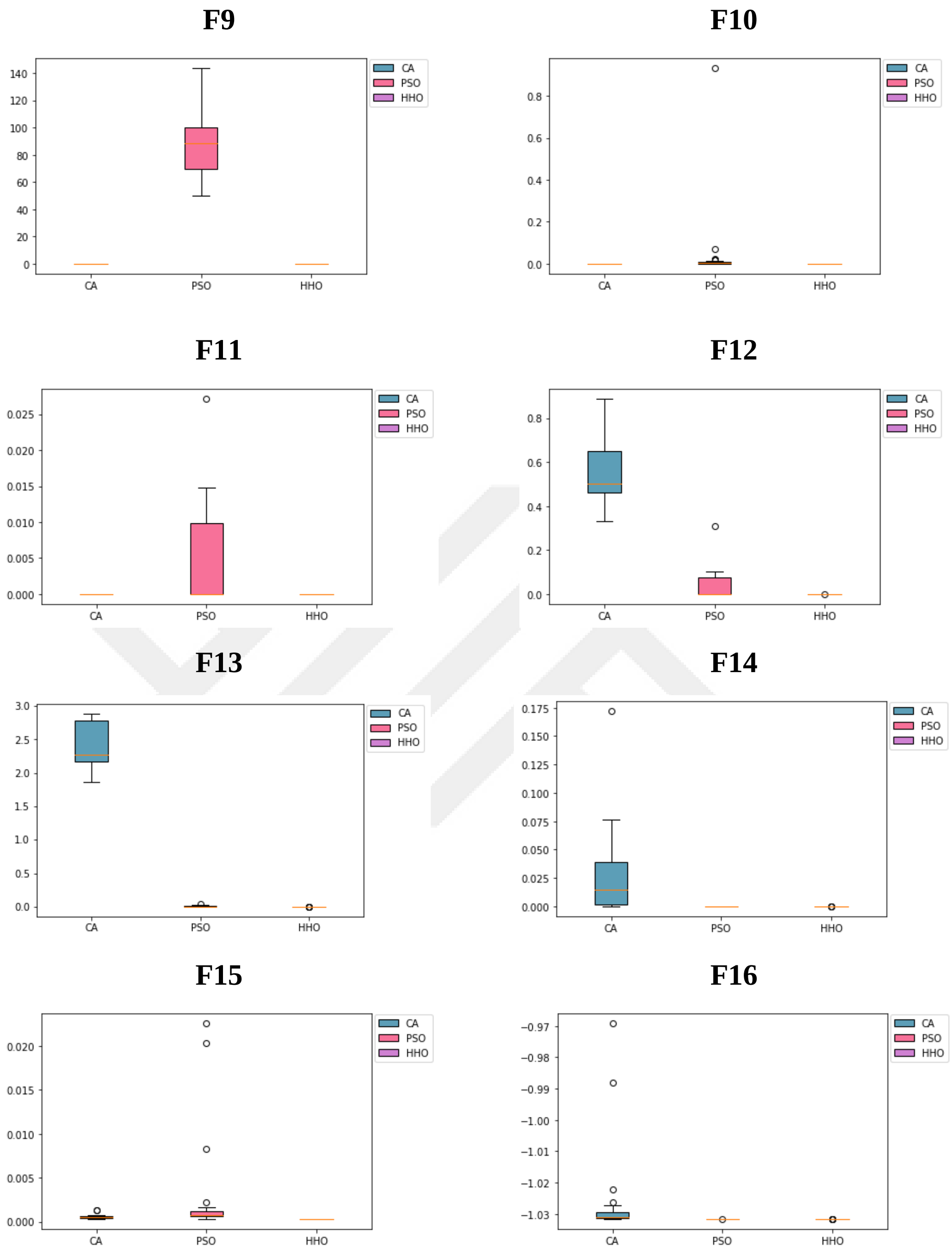

Figure 4.2 Box plots of CA, PSO, and HHO (continued)

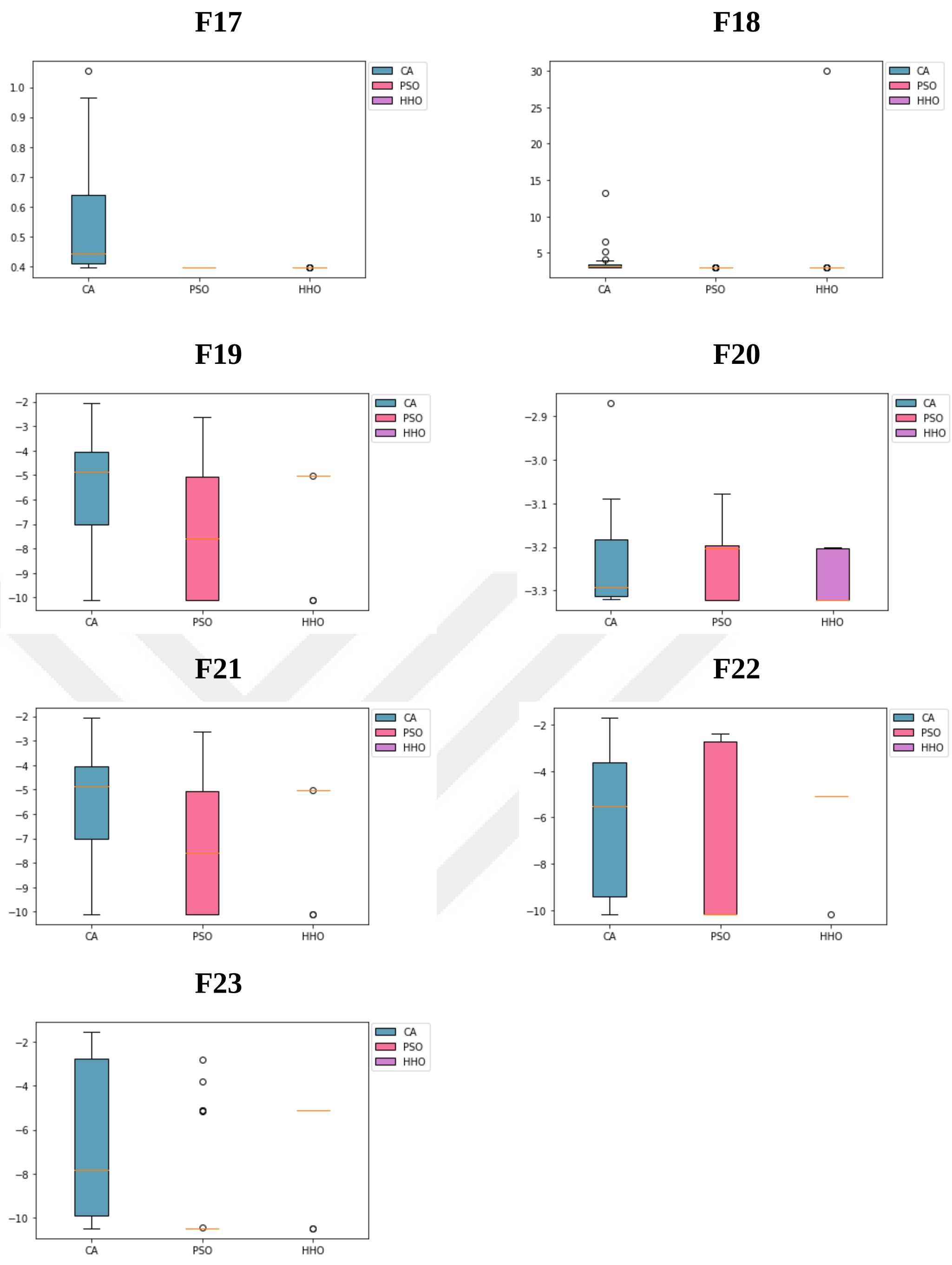

Figure 4.2 Box plots of CA, PSO, and HHO (continued)

Figure 4.1 shows the Convergence Curve of CA, PSO, and HHO. CA, PSO, and HHO are the convergence of algorithms that had the lowest three values according to the MSE calculations of the algorithms. The most noticeable detail in the graphs is that the CA algorithm has a lot of vibration in F17, F19, F21, F22 functions. CA has a converge problem to the best in these functions. Figure 4.2 shows the box plots of

CA, PSO, and HHO. The noticeable detail seen in Figure 4.1 can also be seen in Figure 4.2.

In this thesis, the performance analysis of CA was performed for unconstrained continuous problems. Its performance should also be tested in problems such as constrained, discrete to CA can be among successful algorithms in the literature. Also, it should be tested for real-world problems like job scheduling, assignment problems.

The differences of CA from compared algorithms can be listed as follows:

- The compared algorithms are generally swarm-based algorithms. The solutions are represented such as the particle in PSO, firefly in FFA, the grey wolf in GWO. In CA, solutions are represented as a vessel that goes from the liver to the tissues.

- The other algorithms' solution quality is determined with the positions of particles. In CA, it is determined with cholesterol levels of solution.

- While the other algorithms' structures are more complex, especially CA's structure was kept simple.

- In this thesis, a system has been established on actual cholesterol levels for the CA algorithm. The success of CA algorithm is acceptable with actual cholesterol levels.

The inverse ratio in HDL and LDL levels used in CA can be similar to the reward-penalty system in reinforcement learning. For example, The closer the solution is to the best solution, the higher the amount of HDL or the reward. The worse it is, the higher the amount of LDL or the penalty. This feature can be used for artificial intelligence applications.

Cholesterol ranges such as low, good, and high used to determine the quality of the solutions are similar to the logic of fuzzy. This feature can be used with fuzzy applications.

The vascular system used in CA can be easily adapted to distance problems like TSP problems.

As an extension of CA, a more effective algorithm can be developed by combining the cholesterol mechanism with systems that are important for the human body such as the circulatory system.

CHAPTER V

CONCLUSION

A problem can have hundreds of solutions. Finding the optimum is a topic that researchers have been working on for years. Various methods were developed, but these methods became insufficient as the size, constraints, and objectives of the problem changed. The universe, nature, or animals can solve their problems very easily and in a short time without consciousness. This situation attracts the attention of researchers. For this reason, they have developed many new methods inspired by the universe, nature, or animals, and they continue to do so. These methods are called metaheuristics. Although metaheuristics could not obtain exact results, they were able to produce approximate results in a shorter time. Time is one of the most important limited resources in today's world. For this reason, finding an approximate solution in a shorter time is an important advantage for the method. It is also an important issue that the metaheuristics developed can be easily adapted to many different problems. This is achieved by making the problem.

Today's world problems are also more complex and multidimensional. It is the most vital issue for the method to approach the correct solution in this complexity and hayloft. Various algorithms have been developed inspired by many disciplines in literature. However, an algorithm that holds all the advantages in the subjects listed has not been found yet. For this reason, new algorithms with different features are added to the literature and the search for this algorithm continues by trying different methods in the existing algorithms.

In this thesis, it is aimed to add a new algorithm to the literature. In the literature, there are very few algorithms such as ANN and AIS that have been developed with inspiration from the health field. A new health-based algorithm has been developed using the levels of cholesterol, which is vital in the human body. First of all, a short background about optimization and metaheuristics is given in the thesis. Then, by mentioning the process in which an algorithm is developed, examples from other algorithms in the literature are given. The functions of cholesterol in the human body and the properties of the lipoproteins they are transported are mentioned. It has been described how CA is modeled by inspiration from lipoproteins in the human body.

Lipoproteins that carry cholesterol in the human body are HDL, LDL, and triglycerides. There are certain levels of these lipoproteins that should be in a healthy person. For example, when LDL goes above a certain amount and HDL goes below a certain amount, clogging occurs in the vessels. These lipoproteins must be at certain levels to have a healthy blood flow in the vessel. This situation has been imitated in CA. If the solution is getting better, the amount of LDL decreases, and the amount of HDL increases. Even when the solution worsens, the amount of HDL decreases while the amount of LDL increases. Whether the solution goes for better or worse is determined by the distance of the solution to the best and worst solutions.

The success of the CA algorithm in 23 unconstrained continuous optimization problems was compared with 11 algorithms in the literature. For this, the best, average, and standard deviation values found by the algorithms in the functions, and run times were used. In addition, Wilcoxon signed-rank test was performed to make binary comparisons of the CA algorithm with other algorithms. In addition, the convergence curve and box-plot graphics of the CA were shown with the HHO and PSO algorithms, which had the lowest two values according to the MSE calculations of the algorithms.

CA algorithm was tested only in unconstrained continuous problems in this thesis. Its performance in other problem types like discrete, real-world problems is unknown. CA algorithm had proven its exploitation capability in unimodal functions, but multimodal functions did not achieve superior success to other algorithms. It has a convergence problem in some multi-nonseparable problems and it doesn’t have a more controlled exploration-exploitation mechanism. However, it can compete with other algorithms with its simple structure. In addition, although it did not show much superior success compared to other algorithms in unconstrained continuous optimization problems, it found results close to best-known answers in a shorter time than most of the algorithms.

Finally, for future research, the performance of the CA algorithm will be tested in other different problem types and real-world applications. The parallel structure of the CA and its hybrid states, combining good features of other algorithms, will also be tested.

REFERENCES

[1] Winston, W. L. (2004). Operations Research: Applications and Algorithms, Forth Edition. Hoboken, New Jersey: John Wiley & Sons, Inc.

[2] Gorissen, B. L., Yanikoğlu, I., and den Hertog D. (2015). A practical guide to robust optimization, *Omega-international Journal of Management Science,* **53,** 124-137. doi: 10.1016/j.omega.2014.12.006.

[3] Lodwick, W. A., and Bachman, K. A. (2005). Solving large-scale fuzzy and possibilistic optimization problems, *Fuzzy Optim. Decis. Mak.*, , 257–278 . doi: 10.1007/s10700-005-3663-4.

[4] Yang, X. S. (2010). Engineering Optimization: An Introduction with Metaheuristic Applications. Hoboken, New Jersey: John Wiley & Sons, Inc..

[5] Darwish, A. (2018). Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications, *Futur. Comput. Informatics J.*, **3(2),** 231-246. doi: 10.1016/j.fcij.2018.06.001.

[6] Yang X.S. (2012). Nature-Inspired Mateheuristic Algorithms: Success and New Challenges. *J Comput. Eng. Inf. Technol.*, **1(1),** 1-3. doi:10.4172/2324-9307.1000e101

[7] Odili, J. B., Noraziah, A., Ambar, R.,Wahab, M. H. A. (2018). A Critical Review of Major Nature-Inspired Optimization Algorithms, *Eurasia Proc. Sci. Technol. Eng. Math.,* **2,** 376-394.

[8] Gogna A., Tayal, A. (2013). Meta-heuristics: Review and application, *J. Exp. Theor. Artif. Intell.*, **25(4)**, 503–526. doi: 10.1080/0952813X.2013.782347.

[9] Abdel-Basset, M., Abdel-Fatah, L., Sangaiah, A.K. (2018). Metaheuristic algorithms: A comprehensive review, *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications.* 185-231. https://doi.org/10.1016/B978-0-12-813314-9.00010-4

[10] Sörensen K., Glover F. W. (2013). Metaheuristics, *Encyclopedia of Operations Research and Management Science*, .

[11] Holland, J. H. (1975). Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[12] Kirkpatrick, S., Gelatt, C. D., Vecchi, M.P. (1983). Optimization by simulated annealing. *Science (80-.),* **220(4598),** 671-680. doi: 10.1126/science.220.4598.671.

[13] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, *Comput. Oper. Res.,* **13(5)**, 533-549. doi: 10.1016/0305-0548(86)90048-1.

[14] Colorni, A., Dorigo, M., Maniezzo, V. (1992). Distributed Optimization by ant colonies.

[15] Koza, J. (1992). Genetic programming: on the programming of computers by means of natural selection.

[16] Dasgupta, D. (1993) An Overview of Artificial Immune Systems and Their Applications, *Artificial Immune Systems and Their Applications,* 3-21. https://doi.org/10.1007/978-3-642-59901-9_

[17] Kennedy J., Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-International Conference on Neural Networks,* **4,** 1942-1948.

[18] Storn R., Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **11,** 341-359. doi: 10.1023/A:1008202821328.

[19] Geem, Z. W., Kim, J. H., Loganathan, G.V. (2001). A New Heuristic Optimization Algorithm: Harmony Search, *Simulation.,* **76(2)**, 60-68. doi: 10.1177/003754970107600201.

[20] Passino, K. M. (2002). Biomimicry of Bacterial Foraging for Distributed Optimization and Control, *IEEE Control Syst.,* **22(3),** 52-67. doi: 10.1109/MCS.2002.1004010.

[21] Karaboga, D. (2005). An idea based on Honey Bee Swarm for Numerical Optimization, *Tech. Rep. TR06, Erciyes Univ.*

[22] Mehrabian A. R. Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization, *Ecol. Inform.*, **1(4),** 355-366 doi: 10.1016/j.ecoinf.2006.07.003.

[23] Simon, D. (2008). Biogeography-based optimization, *IEEE Trans. Evol. Comput.,* **12(6)**, 702-713. doi: 10.1109/TEVC.2008.919004.

[24] Yang X.S., Deb, S. (2009). Cuckoo search via Lévy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore,* 210-214 doi: 10.1109/NABIC.2009.5393690.

[25] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm, *Inf. Sci. (Ny).* **179,** 2232–2248. doi: 10.1016/j.ins.2009.03.004.

[26] Yang, X. S. (2009). Firefly algorithms for multimodal optimization., **5792,**

169-178. doi: 10.1007/978-3-642-04944-6_14.

[27] Yang, X. S. (2010). A new metaheuristic Bat-inspired Algorithm, **284,** 65-74. doi: 10.1007/978-3-642-12538-6_6.

[28] Rao, R. V., Savsani, V. J., Vakharia, D. P. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *CAD Comput. Aided Des.,* **43,** 303-315. doi: 10.1016/j.cad.2010.12.015.

[29] Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M. (2012). Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.*, **110-111,** 151-166. doi: 10.1016/j.compstruc.2012.07.010.

[30] Yang, X. S. (2012). Flower pollination algorithm for global optimization. **7445**, 240–249. doi: 10.1007/978-3-642-32894-7_27.

[31] Rajakumar, B. R. (2012). The Lion's Algorithm: A New Nature-Inspired Search Algorithm, *Procedia Technol.*, **6,** 126-135. doi: 10.1016/j.protcy.2012.10.016.

[32] Mirjalili, S., Mirjalili, S. M., Lewis, A. (2014). Grey Wolf Optimizer, *Adv. Eng. Softw.,* **69,** 46-61 doi: 10.1016/j.advengsoft.2013.12.007.

[33] Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowledge-Based Syst.,* **89,** 228-249. doi: 10.1016/j.knosys.2015.07.006.

[34] Mirjalili S., Lewis, A. (2016). The Whale Optimization Algorithm, *Adv. Eng. Softw.,* **95,** 51-67. doi: 10.1016/j.advengsoft.2016.01.008.

[35] Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems, *Knowledge-Based Syst.*, **9,** 120-133. doi:

10.1016/j.knosys.2015.12.022.

[36] Mirjalili, S., Mirjalili, S.M., Hatamlou, A. (2016). Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. *Neural Comput. Appl.,* **27,** 495–513 doi: 10.1007/s00521-015-1870-7.

[37] Uymaz, S.A., Tezel, G., Yel, E. (2015). Artificial algae algorithm (AAA) for nonlinear global optimization, *Appl. Soft Comput. J.* **31,** 153-171. doi: 10.1016/j.asoc.2015.03.003.

[38] Li M. D., Zhao H., Weng X. W., and Han T. (2016). A novel nature-inspired algorithm for optimization: Virus colony search, *Adv. Eng. Softw.*, **92,** 65-88. doi: 10.1016/j.advengsoft.2015.11.004.

[39] Venkata, R. R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, *Int. J. Ind. Eng. Comput.*, **7,** 19-34. doi: 10.5267/j.ijiec.2015.8.004.

[40] Kiran, M. S. (2015). TSA: Tree-seed algorithm for continuous optimization, *Expert Syst. Appl.*, **42,** 6686-6698. doi: 10.1016/j.eswa.2015.04.055.

[41] Askarzadeh, A.(2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, *Comput. Struct.*, **169,** 1-12. doi: 10.1016/j.compstruc.2016.03.001.

[42] Mirjalili S., Gandomi A. H., Mirjalili S. Z., Saremi S., Faris H., and Mirjalili S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Softw.,* **114,** 163-191. doi: 10.1016/j.advengsoft.2017.07.002.

[43] Heidari A. A., Mirjalili S., Faris H., Aljarah I., Mafarja M., and Chen H. (2019). Harris hawks optimization: Algorithm and applications, *Futur. Gener. Comput. Syst.,* **97,** 849-872. doi: 10.1016/j.future.2019.02.028.

[44] Talbi, E. G. (2009). Metaheuristics: From Design to Implementation.

[45] Yang, X.S., Deb, S., Fong, S. (2014). Metaheuristic algorithms: Optimal balance of intensification and diversification. *Appl. Math. Inf. Sci.,* **8(3)**, 977-983. doi: 10.12785/amis/080306.

[46] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man, Cybern. Part B Cybern., **26(1),** 29-41 doi: 10.1109/3477.484436.

[47] Hussain, K., Mohd Salleh, M. N., Cheng, S., Naseem, R. (2017). Common Benchmark Functions for Metaheuristic Evaluation: A Review. *JOIV Int. J. Informatics Vis.,* **1(4-2)**, 218- 223. doi: 10.30630/joiv.1.4-2.65.

[48] Jamil, M., Yang, X.S. (2013). A literature survey of benchmark functions for global optimisation problems, *Int. J. Math. Model. Numer. Optim.,* **4(2),** 150-194,. doi: 10.1504/IJMMNO.2013.055204.

[49] Faris H., Aljarah I., Mirjalili S., Castillo P.A., and J. J. Merelo J. J. (2016), EvoloPy: An open-source nature-inspired optimization framework in python, **1,** 171-177. doi: 10.5220/0006048201710177.

[50] Khurma R. A., Aljarah I., A. Sharieh A., and Mirjalili S. (2020). EvoloPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection,175-182. doi: 10.5220/0008960701750182

[51] Parejo, J.A., Ruiz-Cortés, A., Lozano, S., Fernandez, P. (2012). Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Comput.,* **16**, 527–561. doi: 10.1007/s00500-011-0754-8

[52] Yang X. S. (2013). Bat algorithm: Literature review and applications, *International Journal of Bio-Inspired Computation.,* **5(3),** 141–149. doi: 10.1504/IJBIC.2013.055093.

[53] Joshi A. S., Kulkarni O., Kakandikar G. M., and Nandedkar V. M. (2017). Cuckoo Search Optimization- A Review, **4,** 7262–7269. doi: 10.1016/j.matpr.2017.07.055.

[54] Binitha S. and Sathya S. S. (2012). A Survey of Bio inspired Optimization Algorithms, *Int. J. Soft Comput. Eng.*, **2(2)**, 137-151.

[55] Fister I., Yang X. S., and Brest J. (2013). A comprehensive review of firefly algorithms, *Swarm Evol. Comput.,* **13,** 34-46. doi: 10.1016/j.swevo.2013.06.001.

[56] Karaboga D., Gorkemli B., Ozturk C., and Karaboga N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications, *Artif. Intell. Rev.,* **42,** 21–57. doi: 10.1007/s10462-012-9328-0.

[57] WHO, (2002). Chapter 7 High cholesterol, *Who*.

[58] Cholesterol, chemical compound, Britannica. https://www.britannica.com/science/cholesterol, 12.02.2021.

[59] Bloch, K. (1991). Chapter 12 Cholesterol: Evolution of structure and function, *New Compr. Biochem.*, **20,** 363–381. doi: 10.1016/S0167-7306(08)60340-3.

[60] Birtcher K. K., and C. M. Ballantyne, C. M. (2004). Measurement of Cholesterol, *Circulation*, **110(11),** 296–297. doi: 10.1161/01.cir.0000141564.89465.4e.

[61] Jackson, R. L. , Morrisett, J. D. , and Gotto, A. M. (1976). Lipoprotein structure and metabolism, *Physiological Reviews*., **56(2),** 259-316. doi: 10.1152/physrev.1976.56.2.259.

[62] Ansell, B. J., Watson, K. E., Fogelman, A. M., Navab, M., and Fonarow, G. C. (2005). High-density lipoprotein function: Recent advances, *Journal of the*

American College of Cardiology, **46(10),** 1792-1798. doi: 10.1016/j.jacc.2005.06.080.

[63] Fisher, E. A., Feig, J. E., Hewing, B., Hazen, S. L., and Smith, J. D. (2012). High-density lipoprotein function, dysfunction, and reverse cholesterol transport," *Arterioscler. Thromb. Vasc. Biol.*, **32(12)**, 2813-2820. doi: 10.1161/ATVBAHA.112.300133.

[64] Barter *et al.* P. (2007). HDL Cholesterol, Very Low Levels of LDL Cholesterol, and Cardiovascular Events, *N. Engl. J. Med.*, **357**, 1301-1310. doi: 10.1056/nejmoa064278.

APPENDIX

CA PYTHON CODE

```
# Import required packages
import math
import numpy
import random
import time
from scipy.spatial import distance
from solution import solution

# Definition of used variables
# objf : functions
# lb : lower bound
# ub : upper bound
# dim : dimension
# N : number of vein
# Max_iteration : the number of iteration

# Define CA function with parameters
def CA(objf, lb, ub, dim, n, Max_iteration):
    if not isinstance(lb, list):
        lb = [lb] * dim
    if not isinstance(ub, list):
        ub = [ub] * dim
#Define matrices of solutions and function values
    Sol = numpy.zeros((n, dim))
    new_sol= numpy.zeros((n, dim))
    Func= numpy.zeros(n)
# Initiliaze solution sets and evaluate
    for i in range(dim):
        Sol[:, i] = numpy.random.rand(n) * (ub[i] - lb[i]) + lb[i]
    for i in range(0, n):
        Func[i] = objf(Sol[i, :])
    s = solution()
# Use to measure the elapsed time
    timerStart = time.time()
    s.startTime = time.strftime("%Y-%m-%d-%H-%M-%S")
```

```
# Hold old values and find new best values
    old_func = Func.copy()
    new_func = numpy.zeros(n)
    globalbest = min(old_func)
 # Start iterations
    for i in range(Max_iteration):
        # Define vectors for local best and local worst values
        temp_fitness = []
        local_best = min(old_func)
        local_worst = max(old_func)
        local_worsti= numpy.argmax(old_func)
        local_besti= numpy.argmin(old_func)
# Decide to local best value is lower than global best value or not
        if globalbest>= local_best:
            globalbest=local_best
            best_sol=curr_sol[local_besti,:]
            global_besti=local_besti
  # Testing each vine
        for j in range(n):
            if local_worst!=local_best:
                # Evaluate and optimize HDL and LDL values
               LDL=160*abs(old_fitness[j]-local_best)/abs(local_worst-local_best)
               HDL = 40 + 40 * abs(local_worst-old_fitness[j])/abs(local_worst-
local_best)
                Trig =numpy.random.uniform(0,1)*40
                Total_chol= LDL+HDL+Trig
                cho_leveli[j,0] = Total_chol
                cho_leveli[j,1] = HDL
                cho_leveli[j,2] = Trig/Total_chol
                cho_leveli[j,3] = LDL
            else:
                pass
            if cho_leveli[j,0] >= 150:
                if cho_leveli[j,3] >= 100 and cho_leveli[j,1] <= 50:
                   new_sol[j,:] = best_sol + best_sol *
(cho_levelb[2])*numpy.random.uniform(-1,1,dim)
                else:
                   new_sol[j,:] = curr_sol[j,:] - (curr_sol[j,:] * ((cho_leveli[j,3] /
cho_leveli[j,1]) - (cho_leveli[j,2])))
            else:
                if cho_leveli[j,3] >= 100 and cho_leveli[j,1] <= 50:
                   new_sol[j,:] = best_sol + best_sol *
(cho_levelb[2])*numpy.random.uniform(-1,1,dim)
                else:
                      new_sol[j,:] = curr_sol[j,:] + (curr_sol[j,:] * ((cho_leveli[j,3] /
cho_leveli[j,1]) - (cho_leveli[j,2])))
            new_func[j] = (objf(new_sol[j,:]))
            temp_func.append(objf(new_sol[j])
    return s
```