MODELING HYBRID METAHEURISTIC OPTIMIZATION ALGORITHM FOR CONVERGENCE PREDICTION

By

Noel Jose Thengappurackal Laiju

Bachelor of Technology in Electronics & Communication Engineering

Cochin University of Science and Technology, India, 2008

A project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering

In the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

AUTHOR'S DECLARATION

Project:	MODELING HYBRID METAHEURISTIC OPTIMIZATION ALGORITHM FOR CONVERGENCE PREDICTION.
Degree:	Master of Engineering
Year of convocation:	2019
Name:	Noel Jose Thengappurackal Laiju
Program:	Electrical and Computer Engineering
University:	Ryerson University

ABSTRACT

The project aims at the design and development of six hybrid nature inspired algorithms based on Grey Wolf Optimization algorithm with Artificial Bee Colony Optimization algorithm (GWOABC), Moth Flame Optimization Algorithm with Ant Lion Optimization algorithm (MFOALO), Cuckoo Search Optimization algorithm with Fire Fly Optimization Algorithm(CSFFA), Multi-Verse Optimization algorithm with Particle Swarm Optimization Algorithm (MVOPSO), Grey Wolf Optimization algorithm with Whale Optimization Algorithm (GWOWOA), and Binary Bat Optimization Algorithm with Particle Swarm Optimization Algorithm(BATPSO). Hybrid optimizations assume the implementation of two or more algorithms for the same optimization problem. "Hybrid algorithm" does not refer to simply combining multiple algorithms to solve a different problem but rather many algorithms can be considered as combinations of simpler pieces. The hybrid approach combines algorithms that solve the same problem but differs in other characteristics notably performance. A hybrid optimization uses a heuristic to choose the best of these algorithms to apply in a given situation. The proposed hybrid algorithms are benchmarked using a set of 23 classical benchmark functions employed to test different characteristics of hybrid optimization algorithms. The results of the fitness functions prove that the proposed hybrid algorithms are able to produce better or more competitive output with respect to improved exploration, local optima avoidance, exploitation, and convergence. All these hybrid algorithms find superior optimal designs for quintessential engineering problems engaged, showcasing that these algorithms are capable of solving constrained complex problems with diverse search spaces. Optimization results demonstrate that all hybrid algorithms are very competitive compared to the state-of-the-art optimization methods and validated by fitness function. The hybrid algorithms are applied for optimal efficiency determination in various design challenges based on cantilever beam problem.

ACKNOWLEDGEMENT

In my journey towards this degree, I have found my mentor and inspiration, Professor Dr. Reza Sedaghat for continuous support and encouragement throughout this project. I had an opportunity to learn under his mentorship the subject of optimization which spurred us to pursue further on the subject and experience practical implementation in the real-life project. He consistently guided me in the research and steered me in the right the direction whenever he thought I needed it.

This project would not have been materialized without the great help and guidance of Prathap (Patrick) Siddavaatam. The door to Patrick's office was always open whenever I ran into a trouble spot or had a question about my research. I am extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

I would like to dedicate this work to my mother Mrs. Mary Nelbi whose dreams for me have resulted in this achievement and without her loving upbringing and nurturing; I would not have been where I am today and what I am today. Had it not been for my mother's unflinching insistence and support, my dreams of excelling in education would have remained mere dreams. I am grateful to my sibling Naveen and father Laiju, who have provided me through moral and emotional support in my life.

I would like to thank my wife Aswathy, for all her love and support. She was always keen to know what I was doing and how I was proceeding, although it is likely that she has never grasped what it was all about! I will miss your screams of joy whenever a significant momentous was reached.

Finally, I would like to thank my friends for their support, patience and encouragement. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

GWO	Grey Wolf Optimization algorithm
ABC	Artificial Bee Colony Optimization
WOA	Whale Optimization Algorithm
MFO	Moth Flame Optimization Algorithm
ALO	Ant Lion Optimization algorithm
CS	Cuckoo Search Optimization algorithm
FFA	Fire Fly Optimization
MVO	Multi-Verse Optimization algorithm
BAT	Binary Bat Optimization Algorithm
PSO	Particle Swarm Optimization Algorithm
GWOABC	Hybrid Grey Wolf Optimization algorithm and Artificial Bee Colony Optimization algorithm
GWOWOA	Hybrid Grey Wolf Optimization algorithm and Whale Optimization Algorithm
MFOALO	Hybrid Moth Flame Optimization Algorithm and Ant Lion Optimization algorithm
CSFFA	Hybrid Cuckoo Search Optimization algorithm and Fire Fly Optimization Algorithm
MVOPSO	Hybrid Multi-Verse Optimization algorithm and Particle Swarm Optimization Algorithm
BATPSO	Hybrid Binary Bat Optimization Algorithm and Particle Swarm Optimization Algorithm
DA	dragonfly algorithm

1 INTRODUCTION

1.1 Overview

In the recent years, metaheuristic algorithms are employed as primary techniques for obtaining the optimal solutions of real-world engineering design optimization problems [1-3]. The optimization process is initialized by creating a set of random solutions. These initial solutions are then united, reallocated or derived over a pre-determined number of steps termed as iterations. An algorithm becomes unique in terms of its characteristics in mixing, allocating or evolving these initial solutions during the optimization process. Most of these algorithms take advantages of stochastic operators which makes them unique from deterministic approaches.

A deterministic algorithm arrives at similar results for a given problem with identical initial starting point as these algorithms often get entrapped in a local minimum and fails to arrive at global minimum. Since real world problems have innumerous local solutions, deterministic algorithms are futile and unreliable in find true global optimum. Randomness is the main trait of stochastic algorithms. They employ random operators in order to avoid local minimums and stochastic operators which enable algorithms to obtain different solutions for a given problem in each run.

Meta-heuristic algorithms search for the global optimum in a search space by creating one or more random solutions for a given problem [4]. Hence, these algorithms have following advantages: problem independency, evolution independency, local minimum evasion and natural optimization inspirations makes these algorithms makes it simple and follow a general and common framework, which imparts us scope to improve these algorithms with hybridization. Some of the most popular algorithms in this field used in this paper are: Grey Wolf Optimization algorithm(GWO)[5], Artificial Bee Colony Optimization (ABC)[6], Whale Optimization Algorithm (WOA)[7], Moth Flame Optimization Algorithm (MFO)[8], Ant Lion Optimization algorithm (ALO)[9], Cuckoo Search Optimization algorithm (CS)[10], Fire Fly Optimization (FFA)[11], Multi-Verse Optimization algorithm (MVO)[12], Binary Bat Optimization Algorithm (BAT)[13] and Particle Swarm Optimization Algorithm (PSO)[14].

1.2What is Optimization?

Mathematical optimization or programming is the study of planning and design problems using mathematical tools to find solutions to optimally use resources, time and money under various constraints. With the advent of computers, optimization has become a part of computer-aided design activities. An optimization algorithm is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found.

The generalization of optimization theory and techniques to other formulations constitutes a large area of applied mathematics. More generally, optimization includes finding "best available" values of some objective function given a defined domain (or input), including a variety of different types of objective functions and different types of domains.

1.3Optimization Problems

An optimization problem can be represented in the following generic form

Minimize $x \in R^n$ $f_i(x)$, $(i = 1,2,\dots, M)$ (1.2.1)

subject to $h_j(x) = 0$, $(j = 1,2,\dots, J)$ (1.2.2)

$$g_k(x) \leq 0, \qquad (k = 1,2,\ldots, K) \qquad (1.2.3)$$

Where, f_i(x), h_j(x) and g_k(x) are the functions of design vector

$$x = (x_1, x_2,\ldots, x_n)^T \qquad (1.2.4)$$

Here, the components of x_i of x are called design or decision variables, and they belongs to real continuous, discrete or mixed of these two types.

The functions f_i(x)where i = 1, 2, …, M are called the objective functions or cost functions, where when M = 1 , there is only one single objective. The space occupied by the decision variables are termed as design space or search space R^n, while the space spanned by the objective functions are named response or solution space. The equalities of h_j(x) and g_k(x) are termed as constraints.

In mathematics, conventional optimization problems are usually stated in terms of minimization.

a local minimum is:

$$f(a) \leq f(x) \text{ for all x in the interval} \qquad (1.2.5)$$

While a local minimum is at least as good as any nearby elements, a global minimum is at least as good as every feasible element. Generally, unless both the objective function and the feasible region are convex in a minimization problem, there may be several local minimum. In a convex problem, if there is a local minimum that is interior (not on the edge of the set of feasible elements), it is also the global minimum, but a nonconvex problem may have more than one local minimum not all of which need be global minimum.

A large number of algorithms proposed for solving nonconvex problems – including the majority of commercially available solvers – are not capable of making a distinction between locally optimal solutions and globally optimal solutions and will treat the former as actual solutions to the original problem. Global optimization is the branch of applied mathematics and numerical analysis that is concerned with the development of deterministic algorithms that are capable of guaranteeing convergence in finite time to the actual optimal solution of a nonconvex problem.

A naive optimal design is achieved by comparing a few(limited up to ten or so) alternative solutions created by using a priority problem knowledge. In this method feasibility of each design

solution is first investigated. Thereafter an estimate of underlying objective (cost, profit, etc.,) of each solution is compared and best solution is adopted. It is impossible to apply single formulation procedure for all engineering design problems, since the objective in a design problem and associated therefore, design parameters vary product to product different techniques are used in

different problems. Purpose of formulation is to create a mathematical model of the optimal design problem, which then can be solved using an optimization algorithm.

1.4 Types of Optimization Algorithms

Classification of optimization algorithms can be done in a number of ways. There are two types of distinct types of optimization algorithms used today on the basis of nature of algorithm.

1.4.1 Deterministic Algorithms

They use specific rules for moving one solution to other. These algorithms are in use to suite sometimes and have been successfully applied for many engineering design problems. These algorithms adopt a rigorous method and its path and values of both design variables and repeatable functions. Rigorous methods converge to the global optimum in finite time. Deterministic global optimization methods are typically used when locating the global solution is a necessity (i.e. when the only naturally occurring state described by a mathematical model is the global minimum of an optimization problem), when it is extremely difficult to find a feasible solution, or simply when the user desires to locate the best possible solution to a problem.

1.4.2 Stochastic Algorithms

The stochastic algorithms are in nature with probabilistic translation rules. These are gaining popularity due to certain properties which deterministic algorithms do not have. Randomness is the main trait of stochastic algorithms. They employ random operators in order to avoid local minimums and stochastic operators which enable algorithms to obtain different solutions for a given problem in each run. Stochastic Algorithms are classified into two:

1.4.2.1 Heuristic Algorithms

The word heuristic refers to ‘to find’ or to ‘discover by trial and error’. Quality solutions to a complex optimization problem can be achieved in a reasonable amount of time, but there is no guarantee that optimal solutions are obtained. These algorithms work most of the time, however not at every instance.

1.4.2.2 Meta-Heuristic Algorithms

These algorithms simply perform better than heuristics, hence named meta-heuristic algorithm. They produce ***acceptable solutions within a reasonable feasible time frame***. The complexity of the problem of interest makes it impossible to search every possible solution or combination and the goal is to find acceptable solution within specified time frame.

Nature-inspired meta-heuristic algorithms solve optimization problems by mimicking biological or physical phenomena. They can be grouped in three main categories.

1.4.2.2.1 Evolution based Meta-Heuristic Algorithms

Evolution-based methods are inspired by the laws of natural evolution. The search process starts with a randomly generated population which is evolved over subsequent generations. The strength point of these methods is that the best individuals are always combined together to form the next generation of individuals. This allows the population to be optimized over the course of generations. The most popular evolution-inspired technique is Genetic Algorithms (GA) [43] that simulates the Darwinian evolution. Other popular algorithms are Evolution Strategy (ES) [44] , Probability-Based Incremental Learning (PBIL) [45] , Genetic Programming (GP) [46] , and Biogeography-Based Optimization algorithm (BBO) [47] .

1.4.2.2.2 Physics based Meta-Heuristic Algorithms

Physics-based methods imitate the physical rules in the universe. The most popular algorithms are Simulated Annealing (SA) [48] , Gravitational Local Search (GLSA) [49] , Big-Bang Big-Crunch (BBBC) [50] , Gravitational Search Algorithm (GSA) [51] , Charged System Search (CSS) [52] , Central Force Optimization (CFO) [10] , Artificial Chemical Reaction Optimization Algorithm (ACROA) [53] , Black Hole (BH) [54] algorithm, Ray Optimization (RO) [55] algorithm, Small-World Optimization Algorithm (SWOA) [56] , Galaxy-based Search Algorithm (GbSA) [57] , and Curved Space Optimization (CSO) [58] .

1.4.2.2.3Nature inspired Meta-Heuristic Algorithms

The third group of nature-inspired methods includes swarm- based techniques that mimic the social behavior of groups of animals. Meta-heuristic algorithms search for the global optimum in a search space by creating one or more random solutions for a given problem [4]. Hence, these algorithms have following advantages: problem independency, evolution independency, local minimum evasion and natural optimization inspirations makes these algorithms makes it simple and follow a general and common framework, which imparts us scope to improve these algorithms with hybridization. Some of the most popular algorithms in this field used in this paper are: Grey Wolf Optimization algorithm(GWO)[5], Artificial Bee Colony Optimization (ABC)[6], Whale Optimization Algorithm (WOA)[7], Moth Flame Optimization Algorithm (MFO)[8], Ant Lion Optimization algorithm (ALO)[9], Cuckoo Search Optimization algorithm (CS)[10], Fire Fly Optimization (FFA)[11], Multi-Verse Optimization algorithm (MVO)[12], Binary Bat Optimization Algorithm (BAT)[13] and Particle Swarm Optimization Algorithm (PSO)[14].

This paper proposes following six hybrid algorithms: Hybrid Grey Wolf Optimizer and Artificial Bee Colony Optimizer (GWOABC), Hybrid Grey Wolf Optimizer and Whale Optimization Algorithm (GWOWOA), Hybrid Moth Flame Optimization Algorithm and Ant Lion Optimizer (MFOALO), Hybrid Cuckoo Search Optimizer and Fire Fly Optimization Algorithm(CSFFA), Hybrid Multi-Verse Optimizer and Particle Swarm Optimization Algorithm (MVOPSO), Hybrid Binary Bat Optimization Algorithm and Particle Swarm Optimization Algorithm(BATPSO). Section 2 of this paper state the objective, concept and motivation of project. Section 3 discusses literature review. Section 4, 5, 6, 7, 8, and 9 deals with the concept of hybridization and implementation of hybrid algorithms. Section 11 of this paper showcases benchmarking and results of 23 fitness functions used. Section 12, the hybrid algorithms are applied for optimal efficiency determination in various design challenges based on cantilever beam problem. Section 13 concludes the paper and discusses possible future scope and improvements.

2 HYBRIDIZATION

A hybrid algorithm is an algorithm that combines two or more other algorithms that solve the same problem, either choosing one (depending on the data), or switching between them over the course of the algorithm. This is generally done to combine desired features of each, so that the overall algorithm is better than the individual components.

"Hybrid algorithm" does not refer to simply combining multiple algorithms to solve a different problem – many algorithms can be considered as combinations of simpler pieces – but only to combining algorithms that solve the same problem, but differ in other characteristics, notably performance.

Section 2 of this paper discusses literature review. Section 3 deals with the concept of hybridization and implementation of hybrid algorithms. Section 4 of this paper showcases benchmarking and results of 23 fitness functions used. Section 5 concludes the paper and discusses possible future scope and improvements.

2.1 Motivation

Many researchers have attempted the use of hybridization in order to enhance the performance of these algorithms. This paper proposes following six hybrid algorithms: Hybrid Grey Wolf Optimization algorithm with Artificial Bee Colony Optimization algorithm (GWOABC), Hybrid Grey Wolf Optimization algorithm with Whale Optimization Algorithm (GWOWOA), Hybrid Moth Flame Optimization Algorithm with Ant Lion Optimization algorithm (MFOALO), Hybrid Cuckoo Search Optimization algorithm with Fire Fly Optimization Algorithm(CSFFA), Hybrid Multi-Verse Optimization algorithm with Particle Swarm Optimization Algorithm (MVOPSO), Hybrid Binary Bat Optimization Algorithm with Particle Swarm Optimization

Algorithm(BATPSO). We have selected the above meta-heuristics for hybridization due to the following reasons.

- **No free lunch theorem for optimization** - This theorem has logically proved a particular meta-heuristic may show very promising results on a set of problems, but the same algorithm may show poor performance on a different set of problems.
- **Combination of algorithmic ideas** - Since there is not a single strategy which can be used to solve all kinds of optimization problems.
- **Individual optimization algorithm performance** -Any superior performance in one class of problems generally results in inferior performance over another class.
- **Nature-inspired (simplicity)** - They all share a common idea of using social behavior presented by species for survival or a natural phenomena and are put in as mathematical code to solve engineering problems. Therefore, easy to hybrid both algorithms.
- **Main objective of Hybridization** - the percentage of successful convergence to global optimum should increase as opposed to those obtained by standalone algorithm.
- **Popular Hybrid Algorithms** – Most hybrid algorithms available now are based on Genetic Algorithm (GA) and Particle Swarm Optimization Algorithm (PSO) only as they are the simplest meta-heuristic algorithms. Very few researches have been performed on optimization algorithms we have taken in our project.

We have selected the most popular and most promising meta-heuristic algorithms for hybridization – GWO, ABC, WOA, MFO, ALO, CS, FFA, MVO and BAT algorithms. There are other meta-heuristic algorithms modelled such as Ray Optimization (RO) [55] algorithm, Small-World Optimization Algorithm (SWOA) [56] , Galaxy-based Search Algorithm (GbSA) [57] , Curved Space Optimization (CSO) [58] .etc, but they have a poor rate of convergence, results and also have less real world applications.

Moreover, algorithms we have hybrid together share a similar mathematical model or structure as meta-heuristic algorithms are nature inspired and share a common idea of using social behavior presented by species for survival or a natural-phenomena. This similarity makes meta-heuristic algorithms easy to hybrid, provided they share identical mathematical modelling structure.

2.2 Objective and method of hybridization.

All metaheuristic algorithm share a common search process divided into two phases

- **Exploration phase:** the process of investigating the promising area(s) of the search space as broadly as possible. Algorithm uses stochastic operators at this phase.
- **Exploitation phase:** refers to the local search capability around the promising regions obtained in the exploration phase.

The goal of hybridization is to find a proper balance between these two phases; as it is considered a challenging task due to the stochastic nature of meta-heuristics. The resultant hybrid algorithm combines two or more other algorithms that solve the same problem, either choosing one (depending on the data), or switching between them over the course of the algorithm. This is generally done to combine desired features of each, so that the overall algorithm is better than the individual algorithms used for hybridization.

We hybridize different meta-heuristics in order to improve the balance between exploitation and exploration phase. We propose six hybrid optimization algorithms in this paper and their methods of hybridization to improve exploration-exploitation balance are explained below.

2.2.1 GWOABC

In GWO, hunting (exploitation process are guided by alpha, beta and delta wolves who save first 3 best solutions and oblige the other search agents (including the omegas) to update their positions according to the average position of the alpha, beta and delta wolves give by the equation. $\vec{X}(t+1) = \frac{\vec{X}_1+\vec{X}_2+\vec{X}_3}{3}$. When the algorithm fails to find three best solutions namely alpha, beta and delta; the algorithm tends to fall in local minima and fails to update the position at lower fitness values. i.e. $\vec{X}(t+1) = \frac{\vec{X}_1+\vec{X}_2+\vec{X}_3}{3}$ remains same over n iteration. If the solution does not improve after a limited set of trials (t) or iterations, we incorporate scout bee mechanism in ABC algorithm with which ABC algorithm avoids local minima entrapment. During scout bee phase, scout bee sends alpha, beta and delta wolves repeatedly to generate a new solution if the current solution is not improved within a limited set of trials. (i.e. change of exploitation to exploration phase to find a better global minimum). The scout generated solution inherits some good structures from the discarded solutions and the new solution is better than a randomly generated fitness value.

We also use adaptive parameters of ABC in GWO than relying random numbers to smoothly transit between exploration and exploitation.

Prob = (random.random() * fitness / Alpha_score)+0.1

<u>Advantages of GWOABC hybridization</u>

- Avoid local minima entrapment tendency in GWO by using scout bee exploitation.
- GWO have a better rate of convergence than swarm based ABC optimizer where all fitness of employed bees and onlooker bees are taken into account than 3 best solutions in GWO.

2.2.2 GWOWOA

In WOA, unlike GWO, the position of a search agent in the exploration phase is done according to a randomly chosen search agent instead of the best search agent found so far. At exploitation phase , whales follow a Spiral updating position. This approach assumes that the prey is still moving and therefore we calculate the distance between the whale located at (X , Y) and prey located at (X ∗, Y ∗). A spiral equation is then created between the position of whale and prey to mimic the helix-shaped movement of humpback whales as follows : $\vec{X}(t+1) = \overrightarrow{D'}.\ e^{bl}.\cos(2\pi l) + \overrightarrow{X*}\,(t)$.

We integrate the shrinking encircling mechanism of bubble-net attacking method in WOA to GWO prey chasing, tracking and encircling phase in this hybrid optimization algorithm; as they both share identical mathematical modelling with different coefficient vectors. We also adapt the spiral updating position of whale and prey to mimic the helix shaped movement of humpback whales at higher values of co-efficient vector (A). $^{\rightarrow}$ This bubble-net foraging method is more sophisticated prey encircling method than just updating position vectors with respect to the three best fitness solutions, we were able to achieve faster convergence and better global optimum in most cases.

<u>Advantages of GWOWOA hybridization</u>

- GWO has faster rate of convergence at higher fitness values; WOA stagnates due to its poor randomization technique.

- Encircling mechanism of GWO and WOA has less capability of jumping out of local minima which was overcome by spiral updating position technique to cover a wider search-space to avoid local minima entrapment.

2.2.3 MFOALO

MFO is a population based algorithm inspired by the peculiar navigational habit of a moth called Transverse Orientation. ALO algorithm is inspired by the foraging behavior of antlion's larvae. At MFO exploration phase, in addition to transverse navigation method of moths, we also create the **random walk** of antlions and normalize it according to MFO parameters given by $X_i^t = \frac{(x_i^t - a_i)x(d_i - c_i^t)}{(d_i^t - a_i)} + c_i$

The best antlion obtained so far in each iteration is saved and considered as an elite. Elite antlion position is calculated and is compared with that of moth position as given by the logarithmic spiral equation**(exploitation)** : $\vec{X}(t+1) = \overrightarrow{D'}.\ e^{bl}.\cos(2\pi l) + \overrightarrow{X*}(t)$. If Elite antlion fitness is greater than that of moth-flame fitness; position vectors are updated using both moth flame position and elite ant-lion position.

<u>Advantages of MFOALO hybridization</u>

- High probability of resolving local optima stagnation due to the use of random walks.
- Both are population-based algorithms – easy to hybrid, better exploitation search characteristics.
- The MFO algorithm is a gradient-free algorithm and considers problem as a black box. (less parameters to adjust).
- Elitism maintains best solution attained so far and affects the movement of moths and antlions in subsequent iteration to find better global optimum.

2.2.4 CSFFA

CS algorithm is inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). FFA is a population algorithm inspired by flashing characteristics of fireflies.

In FFA we assume that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function. Instead of using a random probability $P_a \in (0,1)$ at exploitation phase at CS, we use light intensity **I(r)** and attractive coefficient $\boldsymbol{\beta}$ of FFA at exploitation to facilitate proper balance between exploitation and exploration**. I(r)** = $\boldsymbol{\frac{I_s}{r^2}}$ **,** $\boldsymbol{\beta = \beta_0\, e^{-\gamma r^m}}$ **where m ≥ 1**

In CS, movement of cuckoos to generate new solution is a random pattern. It adversely affects its local exploitation capability. Hence, instead of random movement we integrate **Cartesian or Euclidean distance** maintained by fireflies at higher brightness in order to improve local minima avoidance at exploitation phase - $r_{ij} = |x_i - x_j| = \sqrt{\sum_{i=1}^{d}(x_{ik} - x_{jk})2}$

<u>Advantages of CSFFA hybridization</u>

- Improving the balance of exploration and exploitation in CS.
- Elitism characteristics of CS retain best solution found so far.
- Adaptive parameters with respect to best solution in search space provide better exploitation capability.

2.2.5 MVOPSO

MVO is a population-based algorithm are based on three concepts in cosmology: white hole, black hole and wormhole. PSO is a population based meta-heuristics inspired by social behavior of bird flocking or fish schooling.

In PSO, velocity of each agent can be modified by the following equation in inertia weight approach. v[] = w* v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]), 'w' is called as the inertia factor which controls the influence of previous velocity on the new velocity : $w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}}$

Mathematical modeling of MVO consists of two main coefficients - wormhole existence probability (WEP) and travelling distance rate (TDR). $WEP = min + l * \left(\frac{max - min}{l}\right)$, TDR = 1 - $\frac{l^{1/p}}{L^{1/p}}$. We use this inertia factor w in MVO in order to substitute WEP at exploration phase to improve rate of convergence and modify the wormhole tunnel exchange mechanism with respect to TDR to explore search space faster.

Advantages of MVOPSO hybridization

- Computational procedure framework remains intact with that of MVO as PSO is a very simple algorithm based on just relative velocity of search agents.

- Very fast convergence rate inherited by PSO at exploration phase.

- PSO is the simplest optimization algorithm, easy to hybrid and have less computational complexity.

- Overcome premature local minima convergence of PSO.

2.2.6 BATPSO

BAT algorithm is Inspired by the echolocation behavior of bats. PSO is a population based meta-heuristics inspired by social behavior of bird flocking or fish schooling.

In BAT minimum fitness function is located by an array-based greedy algorithm (works best in faster exploitation in high dimensional fitness values and are more stochastic). PSO is based on element-wise pos[I,j] search and updating the velocity to converge to a global minimum. (works best in faster exploitation in lower fitness values and have a tendency to get trapped in local minimum). Both PSO and BAT are run in parallel and they perform a comparison between both minimum fitness function at each iteration. The lowest value is taken and both PSO and BAT is updated with the lowest value and respective positions are updated simultaneously.

<u>Advantages of BATPSO hybridization</u>

- Jumping out of local minima – disadvantage of PSO
- Faster rate of convergence – disadvantage of BAT.
- Accuracy of BAT may be limited if the number of function evaluations is not high which is dealt by PSO.
- BAT algorithm – still in primitive stage and requires more testing and tuning.

Inspiration, mathematical modelling and computational procedure of all hybrid algorithms are explained in detail in subsequent chapters.

3 LITERATURE REVIEW

Most of the hybrid optimization algorithms available at present are based on particle swarm optimization (PSO). These algorithms take advantage of fast convergence speed of PSO and minimize its defects such as premature convergence and local minimum stagnation. Mr. Jianwen Guo proposed a hybrid optimization algorithm based on PSO and Cuckoo Search (CS) algorithm for solving a preventive maintenance period optimization model problem [15].

A hybrid Firefly-Particle Swarm Optimization (FAPSO) is proposed in a paper in which the population of FAPSO is divided into two sub-populations selecting FA and PSO as their basic algorithm to carry out the optimization process, respectively. To exchange the information of the two sub-populations and then efficiently utilize the merits of PSO and FA, the sub-populations share their own optimal solutions while they have stagnated more than a predefined threshold. Secondly, each dimension of the search space is divided into many small-sized sub-regions, based on which much historical knowledge is recorded to help the current best solution to carry out a detecting operator [16].

A hybrid DAPSO algorithm combines the frameworks of the dragonfly algorithm (DA) and particle swarm optimization (PSO) to find the optimized solutions for the power system. The hybrid algorithm adopts the exploration and exploitation phases of the DA and PSO algorithms, respectively, and was implemented to solve the multi-objective optimal power flow problem [17]. A hybrid algorithm based on using moth-flame optimization (MFO) algorithm with simulated annealing (SA), namely (SA-MFO) is explained in one paper. The proposed SA-MFO algorithm takes the advantages of both algorithms. It takes the ability to escape from local optima mechanism of SA and fast searching and learning mechanism for guiding the generation of candidate solutions of MFO [18].

The hybrid algorithm has been constructed using Mean Grey Wolf Optimization algorithm (MGWO) and Whale Optimization algorithm (WOA) utilizing the spiral equation of Whale Optimization algorithm for two procedures in the Hybrid Approach GWO (HAGWO) algorithm:

(i) firstly, the spiral equation in Grey Wolf Optimization algorithm for balance between the exploitation and the exploration process in the new hybrid approach; and (ii) secondly, the equation in the whole population in order to refrain from the premature convergence and trapping in local minimum. [19]

A hybrid algorithm for feature subset selection in high-dimensional datasets using FICA and IWSSr algorithm was developed by Mostafa Moradhkhani. A hybrid method is proposed for efficient subset selection in high-dimensional datasets. The proposed algorithm runs filter-wrapper algorithms in two phases. The symmetrical uncertainty (SU) criterion is exploited to weight features in filter phase for discriminating the classes. In wrapper phase, both FICA (fuzzy imperialist competitive algorithm) and IWSSr (Incremental Wrapper Subset Selection with replacement) in weighted feature space are executed to find relevant attributes. The new scheme is successfully applied on 10 standard high-dimensional datasets, especially within the field of biosciences and medicine, where the number of features compared to the number of samples is large, inducing a severe curse of dimensionality problem. The comparison between the results of our method and other algorithms confirms that our method has the most accuracy rate and it is also able to achieve to the efficient compact subset [60].

The optimal coordination of Directional Overcurrent Relays (DOCRs) is a nonlinear and non-convex optimization problem integrating large number of constraints. In a paper, the hybrid Cuckoo Search Algorithm (CSA) - Firefly Algorithm (FFA) approach is implemented to solve coordination problem of DOCRs. The Artificial Intelligence (AI) based method such as FFA searches large solution space with large deviation in results with different number of simulations. Also it is not assured that result given by FFA is global best. To solve this problem, the preliminary optimal value of Time Multiplier Setting (TMS) and pickup current (Ip) are determined using CSA. The values of these variables are used in FFA as upper bounds which reduce the solution space and give a global optimal solution with very minimum deviation. The obtained results using proposed method are compared with hybrid Genetic Algorithm-Nonlinear Programming (GA-NLP) as well as with conventional CSA and FFA methods. The outcome demonstrates that the proposed method can obtain realizable and global best solution with minimum deviation in results and improved computational efficiency for this complex problem [61].

In a paper, hybrid clustering algorithm that integrates Fuzzy C-Means (FCM) and Whale Optimization Algorithm (WOA) using the Chebshev distance function is proposed. The FCM algorithm uses Euclidean distance to measure the similarity between the data. To avoid the existing disadvantages of the Euclidean distance, all distances in the FCM algorithm is calculated with the Chebsyhev distance function. The BOA algorithm is used to optimize the initial cluster centers. The proposed hybrid algorithm is tested with three different sets of data selected from UCI Machine Learning Repository database. As a result, it is seen that the clustering performance of the proposed algorithm is much better than the FCM algorithm [62].

Real Time Processor Scheduling with no preemption of tasks is a class NP-hard problem. We have attempted to get the best task allocation schedule for sporadic tasks, such that all the tasks are being scheduled without missing its deadline. Bat algorithm was proposed from the motivation behind the bat's echolocation behavior. Bat Searches for the prey in the given search location once found it tries to catch it by converging towards the direction of the prey. The existing system comprises of multiprocessor scheduling using bat algorithm which has been proved to be very efficient. Based on it, proposed modernistic algorithm used to solve the Multi-objective multiprocessor scheduling algorithm of a soft real time scheduling system for both periodic and sporadic tasks that is completely scalable in nature [63].

Gravitational search algorithm (GSA) is an optimization algorithm inspired from Newton's law of gravitation. Moth flame optimization (MFO) is another optimization algorithm, motivated by the locomotion of moths around a light source. Both of these algorithms have tried to model the search agents and altered properties like mass, gravitational constant, fitness, location, etc. in order to find the most optimal value. By hybridizing MFO and GSA, the performance is expected to improve across various measures. This paper presents a hybrid optimization algorithm by using concepts of moth flame optimization and gravitational search algorithm and applies this hybrid algorithm to image segmentation. An optimized K-means algorithm and an optimized thresholding algorithm have been proposed. The results of the segmentation are then used to classify apples into different classes [64]. The concept of hybridization, mathematical modeling and implementation of hybrid algorithms are discussed in next section 3 to 8.

4 HYBRID GREY WOLF OPTIMIZATION ALGORITHM AND ARTIFICIAL BEE COLONY OPTIMIZATION ALGORITHM ALGORITHM (GWOABC)

4.1Grey Wolf Optimization algorithm (GWO)

Grey wolf (Canis lupus) is from the Canidae family. Grey wolves are declared as apex predators, which means they are at the top of the food chain. Generally Grey wolves lives in pack, with average group size of 5- 12 in one group. Interesting thing is that they have a very strict social dominant hierarchy.

Both Female and Male Leaders are called alphas, on the top hierarchy, who generally take decisions about hunting, sleeping place, time to wake up etc. The Alphas generally command to the pack. Still there are some typo behavior where an Alpha Wolf follows other Wolves in the pack. The gatherings called by Alpha Wolf are acknowledged by other wolves in the pack holding their tail down. The Alpha Wolf are also known as Dominant Wolf as they command orders which are to be followed by other wolves in the pack. The Alpha Wolves can only mate within their pack. Alpha Wolves need not be the strongest in the group, it is just for maintaining the discipline and organization of the pack. The alpha wolves are only allowed to mate in the pack.

Beta, the next level of hierarchy, are subordinate wolves that help the alpha in decision-making or other pack activities. They can be either male or female wolf and is the next choice in case if the alpha wolves becomes old or dies. The beta wolf should control the lower level wolves and also obey the alpha wolves. In fact, they should play an advisory role to the alpha wolves and be commanding to the other wolves in the lower level. They have to actually convey the messages and concern between alpha wolves and the lower level wolves.

The lowest wolves in the hierarchy are the omega. The omega plays the role of scapegoat. Omega wolves have to take upon everything what the other wolves does. In fact they are always the last to eat. Even though this category omega, doesn't seems to be important in the pack, they are the cause for all fights in the pack as they vent out violence and frustration of all wolves by the omega. This Omega category assists satisfying the entire pack and maintaining the dominance structure. In some cases, the omega wolves babysit too.

There is one more category in this hierarchy which is neither Alpha, Beta nor Omega and is called subordinate (also known as delta). Delta wolves report to alphas and betas, but they dominate the omega. Scouts, sentinels, elders, hunters, and caretakers belong to this category. Scouts do the job of alerting the pack in case of any danger hence they protect the boundary of the territory. Sentinels protect and guarantee the safety of the pack. Elders are those experienced wolves who used to be alpha or beta. Hunters help the alphas and betas when hunting prey and providing food for the pack. Finally, the care takers are responsible for caring for the weak, ill, and wounded wolves in the pack.

Grey Wolf Optimization algorithm imitate the leadership hierarchy and hunting mechanism of gray wolves. In designing GWO, we consider the fittest solution as the alpha (α), second and third best solutions are named beta (β) and delta (δ) respectively. The rest of the candidate solutions are omega (ω). The GWO optimization hunting mechanism is ruled by α, β and δ. The ω wolves follow these three wolves.

The three main stages of grey wolves hunting mechanism are as follows:

- Chasing, Tracking and drawing near the prey.
- Pursuing, encircling and hassling the pray until it halts.
- Attacking towards the prey.

This hunting technique and the social hierarchy of grey wolves are mathematically modeled in order to design GWO and perform optimization.

4.2Artificial Bee Colony algorithm (ABC)

The ABC Algorithm was developed by Karaboga in 2005 and it is Swarm based meta- heuristic Algorithm. Foraging behavior of honey bees is how this algorithm got inspired by. This Algorithm consists of 3 categories of bees, they are – Employed Bees, Onlooker bees and the scout bees. Employed bees are those connected with specific food sources, onlooker bees monitor the dance of employed bees within the hive and scout bees are those of which who search for food sources in a different pattern. Every food source has only one employed bee. There is only one employed bee for every food source. Which finally means that the number of employed bees are equal to the number of food sources in the hive. The employed bee whose food source has been finished by the bees becomes a scout.

There are basically three steps combined in the cycle. Firstly, moving the onlooker and employed bee's food sources and watching their nectar amounts and counting taking count of the scout bees and directing them to the possible and available food sources. A food source arrangement reflects a possible solution to the problem to be solved. The quality of solution points to the amount of nectar of a food source. For placing Onlookers on the food source Probability- based selection process is used. The probability value with which the food source is preferred by onlookers increases as the nectar amount of food source rises. Every Bee colony has scouts that are the colony's explorers.

The explorers look for food in any pattern, not in any method. First, they are concerned mainly in finding the kind of food source. The scouts are characterized by low search costs and a low average in food source quality as a result of such behavior. Sometimes, the scouts can also discover rich, entirely unknown food sources. The artificial scouts could have the fast discovery of the group of feasible solutions as a task in the case of artificial bees. In ABC, the scout bee is classified by selecting one of the employed bee. The selection is done by a control parameter "limit". The food source is ignored by its employed bee if a solution representing a food source is not improved by a predetermined number of trials and the employed bee is converted to a scout.

An important control parameter of ABC is the number of trials for releasing a food source is equal to the value of "limit". Exploration and exploitation processes must be carried out together in a robust search process. In the ABC algorithm, the scouts control the exploration process while onlookers and employed bees carry out the exploitation process in the search space.

The recruitment rate represents a “measure” of how quickly the bee swarm locates and exploits the newly discovered food source in the case of real honey bees. Artificial recruiting process could similarly represent the “measurement” of the speed with which the feasible solutions or the optimal solutions of the difficult optimization issues can be identified. The survival and progress of the real bee swarm can be based upon the rapid discovery and efficient utilization of the best food resources. Also, the optimal solution of difficult engineering problems is connected to the relatively fast discovery of “good solutions” especially for the problems that need to be solved in real time.

4.3Mathematical Modeling of GWOABC

We take the advantage of scout bees in ABC, so that we can jump out from local minimum at lower fitness values in GWO.A scout is used to generate a new solution in the predefined search scope in the scout bee phase. For a new better solution, we take advantage of the discarded solution. The newly generated solution makes use of some good structures from the discarded solutions and the new solution is always better than a randomly generated fitness value.

In order to mathematically model the algorithm, following equations are used:

$$\vec{D} = |\vec{C}.\overrightarrow{Xp}(t) - \vec{X}(t)| \qquad (3.3.1)$$

$$\vec{X}(t+1) = \overrightarrow{Xp}(t) - \vec{A}\ .\vec{D} \qquad (3.3.2)$$

Where t denotes the current iteration, $\overrightarrow{A}$ and $\overrightarrow{C}$ are co-efficient vectors, $\overrightarrow{Xp}(t)$ is the prey position vector and $\vec{X}$ indicates position of grey wolf

Vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated using below equations:

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \qquad (3.3.3)$$

$$\vec{C} = 2.\vec{r}_2 \qquad (3.3.4)$$

Where $\vec{a}$ linearly decreases from to 2 to 0 and r1 and r2 are random vectors in [0,1]

The fittest solution is saved as alpha. Beta is between alpha and delta, which has lower value than alpha. Once we get the three best solutions after all this we then instruct the other agents in order to update their positions according to the best search agents. For this, the following equations are used.

$$\vec{D}_\alpha = |\overrightarrow{C_1}\vec{X}_\alpha - \vec{X}|,\ \vec{D}_\beta = |\overrightarrow{C_2}\vec{X}_\beta - \vec{X}|,\ \vec{D}_\delta = |\overrightarrow{C_3}\vec{X}_\delta - \vec{X}| \qquad (3.3.5)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.(\vec{D}_\alpha),\ \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.(\vec{D}_\beta),\ \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.(\vec{D}_\delta) \qquad (3.3.6)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \qquad (3.3.7)$$

The algorithm fails to update position and fall in local minimum when the algorithm fails to find three best solutions namely alpha, beta and delta. We incorporate scout bee mechanism in ABC algorithm with which ABC algorithm avoids local minimum entrapment if the solution does not improve after a limited set of trials or iterations.

Scout bee sends alpha, beta and delta wolves repeatedly to generate a new solution if the current solution is not improved within a limited set of trials during scout bee phase. The discarded solution provides some good structures to the scout generated solution the new solution is better than a randomly generated fitness value. The above process is repeated multiple times to generate several new solutions and finally choose the fittest solution among them and update alpha, beta and delta solution fitness accordingly.

Moreover, instead of using random numbers in determining the values of co-efficient vectors $\overrightarrow{A}$ and $\vec{C}$; we assign random numbers as a function of current fitness and best fitness based on probability equation in ABC.

prob=(random.random()*fitness/Alpha_score)+0.1 (3.8)

4.4Computational Procedure of GWOABC Algorithm

```
Initialize grey wolf population $X_i$ $(i = 1,2,3,..,n)$

Initialize $\alpha$, A and C

Calculate fitness for each search agent

$X_\alpha$ = best search agent, $X_\beta$ = second best search agent

$X_\delta$ = third best search agent

While(t < Max number of iterations)

   For each search agent

        Update the position of each search agent by eq 3.3.7
```

```
    End For
    Update $\alpha$, A and C
    Update/reset trial counter if better $\alpha$ not found/found
    Calculate the fitness of each search agents
    Update $X_{\alpha}$, $X_{\beta}$, $X_{\delta}$
    For trial exceed limit(n)
        If $\alpha$ not improved in 'n' continuous iterations then
        Scout $\alpha$ solution generated
        Update $\alpha$, A and C
    End For
    Reset trial
    $t = t+1$
End While
Return $X_{\alpha}$
```

5 HYBRID GREY WOLF OPTIMIZATION ALGORITHM AND WHALE OPTIMIZATION ALGORITHM (GWOWOA)

5.1Whale Optimization Algorithm (WOA)

Whales being the biggest mammals are considered fancy creatures. An adult whale can grow up to 180 t weight and 30 m long. This giant mammals consists 7 different main such as killer, Minke, Sei, humpback, right, finback, and blue. Whales are mostly considered as predators. They have to breathe from the surface of the oceans which is why they never sleep. Actually, half of the brain only sleeps. The interesting thing about the whales is that they are considered as highly intelligent animals with emotions.

According to Hof and Van Der Gucht, whales have spindle cells in certain areas of their brains similar to those of human. These cells are responsible for emotions, judgment and social behaviors in humans. In other words, the spindle cells make us different from other creatures. Whales have twice number of these cells than an adult human which is the main cause of their smartness. It has been proven that whale can think, learn, judge, communicate, and become even emotional as a human does, but obviously with a much lower level of smartness. It has been observed that whales are able to develop their own dialect as well.

The social behavior of whales is another interesting point. They either live alone or in groups. Even though, they are mostly observed living in groups. Some of their species, killer whales for instance, can live in a family over their entire life period. One of the biggest baleen whales is the humpback whales (Megaptera novaeangliae). An adult humpback whale is as big as a school bus. Their favorite preys are krill and small fish herds. The special hunting method is another interesting thing about the humpback whales. This foraging behavior is called bubble-net feeding method. Humpback whales generally prefer hunting school of krill or small fishes that are close to the surface. It has been analyzed that this foraging is done by creating distinctive bubbles along a circle or ‘9’-shaped path. This behavior was only determined based on the observation from surface before 2011. However, Goldbogen investigated this behavior utilizing tag sensors. They conquered 300 tag-derived bubble-net feeding events of 9 individual humpback whales. They named them ‘upward-spirals’ ‘double- loops’ after finding two maneuvers associated with bubble.

Humpback whales dive around12 m down in the former maneuver and then start to create bubble in a spiral shape around the prey and swim up towards the surface. The later maneuver combines three different stages: lobtail, coral loop and capture loop. It is worth mentioning here that Bubble-net feeding is a unique behavior that can only be observed in humpback whales. The spiral bubble-net feeding maneuver is mathematically modeled in order to perform WOA optimization.

The three main phases of Whale Optimization Algorithm are given below:

- Encircling prey
- Bubble-net attacking method (exploitation phase)
- Search for prey (exploration phase)

GWO optimization is already discussed in section 3.1.

5.2Mathematical Modeling of GWOWOA

Encircling prey method in WOA is almost as similar to tracking, chasing and drawing near the prey in GWO. Encircling prey in WOA is represented by following equations:

$$\vec{D} = |\vec{C}.\overrightarrow{X*}(t) - \vec{X}(t)| \qquad (4.2.1)$$

$$\vec{X}(t+1) = \overrightarrow{X*}(t) - \vec{A}\,.\vec{D} \qquad (4.2.2)$$

Where t denotes the current iteration, $\overrightarrow{A}$ and $\overrightarrow{C}$.are co-efficient vectors, $\overrightarrow{X*}$ is the position vector of best solution obtained so far and $\vec{X}$ indicates the position vector, | | is the absolute value and is an element -by- element multiplication.

Vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated using below equations:

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \tag{4.2.3}$$

$$\vec{C} = 2.\vec{r}_2 \tag{4.2.4}$$

We integrate the shrinking encircling mechanism of bubble-net attacking method in WOA to GWO prey chasing, tracking and encircling phase in this hybrid optimization algorithm. The shrinking encircling mechanism of bubble-net attacking method is achieved by decreasing the value of $\vec{a}$ in equation (4.2.3). Note that the fluctuation range of $\vec{A}$is also decreased by $\vec{a}$. Shrinking encircling mechanism is represented by the equation below:

$$\vec{X}(t+1) = \overrightarrow{X*}(t) - \overrightarrow{A}\,.\vec{D} \tag{4.2.5}$$

We also adapt the spiral updating position of whale and prey to mimic the helix shaped movement of humpback whales as given below:

$$\vec{X}(t+1) = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X*}(t) \tag{4.2.6}$$

In search for prey (exploration phase); we update the position of a search agent in the exploration phase by selecting a search agent in a random fashion instead of choosing the search agent with the best solution obtained so far. This technique and |A|>1 underline exploration and allow WOA to carry out a global search.

$$\vec{D} = |\vec{C}.\overrightarrow{X_{rand}} - \vec{X}| \tag{4.2.7}$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}\,.\vec{D} \tag{4.2.8}$$

We use equation (4.2.1), (4.2.5), (4.2.6), (4.2.7) and (4.2.8) to modify and adapt to equation (3.3.1), (3.3.2), (3.3.5) and (3.3.6) in GWO. Since bubble-net foraging method is more sophisticated prey encircling method than just updating position vectors with respect to the three best fitness solutions, we were able to achieve faster convergence and better global optimum in most cases.

5.3Computational Procedure of GWOWOA

```
Initialize grey wolf population $X_i$ (i = 1,2,3,..,n)
Initialize α, A and C
Calculate fitness for each search agent
$X_\alpha$ = best search agent, $X_\beta$ = second best search agent
$X_\delta$ = third best search agent
While(t < Max number of iterations)
    If p < 0.5
        If (|A|<1)
        Update the position of each search agent by eq (4.2.1)
        Else
        Select random search agent ($X_{rand}$)
        Update the position of each search agent by eq (4.2.5)
    If p > 0.5
```

```
            Select random search agent ($X_{rand}$)
            Update the position of search agent by eq (4.2.5) & (4.2.6)
            Re-initiate search iteration if no better solution found
        End If
        Check if any search agent goes beyond the search space and     amendit.
        Update $\alpha$, A and C
        Calculate the fitness of each search agents
        Update $X_{\alpha}$, $X_{\beta}$, $X_{\delta}$
        t = t+1
    End While
    Return $X_{\alpha}$
```

6 HYBRID MOTH FLAME OPTIMIZATION ALGORITHM AND ANT LION OPTIMIZATION ALGORITHM (MFOALO)

6.1Moth Flame Optimization Algorithm (MFO)

Moths are fancy insects, which are highly similar to the family of butterflies. Basically, there are over 160,000 various species of this insect in nature. They have two main milestones in their lifetime: adult and larvae. In Cocoons, larvae are converted to moth. Moths have this special navigation method in the night which is the most interesting fact about them. They have been formed to fly in night under the moon light. They follow a mechanism called transverse orientation for navigation in which a moth flies by maintaining a fixed angle with respect to the moon which is a very effective mechanism for travelling long distances in a straight path. As the moon is far away from the moth, this mechanism ensures flying in straight-line.

The same navigation method can be followed by humans as well. Suppose if the moon is in the south side of the sky and a human wants to go the east side, while walking if human keeps moon on his left side, he can move towards the east on a straight line. It is generally observed that moths fly spirally around the lights, despite the effectiveness of transverse orientation. In fact, moths are tricked by artificial lights and then they react with such behavior, this is due to the inefficiency of the transverse orientation. The Moths try to maintain a similar angle with the light to fly in straight line, when they see human-made artificial light.

Compared to the moon since such a light is extremely close, maintaining a similar angle to the light source causes deadly spiral fly path for moths. It may be observed in that the moth eventually intersects towards the light. An optimization algorithm called Moth-Flame Optimization (MFO) is modeled mathematically to capture this behavior.

The navigation method of moths in nature called transverse orientation is the inspiration for Moth Flame Optimization algorithm (MFO).In this method, a moth flies uses a very productive technique in travelling by maintaining a fixed angle with respect to the moon. This mechanism guarantees flying in a straight path as the moon lies at very large distance from the moth fly. But then, this straight path travelling is hugely affected by man-made artificial light. Since these artificial lights are so close to moths, they often end up in flying spirally around these lights as they may get tricked.

6.2Ant Lion Optimization Algorithm (ALO)

Antlions (doodlebugs) belong to the Myrmeleontidae family and Neuroptera order (net-winged insects). The lifecycle of antlions includes two main phases: larvae and adult. A natural total lifespan can be up to 3 years, which mostly covers in larvae. Antlions undergo metamorphosis in a cocoon to become adult (only3–5 weeks for adulthood).They mostly hunt in larvae and the adulthood period is only for reproduction.

Their names are extracted from their unique way of hunting behavior and based on their favorite prey. An antlion larvae digs a cone-shaped pit in sand with its massive jaw by moving along a circular path and throwing out sands. After digging the trap, the larvae hide underneath the bottom of the cone and waits for insects (preferably ant) to be trapped in the pit.

As the edge of the cone is sharp, insects fall to the bottom of the trap easily. Once the antlion realizes that a prey is in the trap, it tries to catch it. Generally insects usually are not caught immediately and try to escape from the trap. In this case, antlions intelligently throw sands towards the edge of the pit in order to slide the prey into the bottom of the pit. When the prey is caught into the jaw, it is pulled under the soil and eaten up. After eating the prey, antlions throw the leftovers outside the pit and prepare the pit for the next hunt.

Another interesting characteristic that has been observed in lifestyle of antlions is the relevancy of the size of the trap and another two things: level of hunger and shape of the moon. Antlions tend to dig out larger traps as they become hungrier and/or when the moon is full .They have been evolved and adapted this way to improve their chance of survival. It also has been discovered that an antlion does not directly observe the shape of the moon to decide about the size of the trap, but it has an internal lunar clock to make such decisions.

The main inspiration of the ALO algorithm comes from the foraging behavior of antlion's larvae. Ant Lion Optimization algorithm (ALO) is a meta-heuristic algorithm based on the interaction of ants and antlions in nature. Antlions belongs to Myrmeleontidae family and live in two phases of larvae and adult. During their larvae phase, antlions make a small cone shaped trap in order to trap ants. Antlions sit under the pit and wait for ants to be trapped. After feeding on trapped ants, antlions throw the leftovers outside the pit and prepare the pit for the next hunt. It has been observed that antlions dig a bigger bit when they are hungry, and this is the main concept of ALO optimization algorithm.

During optimization, the following conditions are applied:

1. Ants use different random walks to move around the search space.
2. Random walks are applied to all the dimension of every ants.
3. Random walks are affected by the traps of antlions.
4. Antlions can build pits proportional to their fitness (the higher fitness, the larger pit).
5. Antlions having larger pits have the higher probability to catch ants.
6. Each ant can be caught by an antlion in each iteration and the elite (fittest antlion).
7. The range of random walk is decreased adaptively to simulate sliding ants towards antlions.
8. If an ant becomes fitter than an antlion, this means that it is caught and pulled under the sand by the antlion.

9. An antlion repositions itself to the latest caught prey and builds a pit to improve its change of catching another prey after each hunt.

6.3Mathematical Modelling of MFOALO

In the proposed MFO algorithm, it is assumed that the candidate solutions are moths and the problem's variables are the position of moths in the space. Therefore, the moths can fly in 1-D, 2-D, 3-D, or hyper dimensional space with changing their position vectors. Since the MFO algorithm is a population-based algorithm, the set of moths is represented in a matrix as follows:

$$\mathbf{M} = \begin{vmatrix} M_{1,1} & M_{1,2} \ldots\ldots & M_{1,d} \\ M_{2,1} & M_{2,1} \ldots\ldots & M_{2,d} \\ \vdots & \vdots & \vdots \\ M_{n,1} & M_{n,2} \ldots. & M_{n,d} \end{vmatrix} \qquad (5.3.1)$$

where n is the number of moths and d is the number of variables (dimension).

For all the moths, we also assume that an array exists which holds the related fitness solutions of the moths and it is given by the following equation:

$$\mathbf{OM} = \begin{vmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{vmatrix} \qquad (5.3.2)$$

Where n denotes the number of moths.

Another important characteristics of the algorithm are the moth-flames and is given by the below equation:

$$\mathbf{F} = \begin{vmatrix} F_{1,1} & F_{1,2} \ldots\ldots & F_{1,d} \\ F_{2,1} & F_{2,1} \ldots\ldots & F_{2,d} \\ \vdots & \vdots & \vdots \\ F_{n,1} & F_{n,2} \ldots. & F_{n,d} \end{vmatrix} \qquad (5.3.3)$$

For the flames , the corresponding fitness values are stored in the below matrix:

$$\mathbf{OF} = \begin{vmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{vmatrix} \qquad (5.3.4)$$

Similarly for ALO, the algorithm mimics the interaction between the antlions and ants in the trap. In its mathematical model, ants are required to move over the search space, and antlions are allowed to hunt them and become fitter using traps. Since ants move stochastically in nature when searching for food, a random walk is chosen for model lignans movement. The position of ants are stored and used during optimization in the below matrix equation:

$$\mathbf{M_{ant}} = \begin{vmatrix} A_{1,1} & A_{1,2} \ldots\ldots & A_{1,d} \\ A_{2,1} & A_{2,1} \ldots\ldots & A_{2,d} \\ \vdots & \vdots & \vdots \\ A_{n,1} & A_{n,2} \ldots. & A_{n,d} \end{vmatrix} \qquad (5.3.5)$$

The corresponding fitness function are stored in following equation in order to evaluate the fitness of each ant

$$\mathbf{M_{OA}} = \begin{vmatrix} f([A_{1,1}, & A_{1,2}, \ldots\ldots & A_{1,d}]) \\ f([A_{2,1}, & A_{2,1}, \ldots\ldots & A_{2,d}]) \\ \vdots & \vdots & \vdots \\ f([A_{n,1} & A_{n,2} \ldots. & A_{n,d}]) \end{vmatrix} \qquad (5.3.6)$$

We assume that antlions are hiding at some places in the search space and their positions and fitness solutions are saved in below matrix:

$$\mathbf{M_{antlion}} = \begin{vmatrix} Al_{1,1} & Al_{1,2} \ldots\ldots & Al_{1,d} \\ Al_{2,1} & Al_{2,1} \ldots\ldots & Al_{2,d} \\ \vdots & \vdots & \vdots \\ Al_{n,1} & Al_{n,2} \ldots . & Al_{n,d} \end{vmatrix} \tag{5.3.7}$$

where $\mathrm{M_{antlion}}$ is the matrix for saving the position of each antlion,$Al_{i,j}$ shows the j-th dimension's value of i-th antlion, n is the numberof antlions, and d is the number of variables (dimension).

$\mathrm{M_{OAL}}$ is the matrix for saving the fitness of each antlion,$Al_{i,j}$ shows the j-th dimension's value of i-th antlion, n is the numberof antlions, and f is the fitness function.

$$\mathbf{M_{OAL}} = \begin{vmatrix} f([Al_{1,1}, & Al_{1,2}, \ldots\ldots & Al_{1,d}]) \\ f([Al_{2,1}, & Al_{2,1}, \ldots\ldots & Al_{2,d}]) \\ \vdots & \vdots & \vdots \\ f([Al_{n,1} & Al_{n,2} \ldots . & Al_{n,d}]) \end{vmatrix} \tag{5.3.8}$$

In MFOALO; we extract the elitism characteristics of ALO algorithm and harmonize it into MFO algorithm. Elitism is an important characteristic of evolutionary algorithms that allows them to maintain the best fitness(s) obtained at any stage of optimization process. In ALO, the best antlion obtained so far in each iteration is saved and considered as an elite. Since elite is the fittest antlion, it can influence the movements of all ants during iteration. Therefore, we assume that every ants random walks around a selected antlion by the roulette wheel and the elite simultaneously as given by the equation:

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{5.3.9}$$

Where, where R_A^t is the random walk around the antlion selected by the roulette wheel at t-th iteration, R_E^t is the random walk around the elite at t-th iteration, and Ant_i^t indicates the position of i-th ant at t-th iteration [28].

The ALO algorithm can be deduced to a three-tuple function that search for global minimum for optimization as follows:

ALO(A,B,C)

where A is a function that generates the random initial solutions, B manipulates the initial population provided by the function A, and C returns true when the end criterion is satisfied. The functions A, B, and C are defined as follows:

$$\Theta \xrightarrow{A} \{M_{Ant}, M_{OA}, M_{Antlion}, M_{OAL}\} \quad (5.3.10)$$

$$\{M_{Ant}, M_{Antlion}\} \xrightarrow{B} \{M_{Ant}, M_{Antlion}\} \quad (5.3.11)$$

$$\{M_{Ant}, M_{Antlion}\} \xrightarrow{C} \{true, false\} \quad (5.3.12)$$

where M_{Ant} t is the matrix of the position of ants, $M_{Antlion}$ includes the position of antlions, M_{OA} contains the corresponding fitness of ants, and M_{OAL} has the fitness of antlions.

MFO is also three-tuple function that approximates the global optimal of optimization problems. This uncanny similarity between ALO and MFO serves as the basis of our hybridization and is defined as follows:

Where I is a function that generates a random population of moths and corresponding fitness values. P function, which is the main function, moves the moths around the search space. This function received the matrix of M and returns its updated one eventually. The T function returns true if the termination criterion is satisfied and false if the termination criterion is not satisfied. The position of each moth is updated with respect to a flame using the following equation:

$$M_i = S(M_i, F_j) \tag{5.3.13}$$

Where M_i indicate the i-th moth, F_j indicates the j-th flame, and S is the spiral function.

In MFOALO in addition to navigation method of moths, we also create the random walk of antlions and normalize it using following equations:

$$c^t = \frac{c^t}{\mathrm{I}} \tag{5.3.14}$$

$$d^t = \frac{d^t}{\mathrm{I}} \tag{5.3.15}$$

where I is a ratio, c^t is the minimum of all variables at t-th iteration, and d^t indicates the vector including the maximum of all variables at t-th iteration [29].

Elite antlion position is calculated and is compared with that of moth position as given by the logarithmic spiral equation below:

$$\vec{X}(t+1) = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X*}(t) \tag{5.3.16}$$

where D_i. indicates the distance of the i-th moth for the j-th flame, b is a constant for defining the shape of the logarithmic spiral, and t is a random number in [-1, 1].

D is calculated as follows:

$$D_i = |F_j - M_i| \tag{5.3.17}$$

where M_i indicate the i-th moth, F_j indicates the j-th flame, and D_i indicates the distance of the i-th moth for the j-th flame.

If Elite antlion fitness is greater than that of moth-flame fitness; position vectors are updated using equation (5.3.9) using both moth flame position and elite ant-lion position.

6.4Computational Procedure of MFOALO

```
Initialize the number of flames(flame number)
Initialize moth population.
Calculate fitness values.
For all moths
   For all parameters
       Update r and t
       Calculate D using Eq. (5.3.17) with respect to the  corresponding moth.
       Update the matrix M with respect to the corresponding moth using Eq(5.3.13)& Eq (5.3.16)
       Select an antlion using roulette wheel
       Update c and d
       Create random walk and normalize it.
```

```
        Update elite antlion and compare with moth fitness   with Eq(5.3.9)
        Update position
     End For
    Calculate all fitness values
    Update flames
End
```

7 HYBRID CUCKOO SEARCH OPTIMIZATION ALGORITHM AND FIRE FLY OPTIMIZATION ALGORITHM (CSFFA)

7.1 Cuckoo Search Algorithm (CS)

Cuckoo Search Algorithm is developed by Xin-she Yang and Suash Deb in 2009. The algorithm is inspired by the obligate brood parasitism of certain cuckoo species by laying their eggs in the nests of other host birds belonging to different species. Cuckoo search mimics the breeding behavior of some female parasitic cuckoo species which mimics the colors and pattern of eggs of hosts birds, thereby avoiding host birds from identifying their eggs which results in tossing alien eggs or abandoning nests. On top of it, this algorithm is enhanced by incorporating Levy flights rather than just using isotropic random walks.

Cuckoo are fascinating birds, not solely as a result of the attractive sounds they'll build, also additionally as a result of their aggressive replica strategy. Some species like the cuckoo and Guira cuckoos lay their eggs in communal nests, though they may remove other’s eggs to increase the hatching probability of their own eggs. Quite a variety of species interact the obligate brood interdependence by birth their eggs within the nests of different host birds (often different species). There are 3 basic styles of brood interdependence: intra specific brood parasitism, cooperative breeding, and nest takeover. Some host birds will interact direct conflict with the intrusive cuckoos. If a bunch bird discovers the eggs don't seem to be their owns, they'll either throw these alien eggs away or just abandon its nest and build a replacement nest elsewhere. Some cuckoo species just like the New World brood-parasitic Tapera have evolved in such the manner that female parasitic cuckoos are generally very specialized among the mimicry in color and pattern of the eggs of a number of chosen host species . This reduces the likelihood of their eggs being abandoned and therefore will increase their reproductivity. In addition, the temporal arrangement of egg-laying of some species is additionally superb. Parasitic cuckoos usually opt for a nest wherever the host bird simply set its own eggs. In general, the cuckoo eggs hatches lightly earlier than their host eggs. Once the primary cuckoo chick is hatched, the first instinct action it will take is to evict the host eggs by blindly propelling the eggs out of the nest, which will increase the cuckoo chick's share of food provided by its host bird. Studies also show that a cuckoo chick can also mimic the call of host chicks to gain access to more feeding opportunity's algorithm is based on three rules

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest.
- The best nests with high quality of eggs will carry over to the next generation.
- The number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability pa Є (0,1). Discovering operate on some set of worst nests, and discovered solutions dumped from farther calculations [30].

6.1.1Lévy Flights

On the other hand, various studies have shown that flight behavior of many animals and insects has demonstrated the typical characteristics of Lévy flights. A recent study by Reynolds and Frye shows that fruit flies or Drosophila melanogaster, explore their landscape using aseries of straight flight paths punctuated by a sudden 90^{o} turn, leading to a Lévy -flight-style intermittent scale free search pattern. Studies on human behavior such as the Ju/'hoansi hunter-gatherer foraging patterns also show the typical feature of Lévy flights. Even light can be related to Lévy flights [30].

7.2Fire Fly Optimization (FFA)

The flashing light of fireflies is an amazing sight in the summer sky in the tropical and temperate regions. There are about two thousand firefly species, and most fireflies produce short and rhythmic flashes. The pattern of flashes is often unique for a particular species. The flashing light is produced by a process of bioluminescence, and the true functions of such signaling systems are still debating. However, two fundamental functions of such flashes are to attract mating partners (communication), and to attract potential prey. In addition, flashing may also serve as a protective warning mechanism. The rhythmic flash, the rate of flashing and the amount of time form part of the signal system that brings both sexes together. Females respond to a male's unique pattern of flashing in the same species, while in some species such as photuris, female fireflies can mimic the mating flashing pattern of other species so as to lure and eat the male fireflies who may mistake the flashes as a potential suitable mate.

The flashing light can be formulated in such a way that it is associated with the objective function to be optimized, which makes it possible to formulate new optimization algorithms. In the rest of this paper, we will first outline the basic formulation of the Firefly Algorithm (FA) and then discuss the implementation as well as analysis in detail.

Fire-fly optimization Algorithm is also developed by Xin-she Yang inspired by the flashing characteristics of fireflies. The main objective of flashing by a firefly is to act as a signal system to attract other fireflies. FFA algorithm is formulated according following assumptions:

- All fireflies are unisexual, so that any individual firefly will be attracted to all other fireflies.
- Attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by (and thus move towards) the brighter one; however, the intensity (apparent brightness) decrease as their mutual distance increases.
- If there are no fireflies brighter than a given firefly, it will move randomly [31].

For a maximization problem, the brightness can simply be proportional to the value of the objective function. Other forms of brightness can be defined in a similar way to the fitness function in genetic algorithms or the bacterial foraging algorithm (BFA) [69].In the firefly algorithm, there are two important issues: the variation of light intensity and formulation of the attractiveness. For simplicity, we can always assume that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function.

7.3Mathematical Modelling of CSFFA

CS algorithm is based on three rules

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest.
- The best nests with high quality of eggs will carry over to the next generation.
- The number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability pa $\in$ (0,1). Discovering operate on some set of worst nests, and discovered solutions dumped from farther calculations [30].

In CS algorithm, when generating new solutions $x^{(t+1)}$, lets take for cuckoo ‘i’, Levi flight is performed:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \otimes \text{Levy}(\lambda) \qquad (6.3.1)$$

where $\alpha > 0$ is the step size. We use $\alpha = O(L/10)$ where L is the characteristic scale of the problem of interest. In CS, Levy flight performs the random walk whose random step size or length is inherited from a Levy distribution which has infinite mean and infinite variance. The above equation is essentially the stochastic equation for random walk. In general, a random walk is a Markov chain whose next status/location only depends on the current location (the first term in the above equation) and the transition probability (the second term). The product $\oplus$ means entry wise multiplications. This entry wise product is similar to those used in PSO, but here the random walk via Lévy flight is more efficient in exploring the search space as its step length is much longer in the long run.

$$\text{Levy} \sim u = t^{-\lambda} \; (1 < \lambda < 3) \qquad (6.3.2)$$

Here the steps essentially form a random walk process with a power-law step-length distribution with a heavy tail. Some of the new solutions should be generated by Lévy walk around the best solution obtained so far, this will speed up the local search. However, a substantial fraction of the new solutions should be generated by far field randomization and whose locations should be far enough from the current best solution, this will make sure the system will not be trapped in a local optimum.

In the firefly algorithm, there are two important issues: the variation of light intensity and formulation of the attractiveness. For simplicity, we can always assume that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function. In the simplest case for maximum optimization problems, the brightness Iof a firefly at a particular location x can be chosen as $I(x) \propto f(x)$. However, the attractiveness β is relative, it should be seen in the eyes of the beholder or judged by the other fireflies. Thus, it will vary with the distance rij between firefly i and firefly j.

In addition, light intensity decreases with the distance from its source, and light is also absorbed in the media, so we should allow the attractiveness to vary with the degree of absorption. In FFA, the light intensity I(r) varies according to the inverse square law.

$$I(r) = \frac{I_s}{r^2} \tag{6.3.3}$$

Where I_s is the intensity of the source, r is the distance.β is defined as the attractiveness of a firefly and can be approximated as:

$$\beta = \frac{\beta_0}{1+\gamma r^2} \tag{6.3.4}$$

In the implementation, the actual form of attractiveness function β(r) can be any monotonically decreasing functions such as the following generalized form:

$$\beta_0 e^{-\gamma r^m} \quad \text{where } m \geq 1 \tag{6.3.5}$$

For a fixed γ, the characteristic length becomes Γ = γ −1/m → 1 as m → ∞. Conversely, for a given length scale Γ in an optimization problem, the parameter γ can be used as a typical initial value. That is γ = 1 Γm.

The distance between any two fireflies i and j at xi and xj , respectively, isthe Cartesian distance:

$$r_{ij} = |x_i - x_j| = \sqrt{\sum_{i=1}^{d} (x_{ik} - x_{jk})2} \tag{6.3.6}$$

The movement of a fire-fly ‘i’ is attracted to another more attractive (brighter) butterfly j is determined by :

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha \epsilon_j \tag{6.3.7}$$

Where rij is the Cartesian distance between any two fireflies. The β coefficient is determined as 3/2 is CS algorithm which is responsible for controlling the Lewy flights in CS algorithm. Attractiveness of the firefly ‘β’ is responsible for the movement of firefly in FFA algorithm; instead of relying on an arbitrary chosen fraction component; we use attractiveness (6.3.4) and luminous intensity coefficients (6.3.3) of fireflies to perform Levy flights, sigma factor and also to determine step-size in CS algorithm and the hybrid algorithm is realized.

7.4Computational Procedure of CFFFA

```
Objective function $f(x)$, $x = (x1, x2, \ldots, xd)$
Generate an initial population of n host nests xi;
While (t<MaxGeneration) or (stop criterion)
    Get a cuckoo randomly/generate a solution by Levy flights and then evaluate its
    quality/fitness $F_i$.
    Choose a nest among n (say, j) randomly.
    Vary attractiveness with distance r via $\exp[-\gamma r]$
    Evaluate new solutions and update light intensity.
    If ($F_i > F_j$)
            Replace j by the new solution
    End
    A fraction ($p_a$) of worse nests are abandoned and new ones/solutions are
    built/generated.
    Keep the best solutions (or nests with quality solutions)
    Rank the solutions and find the current best
End While
Post process results
```

8 HYBRID MULTI-VERSE OPTIMIZATION ALGORITHM AND PARTICLE SWARM OPTIMIZATION ALGORITHM (MVOPSO)

8.1Multi-Verse Optimization algorithm Algorithm (MVO)

The big bang theory discusses that our universe starts with a massive explosion. According to this theory, the big bang is the origin of everything in this world, and there was nothing before that. Multi-verse theory is another recent and well-known theory between physicists. It is believed in this theory that there are more than one big bang and each big bang causes the birth of a universe. The term multi-verse stands opposite of universe, which refers to the existence of other universes in addition to the universe that we all are living in . Multiple universes interact and might even collide with each other in the multi-verse theory. The multi-verse theory also suggests that there might be different physical laws in each of the universes.

We chose three main concepts of the multi-verse theory as the inspiration for the MVO algorithm: white holes, black holes, and wormholes. A white hole has never seen in our universe, but physicists think that the big bang can be considered as a white hole and may be the main component for the birth of a universe. It is also argued in the cyclic model of multi-verse theory. that big bangs/white holes are created where the collisions between parallel universes occur. Black holes, which have been observed frequently, behave completely in contrast to white wholes. They attract everything including light beams with their extremely high gravitational force. Wormholes are those holes that connect different parts of a universe together. The wormholes in the multi-verse theory act as time/space travel tunnels where objects are able to travel instantly between any corners of a universe (or even from one universe to another).

Every universe has an inflation rate (eternal inflation) that causes its expansion through space. Inflation speed of a universe is very important in terms of forming stars, planets, asteroids, black holes, white holes, wormholes, physical laws, and suitability for life. It is argued in one of the cyclic multi-verse models that multiple universes interact via white, black, and wormholes to reach a stable situation. This is the exact inspiration of the MVO algorithm, which is conceptually and mathematically modelled in the following subsection.

As discussed, a population-based algorithm divides the search process into two phases: exploration versus exploitation. We utilize the concepts of white hole and black hole to explore search spaces by MVO. In contrast, the wormholes assist MVO in exploiting the search spaces. We assume that each solution is analogous to a universe and each variable in the solution is an object in that universe. In addition, we assign each solution an inflation rate, which is proportional to the corresponding fitness function value of the solution. We also use the term time instead of the iteration in this paper since it is a common term in multi-verse theory and cosmology.

The following rules are applied to the universes of MVO during optimization:

- The more inflation rate, the more probability of having white hole.
- The higher inflation rate, lesser the probability of having black holes.
- Universes with higher inflation rate tend to send objects through the white holes.
- Universes with lower inflation rate tend to receive more objects through the black holes.
- The objects in all universes may face random movement towards the best universe through the wormholes regardless of the inflation rate.

When a white/black tunnel is established between two universes, the universe with higher inflation rate is considered to have white hole, wherein the universe with lesser inflation rate is assumed to have black holes. The objects are then transferred from the white holes of the source universe to the black holes of the destination universe. This mechanism allows the universes to easily exchange objects. To improve the whole inflation rate of the universes, we assume that the universes with high inflation rates are highly probable to have white holes. In opposition, the universes with low inflation rates have a high probability of having black holes. Therefore, there is always high possibility to move objects from a universe with high inflation rate to a universe with low inflation rate. This guarantees the improvement of the average inflation rates of the whole universes over the iterations.

8.2Particle Swarm Optimization algorithm Algorithm (PSO)

Particle swarm optimization (PSO) is described as population based stochastic optimization technique which is developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by the social behavior of fish schooling or bird flocking.

PSO have got many similarities with evolutionary computation techniques such as Genetic Algorithms. A population of random solutions and searches for optima by updating generations initializes the system. Though, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, which are called particles, fly through the problem space by following the current optimum particles. The detailed information will be given in following sections as below.

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO is applied in many areas such as function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

Particle Swarm Optimization can sound complicated even though it's really a very simple algorithm. Over many iterations, a group of variables have their values adjusted closer to the member whose value is closest to the target at any given moment. Suppose a flock of birds circling over an area where they can smell a hidden source of food, the one who is closest to the food chirps will be the loudest and the other birds swing around in his direction. If any of the other circling birds comes closer to the target than the first, it chirps louder, and the others move towards him. This pattern continues until one of the birds happens upon the food. It's an algorithm that's simple and easy to implement.

The algorithm keeps track of three global variables:

- Target value or condition
- Global best (gBest) value indicating which particle's data is currently closest to the Target
- Stopping value indicating when the algorithm should stop if the Target isn't found [35].

Each particle consists of:

- Data representing a possible solution
- A Velocity value which indicates how much the Data can be changed
- A personal best (pBest) value indicating the closest the particle's Data has ever come to the Target

The particles' data could be anything. In the flocking birds example on top of, the data would be the X, Y, Z coordinates of each bird. The individual coordinates of every bird would try and move nearer to the coordinates of the bird that is nearer to the food's coordinates (gBest). If the information could be a pattern or sequence, then individual pieces of the data would be manipulated until the pattern matches the target pattern.

The velocity worth is calculated in line with however way somebody's information is from the target. The additional it's, the larger the velocity value. In the birds example, the people furthest from the food would create an endeavor to stay up with the others by flying quicker toward the gBest bird. If the information could be a pattern or sequence, the speed would describe however totally different the pattern is from the target, and thus, what proportion it has to be modified to match the target.

Each particle's pBest value only indicates the closest the data has ever come to the target since the algorithm started. The gBest value only changes when any particle's pBest value comes closer to the target than gBest. Through every iteration of the algorithmic rule, gBest step by step moves nearer and nearer to the target till one in all the particles reaches the target.

It's also common to envision PSO algorithms mistreatment population topologies, or "neighborhoods", which can be smaller, localized subsets of the global best value. These neighborhoods will involve 2 or additional particles that area unit planned to act along, or subsets of the search space that particles happen into during testing. The use of neighborhoods usually facilitate the algorithmic rule to avoid obtaining stuck in native minimum.

A group of birds area unit arbitrarily looking out food in a neighborhood. There is only one food source and all birds do not know location of the food source. But, they know how far is the food source in each iteration and they tend to follow the bird which is nearest to the food source. PSO is perhaps the simplest optimization algorithm and have faster rate of convergence, however PSO often tends to suffer from premature convergence and local minimum stagnation. In PSOMVO, we use the same computational framework as MVO however we fine tune controlling parameters to that of PSO in order to impart the algorithm the advantage of faster convergence speed.

8.3Mathematical Modelling of MVOPSO

In order to mathematically model the white/black hole tunnels and exchange the objects of universes, we utilized a roulette wheel mechanism. At every iteration, we sort the universes based of their inflation rates and chose one of them by the roulette wheel to have a white hole. The following steps are done in order to do this.

Assume that,

$$\mathbf{U} = \begin{vmatrix} x_{1,1} & x_{1,2} \cdots\cdots & x_{1,d} \\ x_{2,1} & x_{2,2} \cdots\cdots & x_{2,d} \\ \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} \cdots\cdot & x_{n,d} \end{vmatrix} \qquad (7.3.1)$$

where d is the number of parameters (variables) and n is the number of universes (candidate solutions),$x_{i,j}$ indicates the jth parameter of ith universe, Ui shows the ith universe, NI(Ui) is normalized inflation rateof the ith universe, r1 is a random number in [0, 1], and $x_{j,k}$indicates the jth parameter of kth universe selected by aroulette wheel selection mechanism.

The selection and determination of white holes are done by the roulette wheel, which is based on the normalized inflation rate. The less inflation rate, the higher probability of sending objects though white/black hole tunnels. Please note that -NI should be changed to NI for the maximization problems. The exploration can be guaranteed using this mechanism since the universes are required to exchange objects and face abrupt changes to explore the search space.

With the above mechanism, the universes keep exchanging objects without perturbations. To maintain the diversity of universes and perform exploitation, we consider that each universe has wormholes to transport its objects through space randomly. It may be observed that the wormholes randomly change the objects of the universes without consideration of their inflation rates. To provide local changes for each universe and have high probability of improving the inflation rate using wormholes, we assume that worm hole tunnels are always established between a universe and the best universe formed so far. The formulation of this mechanism is as follows:

$x_i^j = x_j + TDR * ((ub_j - lb_j) * r4 + lb_j)$ $r3 < 0.5$, $r2 < WEP$ (7.3.2)

$x_i^j = x_j - TDR * ((ub_j - lb_j) * r4 + lb_j)$ $r3 > 0.5$, $r2 < WEP$ (7.3.3)

$x_i^j = x_i^j$, $r2 > WEP$ (7.3.4)

where x_j indicates the jth parameter of best universe formed so far, TDR is a coefficient, WEP is another coefficient, lb_j shows the lower bound of jth variable, ub_j is the upper bound of jth variable, $x_{j,I}$ indicates the jth parameter of ith universe, and r2, r3, r4 are random numbers in [0, 1].

It may be inferred from the pseudocodes and mathematical formulation that there are two main coefficients herein: wormhole existence probability (WEP) and travelling distance rate (TDR). The former coefficient is for defining the probability of wormhole's existence in universes. It is required to increase linearly over the iterations to emphasize exploitation as the progress of optimization process. Travelling distance rate is also a factor to define the distance rate (variation) that an object can be teleported by a wormhole around the best universe obtained so far. In contrast to WEP, TDR is increased over the iterations to have more precise exploitation/local search around the best obtained universe.

The adaptive formula for both coefficients are given as follows:

$$WEP = min + l * \left(\frac{max - min}{l}\right) \quad (7.3.5)$$

where min is the minimum (0.2 in this paper), max is the maximum (1 in this paper), l indicates the current iteration, and L shows the maximum iterations.

$TDR = 1 - l^{(1/p)}/L^{(1/p)}$ (7.3.6)

where p (in this paper equals 6) defines the exploitation accuracy over the iterations. The higher p, the sooner and more accurate exploitation/local search.

Note that WEP and TDR can be considered as constants as well, but we recommend adaptive values according to the results of this hybrid algorithm realized.

In the MVO algorithmic program, the optimization method starts with making a group of random universes. At each iteration, objects in the universes with high inflation rates tend to move to the universes with low inflation rates via white/black holes. Meanwhile, every single universe faces random teleportation in its objects through wormholes towards the best universe. This process is iterated until the satisfaction of an end criterion (a pre-defined maximum number of iterations, for instance).

The computational complexity of the proposed algorithms depends on number of iterations, number of universes, roulette wheel mechanism, and universe sorting mechanism. Sorting universe is done in every iteration, and we employ the Quicksort algorithm, which has the complexity of O(n log n) and O(n2) in the best and worst case, respectively. The roulette wheel selection is run for every variable in every universe over the iterations and is of O(n) or O(log n) based on the implementation.

To see how the proposed algorithm theoretically has the potential to solve optimization problems, some observations are as follows:

White holes are more possible to be created in the universes with high inflation rates, so they can send objects to other universes and assist them to improve their inflation rates.

where p (in this paper equals 6) defines the exploitation accuracy over the iterations. The higher p, the sooner and more accurate exploitation/local search. Note that WEP and TDR can be considered as constants as well, but we recommend adaptive values according to the results of this hybrid algorithm realized.

In the MVO formula, the optimization process starts with creating a set of random universes. At each iteration, objects in the universes with high inflation rates tend to move to the universes with low inflation rates via white/black holes. Meanwhile, every single universe faces random teleportation in its objects through wormholes towards the best universe. This process is iterated until the satisfaction of an end criterion (a pre-defined maximum number of iterations, for instance).

The computational complexity of the proposed algorithms depends on number of iterations, number of universes, roulette wheel mechanism, and universe sorting mechanism. Sorting universe is done in every iteration, and we employ the Quicksort algorithm, which has the complexity of O(n log n) and O(n2) in the best and worst case, respectively. The roulette wheel selection is run for every variable in every universe over the iterations and is of O(n) or O(log n) based on the implementation.

To see how the proposed algorithm theoretically has the potential to solve optimization problems, some observations are as follows:

White holes are more possible to be created in the universes with high inflation rates, so they can send objects to other universes and assist them to improve their inflation rates.

Black holes are more likely to be appeared in the universes with low inflation rates so they have higher probability to receive objects from other universes. This again will increase the possibility of up inflation rates for the universes with low inflation rates. White/black hole tunnels tend to transport objects from universes with high inflation rates to those with low inflation rates, so the overall/average inflation rate of all universes is improved over the course of iterations. Wormholes tend to appear randomly in any universe regardless of inflation rate, so the diversity of universes can be maintained over the course of iterations.

In PSO, W parameter is given by below equation:

w=wMax-(current-iter)*((wMax-wMin)/Max-iter) (7.3.8)

where wMaxis 0.9, wMin is 0.2 . This w parameter is responsible for updating velocity in PSO given by :

v[] = w* v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (7.3.9)

present[] = present[] + v[] (b) (7.3.10)

v[] is the particle velocity, persent[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2 [35].

We use ‘w’ parameter in PSO in equation 7.3.2, 7.3.3 and 7.3.4 and modify TDR parameters according to PSO velocity parameters eq (7.3.8) in order to determine the travelling distance rate to the best possible solution/universe at each iteration. Computational procedure framework remains intact with that of MVO as PSO is a very simple algorithm based on just relative velocity of search agents.

8.4Computational Procedure of MVOPSO

```
for each universe indexed by i
    for each object indexed by j
        r2=random([0,1]);
        if r2<Wormhole_existance_probability
            r3= random([0,1]);
            r4= random([0,1]);
            if r3<0.5
                U(i,j)=Best_universe(j) + Travelling_distance_rate * (( ub(j) -
                lb(j)) * r4 + lb(j));
                // adaptive parameters used
            else
                U(i,j)=Best_universe(j) - Travelling_distance_rate * (( ub(j) -
                lb(j)) * r4 + lb(j));
                // adaptive parameters used
            end if
        end if
    end for
end for
```

9 HYBRID BINARY BAT OPTIMIZATION ALGORITHM AND PARTICLE SWARM OPTIMIZATION ALGORITHM(BATPSO).

9.1Binary Bat Optimization Algorithm (BAT)

Bats are fascinating animals. They are the sole mammals with wings and that they conjointly have advanced capability of fix. It is estimated that there are about 996 different species which account for up to 20% of all mammal species. Their size ranges from the small bee bat (of regarding one.5 to 2g) to the enormous round the bend with length of regarding 2m and weight up to regarding one metric weight unit. Microbats typically have forearm length of about 2.2 to11cm. Most round the bend uses fix to a definite degree; among all the species, microbats are a celebrated example as microbats use fix extensively whereas mega-bats do not.

Most microbats are insectivores. Microbats use a kind of measuring system, called, fix, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a really loud sound pulse and listen for the echo that bounces back from the sur-rounding objects. Their pulses vary in properties and may be related to with their looking ways, reckoning on the species. Most round the bend uses short, frequency-modulated signals to comb through regarding AN octave, while others more often use constant-frequency signals for echolocation. Their signal information measure varies depends on the species, and often redoubled by victimization a lot of harmonics.

Though each pulse only lasts a few thousandths of a second (up to about 8 to 10 ms), however, it's a continuing frequency that is typically within the region of 25kHz to a hundred and fifty kHz. the everyday vary of frequencies for many bat species ar within the region between25kHz and 100kHz, though some species can emit higher frequencies up to 150 kHz.

Each unbearable burst might last usually five to twenty ms, and microbats emit regarding ten to 20such sound bursts each second. When attempting to find prey, the rate of pulse emission can be sped up to about 200 pulses per second when they fly near their prey. Such short sound bursts imply the fantastic ability of the signal processing power of bats. In fact, studies show the combination time of the bat ear is usually regarding 300 to 400 µs.

Amazingly, the emitted pulse may be as loud as one hundred ten dB, and, fortuitously, they are in the ultrasonic region. The loudness conjointly varies from the loudest once searching for prey and to a quieter base once orienting towards the prey. The travelling range of such short pulses are typically a few metres, depending on the actual frequencies.

Microbats will manage to avoid obstacles as tiny as skinny human hairs. Studies show that microbats use the time delay from the emission and detection of the echo, the time difference between their 2 ears, and the loudness variations of the echoes to build up three-dimensional scenario of the surrounding. They can detect the distance and orientation of the target, the type of prey, and even the moving speed of the prey such as small insects. Indeed, studies suggested that bats seem to be able to discriminate targets by the variations of the Doppler effect induced by the wing-flutter rates of the target insects.

Obviously, some round the bend have sensible sight, and most bats also have very sensitive smell sense. In reality, they're going to use all the senses as a mix to maximize the efficient detection of prey and sleek navigation. However, here we are only interested in the echolocation and the associated behavior. Such echolocation behavior of microbats can be formulated in such a way that it can be associated with the objective function to be optimized, and this make it possible to formulate new optimization algorithms.

Bat Algorithm (BAT) was proposed by Xin-She Yang based on the echolocation behavior of bats. The capability of location of microbats is fascinating as these bats can find their prey and discriminate different types of insects. BAT algorithm is based on following assumptions

- All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way.
- Bats fly randomly with velocity vi at position xi with a fixed frequency fmin, varying wavelength λ and loudness A0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r Є [0, 1], depending on the proximity of their target.
- Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A0 to a minimum constant value Amin [36].

Another obvious simplification is that no ray tracing is employed in estimating the time delay and 3-dimensional topography. Though this might be a good feature for the application in computational geometry, however, we will not use this as it is more computationally extensive in multidimensional cases.

In addition to those simplified assumptions, we also use the following approximations, for simplicity. In general, the frequency f in an exceedingly vary [fmin, fmax] corresponds toa range of wavelengths [λmin, λmax]. For example a frequency vary of [20kHz, 500kHz] corresponds to a range of wavelengths from 0.7mm to 17mm.

For a given downside, we are able to additionally use any wavelength for the benefit of implementation. In the actual implementation, we are able to regulate the vary by adjusting the wavelengths(or frequencies), and therefore the detectable vary (or the most important wavelength) ought to be chosen.

Particle Swarm Algorithm (PSO) is already discussed in section 7.2

9.2Mathematical Modelling of BATPSO

In simulations, we use virtual bats naturally. We have to define the rules how their positions x_i and velocities v_i in a d-dimensional search space are updated. The new solutions x_i^t and velocities v_i^t at time step t are given by:

$$f_i = f_{min} + \beta \quad (f_{max} - f_{min}) \qquad (8.2.1)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*) f_i \qquad (8.2.2)$$

$$x_i^t = x_i^{t-1} + v_i^t \qquad (8.2.3)$$

where, β Є [0, 1], is a random vector drawn from a uniform distribution. Here x*is the current global best location (solution) which is located after comparing all the solutions amongst all the n bats.

As the product $\lambda_i f_i$ is the velocity increment, we can use either fi(or λi) to adjust the velocity change while fixing the other factor λi (or fi), depending on the type of the problem of interest. In our implementation, we will use fmin = 0 and fmax = 100, depending the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from [fmin, fmax].

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$X_{new} = x_{old} + \text{Є} A^t \qquad (8.2.4)$$

where Є = [−1, 1] is a random number, while $A_t = \langle A_{ti} \rangle$is the average loudness of all the bats at this time step.

The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization [82] as fi essentially controls the pace and range of the movement of the swarming particles. To a degree, BAT can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate. This is the core basis of our hybridization.

Furthermore, the loudness A_i and rate r_i of pulse emission is given by:

$$x_i^t = x_i^{t-1} + v_i^t \quad (8.2.5)$$

$$A_i^{t+1} = \alpha A_i^t \quad (8.2.6)$$

$$r_i^{t+1} = r_i^0[1-e^{-\gamma t}] \quad (8.2.7)$$

where α and γ are constants. In fact, it is similar to the cooling factor of a cooling schedule in the simulated annealing.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimization algorithm is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (8.2.8)
present[] = present[] + v[] (8.2.9)

v[] is the particle velocity, present[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2 [35].

In BATPSO, In BAT minimum fitness function is located by an array-based greedy algorithm. All fitness function of array pos[i:] parameters are calculated and 'min' function is used to find the fitness minimum. According to the observation, this greedy algorithm works best in higher values; however, finds it difficult in finding the global minimum as it reaches lower values; especially fractional fitness value. PSO is based on element-wise pos[I,j] search and updating the velocity to converge to a global minimum. PSO works best at lower bound boundary fitness values. Therefore, in this hybrid, both PSO and BAT are run in parallel and they perform a comparison between both minimum fitness function at each iteration. The lowest value is taken and both PSO and BAT is updated with the lowest value and respective positions are updated simultaneously.

The numerical efficiency of the WOA algorithm developed in this study was tested by solving 23 mathematical optimization problems explained in the next section. The next section investigates the effectiveness of all hybrid algorithms realized in practice with respect to these 23 benchmark functions.

9.3Computational Procedure of BATPSO

Objective function f(x), x = (x1, ..., xd)^T

Initialize the bat population x_i (i = 1, 2, ..., n) and v_i

Define pulse frequency fi at xi

```
Initialize pulse rates ri and the loudness Ai
while (t <Max number of iterations)
    Generate new solutions by adjusting frequency, and updating    velocities and locations/solutions
[equations (8.2.1) to (8.2.4)]
    if (rand >ri)
        Select a solution among the best solutions
        Generate a local solution around the selected best solution
    end if
    Generate a new solution by flying randomly
    if (rand < $A_i$& $f(x_i) < f(x_*)$)
        Accept the new solutions
        Increase $r_i$ and reduce $A_i$
    end if
    for each dimension d = 1, ..., n do
       Pick random numbers: $r_p, r_g \sim U(0,1)$
       Update the particle's velocity: $v_{i,d} \leftarrow \omega v_{i,d} + \varphi_p r_p (p_{i,}$
      $d - x_{i,} d) + \varphi_g r_g (g_d - x_{i,} d)$
      Update the particle's position: $x_i \leftarrow x_i + v_i$
      if $f(x_i) < f(p_i)$ then
    Update the particle's best known position: $p_i \leftarrow x_i$
```

```
         if $f(p_i) < f(g)$ then
            Update the swarm's best known position: $g \leftarrow p_i$
        End if
    End For
    Rank the bats and find the current best $x_*$
    Compare PSO and BAT solution and update position
end while
Postprocess results
```

10 FITNESS FUNCTIONS AND ANALYSIS

A fitness function is a particular type of objective function that is used to summarize, as a single figure of merit, how close a given design solution is to achieving the set aims. Fitness functions are used in genetic programming and genetic algorithms to guide simulations towards optimal design solutions.

In particular, in the fields of genetic programming and genetic algorithms, each design solution is commonly represented as a string of numbers (referred to as a chromosome). After each round of testing, or simulation, the idea is to delete the n worst design solutions, and to breed n new ones from the best design solutions. Each design solution, therefore, needs to be awarded a figure of merit, to indicate how close it came to meeting the overall specification, and this is generated by applying the fitness function to the test, or simulation, results obtained from that solution.

The reason that genetic algorithms cannot be considered to be a lazy way of performing design work is precisely because of the effort involved in designing a workable fitness function. Even though it is no longer the human designer, but the computer which comes up with the final design, it is still the human designer who has to design the fitness function. If this is designed badly, the algorithm will either converge on an inappropriate solution, or will have difficulty converging at all.

The fitness function must not only correlate closely with the designer's goal, it must also be computed quickly. Speed of execution is very important, as a typical genetic algorithm must be iterated many times in order to produce a usable result for a non-trivial problem.

Fitness approximation may be appropriate, especially in the following cases:

- Fitness computation time of a single solution is extremely high
- Precise model for fitness computation is missing
- The fitness function is uncertain or noisy.

Two main classes of fitness functions exist: one where the fitness function does not change, as in optimizing a fixed function or testing with a fixed set of test cases; and one where the fitness function is mutable, as in niche differentiation or co-evolving the set of test cases.

Another way of looking at fitness functions is in terms of a fitness landscape, which shows the fitness for each possible chromosome.

In this section, 23 classical benchmark functions are discussed which used by many researchers [37- 42] for benchmarking optimization algorithms. all the hybrid algorithms are benchmarked together with their parent algorithms on 23 classical benchmark functions. Despite the simplicity, we have selected the fitness functions to be able to evaluate our results with those of current metaheuristic and evolutionary optimization algorithms. These functions are listed in the table 1-3 below; where 'dim' indicates the dimension of the function, range is the boundary of the functions search space and fmin is the global optimum.

The test functions used in benchmarking are basically minimization functions and can be divided into four groups: unimodal, multimodal, fixed dimension multimodal and composite functions.

10.1 Unimodal Fitness Functions

Functions F1–F7 are unimodal since they have only one global optimum. These functions allow to evaluate the exploitation capability of the investigated meta-heuristic algorithms.

Function	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \Pi_{i=1}^{n} \lvert x_i \rvert$	30	[-10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j^2 \right)$	30	[-100, 100]	0
$f_4(x) = \max_i \{\lvert x_i \rvert, \quad 1 \le i \le n\}$	30	[-100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30, 30]	0
$f_6(x) = \sum_{i=1}^{n} [(x_i + 0.5)^2]$	30	[-100, 100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$	30	[-1.28, 1.28]	0

Table 1 - Unimodal Functions

Their 3D representation of above unimodal fitness functions are given below:

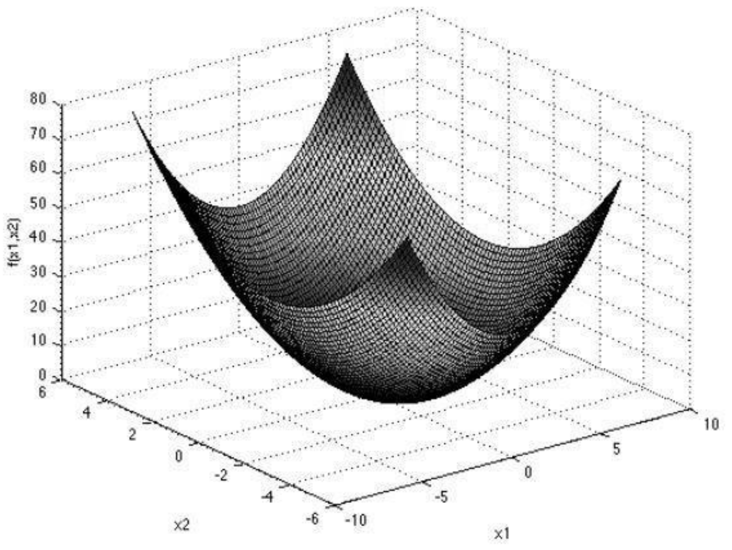

Figure 1-F1(x)

$$\boldsymbol{f_1(x) = \sum_{i=1}^{n} x_i^2}$$

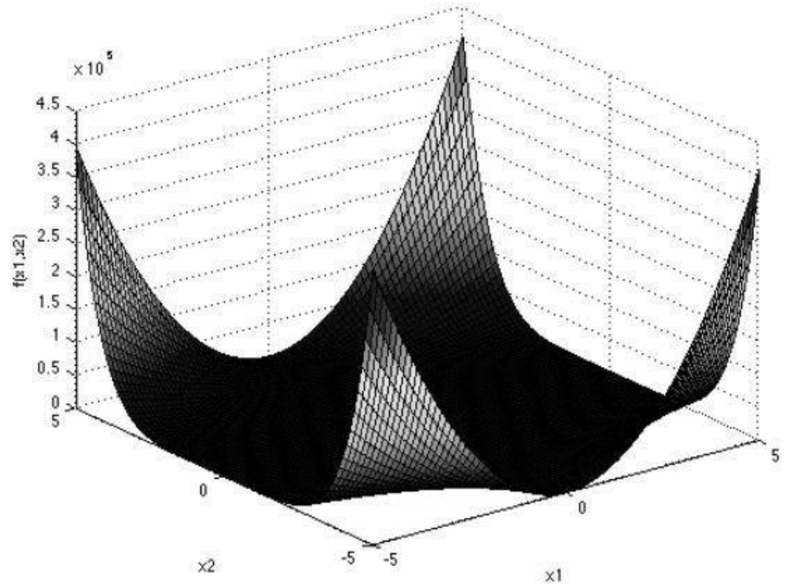

Figure 2-F2(x)

$$f_2(x) = \sum_{i=1}^{n} |x_i| + \Pi_{i=1}^{n} |x_i|$$

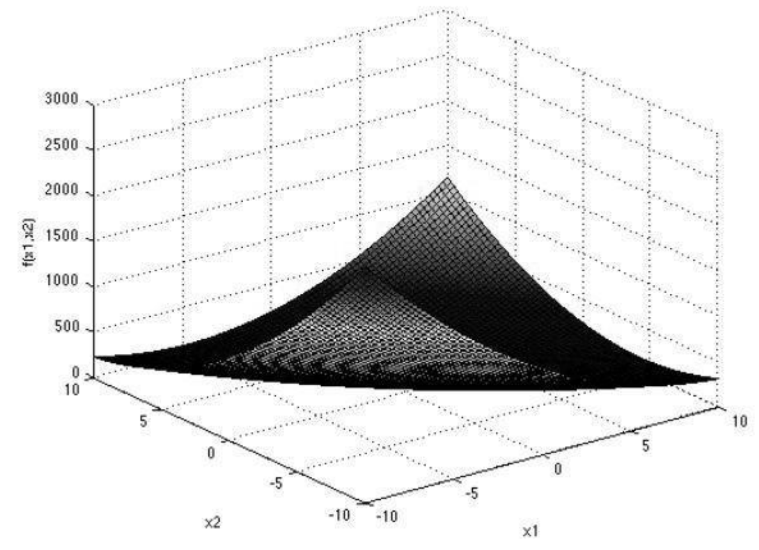

Figure 3-F3(x)

$$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j^2 \right)$$

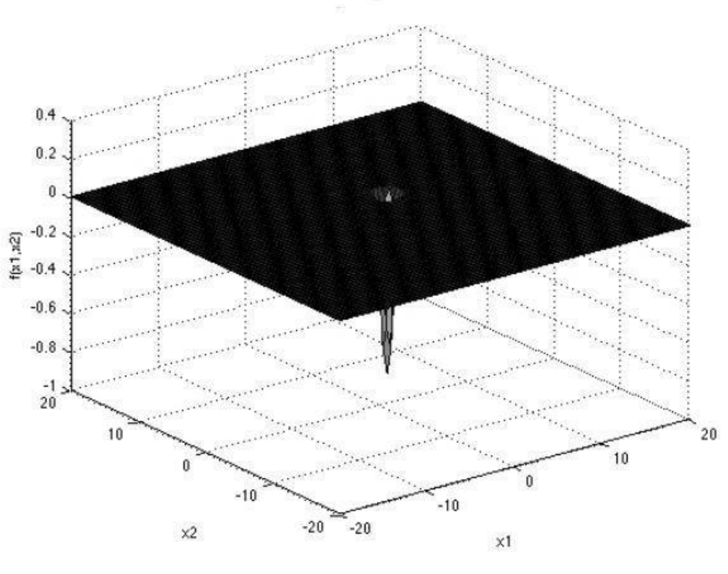

Figure 4-F4(x)

$$f_4(x) = \max_i \{|x_i|, \qquad 1 \le i \ \le n\}$$

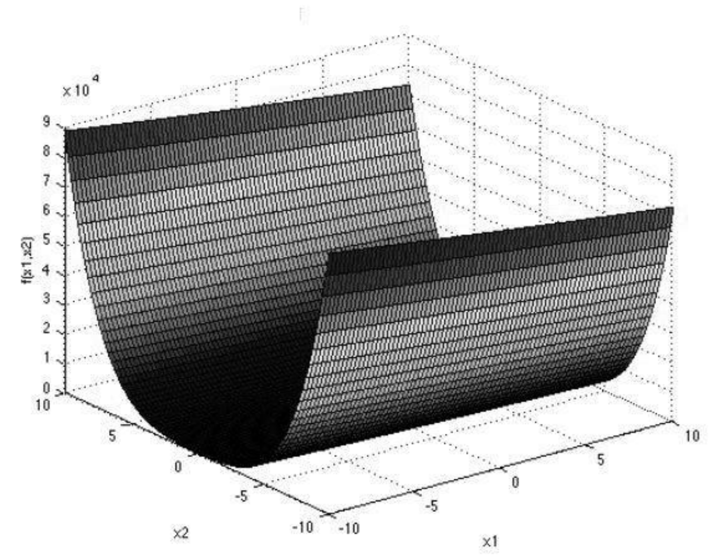

Figure 5-F5(x)

$$f_5(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

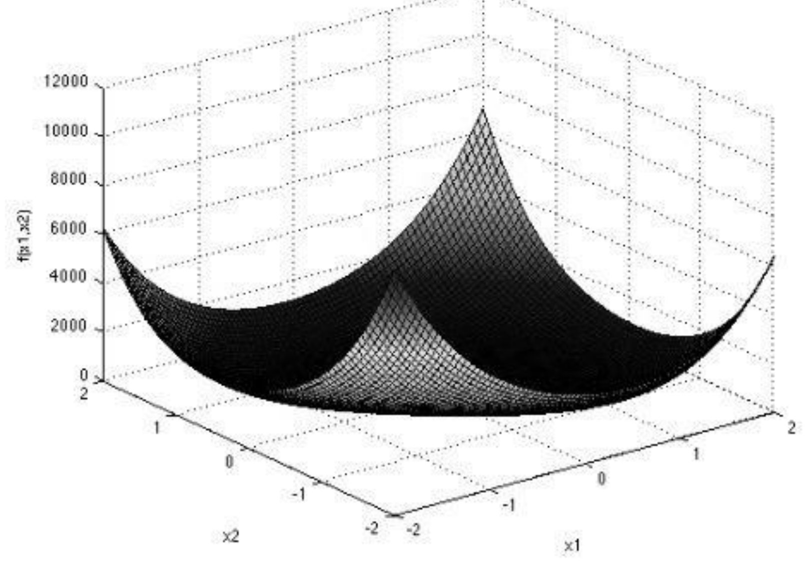

Figure 6-F6(x)

$$f_6(x) = \sum_{i=1}^{n}[(x_i + 0.5)^2]$$

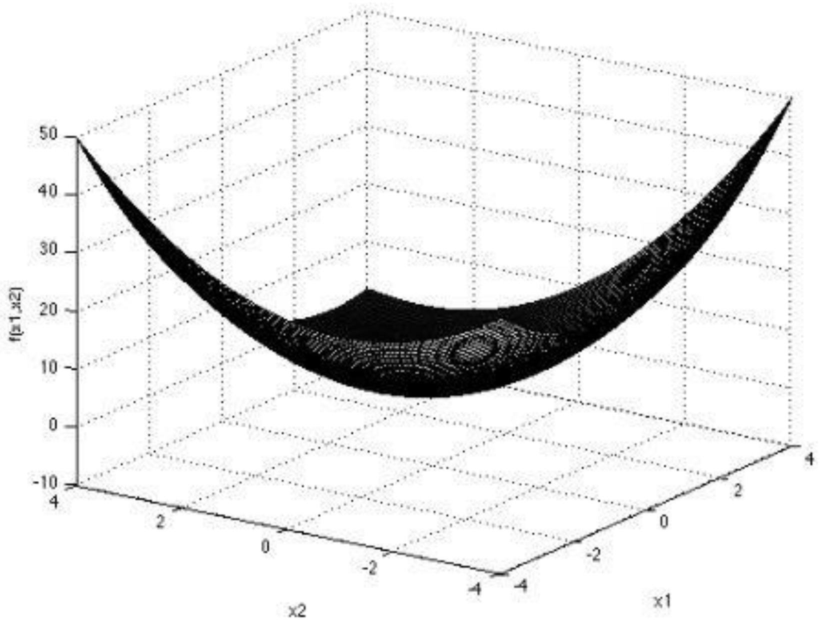

Figure 7- F7(x)

$$f_7(x) = \sum_{i=1}^{n} ix_i^4 + random[0,1]$$

10.2 Multimodal Fitness Functions

In contrast to the unimodal functions, multimodal functions have many local optima with the number increasing exponentially with dimension. This makes them suitable for benchmarking the exploration ability of an algorithm. Functions F8–F13 are multimodal and are specified in the table below:

Function	Dim	Range	f_{min}
$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500, 500]	-418.9289 x5
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10$	30	[-5.12, 5.12]	0
$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n\sum_{i=1}^{n} x_i^2}} - \exp(1n \sum_{i=1}^{n} \cos(2\pi x_i))+$ 20+e	30	[-32, 32]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600, 600]	0
$f_{12}(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i - 1)^2 + [1 + \sin(3\pi x_i + 1)] + (x_n - 1)^2 + \left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	30	[-50, 50]	0
$f_{13}(x) = \sum_{i=1}^{n} sin(x_i).\left(\sin\left(\frac{i.x_i^2}{\pi}\right)\right)^{2m}, m = 10$	30	[0, π]	-4.687

Table 2- Multimodal Fitness Functions

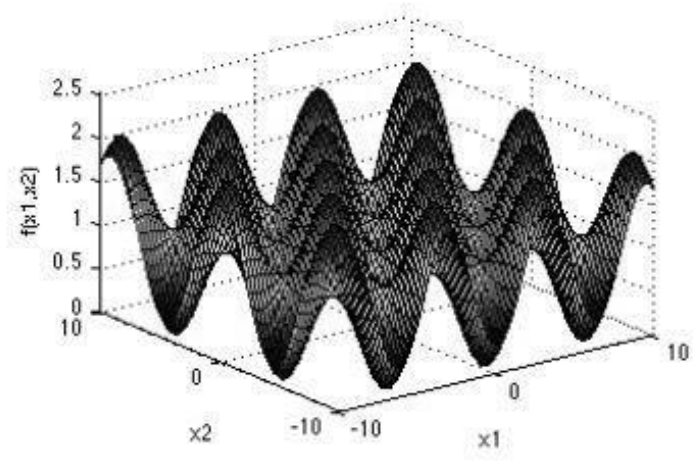

Figure 8 -F8(x)

$$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$$

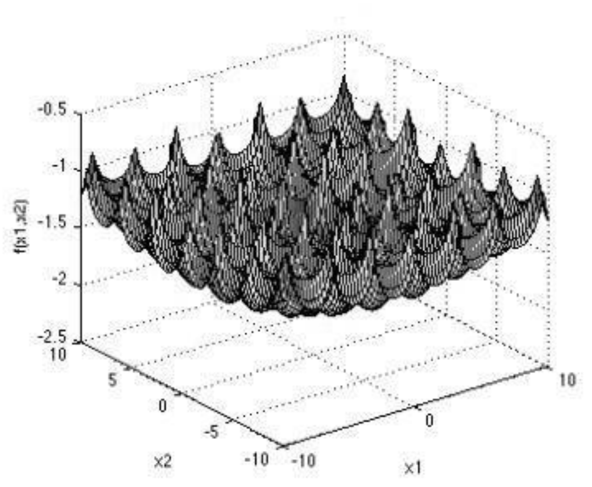

Figure 9-F9(x)

$$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10$$

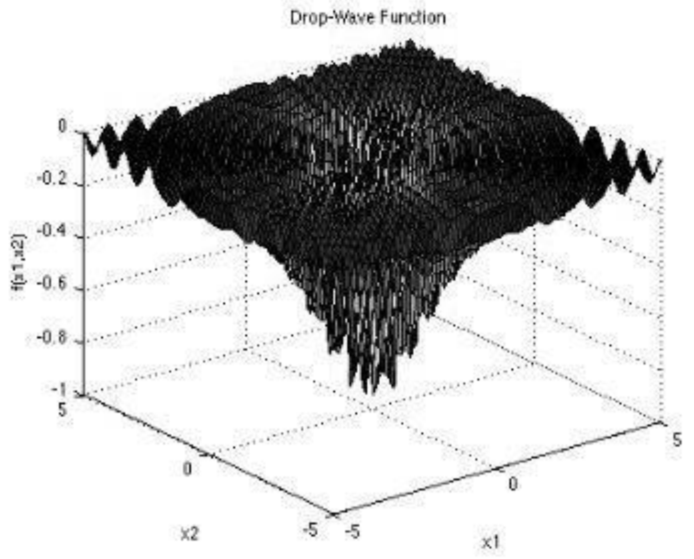

Figure 10-F10(x)

Fig 9.2.3 $f_{10}(x) = -20\exp(-0.2\sqrt{1/n\ \sum_{i=1}^{n} x_i^2} - \exp(1/n\ \sum_{i=1}^{n}\cos(2\pi x_i))$+ 20+e

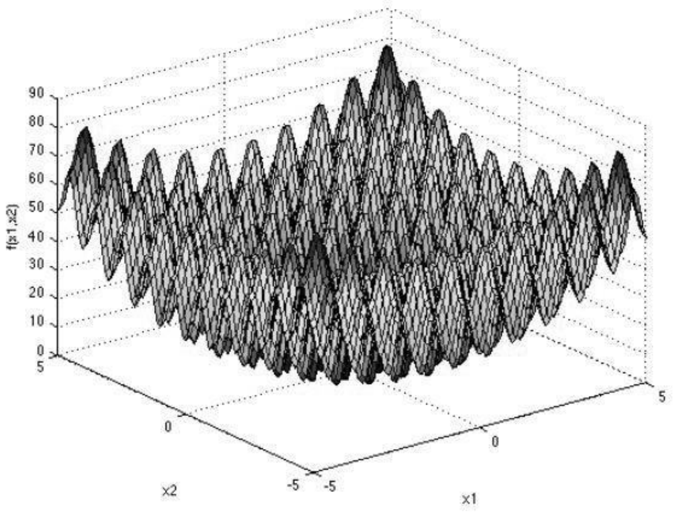

Figure 11 – F11(x)

$$\boldsymbol{f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1}$$

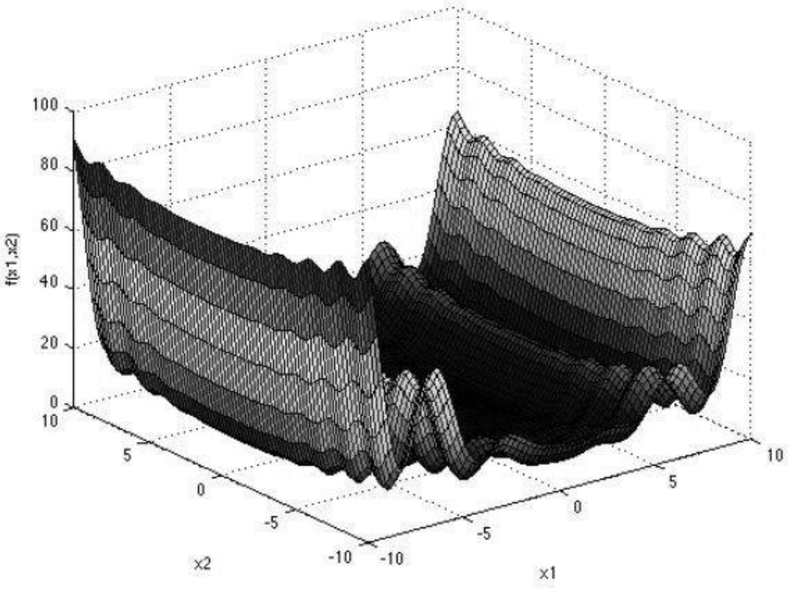

Figure 12-F12(x)

$$f_{12}(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i-1)^2 + [1+\sin(3\pi x_i+1)] + (x_n-1)^2 + [1+\sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5{,}100{,}4)$$

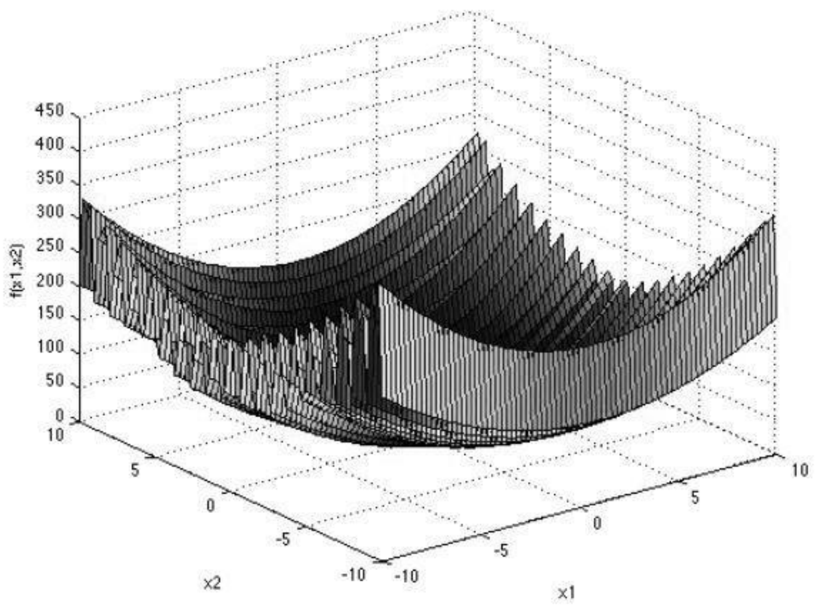

Figure 13-F13(x)

$$f_{13}(x) = \sum_{i=1}^{n} sin\,(x_i).\left(\sin\left(\frac{i.\,x_i^2}{\pi}\right)\right)^{2m}, m = 10$$

10.3 Fixed-dimension Multimodal Fitness Functions

Unlike unimodal functions, fixed-dimension multimodal functions include many local optima whose number increases exponentially with the problem size (number of design variables). Therefore, this kind of test problems turns very useful if the purpose is to evaluate the exploration capability of an optimization algorithm.

Function	Dim	Range	f_{min}
$f_{14}(x) = \left(\frac{1}{500} \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$	2	[-65,65]	1
$f_{15}(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right)^2$	4	[-5,5]	0.00030
$f_{16}(x) = 4x_1^2 + 2.1x_1^4 + \frac{1}{3}x_1^6 - 4x_2^2 + x_1x_2 + 4x_2^4$	2	[-5,5]	-1.0316
$f_{17}(x) = \sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2)$	2	[1,3]	-3.86
$f_{18}(x) = \sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$f_{19}(x) = \sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.5363

Table 3-Fixed-dimension Multimodal Fitness Functions

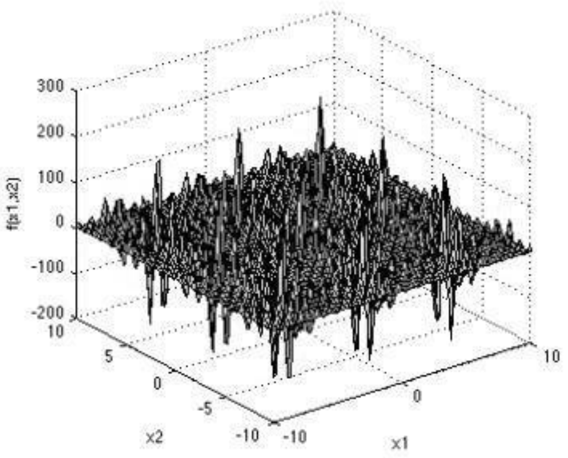

Figure 14-F14(x)

$$f_{14}(x) = \left(\frac{1}{500}\sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$$

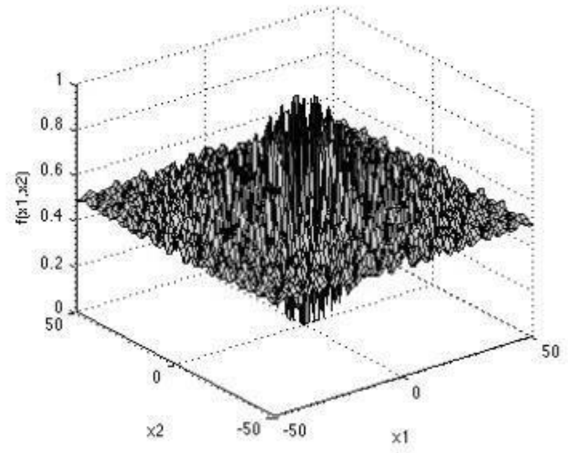

Figure 15-F15(x)

$$f_{15}(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$$

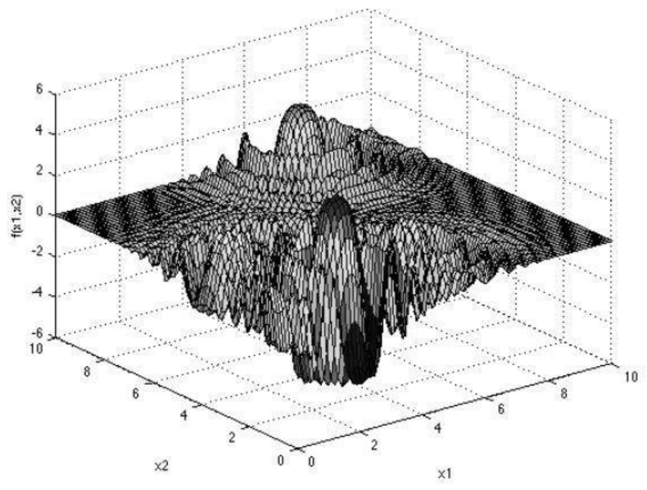

Figure 16-F16(x)

$$f_{16}(x) = 4x_1^2 + 2.1x_1^4 + \frac{1}{3}x_1^6 - 4x_2^2 + x_1x_2 + 4x_2^4$$

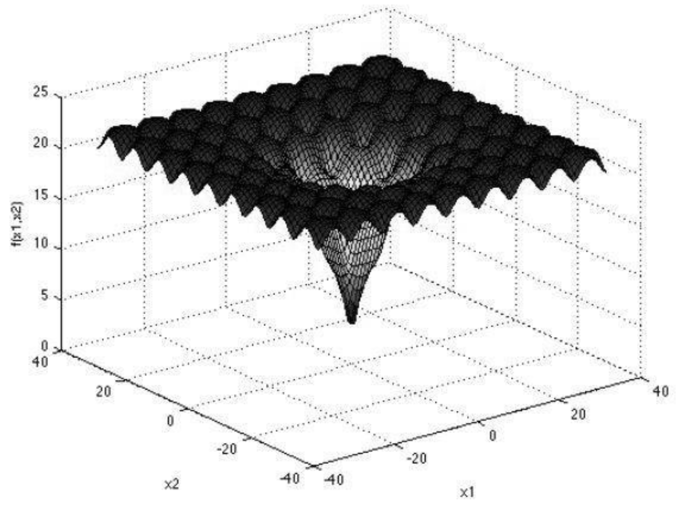

Figure 17-F17(x)

$$f_{17}(x) = \sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2)$$

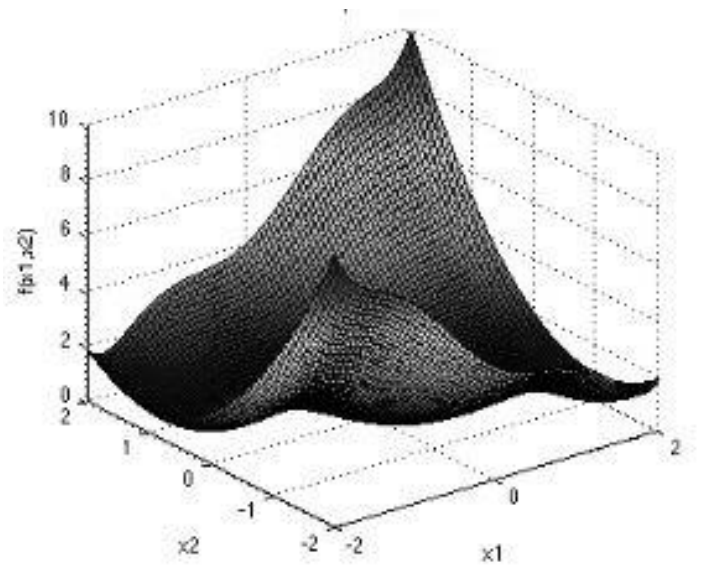

Figure 18-F18(x)

$$f_{18}(x) = \sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$$

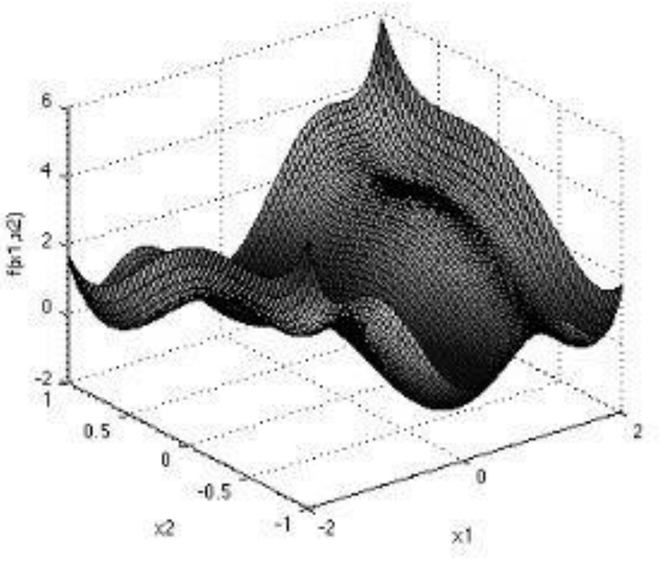

Figure 19-F19(x)

$$f_{19}(x) = \sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$$

10.4 Composite Benchmark Functions

The fourth class of benchmark functions employed includes composite functions, generally very challenging test beds for meta-heuristic algorithms. So, exploration and exploitation can be simultaneously benchmarked by the composite functions. Moreover, the local optima avoidance of an algorithm can be examined due to the massive number of local optima in such test functions.

Function	Dim	Range	f_{min}
$f_{20}(CF1)$: $f_1, f_2, f_3, f_4, \ldots, f_{10} = Sphere\ function$ $[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100,\ldots,5/100]$	10	[-5,5]	0
$f_{21}(CF2)$: $f_1, f_2, f_3, f_4, \ldots, f_{10} = Griewank's\ function$ $[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100,\ldots,5/100]$	10	[-5,5]	0
$f_{22}(CF3)$: $f_1, f_2 = Ackley's\ function$ $f_3, f_4, = Rastrigin's\ function$ $f_5, f_6, = Weierstra's\ function$ $f_7, f_8, = Griewank's\ function$ $f_9, f_{10}, = Sphere\ function$ $[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	10	[-5,5]	0
$f_{23}(CF4)$: $f_1, f_2 = Ackley's\ function$ $f_3, f_4, = Rastrigin's\ function$ $f_5, f_6, = Weierstra's\ function$ $f_7, f_8, = Griewank's\ function$ $f_9, f_{10}, = Sphere\ function$ $[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	10	[-5,5]	0

Table 4-Composite Benchmark Functions

11 RESULTS AND DISCUSSION

In this section, all the hybrid algorithms are benchmarked together with their parent algorithms on 23 classical benchmark functions used by many researchers [37- 42]. Despite the simplicity, we have selected the fitness functions to be able to evaluate our results with those of current metaheuristic and evolutionary optimization algorithms.

The test functions used in benchmarking are basically minimization functions. Unimodal functions allow to evaluate the exploitation capability of the investigated meta-heuristic algorithms. Multimodal and fixed-dimension multimodal functions are used for benchmarking the exploration ability of an algorithm as they include many local optima whose number increases exponentially with the problem size (number of design variables).The fourth class of benchmark functions employed includes composite functions, generally very challenging test beds for meta-heuristic algorithms. So, exploration and exploitation can be simultaneously benchmarked by the composite functions. Moreover, the local optima avoidance of an algorithm can be examined due to the massive number of local optima in such test functions.

We have run our hybrids along with their parent algorithms with which the hybrid algorithm is fabricated. The graphs are plotted below:

11.1 GWOABC fitness convergence graphs

Here, GWOABC (green), GWO (blue) and ABC (red) algorithms are run concurrently, and we have obtained following outputs.

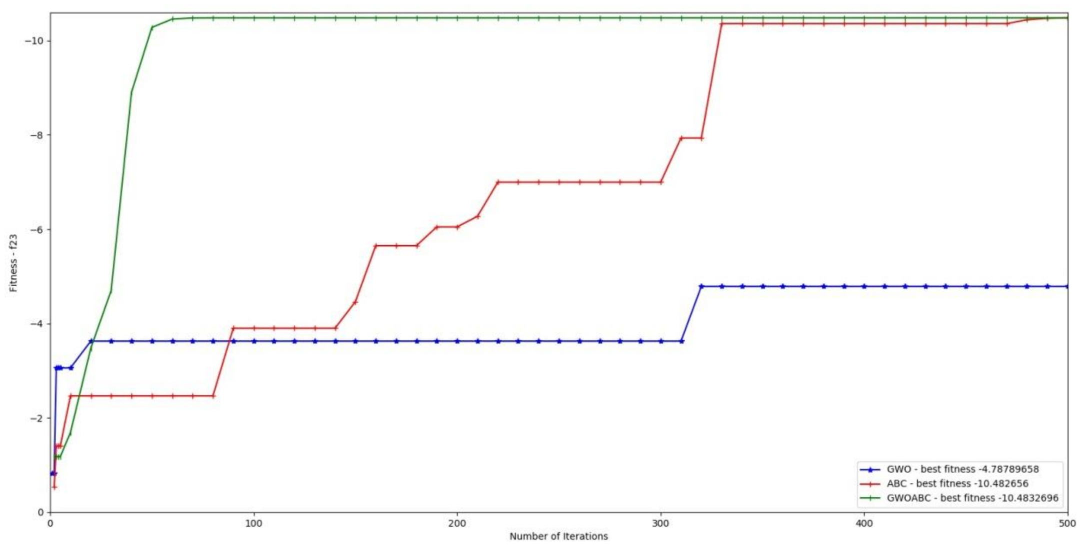

Figure 20-Fitness Function (23) GWOABC

From the graph it is evident that the proposed hybrid algorithm have faster convergence rate and have obtained better global optimum in most of the test function analysis. The graph illustrates the convergence rate in composite constrained benchmark function F23. Composite benchmark functions investigate both exploitation and exploration ability of an optimizer. The convergence line graph shows that GWO-ABC have very acute convergence rate towards lower fitness values and demonstrates better local minima avoidance at lower fitness values.

11.2 GWOWOA fitness convergence graphs

Here, GWOWOA (green), GWO (blue) and WOA (red) algorithms are run concurrently, and we have obtained following outputs.

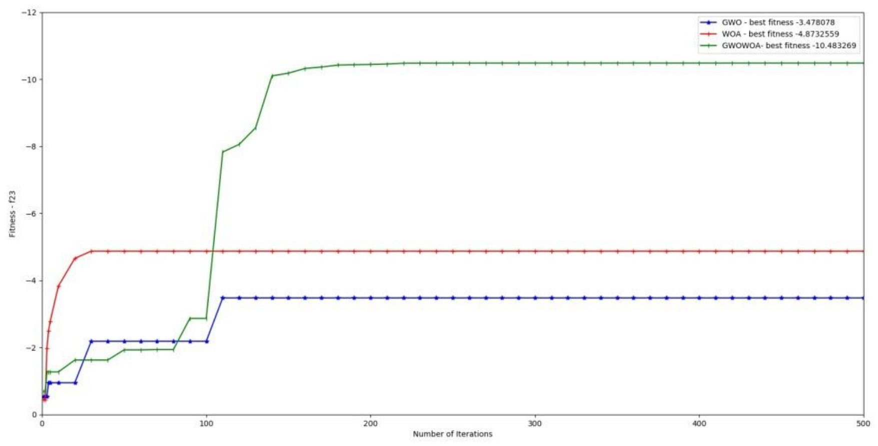

Figure 21- Fitness Function (23) GWOWOA

From the graph it is evident that the proposed hybrid algorithm have faster convergence rate and have obtained better global optimum in most of the test function analysis. The graph illustrates the convergence rate in composite constrained benchmark function F23. Composite benchmark functions investigate both exploitation and exploration ability of an optimizer. The convergence line graph shows that GWO-WOA have very acute convergence rate towards lower fitness values and demonstrates better local minima avoidance at lower fitness values. It starts very slowly and outpace GWO at 80th iteration and WOA around 100th iteration and achieves a superior global optimum value compared to both GWO and WOA.

11.3 MFOALO fitness convergence graphs

Here, MFOALO (green), MFO (blue) and ALO (red) algorithms are run simultaneously, and we have obtained following outputs.

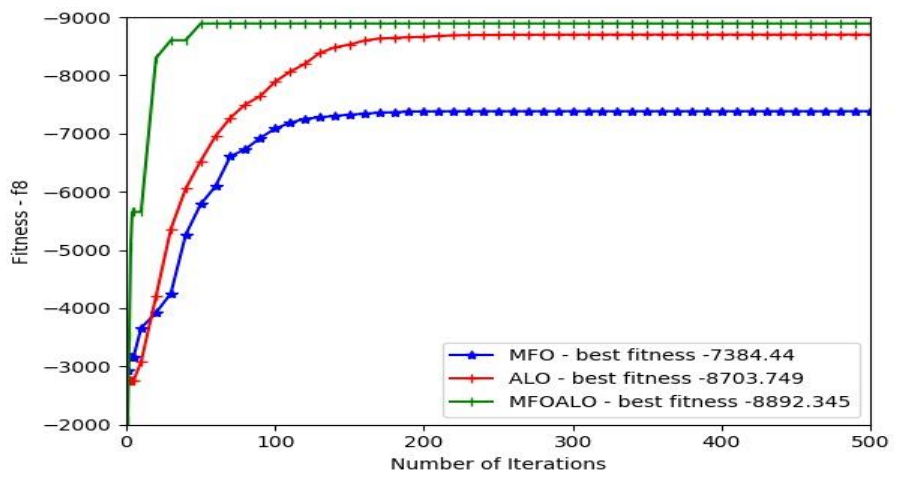

Figure 22-Fitness Function (8) MFOALO

MFOALO has the best performance for this multimodal benchmark function F8. MFOALO achieves a final fitness value of -8892 followed by ALO at -8703, whereas, MFO was able to obtain a value of only -7384. Multimodal functions allow to evaluate the exploration capability of the investigated meta-heuristic algorithms. Exploration consists of probing a much larger portion of the search space with the hope of finding other promising solutions that are yet to be refined. This operation amounts then to diversifying the search in order to avoid getting trapped in a local optimum. The graphical analysis clearly shows that MFOALO showcases immaculate exploration capability in order to jump outside from a local optimum. From the graph it is evident that the proposed hybrid algorithm have faster convergence rate and have obtained better global optimum in most of the test function analysis.

11.4 CSFFA fitness convergence graphs.

Here, CSFFA (green), CS (blue) and FFA (red) algorithms are run simultaneously, and we have obtained following outputs.

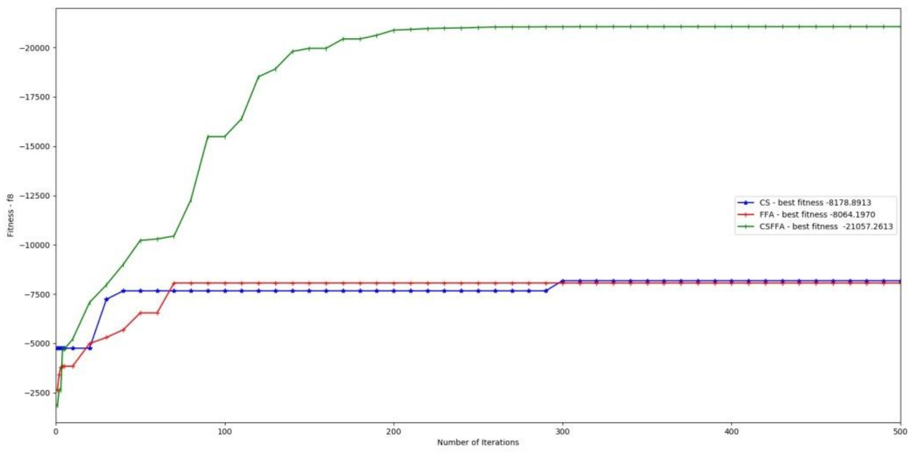

Figure 23- Fitness Function (8) CSFFA

Fig shows the line graphs of convergence of CSFFA hybrid algorithm with respect to their parent CS and FFA algorithms. From the graph, it can be observed that initially CSFFA have the slowest convergence rate compared to CS and FFA. However, CSFFA acquire high convergence rate and outpace both CS and FFA before 50th iteration and obtain a global optimum.

11.5 MVOPSO fitness convergence graphs.

Here, MVOPSO (green), PSO (blue) and MVO (red) algorithms are run concurrently, and we have obtained following outputs.

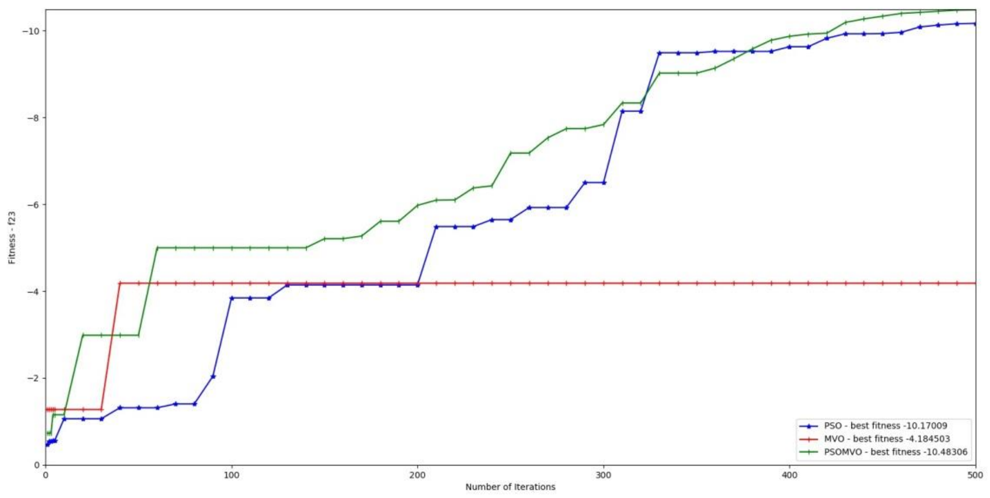

Figure 24-Fitness Function (23) PSOMVO

From the graph it is evident that the proposed PSOMVO hybrid algorithm have faster convergence rate and have obtained better global optimum the test function analysis. MVO get stuck in local minima however PSO experience premature convergence.

11.6 BATPSO fitness convergence graphs.

Here, BATPSO (green), PSO (blue) and BAT (red) algorithms are run concurrently, and we have obtained following outputs.

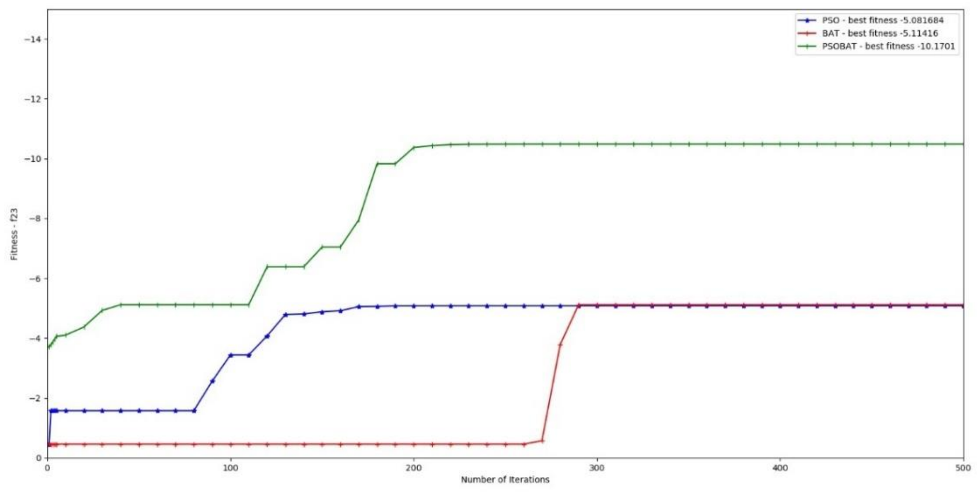

Figure 25-Fitness Function (23) BATPSO

The graph shows the rate of convergence on BATPSO, BAT and PSO algorithms in composite fitness function F23. These constrained benchmark functions test exploration ability of an optimization algorithm in a more complex search space. From the graph it is observed that BAT the fastest and sharp convergence rate in the beginning. PSO has the slowest convergence graph and very bottom out result compared to BAT and BATPSO. BATPSO arrives at improved global minimum than the other algorithm at the end.

11.7 Result Analysis

We use unimodal functions allow to evaluate the exploitation capability, multimodal functions have many local optima with the number increasing exponentially with dimension of the which enable us to assess the exploration capability of algorithm, composite functions which has challenging search space to investigate exploitation, exploration and local minimum avoidance capability of algorithms. A convergence behavior analysis is also investigated in proposed hybrid meta-heuristic algorithms. These hybrids are run 30 times continuously and their convergence graph and results obtained are also considered to verify the robustness of the algorithms proposed. Roman letters in table 5,6 and 7 denotes the rank of algorithm according to their fitness values.

10.7.1 Exploitation Analysis

Unimodal functions are the most suitable for evaluating exploitation ability of an optimization algorithm. Exploitation is to search the surrounding search area nearby the current solution, something like local search. Finding an algorithm that could handle both (exploitation and exploration) is challenging because they are two different objectives.

According to results in Table 5; all hybrid algorithms were able to provide either better or very competitive results compared to their parent algorithms

Hybrid Algorithm, Parent1, Parent 2	Grouping and ranking fitness functions according to performance of algorithms		
	Hybrid	Parent1	Parent 2
GWOABC, GWO, ABC	F1(i), F2(i), F3(i), F4(ii), F5(i), F6(i), F7(iii)	F1(ii), F2(ii), F3(iii), F4(i), F5(ii), F6(iii), F7(ii)	F1(iii), F2(iii), F3(ii), F4(iii), F5(iii), F6(ii), F7(i)
GWOWOA, GWO, WOA	F1(ii), F2(i), F3(i), F4(ii), F5(i), F6(i), F7(i)	F1(i), F2(iii), F3(iii), F4(iii), F5(iii), F6(i), F7(ii)	F1(iii), F2(ii), F3(iii), F4(i), F5(ii), F6(ii), F7(iii)
MFOALO, MFO, ALO	F1(iii), F2(i), F3(i), F4(ii), F5(i), F6(ii), F7(ii)	F1(ii), F2(iii), F3(ii), F4(ii), F5(ii), F6(i), F7(ii)	F1(i), F2(ii), F3(iii), F4(ii), F5(iii), F6(iii), F7(i)
CSFFA, CS, FFA	F1(i), F2(i), F3(i), F4(i), F5(ii), F6(ii), F7(i)	F1(iii), F2(ii), F3(ii), F4(ii), F5(i), F6(i), F7(iii)	F1(ii), F2(iii), F3(iii), F4(iii), F5(iii), F6(iii), F7(ii)
PSOMVO, MVO, PSO	F1(i), F2(i), F3(i), F4(i), F5(ii), F6(ii), F7(ii)	F1(iii), F2(ii), F3(ii), F4(iii), F5(iii), F6(iii), F7(i)	F1(ii), F2(iii), F3(iii), F4(ii), F5(i), F6(i), F7(iii)

PSOBAT, BAT, PSO	F1(iii), F2(i), F3(i), F4(ii), F5(i), F6(iii), F7(ii)	F1(ii), F2(iii), F3(ii), F4(ii), F5(ii), F6(ii), F7(ii)	F1(i), F2(ii), F3(iii), F4(ii), F5(iii), F6(i), F7(i)

Table 5-Comparison of results obtained for unimodal fitness functions.

Exploitation refers to the local search capability around the promising regions obtained in the exploration phase. Finding a proper balance between these two phases is considered a challenging task due to the stochastic nature of meta-heuristics. From the table we find that the hybrid performs the best in at least four to five out of seven unimodal functions and have given competitive results in the remaining unimodal test functions.

10.7.2 Exploration Analysis

In short, exploration is the ability of the algorithm to search for new individuals far from the current individual (current solution in the search space). Multimodal and fixed-dimension multimodal functions have many local optima with the number increasing exponentially with dimension of the which enable us to assess the exploration capability of algorithm. According to results in Table 6 all hybrid algorithms were able to provide either better or very competitive results compared to their parent algorithms.

Hybrid Algorithm, Parent1, Parent 2	Grouping and ranking fitness functions according to performance of algorithms		
	Hybrid	Parent1	Parent 2
GWOABC, GWO, ABC	F8(i), F9(i), F10(i), F11(ii), F12(i), F13(i), F14(i), F15(i), F16(iii), F17(ii), F18(i), F19(i),	F8(ii), F9(ii), F10(ii), F11(iii), F12(ii), F13(iii), F14(iii), F15(iii), F16(i), F17(i), F18(ii), F19(ii),	F8(iii), F9(iii), F10(iii), F11(i), F12(iii), F13(ii), F14(ii), F15(ii), F16(ii), F17(ii), F18(iii), F19(iii),

GWOWOA, GWO, WOA	F8(i), F9(i), F10(iii), F11(ii), F12(i), F13(i), F14(ii), F15(i), F16(iii), F17(i), F18(i), F19(i),	F8(ii), F9(ii), F10(i), F11(iii), F12(ii), F13(iii), F14(iii), F15(iii), F16(i), F17(ii), F18(ii), F19(ii),	F8(iii), F9(iii), F10(ii), F11(i), F12(iii), F13(ii), F14(i), F15(ii), F16(ii), F17(iii), F18(iii), F19(iii),
MFOALO, MFO, ALO	F8(i), F9(i), F10(ii), F11(i), F12(i), F13(ii), F14(iii), F15(ii), F16(iii), F17(ii), F18(i), F19(ii),	F8(ii), F9(iii), F10(i), F11(iii), F12(ii), F13(iii), F14(i), F15(iii), F16(ii), F17(iii), F18(iii), F19(iii),	F8(iii), F9(ii), F10(iii), F11(ii), F12(iii), F13(ii), F14(ii), F15(ii), F16(i), F17(i), F18(ii), F19(i),
CSFFA, CS, FFA	F8(iii), F9(i), F10(ii), F11(i), F12(i), F13(i), F14(iii), F15(i), F16(iii), F17(ii), F18(i), F19(iii),	F8(ii), F9(iii), F10(i), F11(iii), F12(ii), F13(ii), F14(i), F15(ii), F16(ii), F17(iii), F18(iii), F19(i),	F8(i), F9(ii), F10(iii), F11(ii), F12(iii), F13(i), F14(ii), F15(iii), F16(i), F17(i), F18(ii), F19(ii),
PSOMVO, MVO, PSO	F8(i), F9(i), F10(ii), F11(i), F12(i), F13(ii), F14(iii), F15(i), F16(iii), F17(ii), F18(i), F19(iii),	F8(ii), F9(iii), F10(i), F11(iii), F12(ii), F13(iii), F14(i),F15(ii), F16(ii),F17(iii), F18(iii), F19(i),	F8(iii), F9(ii), F10(iii), F11(ii), F12(iii), F13(ii), F14(ii), F15(iii), F16(i), F17(i), F18(ii), F19(ii),
PSOBAT, BAT, PSO	F8(iii), F9(i), F10(ii), F11(i), F12(i), F13(i), F14(ii), F15(i), F16(iii), F17(ii), F18(i), F19(iii),	F8(ii), F9(iii), F10(i), F11(iii), F12(ii), F13(ii), F14(iii), F15(ii), F16(ii), F17(iii), F18(iii), F19(i),	F8(i), F9(ii), F10(iii), F11(ii), F12(iii), F13(i), F14(i), F15(iii), F16(i), F17(i), F18(ii), F19(ii),

Table 6-Comparison of results obtained for multimodal and fixed-dimension multimodal fitness functions.

The exploration phase refers to the process of investigating the promising area(s) of the search space as broadly as possible. An algorithm needs to have stochastic operators to randomly and globally search the search space in order to support this phase. From the table we find that the hybrid performs the best in at least four to 8 out of 11 various multimodal functions and have given competitive results in the remaining test functions.

10.7.3Local Minimum Avoidance and Convergence behavior analysis

The fourth class of benchmark functions employed includes composite functions, generally very challenging test beds for meta-heuristic algorithms. So, exploration and exploitation can be simultaneously benchmarked by the composite functions. Moreover, the local optima avoidance of an algorithm can be examined due to the massive number of local optima in such test functions**.** According to table 7, all hybrid algorithm provides very competitive results on the composite benchmark functions. This demonstrates that hybrid algorithms show a good balance between exploration and exploitation that results in high local optima avoidance. From all the convergence graphs, hybrids show exceptional faster convergence rate compared to their parent hybrids.

Hybrid Algorithm, Parent1, Parent 2	Grouping and ranking fitness functions according to performance of algorithms		
	Hybrid	Parent1	Parent 2
GWOABC, GWO, ABC	F20(i), F21(i), F22(i), F23(ii),	F20(ii), F21(ii), F22(iii), F23(i)	F20(iii), F21(iii), F22(ii), F23(iii)
GWOWOA, GWO, WOA	F20(i), F21(i), F22(i), F23(i),	F20(ii), F21(ii), F22(ii), F23(i)	F20(iii), F21(iii), F22(iii), F23(ii)
MFOALO, MFO, ALO	F20(i), F21(i), F22(i), F23(i),	F20(ii), F21(ii), F22(iii), F23(i)	F20(iii), F21(iii), F22(ii), F23(iii)
CSFFA, CS, FFA	F20(i), F21(i), F22(i), F23(ii),	F20(ii), F21(iii), F22(iii), F23(i)	F20(iii), F21(ii), F22(ii), F23(iii)

PSOMVO, MVO, PSO	F20(i), F21(ii), F22(i), F23(i),	F20(ii), F21(i), F22(iii), F23(i)	F20(iii), F21(iii), F22(ii), F23(iii)
PSOBAT, BAT, PSO	F20(i), F21(i), F22(i), F23(ii),	F20(ii), F21(ii), F22(iii), F23(iii)	F20(iii), F21(iii), F22(ii), F23(i)

Table 7-Comparison of results obtained for Composite benchmark fitness functions.

Optimization results reported in Table 7 show that the hybrid algorithms were the best optimization algorithm in four test problems and was very competitive in the other cases. This proves that the hybrids can well balance exploration and exploitation phases.

12 HYBRID ALGORITHMS FOR CLASSICAL ENGINEERING DESIGN PROBLEM

Engineering design is the method that engineers use to identify and solve problems. In constrained engineering design process, engineers must identify solutions that include the most desired features and fewest negative characteristics. They should also specify the cost functions and their limitations of the given scenario, which could include time, cost, and the physical limits of tools and materials. Constrained engineering design optimization problems are usually computationally expensive due to non-linearity and non convexity of these constraint functions. Evolutionary population based algorithms are widely used to solve constrained optimization problems. Many researchers have implemented many heuristic and meta-heuristic optimization algorithms to solve constrained optimization problems in engineering design.

These meta-heuristic optimization algorithms are of great research interest in recent times due to their ability in finding optimal solutions within short time especially when these real world engineering design problems consists of large number of design variables and multiple constraints which makes the solution search-space larger, complicated and non-linear. Penalty function methods are found to be quite popular due to their simplicity and ease of implementation. In this method, search agents are assigned big objective function values if they violate any of the specified constraints. In this section, we try to solve a real world engineering design problem using hybrid algorithms in order to observe the performance and benchmark the performance .

12.1 Cantilever Beam Design

This is a structural optimization problem [12]. The objective is to design a minimum-mass cantilever beam. A cantilever beam includes five hollow elements with square-shaped cross-section. Since the mass is proportional to the cross-sectional area of the beam, the objective function for the problem is taken as the cross-sectional area. Assuming thickness is constant, there are a total of 5 structural parameters. The mathematical formulation of this problem can be described as follows:

Consider,

$$\vec{z} = [z_1 z_2 z_3 z_4 z_5]$$

Minimize the function,

$$f(\vec{z}) = 0.6224(z_1 + z_2 + z_3 + z_4 + z_5)$$

Subject to,

$$h_1(\vec{z}) = \frac{61}{z_1^3} + \frac{37}{z_2^3} + \frac{19}{z_3^3} + \frac{4}{z_4^3} + \frac{1}{z_5^3} \le 1$$

Where,

$$0.01 \le z_1, z_2, z_3, z_4, z_5 \le 100$$

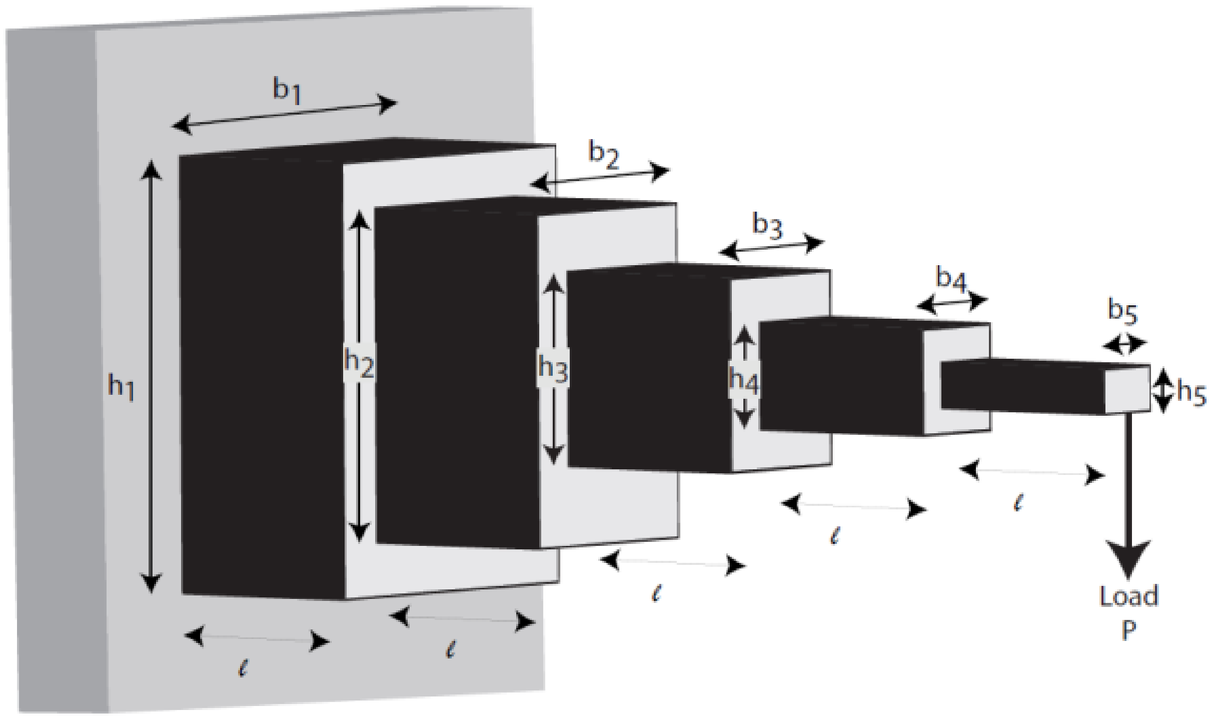

Figure 25-Cantilever Design Problem

Parameters	GWOABC	MFOALO	MVOPSO	CSFFA	GWOWOA	BATPSO
z_1	6.0089	6.0100	6.0101	6.0097	6.0089	6.0091
z_2	5.3001	5.3000	5.02999	5.0300	5.02999	5.03004
z_3	4.4900	4.4900	4.4900	4.49001	4.4899	4.4900
z_4	3.4904	3.4900	3.49006	3.4901	3.4900	3.4900
z_5	2.1601	2.1500	2.1506	2.1508	2.1509	2.1600
Optimum Weight $f(\vec{z})$	1.3400	1.3400	1.3399	1.3401	1.3399	1.3399

Table 8- Cantilever Beam Design results using hybrid algorithms.

Table 8 compares the best solutions for Cantilever Beam Design problem obtained by the proposed hybrid algorithms. As per the result, all hybrid algorithms was able to find the optimal solution which is very competitive to its other benchmarked algorithms. This evidences that the proposed algorithm is able to effectively optimize challenging constrained problems as well. All algorithms

13 CONCLUSION

This paper proposes following six hybrid algorithms: Hybrid Grey Wolf Optimization algorithm and Artificial Bee Colony Optimization algorithm (GWOABC), Hybrid Moth Flame Optimization Algorithm and Ant Lion Optimization algorithm (MFOALO), Hybrid Cuckoo Search Optimization algorithm and Fire Fly Optimization Algorithm(CSFFA), Hybrid Multi-Verse Optimization algorithm and Particle Swarm Optimization Algorithm (MVOPSO), Hybrid Grey Wolf Optimization algorithm and Whale Optimization Algorithm (GWOWOA), Hybrid Binary Bat Optimization Algorithm and Particle Swarm Optimization Algorithm(BATPSO). Twenty-three test functions were employed in order to benchmark the performance of the proposed algorithms in terms of exploration, exploitation, local optima avoidance, and convergence.

The results showed that the hybrids were able to provide highly competitive results compared to well known parent heuristics such as GWO, ABC, WOA, MFO, ALO, CS, FFA, PSO, MVO and BAT. First, the results on the unimodal functions showed the superior exploitation of the hybrid algorithms. Second, the exploration ability of hybrids were confirmed by the results on multimodal functions. Third, the results of the composite functions showed high local optima avoidance. Finally, the convergence analysis of hybrids were confirmed by the comparative convergence of this algorithm. Statistical testing is performed on all hybrid algorithm to validate the results and will be discussed in publishable papers.

Moreover, we used these proposed hybrid optimization algorithms to solve a real-world engineering design problem – Cantilever beam design, with large number of variables and constraints. The results show the capability of hybrid algorithms in handling various real-world con-junctional optimization problems under lower computational efforts. All hybrid algorithms were able to attain the optimal or near optimal solutions better than to most of the existing optimization algorithms subjected to the study. For future work, large scale optimization problems can be studied and realized using these algorithms. A self-adaptive method of choosing parameters can be developed in order to further improve the efficiency of these optimization algorithms. A multi-objective version of these algorithms can also be developed which find itself immense scope in diverse real-world optimization applications

APPENDIX – A

Benchmark Functions Python Code

```
# -*- coding: utf-8 -*-
"""
Created on May 17 12:46:20 2018

@author: Noel Jose ThengappurackalLaiju
"""

import numpy
import math

# define the function blocks
def prod( it ):
    p= 1
    for n in it:
        p *= n
    return p

def Ufun(x,a,k,m):
    y=k*((x-a)**m)*(x>a)+k*((-x-a)**m)*(x<(-a));
    return y

def F1(x):
    s=numpy.sum(x**2);
    return s

def F2(x):
    o=sum(abs(x))+prod(abs(x));
    return o;

def F3(x):
    dim=len(x)+1;
    o=0;
    for i in range(1,dim):
        o=o+(numpy.sum(x[0:i]))**2;
    return o;

def F4(x):
    o=max(abs(x));
    return o;

def F5(x):
    dim=len(x);
    o=numpy.sum(100*(x[1:dim]-(x[0:dim-1]**2))**2+(x[0:dim-1]-1)**2);
    return o;

def F6(x):
    o=numpy.sum(abs((x+.5))**2);
    return o;

def F7(x):
   dim=len(x);

   w=[i for i in range(len(x))]
   for i in range(0,dim):
        w[i]=i+1;
   o=numpy.sum(w*(x**4))+numpy.random.uniform(0,1);
   return o;

def F8(x):
    o=sum(-x*(numpy.sin(numpy.sqrt(abs(x)))));
    return o;

def F9(x):
    dim=len(x);
    o=numpy.sum(x**2-10*numpy.cos(2*math.pi*x))+10*dim;
    return o;


def F10(x):
    dim=len(x);
    o=-20*numpy.exp(-.2*numpy.sqrt(numpy.sum(x**2)/dim))-
numpy.exp(numpy.sum(numpy.cos(2*math.pi*x))/dim)+20+numpy.exp(1);
    return o;

def F11(x):
    dim=len(x);
    w=[i for i in range(len(x))]
    w=[i+1 for i in w];
    o=numpy.sum(x**2)/4000-prod(numpy.cos(x/numpy.sqrt(w)))+1;
    return o;

def F12(x):
    dim=len(x);
    o=(math.pi/dim)*(10*((numpy.sin(math.pi*(1+(x[0]+1)/4)))**2)+numpy.sum((((x[1:dim-
1]+1)/4)**2)*(1+10*((numpy.sin(math.pi*(1+(x[1:dim-1]+1)/4))))**2))+((x[dim-
1]+1)/4)**2)+numpy.sum(Ufun(x,10,100,4));
    return o;

def F13(x):
    dim=len(x);
    o=.1*((numpy.sin(3*math.pi*x[1]))**2+sum((x[0:dim-2]-1)**2*(1+(numpy.sin(3*math.pi*x[1:dim-1]))**2))+
    ((x[dim-1]-1)**2)*(1+(numpy.sin(2*math.pi*x[dim-1]))**2))+numpy.sum(Ufun(x,5,100,4));
    return o;

def F14(x):
     aS=[[-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32],[-32,-32,-32,-
32,-32,-16,-16,-16,-16,-16,0,0,0,0,0,16,16,16,16,16,32,32,32,32,32]];
aS=numpy.asarray(aS);
bS = numpy.zeros(25)
     v=numpy.matrix(x)
     for i in range(0,25):
         H=v-aS[:,i];
```

```
bS[i]=numpy.sum((numpy.power(H,6)));
     w=[i for i in range(25)]
     for i in range(0,24):
        w[i]=i+1;
     o=((1./500)+numpy.sum(1./(w+bS)))**(-1);
     return o;

def F15(L):
    aK=[.1957,.1947,.1735,.16,.0844,.0627,.0456,.0342,.0323,.0235,.0246];
bK=[.25,.5,1,2,4,6,8,10,12,14,16];
aK=numpy.asarray(aK);
bK=numpy.asarray(bK);
bK = 1/bK;
    fit=numpy.sum((aK-((L[0]*(bK**2+L[1]*bK))/(bK**2+L[2]*bK+L[3])))**2);
    return fit

def F16(L):
     o=4*(L[0]**2)-2.1*(L[0]**4)+(L[0]**6)/3+L[0]*L[1]-4*(L[1]**2)+4*(L[1]**4);
     return o

def F17(L):
    o=(L[1]-(L[0]**2)*5.1/(4*(numpy.pi**2))+5/numpy.pi*L[0]-6)**2+10*(1-
1/(8*numpy.pi))*numpy.cos(L[0])+10;
    return o

def F18(L):
    o=(1+(L[0]+L[1]+1)**2*(19-14*L[0]+3*(L[0]**2)-14*L[1]+6*L[0]*L[1]+3*L[1]**2))*(30+(2*L[0]-
3*L[1])**2*(18-32*L[0]+12*(L[0]**2)+48*L[1]-36*L[0]*L[1]+27*(L[1]**2)));
    return o
# map the inputs to the function blocks
def F19(L):
aH=[[3,10,30],[.1,10,35],[3,10,30],[.1,10,35]];
aH=numpy.asarray(aH);
cH=[1,1.2,3,3.2];
cH=numpy.asarray(cH);
    pH=[[.3689,.117,.2673],[.4699,.4387,.747],[.1091,.8732,.5547],[.03815,.5743,.8828]];
    pH=numpy.asarray(pH);
    o=0;
    for i in range(0,4):
     o=o-cH[i]*numpy.exp(-(numpy.sum(aH[i,:]*((L-pH[i,:])**2))));
    return o


def F20(L):
    aH=[[10,3,17,3.5,1.7,8],[.05,10,17,.1,8,14],[3,3.5,1.7,10,17,8],[17,8,.05,10,.1,14]];
aH=numpy.asarray(aH);
cH=[1,1.2,3,3.2];
cH=numpy.asarray(cH);

pH=[[.1312,.1696,.5569,.0124,.8283,.5886],[.2329,.4135,.8307,.3736,.1004,.9991],[.2348,.1415,.3522,.28
83,.3047,.6650],[.4047,.8828,.8732,.5743,.1091,.0381]];
    pH=numpy.asarray(pH);
    o=0;
    for i in range(0,4):
     o=o-cH[i]*numpy.exp(-(numpy.sum(aH[i,:]*((L-pH[i,:])**2))));
    return o
```

```
def F21(L):
    aSH=[[4,4,4,4],[1,1,1,1],[8,8,8,8],[6,6,6,6],[3,7,3,7],[2,9,2,9],[5,5,3,3],[8,1,8,1],[6,2,6,2],[7,3.6,7,3.6]];
cSH=[.1,.2,.2,.4,.4,.6,.3,.7,.5,.5];
aSH=numpy.asarray(aSH);
cSH=numpy.asarray(cSH);
    fit=0;
    for i in range(0,4):
      v=numpy.matrix(L-aSH[i,:])
      fit=fit-((v)*(v.T)+cSH[i])**(-1);
    o=fit.item(0);
    return o

def F22(L):
    aSH=[[4,4,4,4],[1,1,1,1],[8,8,8,8],[6,6,6,6],[3,7,3,7],[2,9,2,9],[5,5,3,3],[8,1,8,1],[6,2,6,2],[7,3.6,7,3.6]];
cSH=[.1,.2,.2,.4,.4,.6,.3,.7,.5,.5];
aSH=numpy.asarray(aSH);
cSH=numpy.asarray(cSH);
    fit=0;
    for i in range(0,6):
      v=numpy.matrix(L-aSH[i,:])
      fit=fit-((v)*(v.T)+cSH[i])**(-1);
    o=fit.item(0);
    return o

def F23(L):
    aSH=[[4,4,4,4],[1,1,1,1],[8,8,8,8],[6,6,6,6],[3,7,3,7],[2,9,2,9],[5,5,3,3],[8,1,8,1],[6,2,6,2],[7,3.6,7,3.6]];
cSH=[.1,.2,.2,.4,.4,.6,.3,.7,.5,.5];
aSH=numpy.asarray(aSH);
cSH=numpy.asarray(cSH);
    fit=0;
    for i in range(0,9):
      v=numpy.matrix(L-aSH[i,:])
      fit=fit-((v)*(v.T)+cSH[i])**(-1);
    o=fit.item(0);
    return o

def getFunctionDetails(a):

    # [name, lb, ub, dim]
    param = {  0: ["F1",-100,100,30],
1 : ["F2",-10,10,30],
2 : ["F3",-100,100,30],
3 : ["F4",-100,100,30] ,
4 : ["F5",-30,30,30],
5 : ["F6",-100,100,30],
6 : ["F7",-1.28,1.28,30],
7 : ["F8",-500,500,30],
8 : ["F9",-5.12,5.12,30],
9 : ["F10",-32,32,30],
10 : ["F11",-600,600,30] ,
11 : ["F12",-50,50,30],
12 : ["F13",-50,50,30],
13 : ["F14",-65.536,65.536,2],
14 : ["F15",-5,5,2],
15 : ["F16",-5,5,4],
16 : ["F17",-5,15,2],
17 : ["F18",-2,2,2] ,
18 : ["F19",0,1,3],
19 : ["F20",0,1,6],
20 : ["F21",0,10,4],
21 : ["F22",0,10,4],
22 : ["F23",0,10,4],
            }
    return param.get(a, "nothing")
```

REFERRENCES

1. John H (1992) Holland, adaptation in natural and artificial systems. MIT Press, Cambridge.
2. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, pp 1942–1948.
3. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
4. Holland JH. Genetic algorithms. Sci Am 1992; 267:66–72.
5. Liang J, Suganthan P, Deb K. Novel composition test functions for numerical global optimization. In: Proceedings 2005 IEEE swarm intelligence symposium, 2005. SIS 2005; 2005. p. 68–75.
6. Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life; 1991. p. 134–42.
7. The Whale Optimization Algorithm. SeyedaliMirjalili , Andrew Lewis a School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia b Griffith College, Mt Gravatt, Brisbane, QLD 4122, Australia :1744-1765
8. K.D. Frank, C. Rich, T. Longcore, Effects of artificial night lighting on moths, in: Ecological Consequences of Artificial Night Lighting, 2006, pp. 305–344.
9. Griffiths D. Pit construction by ant-lion larvae: a cost-benefit analysis. J Anim, Ecol 1986:39–57.
10. Yang X-S , Deb S . Cuckoo search via Lévy flights. In: Proceedings of the world congress on nature & biologically inspired computing, NaBIC 20 09; 20 09. p. 210–14 .
11. Yang X-S . Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput2010;2:78–84 .
12. Multi-Verse Optimization algorithm: a nature-inspired algorithm for global optimization. SeyedaliMirjalili • Seyed Mohammad Mirjalili •AbdolrezaHatamlou 2015:205-268
13. Yang X-S. A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer; 2010. p.65–74.

14. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings IEEE, international conference on neural networks, 1995; 1995. p. 1942–8.
15. Hybrid Optimization Algorithm of Particle Swarm Optimization and Cuckoo Search for Preventive Maintenance Period Optimization, Jianwen Guo,1 Zhenzhong Sun, Hong Tang, Xuejun Jia, Song Wang, Xiaohui Yan, Guoliang Ye, and Guohong Wu. P. 158-178
16. Xia, Xuewen&Gui, Ling & He, Guoliang& Xie, Chengwang& Wei, Bo & Xing, Ying & Wu, Ruifeng& Tang, Yichao. (2017). A hybrid optimization algorithm based on firefly algorithm and particle swarm optimization algorithm. Journal of Computational Science. 26. 10.1016/j.jocs.2017.07.009.
17. A Hybrid DA-PSO Optimization Algorithm for Multiobjective Optimal Power Flow Problems. Khunkitti, Sirote. 85-91
18. A hybrid SA-MFO algorithm for function optimization and engineering design problems by Sayed, Gehad Ismail and Hassanien, Aboul Ell p. 05-09
19. A New Hybrid Whale Optimization algorithm Algorithm with Mean Strategy of Grey Wolf Optimization algorithm for Global Optimization, Narinder Singh and HanaaHachimi, p. 38-45
20. Grey Wolf Optimization algorithm, SeyedaliMirjalilia,⇑, Seyed Mohammad Mirjalili b, Andrew Lewis, a School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane QLD 4111, Australia, 1755-1799
21. Muro C, Escobedo R, Spector L, Coppinger R. Wolf-pack (Canis lupus) hunting, strategies emerge from simple rules in computational simulations. Behav, Process 2011;88:192–7.
22. Chong, C. S., Sivakumar, A. I., Malcolm Low, Y. H., Gay, K. L. (2006). A bee colony optimization algorithm to job shop scheduling. In Proceedings of the 38th conference on Winter simulation WSC '06, pages 1954-1961, California.95-101
23. A Discrete Artificial Bee Colony for Distributed Permutation Flowshop Scheduling Problem with Total Flow Time Minimization Jia-Qi Pan , Wen-Qiang Zou, Jun-huaDuan
24. Watkins WA ,Schevill WE . Aerial observation of feeding behavior in four baleen whales: Eubalaenaglacialis ,Balaenoptera borealis , Megapteranovaean- gliae , and Balaenopteraphysalus . J Mammal 1979:155–63.

25. Goldbogen JA, Friedlaender AS, CalambokidisJ ,Mckenna MF , Simon M , Nowacek DP . Integrative approaches to the study of baleen whale diving be- havior, feeding performance, and foraging ecology. BioScience2013;63:90–100.
26. Gaston, Kevin J., et al. "The ecological impacts of nighttime light pollution: a mechanistic appraisal." Biological reviews 88.4 (2013): 912-927
27. Scharf I, Ovadia O. Factors influencing site abandonment and site selection in a sit-and-wait predator: a review of pit-building antlion larvae. J Insect Behav, 2006;19:197–218.
28. Kaveh A, Mahdavi V. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014;139:18–27.
29. Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm EvolutComput2011;1:3–18.
30. X.-S. Yang; S. Deb (December 2009). Cuckoo search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214. arXiv:1003.1594v1.
31. Yang, X. S. (2008). Nature-Inspired Metaheuristic Algorithms. Luniver Press. ISBN 1-905986-10-6.
32. X. S. Yang, Metaheuristic optimization: algorithm analysis and open problems, in: Experimental Algorithms (SEA2011), Eds (P. M. Pardalos and S. Rebennack), LNCS 6630, pp.21-32 (2011).
33. Multi-Verse Optimization algorithm: a nature-inspired algorithm for global optimization, SeyedaliMirjalili • Seyed Mohammad Mirjalili • AbdolrezaHatamlou, pp-301-322
34. J. Kennedy and R. Eberhart. Particle swarm optimization. Proceeding of IEEE International Conference on Neural Networks, 1995, 1942-1948.
35. Eberhart, R. C. and Kennedy, J. A new optimization algorithm using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan, 1995.
36. A New Metaheuristic Bat-Inspired Algorithm, Xin-She Yang. Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK, pp. 12-34
37. Yao X, Liu Y, Lin G. Evolutionary programming made faster. EvolutComput, IEEE Trans 1999;3:82–102.

38. Digalakis J, Margaritis K. On benchmarking functions for genetic algorithms. Int J Comput Math 2001;77:481–506.
39. Molga M, Smutnicki C. Test functions for optimization needs. Test functions for optimization needs; 2005. 156-171
40. Yang X-S. Test problems in optimization, arXiv, preprint arXiv:1008.0549; 2010.1056-1088
41. Mirjalili S, Lewis A. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm EvolutComput2013;9:1–14.
42. Mirjalili S, Mirjalili SM, Yang X. Binary bat algorithm. Neural ComputAppl, in press, DOI: 10.1007/s00521-013-1525-5.
43. Holland JH. Genetic algorithms. Sci Am 1992;267:66–72.
44. RechenbergI .Evolutionsstrategien. Springer Berlin Heidelberg; 1978. p. 83–114.
45. Dasgupta D, Zbigniew M, editors. Evolutionary algorithms in engineering ap- plications. Springer Science & Business Media; 2013. Pp 91-133
46. J.R. Koza, "Genetic programming,"1992. Pp 08-31
47. Simon D . Biogeography-based optimization. IEEE Trans EvolComput2008;12:702–13.
48. Kirkpatrick S ,Gelatt CD , Vecchi MP . Optimization by simmulated annealing. Science 1983;220:671–80.
49. Webster B , Bernhard PJ . A local search optimization algorithm based on natural principles of gravitation. In: Proceedings of the 2003 interna- tional conference on information and knowledge engineering (IKE'03); 2003. p. 255–61.
50. Erol OK ,Eksin I . A new optimization method: big bang–big crunch. Adv EngSoftw2006;37:106–11.
51. RashediE ,Nezamabadi-Pour H , Saryazdi S . GSA: a gravitational search algo- rithm. Inf Sci 2009;179:2232–48.
52. Kaveh A ,Talatahari S . A novel heuristic optimization method: charged system search. Acta Mech 2010;213:267–89 .
53. FormatoRA . Central force optimization: A new metaheuristic with applica- tions in applied electromagnetics. Prog Electromag Res 2007;77:425–91.
54. AlatasB . ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. Expert Syst Appl 2011;38:13170–80.

55. HatamlouA . Black hole: a new heuristic optimization approach for data clus- tering. Inf Sci 2013;222:175–84.
56. Kaveh A ,Khayatazad M . A new meta-heuristic method: ray optimization. Comput Struct 2012;112:283–94.
57. Du H , Wu X , Zhuang J . Small-world optimization algorithm for function opti- mization. Advances in natural computation. Springer; 2006. p. 264–73.
58. Shah-Hosseini H . Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int J Comput Sci Eng2011;6:132–40.
59. Moghaddam FF, Moghaddam RF, Cheriet M. Curved space optimization: A random search based on general relativity theory. 2012. arXiv: 1208.2214.
60. A hybrid algorithm for feature subset selection in high-dimensional datasets using FICA and IWSSr algorithm by Mostafa Moradkhania, Ali Amiria, Mohsen Javaherian, Hossein Safarib.pp. 56-61
61. V. N. Rajput, K. S. Pandya and K. Joshi, "Optimal coordination of Directional Overcurrent Relays using hybrid CSA-FFA method," 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Hua Hin, 2015, pp. 1-6.
62. H. Arslan and M. Toz, "Hybrid FCM-WOA data clustering algorithm," 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, 2018, pp. 1-4.
63. N. B. Arunekumar, A. Kumar and K. S. Joseph, "Hybrid bat inspired algorithm for multiprocessor real-time scheduling preparation," 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, 2016, pp. 2194-2198.
64. A. Sarma, A. Bhutani and L. Goel, "Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality," 2017 Intelligent Systems Conference (IntelliSys), London, 2017, pp. 52-60.
65. Grey Wolf Optimization algorithm by SeyedaliMirjalili, Seyed Mohammad Mirjalili, Andrew Lewis. pp 2103-2205
66. T. D. Seeley, Visscher P.K., “Assessing the benefits of cooperation in honeybee foraging: search costs, forage quality, and competitive ability”, Behav. Ecol.
Sociobiol., 22: 229-237, 1988.

67. Hof PR , Van Der Gucht E . Structure of the cerebral cortex of the humpback whale, Megaptera novaeangliae (Cetacea, Mysticeti, Balaenopteridae). Anat Rec 2007;290:1–31 .
68. Payne R. B., Sorenson M. D., and Klitz K., The Cuckoos, Oxford University Press, (2005)
69. Passino, K. M.: Biomimicrt of Bacterial Foraging for Distributed Optimization, University Press, Princeton, New Jersey (2001). Pp 134-138
70. Khoury J, Ovrut BA, Seiberg N, Steinhardt PJ, Turok N (2002), From big crunch to big bang. Phys Rev D 65:086007
71. Tegmark M (2004) Parallel universes. In: Barrow JD, Davies, PCW, Harper CL Jr (eds) Science and ultimate reality: Quantum theory, cosmology, and complexity. Cambridge University Press, pp 459–491
72. Eardley DM (1974) Death of white holes in the early Universe. Phys Rev Lett 33:442
73. Steinhardt PJ, Turok N (2002) A cyclic model of the universe. Science 296:1436–1439
74. Davies PC (1978) Thermodynamics of black holes. Rep Prog Phys 41:1313
75. Morris MS, Thorne KS (1988) Wormholes in spacetime and their use for interstellar travel: a tool for teaching general relativity. Am J Phys 56:395–412
76. Guth AH (2007) Eternal inflation and its implications. J Phys A, Math Theor 40:6811
77. Steinhardt PJ, Turok N (2005) The cyclic model simplified. New, Astron Rev 49:43–57
78. Altringham, J. D.: Bats: Biology and Behaviour, Oxford Univesity Press, (1996).21-35
79. Colin, T.: The Varienty of Life. Oxford University Press, (2000). Pp405-452
80. Richardson, P.: Bats. Natural History Museum, London, (2008).pp.15-23
81. Richardson, P.: The secrete life of bats. http://www.nhm.ac.uk pp-08-16
82. Kennedy, J. and Eberhart, R.: Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks. Perth, Australia, 1942-1945 (1995). Pp -53-59
83. Cavazos, John & Moss, Eliot & F. P. O'Boyle, Michael. (2006). Hybrid Optimizations: Which Optimization Algorithm to Use?. Pages 34-43