A

Report on

Design and Implementation of Cheetah Optimizer Algorithm for Solving Optimization Problem

Submitted in fulfillment for

Bachelor of Technology
in
Information Technology

Submitted by

Mr. Rahul Kumar	**PRN: 2019BTEIT00002**
Mr. Pratik Umesh Pawar	**PRN: 2019BTEIT00006**
Miss. Arundhati Jagdish Wandhekar	**PRN: 2019BTEIT00013**
Miss. Pranjali Vasant Nanaware	**PRN: 2019BTEIT00067**
Miss. Sakshi Mohan Kamble	**PRN: 2020BTEIT00205**

Under the guidance of

Dr. A. J. Umbarkar

Department of Information Technology,

Walchand College of Engineering, Sangli

Maharashtra, India. 416415

2022-23

DECLARATION

We, hereby declare that the dissertation report entitled **"Design and Implementation of Cheetah Optimizer Algorithm for Solving Optimization Problem"** submitted by us to **Walchand College of Engineering, Sangli** in fulfillment of the requirement for the award of the degree of **B.Tech** in **Information Technology** is a record of bonafide project work carried out by us under the guidance of **Dr . A. J. Umbarkar.**

We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. We declare that this dissertation report reflects our thoughts about thesubject in our own words. We have sufficiently cited and referenced the original sources, referred to or considered in this work. We have not misrepresented or fabricated or falsified anyidea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute.

Name: Rahul Kumar

PRN: 2019BTEIT00002

Name: Pratik Umesh Pawar

PRN: 2019BTEIT00006

Name: Arundhati Jagdish Wandhekar

PRN: 2019BTEIT00013

Name: Pranjali Vasant Nanaware

PRN: 2019BTEIT00067

Name: Sakshi Mohan Kamble

PRN: 2020BTEIT00205

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. A. J. Umbarkar for his guidance and continuous support, encouragement, and help extended at every stage of this project work. We express our gratitude and earnest thanks to Dr. A. J. Umbarkar, Head of the Information Technology Department, Walchand College of Engineering, Sangli for providing us with all facilities throughout the project work. We extend our sincere thanks to our parents for providing us with all possible help and inspiration

Last, but not least we extend our sincere thanks to other faculty members especially Prof. M. B. Narnaware as well as non-teaching staff of the Department of Information Technology and to all our friends for their valuable advice in every stage of this project report.

CERTIFICATE

This is to certify that the project dissertation work entitled

"Design and Implementation of Cheetah Optimizer Algorithm for SolvingOptimization Problem"

Submitted by

Miss. Arundhati Jagdish Wandhekar

Mr. Rahul Kumar

Mr. Pratik Umesh Pawar

Miss. Pranjali Vasant Nanaware

Miss. Sakshi Mohan Kamble

In fulfillment of the requirement for the degree of

Bachelor of Technology
In
INFORMATION TECHNOLOGY

From

Walchand College of Engineering, Sangli
(An Autonomous Institute)

Affiliated to Shivaji University, Kolhapur

This project dissertation work is a record of students' own work carried out by them under mysupervision and guidance during the session 2022-23.

Dr. A. J. Umbarkar (Project Guide)	External Examiner	Dr. R. R. Rathod (HOD)

Abstract

Optimization problems are NP-Hard problems. Complexity of these problems cannot be expressed in polynomial time. Manual calculation of NP-Hard problem is not possible, but validation of the results is easy. For the calculation of such problems computational power is needed. This calculation should be optimized. In this project, 2 NP-Hard problems have been solved using Cheetah Optimization algorithm (COA). These problems are:

1. **ELD**: ELD problem entails the optimization of generator power output, considering real and reactive power, within specified boundaries to meet load demands at minimal fuel cost, while minimizing the occurrence of copied content.
2. **Sphere Function Problem**: The sphere function problem is a suitable example of a single-objective optimization task where a function is involved with a singular objective. It exhibits a unimodal characteristic, indicating the presence of a primary mode and a single global optimum.

The results obtained by applying COA on these problems were better, compared with the results obtained from various algorithms used for solving these problems. This makes the use of COA broader in solving different optimization problems. Further, use of technologies such as Dockers, Hadoop and Sparks would suffice the need of handling large data and computation.

Table of Contents

DECLARATION i

ACKNOWLEDGEMENT ii

CERTIFICATE iii

Abstract iv

Table of Contents v

List of Figures vii

List of Tables viii

List of Abbreviations ix

Introduction 1

1.1 Overview and Problem Statement 1

1.1.1 Algorithm Optimization 2

1.1.2 Engineering Optimization 2

1.1.3 Greedy Algorithm 2

1.2 Objectives 2

2. Literature Review 3

2.1 Summary of chapter 3

3.Methodology 4

3.1 Cheetah Optimization Algorithm (COA) 4

3.1.1 Natural Process of the Cheetah Optimization Algorithm 4

3.1.2 Algorithm of COA [22] 6

3.1.3 Search strategy 7

3.1.4 Sit-and-wait strategy 7

3.1.5 Attack Strategy 7

3.1.6 Pseudo code Cheetah Optimization Algorithm(COA) 8

3.1.7 Flowchart of Cheetah Optimization Algorithm(COA) 9
3.2 Case Study 12
3.2.1 Case Study 1: Sphere Function 12
3.2.2 Case Study 2: Economic Load Dispatch problem 12
3.3 Summary of the chapter 13
4. Results and Discussion 14
4.1 CASE STUDY 1: Sphere Function (With random value input) 14
4.1.1 Calculation for iteration 1: 15
4.1.2 Calculations for iteration 2: 16
4.2 CASE STUDY 2: Sphere Function (input from TLBO) 19
4.2.1 Calculation for iteration 1: 19
4.2.2 Calculations of iteration 2 21
4.3 CASE STUDY 3: Economic Load Dispatch Problem 24
4.4 Summary of chapter 28
5. Conclusion 29
6. References 30
Appendix 32
A – Codes 32
Plagiarism Report 37
Vitae 38

List of Figures

1. Figure 3.1: Flowchart of COA according to research paper……12
2. Figure 3.2: Flowchart according to algorithm……14
3. Figure 3.2: Economic Load Dispatch problem……15
4. Figure 4.1: Graphical representation of 30 readings for 500 MW……...30
5. Figure 4.2: Graphical representation of 30 readings for 600 MW……...31
6. Figure 4.3: Graphical representation of 30 readings for 700 MW……...31

List of Tables

1. Table 2: Literature Survey of problems and Algorithms……...6
2. Table 3.1: Characteristics of COA…….8
3. Table 4.1: Initial values of variables for sphere function…….17
4. Table 4.2: Iteration 1 values of variables for sphere function……..19
5. Table 4.3: Iteration 2 values of variables for sphere function……..21
6. Table 4.4: Final values of variables for sphere function……..21
7. Table 4.5: Initial values of variables for sphere function…….22
8. Table 4.6: Result of iteration 1…….24
9 Table 4.7: Iteration 2 values of variables for sphere function……...26
10. Table 4.8: Final values of variables for sphere function……26
11. Table 4.9: Input Parameters of Algorithm……..27
12. Table 4.10: Comparison of generator results with various optimization Algorithms……27
13. Table 4.11: Comparison of Fuel cost results with various optimization Algorithms…….28
14. Table 4.12: 30 readings for 500 MW demand……28
15. Table 4.13: 30 readings for 600 MW demand……....29
16. Table 4.14: 30 readings for 700 MW demand……30

List of Abbreviations

COA	Cheetah Optimization Algorithm
ELD	Economic Load Dispatch
GWO	Grey Wolf Optimizer
OA	Optimization Algorithm
BOA	Butterfly optimization algorithm

Introduction

The Cheetah Optimization Algorithm (COA) is a recently developed population-based optimization technique that draws inspiration from the hunting behavior of cheetahs. By mimicking the speed and agility of cheetahs during hunting, the COA efficiently explores complex optimization problems to find optimal solutions. As a metaheuristic algorithm, the COA does not rely on explicit mathematical models of the problem at hand. Instead, it employs a population of candidate solutions and iteratively enhances them through a combination of local and global search strategies.

One of the key features of the COA is its capacity to achieve a harmonious equilibrium between explore and exploit. This is accomplished by combining random exploration and guided exploitation methods, ensuring that the algorithm avoids being confined to explores the entirety of the solution space. Furthermore, the COA excels at handling high-dimensional optimization problems. Many real-world optimization challenges involve numerous variables, making it difficult for traditional methods to find optimal solutions. The COA addresses this by utilizing a dynamic search space partitioning approach, breaking down the high-dimensional space into smaller sub-spaces.

The effectiveness of the COA has been demonstrated through extensive testing on various benchmark optimization problems. By effectively managing the trade-off between explore and exploit, the COA algorithm demonstrates its capacity to strike a balance, as well as its proficiency in handling high-dimensional spaces, the COA shows great promise as an optimization algorithm that can significantly enhance efficiency and effectiveness in a wide range of applications.

1.1 Overview and Problem Statement

“Design and Implementation of Cheetah Optimizer Algorithm for Solving Optimization Problem”

Main topic- Algorithm Optimization

Subtopic- Solve Real World Problem

Many algorithms have been used to solve the optimization problems. Every algorithm gives adifferent output. Most efficient optimization algorithm should be used to solve the requiredproblem. Here chosen problem is Economic Load Dispatch. In the proposed project, results of this problems using Cheetah Optimization Algorithm (COA) are compared with results of other algorithms.

1.1.1 Algorithm Optimization

Engineering design problems are prevalent in both academic research and industrial applications across various research disciplines. These problems often require the utilization of optimization algorithms to find optimal solutions. However, as the scale and complexity of these problems increase, the performance of traditional optimization algorithms tends to decline.

To address these challenges, researchers have proposed several specialized versions and adaptations of optimization methods tailored specifically for engineering design problems. These algorithms aim to improve the efficiency and effectiveness of optimization in these specific contexts.

1.1.2 Engineering Optimization

Engineering optimization, also known as design optimization, is a field that utilizes optimization techniques to achieve desired design goals in engineering. Optimization involves finding the best possible solution from a set of feasible solutions. Problems of optimization can be classified into various sub-categories, including constrained optimization, unconstrained optimization, unimodal optimization, multimodal optimization, composite optimization, and more.

1.1.3 Greedy Algorithm

A greedy algorithm is a problem-solving approach that focuses on selecting the most advantageous option in the present moment, without concerning itself with the future outcomes.

1.2 Objectives

1) To design Cheetah Optimization Algorithm (COA).
2) To implement COA for Optimization
3) To compare and analyze the performance of COA with other algorithms.

2. Literature Review

The Cheetah Optimization Algorithm (COA) has recently emerged as a population-based optimization algorithm drawn from the hunting behavior of cheetahs. It has gained attention in various research fields due to its unique characteristics and capabilities. In the field of engineering design optimization, Chen and Li proposed a modified version of the COA and applied it to optimize truss structures. Their study showcased that the COA exhibited superior convergence speed and solution quality compared to traditional algorithms. The COA's competence in handling high-dimensional optimization problems proved particularly advantageous in this context.

Sr. No.	Author	Work Carried Out
1	Ling Wang, Ji Pei, Yalan Wen, Jiaxing Pi, Minrui Fei, Panos M. Pardalos	An improved adaptive human learning algorithm for engineering Optimization [15]
2	J.S. Arora	Introduction to optimum Design [16]
3	Yuksel Celik, Hakan Kutucu	"Solving the tension/compression spring design problem by an improved firefly algorithm." Adaption of the neighborhood method to FA and propose an improved firefly algorithm (IFA) to solve a well-known engineering problem, the so-called Tension or Compression Spring Design. [17]
4	H. Liu, Z. Cai, Y. Wang	"Hybridizing particle swarm optimization with differential evaluation for constrained numerical and engineering optimization" Appl. Soft Comput. [18]
5	A. Bouzidi, M. Riffi	Cat Swarm Optimization to solve Flow Shop Scheduling Problem [19]
6	N. Bacarisas, J. Paul	The Effects of Varying the Fitness Function on the Efficiency of the CSO algorithm in solving the Graph Coloring Problem [20]

Table 2: Literature survey of Problems and Algorithms

2.1 Summary of chapter

In this chapter a basic introduction of COA and Economic Load Dispatch is given. The table above shows literature survey of COA, Economic Load Dispatch problem and Gear Train Problem.

3.Methodology

3.1 Cheetah Optimization Algorithm (COA)

Cheetah Algorithm is based on behaviors of cheetah namely searching, sit and waits andattack strategy. It is based on cheetahs that attack prey and find the best cheetahs among them as a leader.

3.1.1 Natural Process of the Cheetah Optimization Algorithm

The COA is an optimization algorithm drawn by the hunting behavior of cheetahs. It aims to solve optimization problems by simulating the hunting strategies and speed of cheetahs in nature. While there is no widely recognized algorithm with this specific name, I can provide you with a general description of how an optimization algorithminspired by cheetahs could work. Please note that this description is a hypothetical representationbased on the behavior of cheetahs and does not refer to a specific algorithm.

1. **Initialization**: Begin by initializing a population of potential solutions to the problem. Each solution represents a potential candidate solution.

2. **Speed and agility**: Cheetahs are known for their incredible speed and agility. In the optimization algorithm, each candidate solution could be associated with a velocity vector that determines its movement within the search space.

3. **Exploration and exploitation**: Cheetahs alternate between exploration (searching for prey) and exploitation (closing in for the kill). Similarly, the algorithm could employ different strategies for exploration and exploitation. During exploration, the candidate solutions would explore different regions of the search space, aiming to discover areas that may contain optimal solutions. Exploitation involves focusing on the most promising areas to refine the solutions.

4. **Fitness evaluation**: Evaluate the fitness of each candidate solution based on a fitness function that quantifies how well the solution satisfies the optimization objective. The fitness function is problem-specific and defines the criteria for determining the quality of solution.

5. **Hunting and chasing**: Cheetahs are skilled hunters and excel at chasing down their prey.

In the optimization algorithm, candidate solutions that demonstrate higher fitness are considered successful "prey" that should be pursued. Solutions with higher fitness are given more importance are selected for further exploration or exploitation.

6. **Adaptation and learning**: Cheetahs adapt their hunting strategies based on the success or failure of their previous attempts. Similarly, the algorithm could incorporate mechanisms for learning and adapting the velocity vectors associated with the candidate solutions. This adaptation could be based on the performance of the solutions and could involve adjusting the velocity vectors to guide the search towards better solutions.

7. **Iteration and convergence**: The optimization process continues through multiple iterations. During each iteration, the candidate solutions evolve, adapt, and explore the search space. Over time, the algorithm converges towards a solution that satisfies the optimization objective or reaches a stopping criterion.

It's important to note that the specific details and variations of a cheetah-inspired optimization algorithm may vary depending on the design choices made by the algorithm developer. The description above provides a general framework based on the behavior of cheetahs, but the actualimplementation details can differ.

General Algorithm	Cheetah Optimization Algorithm
Decision Variable	Cheetahs position in each dimension
Solution	Prey killed by each cheetah
Old solution	Cheetahs position in sit and wait strategy
New solution	New position of cheetah for attack
Best solution	Any cheetah with the best fitness
Fitness function	Distance between cheetah and prey
Initial solution	Random position of cheetah
Survival Selection	-
Process of generating new solution	Searching and attacking a prey

Table 3.1: Characteristics of COA[22]

3.1.2 Algorithm of COA [22]

When a cheetah is actively surveying or observing its environment, there is a possibility of detecting potential prey. Once the cheetah becomes aware of the prey's presence, it can consider its next move. Upon sighting the prey, the cheetah may choose to remain stationary and wait for the prey to come closer before initiating an attack. The attack phase involves two crucial steps: rushing and capturing. However, there are instances where the cheetah may abandon the hunt due to various factors, such as limited energy or the prey's rapid escape. In such cases, the cheetah may return home to rest and prepare for a new hunting attempt. The cheetah intelligently selects one of these strategies based on factors like prey assessment, its own condition, the area, and the distance to the prey. The COA algorithm leverages these hunting strategies during iterations to optimize the search process.

The strategies employed in the COA algorithm can be summarized as follows:

1. **Searching**: Cheetahs actively scan or search their territories or the surrounding area to locate prey. This corresponds to exploring the search space in the algorithm.

2. **Sitting-and-waiting**: If the prey is detected but the conditions are not favorable, cheetahs may choose to patiently sit and waiting for the prey to approach or for the situation to improve.

3. **Attacking**: This strategy involves two key steps. First, when the cheetah decides to attack, it rapidly rushes toward the prey at maximum speed. Second, employing its speed and agility, the cheetah captures the prey by closing in on it.

4. **Returning home**: This strategy is executed in two scenarios. Firstly, if the cheetah failed to hunt the prey, it may change its position or return to its territory. Secondly, if there have been no successful hunting actions within a certain time interval, the cheetah may move to the last known prey location and resume searching in that vicinity.

3.1.3 Search strategy

Cheetahs employ two distinct strategies to search for prey. The first involves scanning the environment while either sitting or standing, which is particularly useful when the prey is densely concentrated and grazing on open plains. The second approach involves actively patrolling the surrounding area. This method requires more energy expenditure compared to the searching mode but is more effective when the prey is scattered and actively moving. Consequently, during the hunting period, cheetahs may alternate between these two search modes based on factors such as prey availability, the size of the hunting area, and their own physical condition.

$$X^{t+1}_{i,j} = X^{t}_{i,j} + r^{\wedge -1}_{i,j} * \alpha^{t}_{i,j} \quad (1)$$

3.1.4 Sit-and-wait strategy

When in search mode, the prey can become visible within the cheetah's field of vision. However, any movement by the cheetah might alert the prey and cause them to flee. In this mode, the cheetah remains stationary and patiently waits for the prey to approach closer before making a move.

$$X^{t+1}_{i,j} = X^{t}_{i,j} \quad (2)$$

3.1.5 Attack Strategy

Cheetahs rely on two crucial elements, namely speed and agility, when launching an attack on their prey. Once a cheetah decides to initiate an attack, it swiftly accelerates towards the prey with tremendous speed. As the chase unfolds, the prey becomes aware of the cheetah's pursuit and attempts to escape. The cheetah, with its sharp eyesight, closely tracks the prey's movements and adjusts its own trajectory to intercept and cut off the prey's path. By closing in on the prey at maximum velocity, the cheetah forces the prey to make sudden evasive maneuvers to survive. The cheetah's next position is strategically positioned near the prey's previous location since it has reached a close distance during the initial burst of speed. This hunting strategy perfectly utilizes the cheetah's speed and agility to secure a successful capture.

$$X^{t+1}_{i,j} = X^{t}_{\beta,j} + \check{r}_{i,j} * \beta^{t}_{i,j} \quad (3)$$

3.1.6 Pseudo code Cheetah Optimization Algorithm(COA)

1. Initialize population of candidate solutions
2. Set maximum number of iterations
3. Set exploration rate
4. Set exploitation rate
5. While iterations < maximum number of iterations:
6. Evaluate fitness for each candidate solution
7. Sort candidate solutions based on fitness
8. Select top solutions for exploitation
9. For each candidate solution:
10. If random number < exploration rate:

        Perform exploration by randomly updating solution position

11. Else:

        Perform exploitation by updating solution position based on the best solutions

12. Apply boundary constraints to the updated position
13. Evaluate fitness for the updated solution
14. If updated solution has higher fitness:

        Replace current solution with the updated solution

15. Reduce exploration rate over time (optional)
16. Increment iterations
17. Return best solution found

3.1.7 Flowchart of Cheetah Optimization Algorithm(COA)

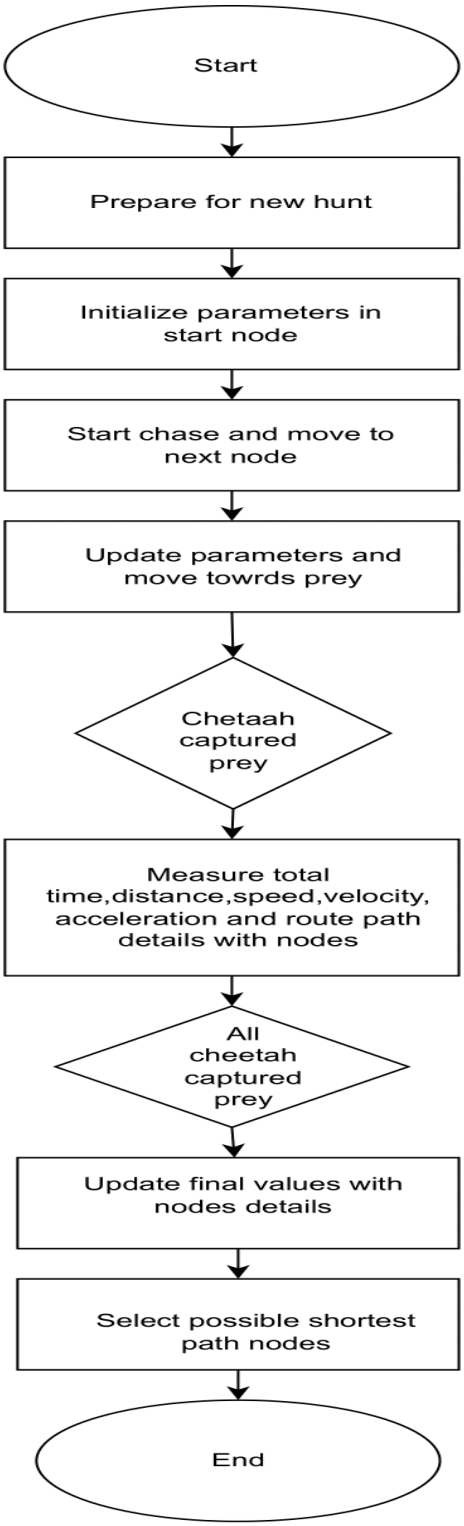
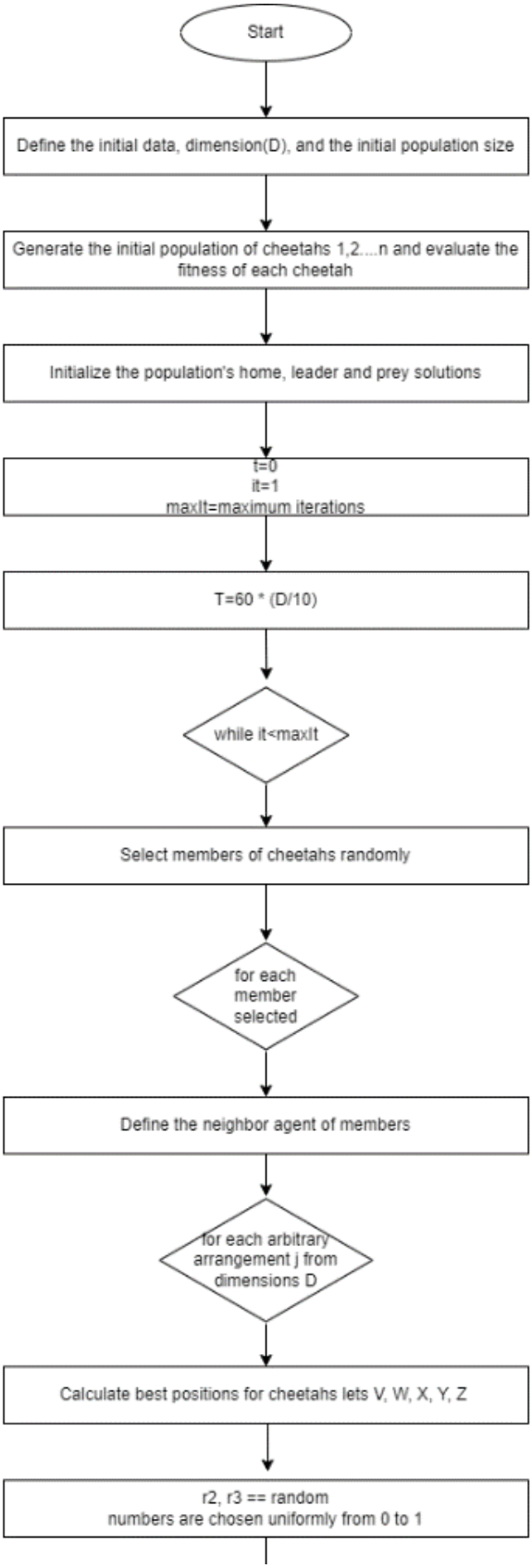


Figure 3.1: Flowchart of COA according to research paper


Start
Define the initial data, dimension(D), and the initial population size
Generate the initial population of cheetahs 1,2....n and evaluate the fitness of each cheetah
Initialize the population's home, leader and prey solutions
t=0
it=1
maxIt=maximum iterations
T=60 * (D/10)
while it<maxIt
Select members of cheetahs randomly
for each member selected
Define the neighbor agent of members
for each arbitrary arrangement j from dimensions D
Calculate best positions for cheetahs lets V, W, X, Y, Z
r2, r3 == random numbers are chosen uniformly from 0 to 1

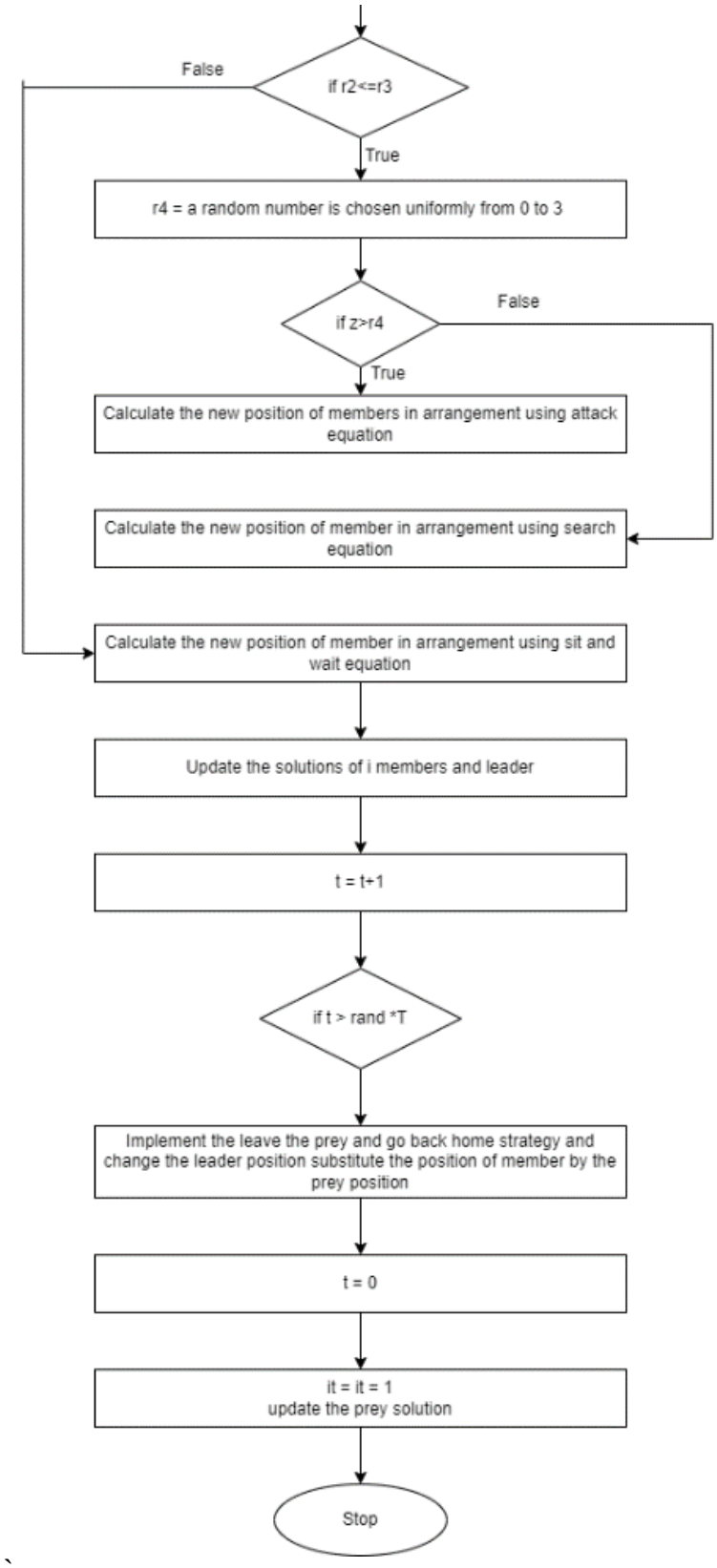


Figure 3.2: Flowchart according to Algorithm

3.2 Case Study

3.2.1 Case Study 1: Sphere Function

The sphere function is given by:

$$f(x) = \sum_{i=1}^{n} X_i^2 \quad (4)$$

The Sphere function can be formulated in an n-dimensional space, exhibiting characteristics of continuity, convexity, and unimodality. Although it can be defined over various input domains, it is commonly evaluated within the hypercube range of $x_i \in [-5.12, 5.12]$ for each dimension i=1...n. In this function, there exists n local minima, distinct from the global minimum, where the number of local minima corresponds to the number of dimensions.

3.2.2 Case Study 2: Economic Load Dispatch problem

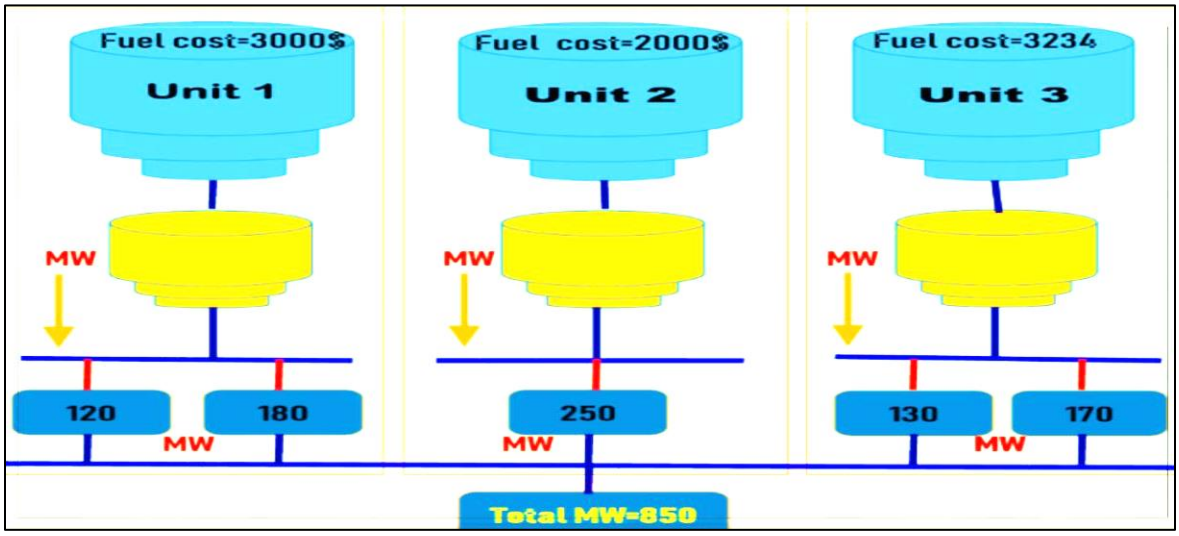


Figure 3.3:Economic load dispatch problem [23]

The ELD is a significant optimization challenge encountered in power systems engineering. It involves determining the most optimal allocation of power output among multiple generators in a power system, aiming to satisfy the load demand while minimizing the overall generation cost.

The primary objective of ELD is to distribute the load demand among available generators in a manner that minimizes the total operating cost, encompassing factors such as fuel costs and operational constraints. This optimization problem takes into consideration various aspects, including generator characteristics, transmission line losses, and system stability limitations.

When addressing the ELD, both the real power (active power) and reactive power of the generators are considered, while adhering to specified limits and operational constraints. The ultimate goal is to identify the optimal set points for the generators, ensuring the lowest possible operating cost while meeting the load demand.

To tackle the ELD, diverse approaches such as mathematical programming methods, evolutionary algorithms, and heuristics are employed. These techniques aim to explore the solution space, considering the various constraints, in order to identify the most cost-effective dispatch strategy. The ELD plays a vital role in power system operations as it directly impacts the cost associated with electricity generation and the stability of the system. Effective solutions to this problem contribute to efficient utilization of resources, reduced operational costs, and enhanced reliability within power systems.

3.3 Summary of the chapter

In this chapter a detailed introduction of COA along with the computational procedures and modes in COA is given.

Case study 1: Sphere function.

Case study 2: Economic Load Dispatch problem.

4. Results and Discussion

4.1 CASE STUDY 1: Sphere Function (With random value input)

X_1 and X_2 are coordinates of Cheetah, and F(X) is the fitness function i.e., Sphere Function, shows in Eq 4.1. Sphere function Optimization problem is solved using population size: 4, no. of catsin tracing mode: 1, position size of all 4 cheetahs and search space range [-5, 5] and no. of iterations: 3.

$$Min\ F(x) = \sum_{i=0}^{n} (X_i)^2 \qquad (4.1)$$

Steps to solve sphere function using cheetah optimizer algorithm.

Step 1: Initialize the population of cheetah, let's assume the population size 4 and initializethe cheetah randomly within search space, shows in the table 4.1:

Sr. No.	X_1	X_2
1	-3.50	1.20
2	2.30	-4.00
3	0.80	3.90
4	-2.60	-0.90

Table 4.1: Initial values of Variables for Sphere Function

Step 2: Evaluate the fitness for each cheetah in the population using sphere function.

Step 3: Determine the best position of cheetah in the population based on their fitness value.

Step 4: Update the position of each cheetah in the population using the formula.

$$X_{old} = X_{old} + r * (X_{best} - X_{old})$$

Where,

X_{old} → old position of cheetah

X_{best} → position of best cheetah

r → random number between 0 to1 (0.5)

4.1.1 Calculation for iteration 1:

1. Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c1} = -3.5$

$X_{1new} = -3.5 + (0.5) * (-2.6 - (- 3.5))$

$= -3.05$

For $X_{c2} = 1.2$,

$X_{2new} = 1.2 + (0.5) * (-0.9 – (1.2))$

$= 1.05$

Calculate the fitness value using below equation

$F(x) = -3.5^2 + 1.2^2 = 11.68$

2. Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c2} = 2.3$

$X_{1new} = 2.3 + (0.5) * (-2.6 - (2.3))$

$= -0.15$

For $X_{c2} = -4$,

$X_{2new} = -4 + (0.5) * (-0.9 – (4))$

$= -1.55$

Calculate the fitness value using below equation

$F(x) = 2.3^2 + 4^2 = 21.29$

3. Calculate the updated position of each cheetah using equation

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c3} = 0.8$

$X_{1new} = 0.8 (0.5) * (-2.6 - (0.8))$

$= -0.9$

For $X_{c3} = 3.9$,

$X_{2new} = 3.9 + (0.5) * (-0.9 – (3.9))$

$= 1.5$

Calculate the fitness value using below equation

$F(x) = 0.8^2 + 3.9^2 = 15.85$

4. Calculate the updated position each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c1} = -2.6$

$X_{1new} = -2.6 + (0.5) * (-2.6 - (- 2.6))$

$= -2.6$

For $X_{c2} = -0.9$,

$X_{2new} = -0.9 + (0.5) * (-0.9 – (-0.9))$

$= -0.9$

Calculate the fitness value using below equation

$F(x) = -2.6^2 + -0.9^2 = 7.57$

Finally in Table 4.2 shows iteration 1:

Sr. No.	X1	X2	F(x)
1	-3.50	1.20	11.68
2	2.30	-4.00	21.29
3	0.80	3.90	15.85
4	-2.60	-0.90	7.57

Table 4.2: Iteration 1 Values of Variables for Sphere Function

4.1.2 Calculations for iteration 2:

1. Calculate the updated position of cheetah using equation

Xold = Xold+ r * (Xbest - Xold)

For Xc1 = -3.05

X1new = -3.05 + (0.5) * (-0.9 - (- 3.05))

= -1.975

For Xc1 = 1.05,

X2new = 1.05 + (0.5) * (1.5 – (1.05))

= 1.275

Calculate the fitness value using below equation

$F(x) = -3.05^2 + 1.05^2 = 10.405$

2. Calculate the updated position each cheetah using equation

Xold = Xold+ r * (Xbest - Xold)

For Xc2 = -0.15

X1new = -0.15 + (0.5) * (-0.9 - (0.15))

= -0.525

For Xc2 = -1.55,

X2new = -1.55 + (0.5) * (1.5 – (1.55))

= -0.025

Calculate the fitness value using below equation

$F(x) = -0.15^2 + 1.55^2 = 2.38$

3 Calculate the updated position each cheetah using equation

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c3} = -0.90$

$X_{1new} = -0.90 + (0.5) * (-0.90 - (- 0.90))$

$= -0.90$

For $X_{c3} = -1.50$,

$X_{2new} = -1.50 + (0.5) * (1.5 – (-1.50))$

$= -1.50$

Calculate the fitness value using below equation

$F(x) = -0.90^2 + 1.50^2 = 1.44$

4. Calculate the updated position each cheetah using equation

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c4} = -2.60$

$X_{1new} = -2.60 + (0.5) * (-0.9 - (- 2.60))$

$= -1.75$

For $X_{c4} = -0.90$,

$X_{2new} = -0.90 + (0.5) * (1.5 – (0.90))$

$= 0.30$

Calculate the fitness value using below equation

$F(x) = -2.60^2 + 0.90^2 = 7.57$

Finally in Table 4.3 shows iteration 2:

Sr. No.	X_{1new}	X_{2new}	F(x)
1	-3.05	1.05	10.405
2	-0.15	-1.55	2.38
3	-0.90	1.50	1.44
4	-2.60	-0.90	7.57

Table 4.3: Iteration 2 Values of Variables for Sphere Function

Similarly, iteration 3 is done and final table 4.4 shows positive results:

Sr. No.	X_1	X_2	F(x)
1	-1.975	1.275	-2.275
2	-0.525	-0.025	-0.275
3	-0.90	1.50	1.44
4	-1.75	0.30	-2.975

Table 4.4: Final Values of Variables for Sphere Function

4.2 CASE STUDY 2: Sphere Function (input from TLBO)

Step 1: Initialize the population of cheetah, let's assume the population size 4 and standard initialized values from TLBO algorithm, the table 4.5 shows initial population:

Sr. No.	X1	X2
1	67.8907	25.6857
2	62.5171	-96.4914
3	-89.1315	71.0286
4	-29.2615	-89.49
5	22.0788	0.4924

Table 4.5: Initial values of Variables for Sphere Function

Step 2: Evaluate the fitness for each cheetah in the population using sphere function.

Step 3: Determine the best position of cheetah in the population based on their fitness value.

Step 4: Update the position of each cheetah in the population using the formula.

$$X_{old} = X_{old} + r * (X_{best} - X_{old})$$

Where,

X_{old} = Old position of cheetah

X_{best} = position of best cheetah

r = random number between 0 to 1 (0.5)

4.2.1 Calculation for iteration 1:

1. Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c1} = 67.8907$

$X_{1new} = -67.8907 + (0.5) * (22.0788 - (-67.8907))$

$= -22.90$

For $X_{c2} = 25.6857$,

$X_{2new} = 25.6857 + (0.5) * (-0.4925 – (25.6857))$

$= 12.59$

Calculate the fitness value using below equation

$$\mathbf{F(x) = 67.8907^2 + 25.6857^2 = 5268.90}$$

2 Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c2} = 62.5171$

$X_{1new} = 62.5171 + (0.5) * (22.0788 - (62.5171))$

$= 42.29$

For $X_{c2} = -96.4914$,

$X_{2new} = -96.4914 + (0.5) * (0.4925 – (-96.4914))$

$= -47.99$

Calculate the fitness value using sphere function

$F(x) = 62.5171^2 + -96.4914^2 = 13218.97$

3 Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c3} = -89.1315$

$X_{1new} = -89.1315 + (0.5) * (22.0788 - (-89.1315))$

$= -33.52$

For $X_{c3} = 71.0286$,

$X_{2new} = 71.0286 + (0.5) * (0.4925 – (71.0286))$

$= 35.76$

Calculate the fitness value using below equation

$F(x) = -89.1315^2 + 71.0286^2 = 12989.48$

4 Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c1} = -29.2615$

$X_{1new} = -29.2615 + (0.5) * (22.0788 - (- 29.2615))$

$= -3.59$

For $X_{c2} = -89.49$,

$X_{2new} = -89.49 + (0.5) * (0.4925 – (-89.49))$

$= -44.49$

Calculate the fitness value using below equation

$F(x) = -29.2615^2 + -89.49^2 = 8864.69$

5 Calculate the updated position of each cheetah using equation.

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c3} = 22.0788$

$X_{1new} = 22.0788 + (0.5) * (22.0788 - (22.0788))$

$= 22.0788$

For $X_{c3} = 0.4925$,

X_{2new} = 0.4925 + (0.5) * (0.4925 – (0.4925))
= 0.4925

Calculate the fitness value using below equation

$F(x) = 22.0788^2 + 0.4925^2 = 487.71$

Finally, the results of population after iteration 1 is:

Sr. No.	X1	X2	F(x)
1	67.8907	25.6857	5268.90
2	62.5171	-96.4914	13218.99
3	-89.1315	71.0286	12989.49
4	-29.2615	-89.49	8864.71
5	22.0788	0.4925	487.71
Mean	**6.8187**	**-17.7549**	

Table 4.6: Result of iteration 1

4.2.2 Calculations of iteration 2

1. Calculate the updated position of cheetah using equation

Xold = Xold+ r * (Xbest - Xold)
For Xc1 = -22.90
X1new = -22.90+ (0.5) * (22.07 - (- 22.90))
= -0.415
For Xc1 = 12.59,
X2new = 12.59 + (0.5) * (0.4925 – (12.59))
= 6.54

Calculate the fitness value using below equation

$F(x) = -22.90^2 + 12.59^2 = 682.91$

2. Calculate the updated position each cheetah using equation

X_{old} = X_{old}+ r * (X_{best} - X_{old})
For X_{c2} = 42.29
X_{1new} = 42.29 + (0.5) * (22.07 - (42.29))
= 32.18
For X_{c2} = -47.99,
X_{2new} = -47.99 + (0.5) * (0.4925 – (-47.99))
= -23.74

Calculate the fitness value using below equation

$F(x) = 42.29^2 + -47.99^2 = 4091.48$

3. Calculate the updated position each cheetah using equation

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c3} = -33.52$

$X_{1new} = -33.52 + (0.5) * (22.07 - (-33.52))$

$= -5.72$

For $X_{c3} = 35.76$,

$X_{2new} = 35.76 + (0.5) * (0.4925 – (35.76))$

$= 18.12$

Calculate the fitness value using below equation

$F(x) = -33.52^2 + 35.76^2 = 2402.36$

3. Calculate the updated position each cheetah using equation

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c4} = -3.59$

$X_{1new} = -3.59 + (0.5) * (22.07 - (- 3.59))$

$= 9.24$

For $X_{c4} = -44.49$,

$X_{2new} = -44.49 + (0.5) * (0.4925 – (-44.49))$

$= -66.48$

Calculate the fitness value using below equation

$F(x) = -3.59^2 + -44.49^2 = 1992.24$

4 Calculate the updated position each cheetah using equation

$X_{old} = X_{old} + r * (X_{best} - X_{old})$

For $X_{c4} = 22.07$

$X_{1new} = 22.07 + (0.5) * (22.07 - (22.07))$

$= 22.07$

For $X_{c4} = 0.4925$,

$X_{2new} = 0.4925 + (0.5) * (0.4925 – (0.4925))$

$= 0.4925$

Calculate the fitness value using below equation

$F(x) = 22.07^2 + 0.4925^2 = 487.32$

Table 4.7 shows result after iteration 2:

Sr. No.	X1new	X2new	F(x)
1	-22.90	12.59	682.91
2	42.29	-47.99	4091.48
3	-33.52	35.76	2402.36
4	-3.59	-44.49	1992.24
5	22.07	0.4925	487.32

Table 4.7: Iteration 2 Values of Variables for Sphere Function

Table 4.8 shows results after iteration 3 i.e., final results:

Sr. No.	X_1	X_2	F(x)
1	-0.415	6.54	42.94
2	32.18	-23.74	1599.14
3	-5.72	18.12	361.05
4	9.24	-66.48	4504.96
5	22.07	0.4925	487.32

Table 4.8: Final Values of Variables for Sphere Function

4.3 CASE STUDY 3: Economic Load Dispatch Problem

Table 4.9 shows given input parameter of ELD problem,

Unit	ai	bi	ci	P_{min}	P_{max}
1	0.007	7	240	100	500
2	0.0095	10	200	50	200
3	0.009	8.5	220	80	300
4	0.009	11	200	50	150
5	0.008	10.5	220	50	200
6	0.0075	12	120	50	120

Table 4.9:Input Parameters of Algorithm

Table 4.10 shows comparison of generators with power demand 500, 600 and 700 respectively.

POWER DEMAND (MW)	TECHNIQUE	POWER GENERATED BY EACH GENERATORS (MW)					
		G1	G2	G3	G4	G5	G6
500	BOA[21]	225.54	50	80	50	50	50
	GA[21]	213.62	55.66	80.35	50.232	55.036	50.743
	PSO[21]	221.19	50	84.39	50	50	50
	COA	161.46	55.36	91.12	56.65	69.31	63.1
600	BOA[21]	283.28	50.61	121.6	50.018	50.98	51.28
	GA[21]	263.4	62.15	106.7	50.475	60.18	65.18
	PSO[21]	280.59	50	127.3	50	50	50
	COA	131.16	68.16	90.98	59.78	73.31	59.55
700	BOA[21]	321.93	78.186	151	53.413	55.983	50.134
	GA[21]	301.07	56.44	159.8	51.996	67.44	74.49
	PSO[21]	323.51	76.823	158.34	50	52.045	50
	COA	101.72	87.72	94.45	67.68	50.24	55.99

Table 4.10: Comparison of generator Results with various Optimization Algorithms

Table 4.11 gives comparison of fuel cost with various algorithms:

(MW)	FUEL COST ($)			
	BOA[21]	GA[21]	PSO[21]	COA
500	6132.5	6147.6	6132.2	6078.8
600	7206.9	7250.5	7203.4	6056.3
700	8355.2	8402.3	8352.6	5846.9

Table 4.11:Comparison of fuel cost Results with various Optimization Algorithms

Readings	Solution	Total Cost
1	[161.46967206305035, 55.36791470665854, 91.12067262551818, 56.65764005365597, 69.31494345449939, 63.1005226973402]	6150.286877
2	[111.24291882565984, 79.65833481395013, 96.37969878595985, 66.51747079612555, 91.49562368318603, 51.8192958307231]	6266.179155
3	[100.90275767227452, 54.15151793598145, 80.07810972491593, 73.70436371363036, 90.11600278136794, 61.71877141239589]	5925.356678
4	[468.916905155013, 107.46067256413284, 86.56739412052539, 53.63995343466547, 65.05364223440867, 71.37201353161399]	4154.231553
5	[119.10751048394857, 82.07197811347544, 118.373921638093, 71.69044461284656, 88.94783924490416, 61.067571733599294]	6742.93539
6	[146.94981954647017, 85.98949367667403, 96.42738536400189, 67.23879761925669, 65.50315706847991, 75.74940591793879]	6667.717595
7	[102.28959854645798, 80.93474528112615, 84.68157271501542, 52.25155288207966, 63.260711394554534, 83.19251641378936]	5990.98755
8	[125.87334551799302, 94.68551430775068, 85.11731531225124, 63.66793135649225, 72.83835746057046, 52.96132971656343]	6213.398428
9	[106.52771088987936, 80.15411764764522, 103.76631242771447, 114.09293199313925, 56.618845713786044, 53.745535310929 25]	6525.558642
10	[151.59370329433543, 92.03648669530794, 85.74903657971603, 67.91919147454223, 62.84136693216716, 56.61220075651421]	6401.3384
11	[102.06137199219162, 139.48236526459476, 88.97362987415211, 83.09271076743262, 55.83961697605842, 57.966869143002796]	6702.740675
12	[116.30502196326154, 83.67110080811146, 89.75174690866993, 82.9597318542915, 58.37140629837348, 58.464090167632 46]	6189.290402
13	[103.4613064940436, 80.45323651628529, 91.81171278904849, 70.89370392782587, 72.17187149782049, 85.27701676429837]	6653.379786
14	[159.41402081046783, 72.07270752409396, 98.05610197504936, 69.27046799674669, 54.46009999878351, 54.75059465787484]	6264.082745
15	[100.68895131618065, 51.384371993842755, 135.11441789722136, 53.06285297040806, 60.782316040192875, 74.14854571111803]	6035.313517
16	[159.34664160356067, 59.29768113532421, 110.1814140538126, 77.05477590616842, 53.572629265911516, 54.114380451179045]	6323.195939
17	[115.38854464923709, 73.89017696611516, 133.72825225163723, 63.721085293116545, 65.89496301683306, 80.87606419117094]	6673.009665
18	[133.20444654333463, 54.31492442207186, 94.38914069790144, 84.65504706438239, 67.12993640684542, 60.15826715167221]	6195.963286
19	[103.08127496797694, 51.91815519781203, 115.31874333578935, 132.46377918282428, 56.773954200239444, 60.11464992207465]	6626.046257
20	[129.38955370318186, 77.27527506629168, 121.26133626503116, 68.78870255262729, 68.2488414589289, 50.181461091455915]	6389.662978
21	[104.88047387976142, 62.026880838296734, 96.80819756762371, 63.75097192964543, 72.11412026840202, 63.012603701623505]	5897.768177
22	[100.12887455847715, 86.20107774237178, 82.7373690647935, 52.80624815526978, 66.66309253462605, 56.188292160058445]	5707.978653
23	[112.12319432812663, 51.784932092847654, 101.25400287581279, 50.57716129400476, 64.15630445604835, 113.93101785030584]	6319.584247
24	[128.25274739115576, 69.28552095370375, 109.61852956945069, 76.77744948953239, 72.30985172500323, 56.55011140303095]	6392.547699
25	[138.2748624115547, 51.813672077125005, 93.75938204319996, 60.79500934022076, 111.83920054592932, 53.81627642069511]	6365.378517
26	[163.26586814540713, 53.22694061682443, 116.21851678802977, 54.60997464027161, 57.53038054465676, 50.19014603890822]	6077.324844
27	[121.78795817722236, 69.73734848855727, 108.31330898694846, 79.07731937813857, 58.0578083557907, 60.486747137177 77]	6242.148742
28	[124.07785010795544, 59.78844491636919, 125.0425057482224, 90.97813406095459, 50.3414278213669, 68.5090233851759]	6493.158953
29	[137.59644003345576, 58.838863579853154, 100.48940615919143, 63.071752202838816, 67.74054650354866, 71.39942071375287]	6234.630137
30	[102.2764188492161, 67.40215328291657, 94.18459272537372, 68.42238413106685, 71.77394655273514, 66.95441213071146]	6013.438288

Table 4.12: 30 readings for 500 MW demand

Readings	Solutions	Total cost
1	[109.94493340988937, 57.65453719125631, 87.49805024703755, 90.01999102600885, 55.21104569117159, 76.41646319152203]	6103.038
2	[109.94493340988937, 57.65453719125631, 87.49805024703755, 90.01999102600885, 55.21104569117159, 76.41646319152203]	6103.038
3	[171.4875127096782, 75.91722042508405, 127.7698847013391, 50.99913867583219, 68.20320001697087, 53.53078442045763]	6654.77
4	[127.42362750205581, 65.43676775424815, 100.72434188023351, 55.47611335680003, 90.12441892610079, 67.12269024925357]	6336.619
5	[139.9481719755978, 70.25036462918845, 88.45885809200416, 68.68325786691544, 71.7940418887863, 68.82589117011523]	6342.931
6	[136.02914104061253, 85.53763967396063, 107.81653433502956, 52.44618557256779, 87.69571101362439, 66.14519275034226]	6638.226
7	[155.49023893791804, 66.60189681296384, 83.99380370610874, 57.0574243687662, 92.45845203065277, 55.67624317730663]	6330.771
8	[150.65614075412216, 89.77808995189204, 128.34942387536154, 55.1078604394073, 51.35655039251502, 56.55694586976526]	6523.594
9	[100.03014316273205, 68.73032248869106, 81.6694954767161, 68.93223791865877, 73.0202694022214, 66.74735760501538]	5901.423
10	[126.18599138144532, 106.07533050843193, 95.4462389060049, 50.66953530948881, 84.55078582759599, 53.10804633515727]	6439.588
11	[100.01391002531004, 78.97654998971818, 153.26455888982397, 56.36605294357868, 53.71819568130802, 73.19304792986136]	6487.538
12	[105.58838258156248, 64.17297149763372, 85.8014467011518, 69.95573186911454, 56.10908483668538, 79.95016824263033]	5928.813
13	[179.09113288755964, 67.73655298748093, 81.15625448974286, 57.906293106586105, 95.54425398769355, 58.28116022807403]	6616.453
14	[179.09113288755964, 67.73655298748093, 81.15625448974286, 57.906293106586105, 95.54425398769355, 58.28116022807403]	6616.453
15	[114.48368839003007, 59.93702019946374, 126.80938907874884, 51.192233250650844, 95.07968031557965, 52.989292603978605]	6263.524
16	[154.71847261532986, 60.91198026657182, 106.55355044524147, 94.54094416781109, 52.980255553614874, 53.949791291139064]	6471.217
17	[141.58352225483327, 50.89246398792293, 101.41604944053874, 71.77468568723201, 52.962474721152326, 66.6937363104378]	6067.657
18	[141.57471421329996, 60.43458959280174, 88.3640013342683, 77.09142748024335, 71.62621513615856, 65.76760770121108]	6308.001
19	[188.85895651604613, 83.79151552982142, 105.59332929188976, 54.015523792703576, 62.549573522611595, 54.86923260455334]	6663.705
20	[133.35564478212225, 80.77542702046158, 88.14440441578128, 58.07424265513568, 71.18547286816488, 51.65597315039204]	6043.907
21	[119.25846646588866, 63.718756078387486, 96.27825889784792, 80.88491887659934, 77.70897621306418, 54.439425263074206]	6200.286
22	[125.97216261106566, 56.98938526983111, 169.7758054701601, 61.49487259042354, 63.67377408154671, 51.92395220433469]	6550.94
23	[109.55486942568649, 63.27780755917571, 94.46784006889916, 57.86341446871519, 50.33000107564019, 70.79951903407613]	5707.561
24	[103.34609574194485, 68.14364607527077, 86.19138895172212, 81.19666184534239, 51.42611413555841, 56.00949691906589]	5732.496
25	[127.87939780317545, 54.48954778482242, 89.6127145918196, 70.07415047105238, 68.10741860103084, 66.29893784146067]	6012.512
26	[203.38142313332446, 55.39300839299151, 109.50499723477172, 65.06569477271695, 51.85139372889525, 70.51628340303208]	6738.275
27	[107.61617741721552, 57.745526258163544, 168.5237952126057, 67.82778257911649, 58.768526374554796, 50.71419741817124]	6391.64
28	[106.85224446445206, 57.36568782761452, 89.45053832548345, 80.7358473122531, 84.12688304077199, 50.02088176431862]	5970.875
29	[153.05436911063617, 63.30481080463246, 86.3177785796011, 68.83127934751948, 79.99494124004147, 58.44496532215358]	6325.12
30	[121.64481816438322, 55.01715155370085, 107.32544595811532, 54.492641302483406, 62.18648970936384, 56.27963834963427]	5759.109

Table 4.13: 30 Readings for 600 MW demand

Readings	Solutions	Total cost
1	[101.52481528952886, 54.63107717376544, 138.9389884194103, 64.63904591858116, 65.13941939045382, 50.8890225762038]	6008.84
2	[157.78474569232955, 66.67899250235959, 80.7799133082958, 57.141501633843504, 104.30181802168647, 51.06301013536398]	6405.606
3	[128.57810144297468, 55.9165525918859, 124.8278150039967, 50.662353435749274, 85.65867882935854, 78.95902461067116]	6538.685
4	[112.82827194249268, 67.86673803125191, 83.45995054617285, 85.49105490074083, 59.01512350622376, 56.62478961886239]	5930.679
5	[109.91697667751498, 58.95525279823627, 97.08992638113503, 75.91237163250352, 96.19077155980044, 64.71371528321559]	6365.564
6	[144.94120716235028, 53.54605761367956, 111.99606589864266, 71.91692503407843, 50.32910326523009, 64.34124554492446]	6178.695
7	[109.91515215076797, 52.41109102326072, 103.31293697388655, 83.53494197933959, 100.84686753506102, 52.73799993264467]	6354.06
8	[123.21068360610482, 81.29236449280978, 109.12201726617806, 53.11554470456106, 66.11956783932308, 82.79332721489371]	6462.973
9	[114.79334360472811, 58.08597664560253, 93.78755337175872, 72.39623470045412, 71.68095303908665, 53.42042604551617]	5884.801
10	[171.21938564654812, 65.5891409820632, 117.16503166406684, 53.21108684448983, 50.99159895975879, 57.58519971142413]	6302.87
11	[109.73360909013734, 57.684367059758785, 83.79178718201521, 58.20555341396445, 85.88139368647776, 76.91764056480989]	6035.196

12	[164.89129685908156, 53.895335217238156, 98.22575767660909, 51.57362556073034, 65.67760090804666, 67.04134503036397]	6186.442
13	[124.87635169938937, 60.997886358139155, 83.6996378877133, 68.12532066804505, 77.75820679998955, 72.21209531767721]	6164.751
14	[119.63550558319, 53.073483916812506, 89.99040396759197, 82.5277533708639, 50.32851109378586, 59.41785568152493]	5790.243
15	[106.88749491335088, 55.795712938205185, 101.8207122556531, 64.68264108342835, 51.836712719430075, 107.85653860122454]	6270.974
16	[113.00456869851291, 75.51903001508236, 98.59256225471287, 62.39261015397997, 69.85655061526097, 58.659841794914364]	6038.926
17	[106.94964294505525, 55.41474722112665, 91.90785748851938, 83.38690103352035, 73.63251367166785, 65.94063978974089]	6089.526
18	[140.85473059042272, 57.546936270108134, 80.05424809545437, 62.684953328366845, 60.473301252903624, 100.26569452508741]	6337.645
19	[119.52442620760007, 64.41063971663115, 91.88420743083779, 59.74750584433443, 54.42401202214568, 111.16508377050864]	6388.355
20	[192.27396027687607, 64.69745652086954, 81.28045145747339, 57.123890347412775, 53.52092173823042, 85.98883513693066]	6571.723
21	[105.12459822796166, 57.57585811144815, 105.82771428000427, 66.50213301904992, 64.11175989529984, 60.47473880101869]	5851.321
22	[122.54896017410006, 95.95147977129362, 84.63069815415956, 57.84362772544202, 50.25977033496849, 66.7014230514369]	6041.885
23	[100.02079990033516, 75.29803004140257, 87.1282928169966, 58.6343123493334, 60.63089564687857, 78.11974804239732]	5911.09
24	[137.08322943654196, 90.93692756433413, 95.78326245057771, 55.94159991456833, 52.043258104401495, 54.004619042244 77]	6057.357
25	[156.98611199387364, 61.37175901322344, 85.2187080045984, 60.01632946353722, 58.5204218002405, 81.39572741139271]	6271.531
26	[110.27686951909973, 68.46502317821128, 83.70024254837791, 60.32149847851909, 124.61154661498644, 52.61868558634806]	6341.87
27	[105.8914163344407, 72.80428263073932, 111.27953551534355, 58.93216686398088, 51.10443567293455, 53.1216578662069]	5751.077
28	[105.02150961982264, 62.1863166957703, 134.66062803292664, 57.848144350028974, 62.165207062246324, 54.846199139308986]	6009.588
29	[107.27548091994512, 109.7060730405446, 81.74198845560669, 78.89090656990084, 64.50830054400001, 57.83263147489124]	6351.342
30	[120.32725896450947, 73.372810473278, 95.84082549439624, 84.00049237489901, 66.0227965597035, 52.12960689995929]	6187.388

Table 4.14: 30 Readings for 700 MW demand

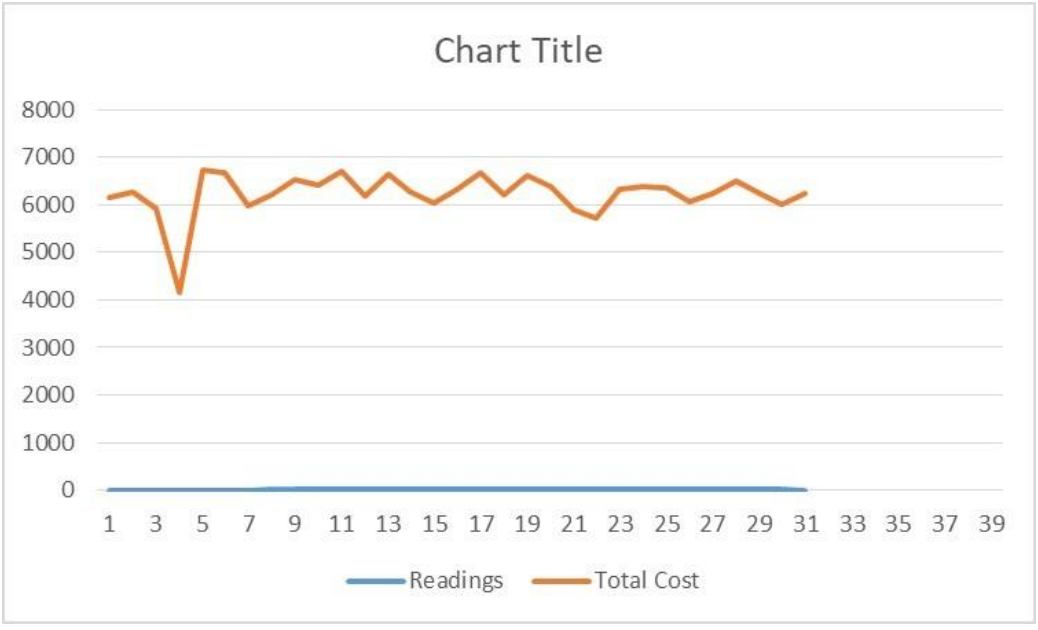


Figure 4.1: Graphical representation of 30 readings for 500MW

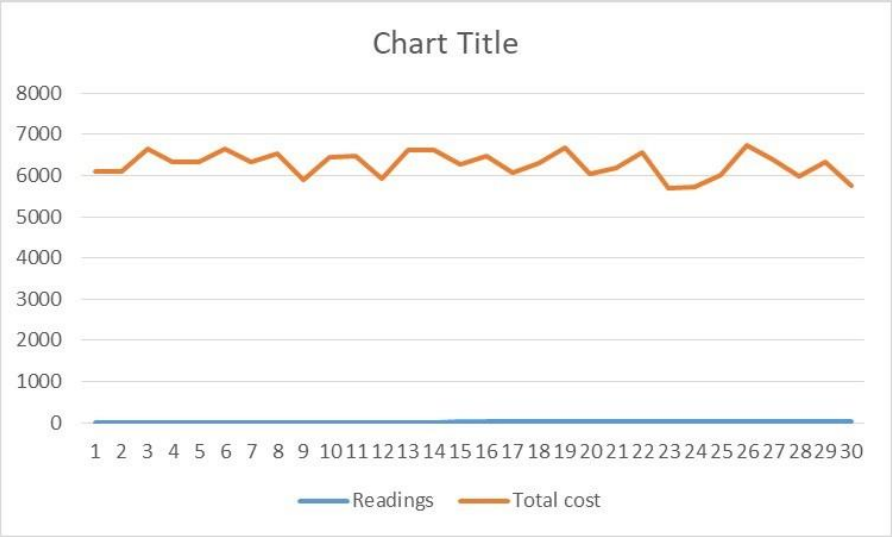


Figure 4.2: Graphical representation of 30 readings for 600 MW

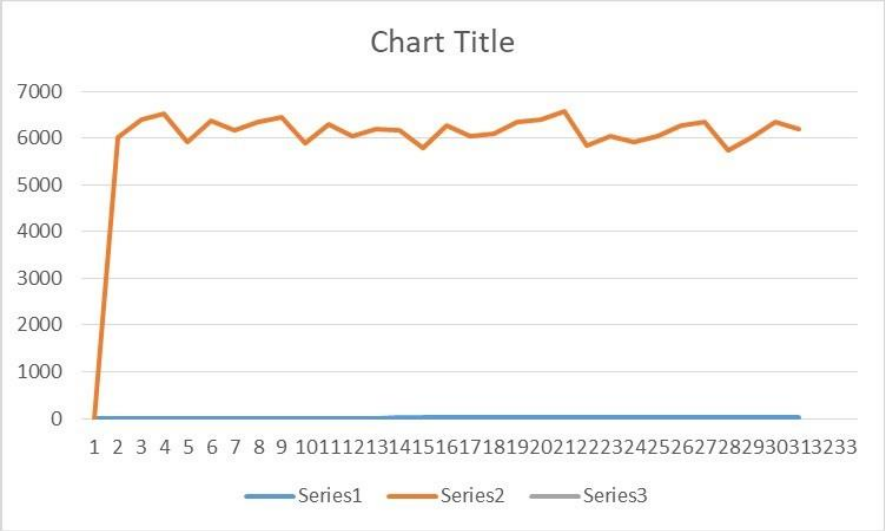


Figure 4.3: Graphical representation of 30 readings for 700 MW

4.4 Summary of chapter

In this chapter 3 iterations of Sphere function solved using COA. Table 4.10 gives details of the result of COA and compares with BOA, GA and PSO. Table 4.11 shows Comparison of various algorithm for Economic Load Dispatch problem.

5. Conclusion

COA is inspired from cheetah behavior based on strategies of search strategy, sit, and wait strategy, attack strategy. It is one of the competitive algorithms of type swarm based. COAis successfully applied to Sphere function and Economic Load Dispatch. In the proposed experiment elitism is applied in COA.

Results show that, COA is performing better than BOA, PSO, GWO, GA. Results are compared with some well-known metaheuristics algorithms. The case study of sphere function and Economic load dispatch is compared with other evolutionary algorithms and found out to be performing better.

6. References

[1] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Int. J. Adv. Sci. Eng. Inf. Technol. 16 (2002) 193–203.

[2] M. Mahdavi, M. Fesanghary, E. Damangir, an improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2)(2007) 1567–1579.

[3] Y. Wang, Z. Cai, Y. Zhou, accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization, Int. J. Numer. Methods Eng. 77 (11) (2009) 1501–1534.

[4] A. Sadollah, A. Bahreininejad, H. Eskandar, et al., Mine blast algorithm: a new population-based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (5) (2013) 2592–2612.

[5] G.A.O. Wei-Shang, S. Cheng, Iterative dynamic diversity evolutionary algorithm for constrained optimization, Acta Autom. Sin. 40 (11) (2014)2469–2479.

[6] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69(2014) 46–61.

[7] W. Yi, X. Li, L. Gao, et al., constrained differential evolution with pre-estimated comparison using gradient-based approximation for constrained optimization problems, Expert Syst. Appl. 44 (2016) 37–49.

[8] C. Zhang, Q. Lin, L. Gao, et al., Backtracking search algorithm with three constraint handling methods for constrained optimization problems, Expert Syst. Appl. 42 (21) (2015) 7831–7845.

[9] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, et al., Salp Swarm algorithm: abio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114(2017) 163–191.

[10] N.B. Guedria, Improved accelerated PSO algorithm for mechanical engineering optimization problems, Appl. Soft Comput. 40 (2016) 455–467.

[11] L. Wu, Q. Liu, X. Tian, et al., A new improved fruit fly optimization algorithm IAFOA and its application to solve engineering optimization problems, Knowl. Syst. 144 (2018) 153–173.

[12] W. Long, J. Jiao, X. Liang, et al., Inspired grey wolf optimizer for solving large-scale function optimization problems, Appl. Math. Modell. 60 (2018)112–126.

[13] L. Wang, H. Ni, R. Yang, et al., A simple human learning optimization algorithm, computational intelligence, in: Networked Systems and Their applications, Springer, Berlin Heidelberg, 2014, pp. 56–65.

[14] L. Wang, H. Ni, R. Yang, et al., An adaptive simplified human learningoptimization algorithm, Inf. Sci. (Ny) 320 (2015) 126–139.

[15] L. Wang, Ji Pei, Y. Wen, J. Pi , M. Fei , P. M. Pardalo et al, An improved adaptive human learning algorithm for engineering optimization, Applied Soft Computing 71 (2018) 894–904.

[16] J.S. Arora, Introduction to optimum Design, McGraw-Hill, New York, 1989.

[17] Y. Celik, H. Kutucu. Solving the tension/compression spring design problem by an improved firefly algorithm.

[18] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput., 2010.

[19] A. Bouzidi, M. Riffi, Cat Swarm Optimization to solve Flow Shop Scheduling Problem, Journal of Theoretical and Applied Information Technology, 20th Feb 2015.Vol.72.No.2

[20] N. Bacarisas, J. Paul the Effects of Varying the Fitness Function on the Efficiency of the CSO algorithm in solving the Graph Coloring Problem, Anale Seria Informatica, Vol.9 fasc. 2-2011

[21] Economic Load Dispatch Problem Using Butterfly Optimization Algorithm (European Journal of Molecular & Clinical Medicine ISSN 2515-8260 Volume 7, Issue 4, 2020)

[22] The cheetah optimizer: a nature-inspired metaheuristic algorithm for large-scale optimization problems (www.nature.com/scientificreports)

[23] https://link.springer.com/article/10.1007/s00521-019-04284-9

Appendix

A – Codes

A1. Python Code for Implementation of Cheetah Optimization Algorithm with Economic Load Dispatch

ELD.COA.py

#This code of CO with Economic Load Dispatch problem is developed by:

1.Rahul Kumar, Student, WCE

2.Pratik Umesh Pawar, Student, WCE

3.Arundhati Jagdish Wandhekar, Student, WCE

4.Pranjali Vasant Nanaware, Student, WCE

5.Sakshi Mohan Kamble, Student, WCE

6.Dr. A. J. Umbarkar, Faculty, WCE, in Academic Year 2022-23. Any user of this code should acknowledge the team.

```
import csv
import numpy as np
import random

# Cheetah Optimization Algorithm
def cheetah_optimization():
    # Initialize the population
    def
    initialize_population():
        population = [ ]
        for _ in range(population_size):
            solution = [ ]
            for i in range(num_generators):
                solution.append(random.uniform(p_min[i],
                p_max[i]))
            population.append(solution)
        return population

    # Evaluate the fitness of the population
    def evaluate_fitness(population):
        fitness_values = []
        for solution in population:
            cost =
            calculate_total_cost(solution)
            fitness_values.append(cost)
        return fitness_values
```

```
# Calculate the total cost of a solution
def calculate_total_cost(solution):
    total_cost = 0
    for i in range(num_generators):
        cost = a[i] * solution[i] ** 2 + b[i] * solution[i] + c[i]
        total_cost += cost
    return total_cost
# Select a parent based on fitness values using roulette wheel selection
def select_parent(population, fitness_values):
    total_fitness = sum(fitness_values)
    probabilities = [fitness / total_fitness for fitness in fitness_values]
    returnpopulation[np.random.choice(range(population_size), p=probabilities)]
# Perform crossover between two parents
def crossover(parent1, parent2):
    offspring = []
    for i in range(num_generators):
        if random.random() < 0.5:
            offspring.append(parent1[i])
        else:
    offspring.append(parent2[i])return offspring
# Perform mutation on an offspring
def mutate(offspring):
    for i in range(num_generators):
        if random.random() < mutation_rate:
    offspring[i] = random.uniform(p_min[i], p_max[i])return offspring
# Main program
num_generators = int(input("Enter the number of generators: "))
p_min = []
p_max    =
[]a = []
b = []
c = []
for i in range(num_generators):
p_min.append(float(input(f"Enter the minimum power output of generator {i+1}: ")))
p_max.append(float(input(f"Enter the maximum power output of generator {i+1}: ")))
a.append(float(input(f"Enter the coefficient 'a' for generator {i+1}: "))) b.append(float(input(f"Enter the
coefficient 'b' for generator {i+1}: "))) c.append(float(input(f"Enter the coefficient 'c' for generator {i+1}: ")))
demand = float(input("Enter the total power demand: "))
population_size = int(input("Enter the population size: "))
max_iterations = int(input("Enter the maximum number of iterations: "))
mutation_rate = float(input("Enter the mutation rate: "))
```

```
    population = initialize_population()

    best_solutions = [ ]
    total_costs = [ ]

    for iteration in range(max_iterations):
        fitness_values = evaluate_fitness(population)
        best_solution = population[np.argmin(fitness_values)]

        new_population = [ ]
        new_population.append(best_solution)

    while len(new_population) < population_size: parent1 = select_parent(population, fitness_values)parent2 =
    select_parent(population, fitness_values)

            offspring = crossover(parent1,
            parent2)offspring = mutate(offspring)
            new_population.append(offspring)

        population = new_population


        best_solution =
        population[np.argmin(evaluate_fitness(population))]total_cost =
        calculate_total_cost(best_solution)

        best_solutions.append(best_solution
        )total_costs.append(total_cost)

        # Export current iteration results to CSV file
        results = [iteration + 1, best_solution,
        total_cost]
        headers = ["Iteration", "Best Solution", "Total Cost"]

        with open("eld_results.csv", "a", newline="") as csvfile:

            writer = csv.writer(csvfile)
            if iteration == 0:
                writer.writerow(headers)
            writer.writerow(results)
    return best_solutions,
total_costs# Example usage
    best_solutions, total_costs = cheetah_optimization()

    print("Results exported to eld_results.csv")
```

A2. Python Code for Implementation of Cheetah Optimization Algorithm with Sphere function

sphere.py

```
import csv
import numpy as np

def sphere_function(x):
    # Calculate the sum of squared
    componentsreturn np.sum(np.power(x, 2))

def initialize_population(population_size, dimension,
    search_space):# Initialize the population within the search space
    population = np.random.uniform(search_space[0], search_space[1], (population_size,
    dimension))return population

def evaluate_population(population):
    # Evaluate the fitness of each solution in the population
    fitness_values = np.apply_along_axis(sphere_function, 1,
    population)return fitness_values

def cheetah_optimization_algorithm(population_size, dimension, search_space, max_iterations)
    population = initialize_population(population_size, dimension, search_space)
    best_solution = None
    best_fitness = np.inf

    # Create and open the CSV file
    filename = "coa_results.csv"
    csvfile = open(filename, "w", newline="")
    writer = csv.writer(csvfile)
    writer.writerow(["Iteration", "Best Solution", "Best Fitness"])

for iteration in range(max_iterations):
    fitness_values = evaluate_population(population)

    # Update the best solution min_fitness =
    np.min(fitness_values)
    min_index = np.argmin(fitness_values)if
    min_fitness < best_fitness:
        best_fitness = min_fitness best_solution =
        population[min_index]
```

```
        # Append current iteration results to the CSV file
        writer.writerow([iteration, *best_solution, best_fitness])
        # Generate new candidate solutions using search operators
        # You can use various operators like mutation, crossover, etc.#
        Update the population with the new solutions
    # Close the CSV file
    csvfile.close()

        return best_solution, best_fitness

    # User input for required parameters
    population_size = int(input("Enter the population size: "))
    dimension = int(input("Enter the dimension: "))
    search_space_low = float(input("Enter the lower bound of the search space: "))
    search_space_high = float(input("Enter the upper bound of the search space: "))
    max_iterations = int(input("Enter the maximum number of iterations: "))

    search_space = (search_space_low, search_space_high)

    # Run the Cheetah Optimization Algorithm
    best_solution, best_fitness = cheetah_optimization_algorithm(population_size, dimension, search_space,
    max_iterations)


print("Best solution:", best_solution)print("Best fitness:", best_fitness)
```

Plagiarism Report

turnitin

Similarity Report ID: oid:27829:36238082

PAPER NAME	AUTHOR
REPORT 26 May.docx	Sakshi Kamble

WORD COUNT	CHARACTER COUNT
7203 Words	42547 Characters

PAGE COUNT	FILE SIZE
48 Pages	1.2MB

SUBMISSION DATE	REPORT DATE
May 26, 2023 1:46 PM GMT+5:30	May 26, 2023 1:46 PM GMT+5:30

15% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 0% Publications database
- 12% Submitted Works database

Excluded from Similarity Report

- Manually excluded text blocks

Vitae

Name of student	Rahul Kumar
Native Place	Udhampur, Jammu and Kashmir
Date of Birth	29th March 2002
Address	Udhampur, Jammu and Kashmir
Email	krahul23779@gmail.com
Objective	I am determined to develop my skill set .
Areas of Interest	C++,Data structure, problem solving .
Mobile No	9682638947

Name of student	Pratik Umesh Pawar
Native Place	Karmala, Solapur, Maharashtra
Date of Birth	28th May 2001
Address	Karmala, Solapur, Maharashtra
Email	pawarpratik80009@gmail.com
Objective	Seeking a challenging career to put my skills to use
Areas of Interest	JAVA, Data Structure, Database Management
Mobile No	9022871939

Name of student	Arundhati Jagdish Wandhekar
Native Place	Shevgaon,Ahmednagar, Maharashtra
Date of Birth	25th July 2001
Address	Shevgaon,Ahmednagar, Maharashtra
Email	arundhatiwandhekar25@gmail.com
Objective	A motivated professional seeking challenges wherein my skills and knowledge can be utilized for self-improvement.
Areas of Interest	JAVA, Database Management
Mobile No	9307985740

Name of student	Pranjali Vasant Nanaware
Native Place	Karmala, Solapur, Maharashtra
Date of Birth	15th September 2001
Address	Karmala, Solapur, Maharashtra
Email	nanawarepranjali@gmail.com
Objective	To build career as a Software Developer using all my knowledge and skills to achieve organizational as well as personal goals and further enhance skills
Areas of Interest	Python, Networking, Database Management
Mobile No	9372549849

Name of student	Sakshi Mohan Kamble
Native Place	Sangola, Solapur, Maharashtra
Date of Birth	13th September 2001
Address	Sangola, Solapur, Maharashtra
Email	kamblesakshi1210@gmail.com
Objective	To associate with an organization which progress dynamically that gives me a chance to update my knowledge and enhance my skills and to be a part of team excels in work towards the growth of the organization.
Areas of Interest	Java,Python,Networking
Mobile No	8624925901