Performance Analysis of Bio-inspired Optimization Algorithms and their applications to Data Clustering

A thesis

submitted in partial fulfillment of the requirement for the Degree of

Master of Technology in Computer Technology of

Jadavpur University

By

Utshab Saha

Registration No.: 137111 of 2016-2017

Examination Roll No.: M6TCT19014

Under the Guidance of

Prof. Susmita Ghosh

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2019

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that the dissertation entitled "Performance Analysis of Bio-inspired Optimization Algorithms and their applications to Data Clustering" has been carried out by Utshab Saha (University Registration No.: 137111 of 2016-17, Examination Roll No.: M6TCT19014) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Technology in Computer Technology. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

……

Dr. Susmita Ghosh (Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata-32

Countersigned

……

Prof. Mahantapas Kundu

Head, Department of Computer Science and Engineering Jadavpur

University, Kolkata-32.

……

Prof. Chiranjib Bhattacharjee

Dean, Faculty of Engineering and Technology Jadavpur

University, Kolkata-32.

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Certificate of Approval

This is to certify that the thesis entitled "Performance Analysis of Bio-inspired Optimization Algorithms and their applications to Data Clustering" is a bona-fide record of work carried out by Utshab Saha in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Technology in the Department of Computer Science & Engineering, Jadavpur University during the period of June 2018 to May 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

……..

Signature of Examiner 1 Date:

……..

Signature of Examiner 2 Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "Performance Analysis of Bio-inspired Optimization Algorithms and their applications to Data Clustering" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Technology in Computer Technology.

All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Utshab Saha

Registration No: 137111 of 2016-2017

Exam Roll No.: M6TCT19014

Thesis Title: Performance Analysis of Bio-inspired Optimization Algorithms and their applications to Data Clustering

…..……..

Signature with Date

Acknowledgement

I would like to start by thanking the holy trinity for helping me deploy all the right resources and for shaping me into a better human being.

I would like to express my deepest gratitude to my advisor, **Dr. Susmita Ghosh**, Department of Computer Science and Engineering, Jadavpur University for her admirable guidance, care, patience and for providing me with an excellent atmosphere for doing research. Our numerous scientific discussions and her many constructive comments have greatly improved this work.

I am immensely grateful to my lab mates Mr. Sayantan Maity, and Miss Payel Pramanik who were always present to motivate, guide me during the result analysis process. Without all of them it would surely be a very lonely lab. I sincerely thank my seniors Miss Tithi Bhakta and Mr. Ranajoy Banerjee for their support in completion of the thesis documentation.

Most importantly none of this would have been possible without the love and support of my family. I extend my thanks to my parents, especially to my father whose forbearance and whole hearted support helped this endeavor succeed. This thesis would not have been completed without the inspiration and support of a number of wonderful individuals — my thanks and appreciation to all of them for being part of this journey and making this thesis possible.

……..

Utshab Saha

Exam Roll No.: M6TCT19014

Registration No: 137111 of 2016-2017

Department of Computer Science & Engineering

Jadavpur University

Abstract

In the present work an investigation has been carried out to study four different bio-inspired optimization algorithms namely, *Bat Algorithm, Bee Colony Optimization, Cuckoo Search* and *Firefly Algorithm*. Experiment has been carried out to study the performance of these algorithms for solving various function optimizations. Nine different benchmark functions are considered. Comparison has been made on the performance of these algorithms. Effectiveness of *Cuckoo Search Algorithm* for clustering datasets is also shown.

Date :

Place :

Utshab Saha

Exam Roll No.: M6TCT19014

Registration No: 137111 of 2016-17

Department of Computer Science & Engineering.

Jadavpur University.

1. Introduction

1.1. Bio-inspired computing

Bio-inspired algorithms are the subset of the *natural* computing. These are inspired by the social behavior of natural creatures. They are mainly based on the mechanism of the brain of living creatures and mimicking their cognitive ability for betterment of the population in terms of finding solution generation after generation.

1.2. Relation between Bio-inspired Computing & Optimization

Optimization is a mathematical process, where we evaluate some function and find an optimum value (*maximum or minimum*) using some complex mathematical calculation. Real-world optimization problems are often challenging to solve, and many applications have to deal with NP-hard problems. To solve such problems, optimization tools have to be used, though there is no guarantee that the optimal solution can be obtained. In fact, for NP-problems, there are no efficient algorithms at all. As a result, many problems have to be solved by trial and errors using various optimization techniques. In addition, new algorithms have been developed to see if they can cope with these challenging optimization problems. Among these new algorithms, many algorithms such as particle swarm optimization, cuckoo search, firefly algorithm etc, have gained popularity due to their high efficiency of searching strategy of the solution space and reducing the time complexity [1].

1.2.1. Inspiration

Nature has inspired many researchers in many ways and thus is a rich source of inspiration. Nowadays, most new algorithms are nature-inspired, because they have been developed by drawing inspiration from nature. Even with the emphasis on the source of inspiration, we can still have different levels of classifications, depending on how details and how many sub-sources we would wish to use. For simplicity, we would use the highest level sources such as biology, physics or chemistry. In the most generic term, the main source of inspiration is Nature. By far the majority of nature-inspired algorithms are based on some successful characteristics of biological system. Therefore, the largest fraction of nature-inspired algorithms are biology-inspired, or bio-inspired for short. Among bio-inspired algorithms, a special class of algorithms have been developed by drawing inspiration from swarm intelligence and swarm-intelligence based [1].

1.3. Classification of bio-inspired computing

It is really a challenging task to classify these algorithms systematically. Obviously, the classifications can largely depend on the criteria, and there is no easy guideline in the literature to set out the criteria.

Major tow classes are *swarm intelligence (SI)* and *bio-inspired (but not SI-based).* For example, if the focus and perspective are about the trajectory of the search path, algorithms can be classified as trajectory-based and population-based. Simulated annealing is a good example of trajectory-based algorithm, while particle swarm optimization and firefly algorithms are population-based algorithms. If our emphasis is placed on the interaction of the multiple agents, algorithms can be classified as attraction-based or non-attraction-based. Firefly algorithm (FA) is a good example of attraction-based algorithm because FA uses the attraction of light and attractiveness of fireflies, while genetic algorithms are non-attraction-based since there is no explicit attraction used. On the other hand, if the emphasis is placed on updating equations of population movement then algorithms can be divided into rule-based and equation-based. For example, particle swarm optimization and cuckoo search are equation-based algorithms because both use explicit updating equations, while genetic algorithms do not have explicit equations for crossover and mutation. However, in this case, the classifications are not unique. For example, firefly algorithm uses three explicit rules and these three rules can be converted explicitly into a single updating equation which is nonlinear. Hence it is understood that classification depends on actual perspective and motivation [1].

1.3.1. Swarm intelligence based

Swarm intelligence (SI) deals with the collective emerging behavior of multiple interacting agents who follow simple rules. Some agents maybe unintelligent, but the whole system of multiple intelligence show some self-organization behavior.

All the SI-based algorithms are inspired by the social insects, like ants, bees, wasps and termites, as well as from other animal societies like flocks of birds or fish. The most common *particle swarm optimization (PSO)* imitates the swarming behavior of fish and birds whereas *firefly (FA)* uses the flashing behavior of the fireflies, *cuckoo search (*CS*)* imitates brooding parasitism of cuckoo species and *bat algorithm (BA)* is inspired by the *echolocation foraging. Ant colony optimization (ACO)* uses the interaction of social insects (e.g., ants), while the class of *bee algorithms* are all based on the foraging behavior of honey bees.

SI-based algorithms are among the most popular and widely used. There are many reasons for such popularity, one of the reasons is that SI-based algorithms usually share information among multiple agents, so that self-organization, co-evolution and learning during iterations may help to provide the high efficiency. Another reason is that multiple agents can be parallelized easily so that large-scale optimization becomes more practical from the implementation point of view [1].

1.3.2. Bio-inspired, but not Swarm Intelligence (SI)-based

SI-based algorithms belong to a wider class of algorithms, called bio-inspired algorithms. In fact, bio-inspired algorithms form a majority of all nature-inspired algorithms.

$$\text{SI-based} \subset \text{bio-inspired} \subset \text{nature-inspired}.$$

Conversely, not all nature-inspired algorithms are bio-inspired, and some are purely physics and chemistry based algorithms. Some algorithms are there which are bio-inspired but they do not use swarming behavior that is why they are called bio-inspired but not SI-based. For example, genetic algorithms are bio-inspired, but not SI-based. However, it is not easy to classify certain algorithms such as differential evolution (DE). DE is not bio-inspired because there is no direct link to any biological behavior. However, as it has some similarity to genetic algorithms and also has a key word 'evolution', we tentatively put it in the category of bio-inspired algorithms [1].

1.4. Pattern Recognition

Pattern can be described as the persisting characteristic or trait that helps in the identification of a phenomenon and serves as an indicator or model for predicting its future behavior. A pattern can be anything; it can be a fingerprint image, handwritten letters or human face that could be given a name. Recognition is a way that finds the association between the patterns [2].

Hence it is necessary to recognize the pattern in the data with fair enough accuracy for the improvement of the result for the task which is specified by user or the analysis which user wants to do. Pattern Recognition is the science (mainly in machine learning) of making a presumption based on data [2]. The main goal in Pattern Recognition is to assign an object or a pattern to a single or multiple categories or classes based on features derived to accentuate commonalities.

Pattern Recognition involves three types of learning:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning

Supervised learning as name concerns a presence of supervisor as teacher. Basically supervised learning is a learning in which we teach or train a machine with data which is well labeled (that means some data is already tagged with correct answer). After that, machine is provided with new set of data so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data. For instance, suppose we are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this:

- If shape of object is rounded and depression at top having color Red then it will be labelled as –**Apple**.
- If shape of object is long curving cylinder having color Green-Yellow then it will be labelled as –**Banana**.

Now suppose after training the data, you have given a new separate fruit say Banana from basket and asked to identify it. Since machine has already learnt the things from previous data and this time have to use it wisely. It will first classify the fruit with its shape and color, and would confirm the fruit name as BANANA and put it in Banana category. Thus machine learns the things from training data (basket containing fruits) and then apply the knowledge to test data (new fruit).

Unsupervised learning is the training of a machine using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Grouping together or clustering (the process of grouping a set of objects into classes of similar objects) is done by clustering algorithms [3].

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore machine is restricted to find the hidden structure in unlabeled data by it-self. In classification, the task is to approximate a mapping function (f) from input variables (x *d-dimensional vectors*) to discrete output variables (y).The output variables are often called labels or categories. The mapping function predicts the class or category for a given observation. The key idea of ensemble learning is to learn a set of classifiers and to give them priorities. It helps to improve the predictive accuracy.

Semi supervised learning, deals with methods for exploiting unlabeled data and labeled data automatically for improving the learning performance without human intervention. There are four types of semi supervised learning. These are, deep learning [4], low density separation [5], graph based methods [6] and heuristic approach [7].

Deep Learning is a newer field of research that uses a smaller set of principles in a more complex neural network that yields results which can blur the lines between artificial and human thought. Deep learning has ability to analyze new data and continuously learn new things.

In Low density separation, the assumption is the final cluster boundary should not cross high density regions, but instead lie in low density regions.

In Graph based methods, the key idea is to construct a graph with edges between very similar data. Unlabeled data can help find the objects of the same class together.

In Heuristic approach, it starts by heavily penalizing solutions with points falling within the margin and then relaxes this requirement in order to find solutions with wider margin where margin is the sum of distances from the closest data points.

1.5. Scope of the thesis

An investigation among various bio-inspired algorithms in the field of optimization has been made in this work. This work asses the statistical results obtained using four different

algorithms and analyze the efficiency of each algorithm. Also *Cuckoo Search Algorithm* is experimented in data clustering and a study and analysis has been done.

1.6. Organization of thesis

This thesis is segmented in six sections. First section covers the introduction and some general information about the thesis work. Second section covers the survey on the bio-inspired algorithms addressed in this thesis with their detailed over view and clarification. Third section describes about the clustering some conventional approaches and how bio-inspired algorithms helps in clustering along with that how *Cuckoo Search* is incorporated in data clustering is also described. Fourth section includes the parameter details of individual algorithms, optimization functions used in the experiments also the datasets used in the experiments. Fifth section consist of experimental results and analysis. Sixth sections concludes with the limitations and future scope of the thesis.

2. Survey on bio-inspired computing techniques

We have always received beautiful gifts from nature. In computer science also nature did not disappoint us. We are inspired by the biological phenomenon of the animals in the nature. Optimization has been encountered in mathematical problems of all engineering and various scientific fields. Due to the wide range of problems, optimization has always been an active research topic of all time. Many optimization algorithms have been computationally inefficient hence they were tend to fail due to the problem size. In this case nature has offered us the biological intelligence of natural animals, which often optimizes the real problems in efficient ways. Bio-inspired optimization algorithms are Meta-heuristic approach which are based upon iterative improvements, randomizations and local search to the given problem [8]. There are several bio-inspired algorithm listed in a hierarchical tree [8] in *Fig 2.1*. This thesis addresses the study of *Swarm Based* four algorithms namely *Bat Algorithm, Bee Colony Optimization, Cuckoo Search* and *Firefly Algorithm* which show a convergent social phenomenon.

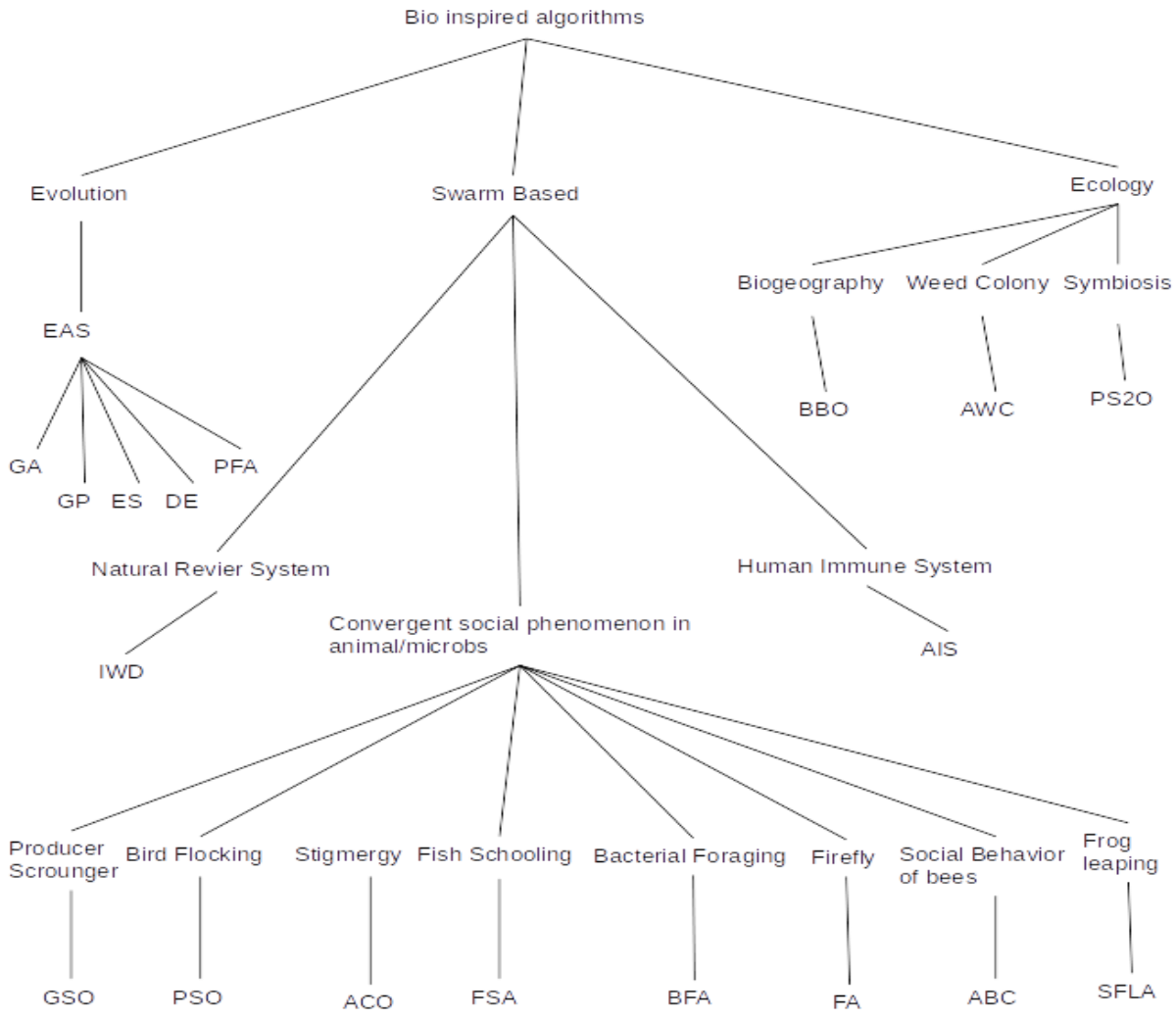

Fig 2.1: Taxonomy and nomenclature of various bio inspired optimization algorithms grouped by the area of inspiration

2.1. Bat Algorithm

Bat Algorithm (BA) is a meta-heuristics algorithm based on echolocation while bats are hunting the prey. Meta-heuristics has the ability to converge not being trapped in local optima. Bats use echolocation to fly, avoid obstacles and hunt for the prey. They create loud sound pulse, emits it and listen to its echo which helps them to identify their surroundings, any obstacles in their way or the prey they are hunting for. They can adjust the loudness, frequency of these sound waves. Using this they search for their prey, they also identify the moving objects, even the velocity of the moving objects. Using this technique they adjust their own velocity as well to hunt down the prey.

Imitating this *echolocation* technique by the virtual bats optimization is formulated in association with an objective function. This objective function value corresponds to the distance between the bat and the prey [9].

2.1.1. Algorithm

Bat Algorithm has various number of parameters which are initialized randomly. Right after that manipulating those parameters virtual bats search for the most obvious solution according to the fitness/objective function. Here manipulation means the movement of the bats. The significant parameters used by the bats are *frequency, velocity, loudness.* Each bats fly with random velocity *v* at *x* position with a frequency *f* and a variable loudness *A* to search for a prey. To do that each bat emits the sound and automatically adjust their frequency depending upon the proximity of the prey and hence adjust the velocity.

- **Movement of the bat**

The movement of virtual bats are formulated as following

$$f_i = f_{min} + (f_{max} - f_{min})\beta \qquad (1)$$

$$v_i^t = v_i^{t-1} + [x_i^t - x_*]f_i \qquad (2)$$

$$x_i^t = x_i^{t-1} + v_i^t \qquad (3)$$

above formula describes the primary movement of the virtual bats in the search space of the problem, which they are looking for the global optima i.e. the best prey or the best bats location. Now the $\boldsymbol{f}$ in above equation is the pulse frequency, $\boldsymbol{x}$ and $\boldsymbol{v}$ are $\boldsymbol{d}$-dimensional vectors, refereeing to the virtual bats location and the velocity. Now $\boldsymbol{f}$ is taken from a range of $[\boldsymbol{f_{min}}, \boldsymbol{f_{max}}]$ and other two are initialized randomly at first and $\boldsymbol{\beta} \in [0, 1]$ taken randomly from a uniform distribution. Then the algorithm evaluate value of $\boldsymbol{x}$ the present position and verifies it by the objective function. Then with respect to the other parameters it updates the $\boldsymbol{x}$. Hence it is obvious when they find the better value for $\boldsymbol{x}$ they retain it until search process is done.

Now when virtual bats are searching for the global optima, each bat also does a local search using the *random walk* method and it is formulated as follows

$$x_{new} = x_{old} + \epsilon A^t \qquad (4)$$

where $\epsilon \in [-1, 1]$ is a random number and $\boldsymbol{A^t}$ is the average loudness at this time stamp. Now likewise the real bats, virtual bats also needs to know that they are approaching to their prey or the better solutions. To do that virtual bats also need to update their pulse rate and loudness accordingly. They are formulated as follows

$$A_i^{t+1} = \alpha A_i^t \qquad (5)$$

$$r_i^{t+1} = r_i^0[1 - e^{-\gamma t}] \qquad (6)$$

where α and γ are constants ***r*** is the pulse emission rate [9].

The following is the pseudo code [10].

- **Pseudo code**

Objective function f(x), $x = (x_1, \ldots\ldots, x_d)^T$
Initialize the bat population x_i $(i=1,2,\ldots.,n)$ and v_i
Define pulse frequency f_i at x_i
Define pulse rate r_i and loudness A_i
while *($t <$ Max number of iteration)*
    Generate new solution by adjusting frequency,
    and updating velocity and location/solution [equations (2) to (4)]
    if *($rand > r_i$)*
        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
    Rank the bats and find the current best $\boldsymbol{x_*}$
end while
Post process result and visualization

Fig 2.2: Pseudo code of Bat Algorithm

2.1.2. Applications

BA is being largely used to solve various problems such as engineering problems, parameter estimation, feature selection, inverse problems, vehicle routing, numerical problems etc. In addition to that variations of BA is also being developed such as Multi-Objective BA, Binary BA, Fuzzy Logic BA [9] etc.

2.2 Bee Colony Optimization Algorithm

Many species due to *biological needs*, stay together. This is identifies as *swarm behavior*. Communication among the individual insects are inspiring. The communication system among these insects contributes to the *swarm intelligence* which denote to the *collective intelligence* and a part of *Artificial Intelligence*. This is a *Multi Agent* system, which consist of physical individual or virtual individuals who communicate among themselves, exchange information and perform a task [4].

Lučić and Teodorović discovered the metaheuristic approach *Bee Colony Optimization* as a new direction to swarm intelligence. This is a bottom-up approach, where artificial bees collaboratively solve complex combinatorial optimization problems [11].

2.2.1. Algorithm

The BCO algorithm was inspired by the social behavior of the natural bees. The goal was to build a multi agent system which can solve some combinatorial optimization problem imitating the bee's behavior. The agents or virtual bees behaves somewhat alike and partially different from the real ones. The natural bee's behavior cab be better understood by the example of nectar collection.

The bees in the hive follow a nest mate who have found a patch of flowers. Then it follows that bee to the patch and then it collects the nectar and return to the hive. Then it relinquish the collected nectar to the food keeper bee. Right then that foraging bee can have three options to choose,

1. Abandon the source and become uncommitted follower.
2. Try to convince other nest mates by dancing on the dance floor of the hive.
3. Continue to foraging without recruiting other follower.

In above process bees advertise the different food source they visit on the dancefloor. How other bees choose, which bee to follow is not clear much, but it is seemed to be depending on some probability which is a function of the quality or quantity of the nectar [11].

Now coming to the algorithm BCO is a population based metaheuristic algorithm, where *virtual bees* searches for the optimal solution. Each bee generates a solution to the problem. The algorithm consists of two main phase i.e. *forward pass* and *backward pass*.

In *forward pass* each bee searches the solution space and find a solution and apply some predefined move and construct or improve a solution of generated a new solution. Then they go back to the hive and starts second phase i.e. *backward pass*.

Now in case of *backward pass* natural bees do the dancing to share the information about the food source they have found. Then the nest mates choose if follow the dancer bee or not. Now in case of virtual bees this selection process depends on the probability, the more the quality of the source better the chance of getting selected. Here the quality stands for the value of the objective function. Now this two pass continues iteratively until it meets terminating condition or the best solution [11].

In BCO algorithm there are mainly two types of bees i.e. *employee* and *follower* bee. Each employee bee searches for the local solutions around the solution it has found. This process is formulated as equation (7). Then it shares the information along with the follower bees in the hive and they choose which employee to be followed. In here it is done by roulette wheel selection which enables follower bees to choose randomly yet with better chance of optimality [12].

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \qquad (7)$$

Here $i \in (1, \dots, n)$ where n is number of population and $j \in (1, \dots, d)$ where d is *d-dimensional vector* and k is the number of employee bee and k is chosen randomly and it has to be different from i. v is the local solution. Φ is the random variable which lies between the range [-1, 1]. The following is the pseudo code [11].

- **Pseudo code**

1. Initialization: every bee is set to an empty solution;
2. For every bee do the forward pass:
a) Set k = 1; //counter for constructive moves in the forward pass;
b) Evaluate all possible constructive moves;
c) According to evaluation, choose one move using the roulette wheel;
d) k = k + 1; If k ≤ NC Go To step b. // NC is the number of constructive steps
3. All bees are back to the hive; // backward pass starts;
4. Sort the bees by their objective function value;
5. Every bee decides randomly whether to continue its own exploration and become a recruiter, or to become a follower (bees with higher objective function value have greater chance to continue its own exploration);
6. For every follower, choose a new solution from recruiters by the roulette wheel;
7. If the stopping condition is not met Go To step 2;
8. Output the best result.

Fig 2.3: Pseudo code Bee Colony Optimization

2.2.2. Applications

There are several domain where BCO has been used such as, Traveling Salesman Problem, Ride-Matching Problem, Routing and Wavelength Assignment in All-Optical Networks Scheduling Independent Tasks [11].

2.3. Cuckoo Search Algorithm

Cuckoo Search Algorithm (CS) is another bio-inspired algorithm which confidently helps to solve the optimization problems. Cuckoo being a bird with sweet voice it has an amazing technique of breeding which inspired computer scientists to formulate and imitate their technique in computer science. So down in the next section natural breeding technique of cuckoo is discussed.

2.3.1. Cuckoo Breeding Technique

It is noticed that cuckoos have an aggressive clever reproduction strategy. They lay their eggs to some other bird's nest also they sometimes remove host bird's eggs to increase the hatching possibility of their own eggs. There are three basic type of brood parasitism: instar-specific brood parasitism, cooperative brooding and nest takeover. Some hosts can engage to direct conflicts as well. If the host bird can identify the intruder egg they either throw the egg away or abandon the nest [14].

Some cuckoo species such as *Tapera* has evolved in such a way that female cuckoo can change the color of their egg similar to the host bird's egg at the time of laying egg. Apart from that cuckoos also careful about the time of laying eggs. They choose to lay eggs when their chosen host bird has just laid their eggs. Cuckoo's eggs hatching time is usually a little earlier than the host bird, so after hatching of a cuckoo egg the cuckoo chick's first instinct is to propel the host eggs out of the nest. This increase the probability of getting more food share from the host bird. Also cuckoo chick can mimic the call of host bird which increases of getting more food share [14].

2.3.2. Lèvy Flights

Lèvy Flights is an important phenomenon in this algorithm which helped to formulate the flight patterns of cuckoo. It has been noticed that many animals and insects shows the characteristics of Lèvy Flights while searching their search space [14]. Hence it has been considered as an appropriate method for formulating the flight pattern during the optimization.

2.3.3. Algorithm

Coming to the algorithm, three idealizations can be used. 1) Each cuckoo lays one egg at a time. 2) Best nest with high quality egg will go over the next generation. 3) The number of host nest is fixed and the probability of egg laid by cuckoo being discovered by host bird is $p_a \in [0, 1]$. Now for simplicity of last assumption can be approximated such way that fraction p_a of n nests will be replaced by new nests i.e. randomly generated new solutions. So each nests are solutions in the searching process and the fitness of a solution is proportional to the value of objective function [14].

Each nests are considered as the solutions of search space where cuckoo lays egg. Now for simplicity the host nest and egg refers to the same solution. Randomly a cuckoo is chosen and searches for a solution by *Lèvy Flight*. Quality of it is evaluated and replaces the previous solution of that cuckoo if quality is better. This movement is formulated in equation (8). It is repeated for all the nests. Then after all this process a fraction of p_a nests are abandoned and new solutions are generated.

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \,.\, \text{Lèvy}(\lambda) \qquad (8)$$

Where $\alpha > 0$ is the step size which is dependent to the problem. In most cases $\alpha = 1$.

Now coming to the *Lèvy* it is described by following equation,

$$\text{Lèvy} \sim u = t^{-\lambda} \qquad (9)$$

where $\lambda \in [1, 3]$. *Lèvy* provides a random walk where step length is draw from *Lèvy* distribution which has infinite variance and mean. This helps in searching a solution around the best one and this speeds up this local search process [14]. The following is the pseudo code [15].

- **Pseudo code**

Begin
Objective function f(x), $x = (x_1, x_2 \ldots\ldots x_d)^T$
Generate initial population of n host nests x_i (i = 1, 2 …… n)
while*(t < MaxGeneration) or (Stopping criterion)*
Get a cuckoo randomly by L´evy flights evaluate its quality/fitness F_i
Choose a nest among n (say, j) randomly
if *($F_i > F_j$)*
replace j by the new solution;
end
A fraction (pa) of worse nests are abandoned and new ones are built;
Keep the best solutions (or nests with quality solutions);
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

Fig 2.4: Pseudo code of Cuckoo Search

2.3.4. Applications

Cuckoo search has been used for various task such as finding optimal features, optimizing parameters of various classifiers including SVM, Neural Network, RBF. It has been used for finding optimizing cluster centroids, job scheduling and many more including industry, wireless sensor networks, health sector etc [16].

2.4. Firefly Algorithm

Another meta-heuristic nature inspired algorithm is *firefly algorithm*. It is motivated by the flashing patterns of the fireflies. There are almost two thousand species of firefly which shows short and rhythmic flashes in sky [17]. Fireflies use these flash patterns on several purpose such as attracting the potential mate, to attract prey and for protective warning as well. Due to this flashing pattern two firefly of opposite sex gets attracted to each other or there are some species where female firefly can mimic the pattern of different species to lure and eat the other species male who mistakes the flashes as a suitable mate [17].

2.4.1. Algorithm

Now due to formulation of the algorithm three idealizations [17, 18] are considered as follows,

- Fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex.
- The attractiveness is proportional to the brightness, and they both decrease as their distance increases. Thus for any two flashing fireflies, the less bright one will move towards the brighter one. If there is no brighter one than a particular firefly, it will move randomly.
- The brightness of a firefly is determined by the landscape of the objective function.

Brightness of firefly is associated with the objective function. Suppose x_i is the solution of firefly i then brightness of that particular solution or the fitness can represented by equation (10) [21].

$$I_i = f(x_i) \qquad (10)$$

Less attractive firefly is attracted and moved to the brighter one, each firefly has its own attractiveness value which is depends on the distance between two firefly. Attractiveness is formulated as follows,

$$\beta(r) = \beta_0 e^{-\gamma r^2} \qquad (11)$$

where r is the distance between firefly i and j represented as equation (12) [19] and γ is the light absorption coefficient of the medium. β_0 is the attractiveness at $r = 0$ [18, 19].

$$r_{ij} = \left|\left| x_i - x_j \right|\right| = \sqrt{\sum_{k=1}^{D} (x_{i,k} - x_{j,k})^2} \qquad (12)$$

Where $x_{i,k}$ is the k^{th} component of i^{th} firefly. D represents each solution is D-dimensional vector.

The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined by equation (13) [10].

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} \left(x_j^t - x_i^t\right) + \alpha_t \epsilon_i^t \quad (13)$$

Where the second term is due to the attraction. The third term is randomization with α_t being the randomization parameter, and ϵ_i^t is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at time t. If $\beta 0 = 0$, it becomes a simple random walk. On the other hand, if $\gamma = 0$, it reduces to a variant of particle swarm optimization [22]. The following is the pseudo code [20].

- **Pseudo Code**

1: Initialize the fireflies;
Calculate the fitness of the fireflies and save the fitness in Light;
2: repeat
3: Flash to attract phase
4: For any two fireflies(x_i , x_j)
5: If Light(x_i) less than Light(x_j)
x_i fly towards x_j according to the expression(2);
Calculate the fitness of the firefly x_i and update Light(x_j);
End if
End for
6: Memorizes the best result found so far;
7: until Termination Condition

Fig 2.5: Pseudo code of FireFly

2.4.2. Applications

Firefly algorithm has plenty of applications [23, 24, 25, 26, 27, 28, 29]. Horng demonstrated that firefly-based algorithm used least computation time for digital image compression [26, 27], while Banati and Bajaj used firefly algorithm for feature selection and showed that firefly algorithm produced consistent and better performance in terms of time and optimality than other algorithms [30]. In the engineering design problems, Gandomi [24] and Azad and Azad [31] confirmed that firefly algorithm can efficiently solve highly nonlinear, multimodal design problems. Basu and Mahanti [32] as well as Chatterjee et al. have applied FA in antenna design optimization and showed that FA can outperform artificial bee colony (ABC) algorithm [24].

3. Clustering

Clustering [33] is a way of grouping together of data samples that are similar in some way with some specific criteria i.e. organizing data into cluster such that there is high intra-cluster similarity and low inter-cluster similarity. It is a form of unsupervised learning technique, as we generally do not have examples demonstrating how the data should be grouped together. So, it is a method of data exploration, a way of looking for patterns or structure in the data that are of interest.

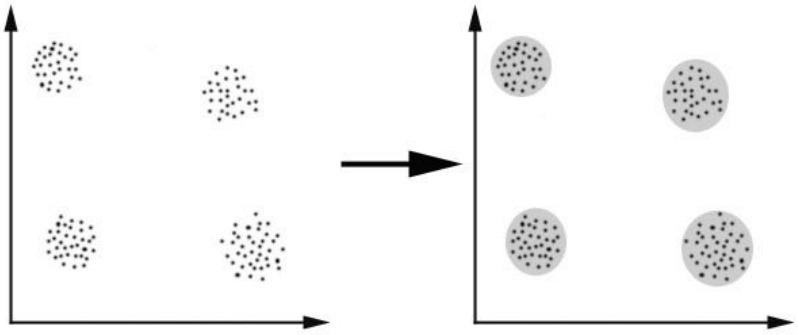

Fig 3.1: Example of exploring and grouping of data

Following is an unsupervised grouping of the patterns i.e. this method has no prior knowledge or information about the class label of the data.

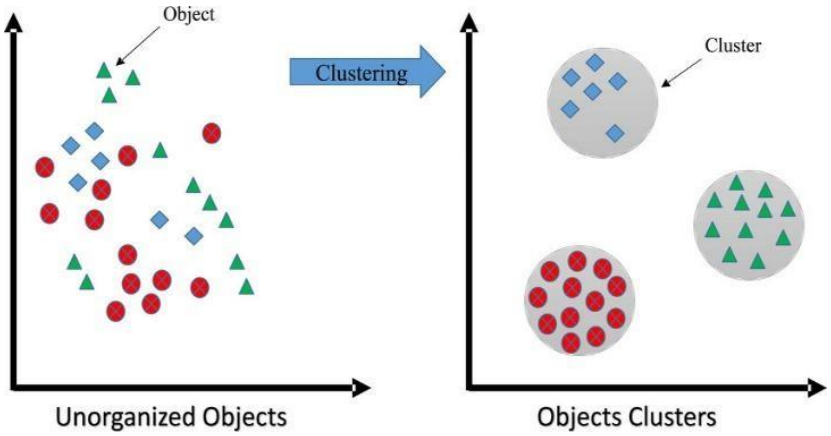

Fig 3.2: Clustering of data

There are several clustering algorithms for performing clustering [34]. These can be categorized into

- Partitioning methods
- Hierarchical methods
- Density based methods
- Grid-based methods

In partitioning method, our objective is partitioning of a dataset *D* of *n* objects into a set of *k* clusters such that each cluster contains at least one object and each object belongs to exactly one

cluster. Examples are *k*-Means developed by MacQueen on 1967 [35], where each cluster is represented by the center of the cluster (centroid) and *k*-medoids (developed by Kaufman & Rousseeuw on 1987) [36], each cluster is represented by one of the objects in the cluster (medoid).

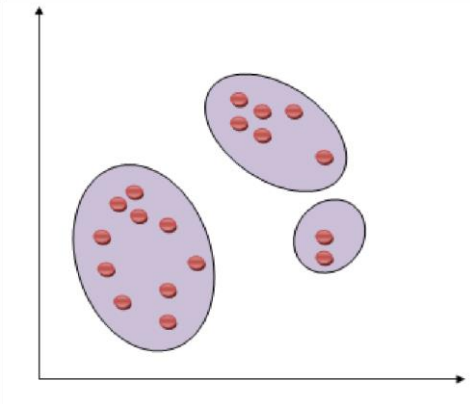

Fig 3.3: Partitioning based Clustering

In hierarchical method construction of a hierarchy of clustering is not just a single partition of objects. Here the number of clusters *k* is not required as an input. It uses a distance matrix as clustering criteria and a termination condition can be used (e.g. A number of clusters). There are two types of hierarchical clustering techniques agglomerative (bottom-up) and divisive (top-down).

- **Agglomerative:** It is a bottom-up approach where clustering starts considering a single data as a cluster. Two clusters are merged when they move up the hierarchy.
- **Divisive:** It is a to-down approach where all the data are considered to be in single cluster. When they move down the hierarchy they are splited.

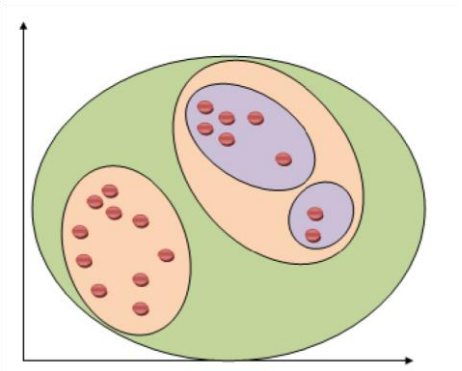

Fig 3.4: Hierarchical based Clustering

In Density based methods clustering is done based on connectivity, density functions and boundary. Here a cluster is defined as a connected dense component which can grow in any direction that density leads. It generates good scalable arbitrarily shaped clusters. The main advantage is that it can handle noise. There are two major types of density based clustering algorithms, Connectivity based: DBSCAN [37], GDBSCAN [38], OPTICS and DBCLASS. Density function based: DENCLUE(developed by Hinneburg & Keim).

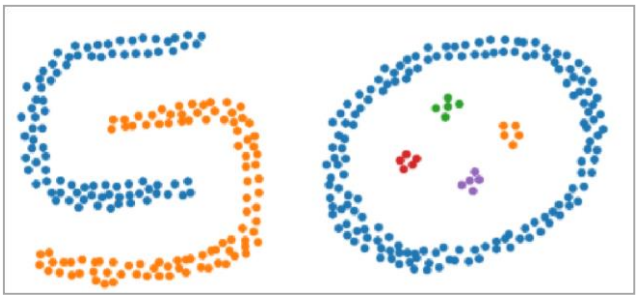

Fig 3.5: Density based Clustering

In Grid-based methods multi-resolution grid data structure is used. Here clustering complexity depends on the number of populated grid cells and not on the number of objects in the dataset. There are several methods in Grid-based method. These are STING (Statistical Information Grid), developed by Wang, Yang and Muntz, WaveCluster developed by Sheikholeslami, Chatterjee, and Zhang and A multi resolution clustering approach using wavelet methodCLIQUE, developed by Agrawal.

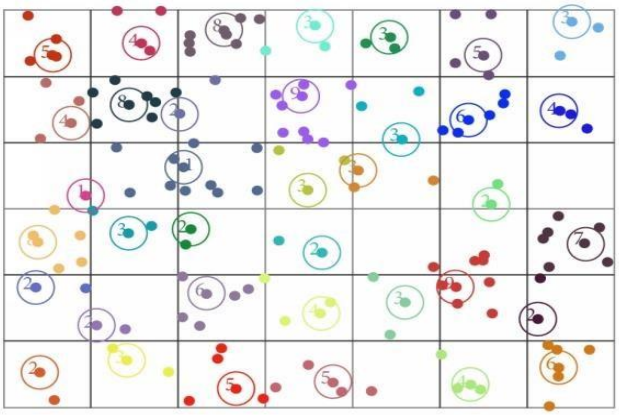

Fig 3.6: Grid based Clustering

Disadvantages of the above stated algorithms are that they are largely heuristic and not based on formal models i.e. formal inference is not possible there. In Model based methods developed by Wolfe [39], sample observations arise from the distribution that is a mixture of two or more components. Each component is depicted with a density function and has an associated probability or weight in the mixture.

Application of clustering: Clustering [40] is useful in machine learning, data mining, as it places similar data objects in a similar group which in turn aids in classification, pattern finding, hypothesis generation and testing. Also used in data summarization, compression, reduction (in image processing and vector quantization), detection of dynamic trend (clustering steaming data and detecting trend and pattern of it), in marketing (for collaborating filtering, recommendation

system or customer partitioning), multimedia data analysis (clustering image, video or audio sequences), detection of biology of plant and animal with respect to their features & clustering biological data i.e. genes or protein sequences, ordering of book for library system, in planning of city (identifying groups of houses according to their house type, value, and geographical location), in land use (identifying of areas of similar land use in an earth observation database) etc.

3.1. Conventional approaches

There are few popular conventional approaches available which are very easy to understand and implement such as

- ***k-means***

  It is one of the simplest unsupervised learning algorithms. The procedure follows a simple way to classify a given dataset through a certain number of clusters (assume *k* clusters) fixed a priori. The main idea is to identify *k* centroid for each cluster. It is preferable that the selected centroids are as much as possible far away from each other. The next step is to take each point belonging to a given dataset and associate it to the nearest centroid. When no point is pending, the first step is completed and an early clustering is done. At this point, we need to recalculate *k* new centroids. After we have these *k* new centroids, then repeat the same process iteratively. As a result of this iterative process, we may notice that the *k* centroids change their location step by step until no more changes are done. In other words, centroids do not move any more. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2 \qquad (14)$$

where, n= number of data points $x = \{x_1, x_2 \ldots, x_n\}$ set of data points $c = \{c_1, c_2 \ldots, c_k\}$ set of centroids.

- ***K-medoids***

  The main disadvantage of *k-M*eans algorithm is that if the dataset is large, increasingly converges to local minima and even after repeated application the global optimum cannot be obtained. To overcome these drawbacks *k-M*edoids in introduced. It uses a different distance criterion to determine the best representative of each cluster. In contrast to the *k-M*eans algorithm, *k-M*edoids algorithm chooses points as centers that belong to the dataset.

  Here we need to find *k* representative objects which are called medoids. *k*-Medoids can handle categorical data. Here we replace means of clusters with modes.

Given n records in cluster, mode is record made up of most frequent. Using new dissimilarity measures to deal with categorical objects.

3.2. Clustering using bio-inspired algorithms

Data mining is the process of extracting previously unknown and valid information from large databases. Clustering is an important data analysis and data mining method. It is the unsupervised classification of objects into clusters such that the objects from same cluster are similar and objects from different clusters are dissimilar. Data clustering is a difficult unsupervised learning problem because many factors such as distance measures, criterion functions, and initial conditions have come into play. Many algorithms have been proposed in literature. However, some traditional algorithms have drawbacks such as sensitive to initialization and easily trapped in local optima. Recently, bio-inspired algorithms such as ant colony algorithms (ACO) and particle swarm optimization algorithms (PSO) have found success in solving clustering problems [41].

The main aim of clustering is to group sets of objects into classes such that similar objects are placed in the same cluster while dissimilar objects are in separate clusters. It is a problem of finding groups in data sets by minimizing some measure of dissimilarity. Over the last decade, bio-inspired algorithms like PSO and ACO have found success in solving clustering problems. Self-organization, cooperation, communication, and flexibility are some of the important characteristics of bio-inspired algorithms which helped in the clustering process [41].

In this thesis work some of the above bio-inspired algorithms studied, apart from traditional ones are experimented and analyzed for clustering as an application of those optimization algorithms. Researchers have found that these optimization techniques can be modeled with clustering in several ways. In this work the optimization strategies are modeled with the cluster quality measure i.e. the main goal is to use the optimization technique to optimize the cluster quality index to find out the optimum better quality clusters. In this work ***Davies–Bouldin index*** has be used to model the optimization.

- **Davies-Bouldin Index**

  The **Davies–Bouldin index (DBI)** is a metric for evaluating clustering algorithms. This is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset.

  Davies-Bouldin index(DBI) is defined as follows

$$DB = \frac{1}{K} \sum_{k=1}^{K} R_{k,t} \qquad (15)$$

  where *K* is the number of clusters and *R* is defined as follows

$$R_{k,t} = \max_{j,\ j \neq k} \left\{ \frac{S_k + S_j}{d_{kj,t}} \right\} \qquad (16)$$

R is computed for each of the k^{th} cluster. Now $d_{kj,t}$ is the *Minkowski distance* used in [39] but here in this thesis *Euclidian distance* is considered. It is the *Euclidian distance* of order *t* between the k_{th} and j_{th} centroids. Because of *Euclidian distance t=2* is chosen and defined as

$$d_{kj,t} = \sqrt{\sum_{l=1}^{n} (x_{kl} - x_{jl})^{t=2}} \qquad (17)$$

where *l* denotes the number *l* dimension i.e. is *n* number of features.

S_i is a measure of scatter within the cluster defined as

$$S_i = \left(\frac{1}{T_i} \sum_{j=1}^{T_i} \left|X_j - A_i\right|^p\right)^{1/p} \qquad (18)$$

here A_i is the centroid of cluster *i* and T_i is the size of the cluster *i*. Usually the value of *p* is 2, which makes this a Euclidean distance function between the centroid of the cluster, and the individual feature vectors.

3.2.1 Cuckoo Search clustering algorithm (CSCA)

As studied that cuckoo has an aggressive breeding strategy [14, 42] for finding the best host nest for their egg. Due to this CS algorithm yields good results on benchmark dataset [42]. CSCA makes use of ***DBI*** as fitness function, described in ***Fig 13*** also a new method ***New_Cuckoo*** for generation of new host is used [42] described in ***Fig 14***.

- **New_Cuckoo**

  The cuckoo laid eggs which correspond to a new solution set. The new set of clusters is created using the current best solution. For finding better solutions a pseudorandom proportional rule is used. In this rule, *q € [0, 1]* is the standard CSCA parameter and *q1* is a random value in *[0, 1]*. This rule helps to exhibit exploration and exploitation way to search the solutions.

1. Begin
2. Consider ***NH*** *host nests containing* ***1*** *egg (solution) each*
3. ***for each solution*** *of host i*
4. Initialize x_i *contain* ***K*** *randomly selected cluster centroids (corresponding to* ***K clusters****), as* $x_i = (m_{i,1} \ldots m_{i,k} \ldots m_{i,k})$ *where* $m_{i,k}$ *represent the* k^{th} *cluster centroid vector of* i^{th} *host.*
End for loop
5. ***for t iterations***
6. ***for each solution of host i*** *of the population*
7. ***for each data point*** Z_p
8. Calculate distance $d(Z_p, m_{i,k})$ *from all cluster centroids* $C_{i,k}$ *using*

$$d(Z_p,\ m_{i,k}) = \sqrt{\sum_{j=1}^{N_d} Z_{pj} - m_{(i,\ k)j}}$$

Where, N_d *is the dimension of the data points.*
9. Assign Z_p *to* $C_{i,k}$ *where*

$$d(Z_p,\ m_{i,\ k}) = \min_{\forall\, k=1\ldots k} \{d(Z_p,\ m_{i,\ k})\}$$

End for loop
10. Calculate fitness function $f(x_i)$ *for each host nest* ***i*** *using* ***Davies – Bouldin index (DBI)*** *i.e.* ***equation 15.***
End for loop
11. Select a fraction (p_a) *of worse nests* (p_a *= abandoned host nest probability :* $p_a \in [0, 1]$)
12. ***for each*** $y \in p_a$
13. Build new ones using ***New_Cuckoo*** *function*
End for loop
14. Keep the best solutions (or nests with quality solution)
15. Find the current best solution
End outermost for loop
16. Consider the clustering solution represented by the best solution
17. ***END***

Fig 3.7: Pseudo code of CSCA

```
q ϵ {0, 1}
q1 = random number ϵ {0, 1}
if (q1 < q)
        for each cluster centroid m_new, k
                for each dimension n_d
                        x_new (m_new, k , n_d) = x_best + pow(-1, n_d) * rand(1)
                        where, x_best is the best solution of current iteration
                end
        end
else
        select another set of clusters randomly from the search space
end
```

Fig 3.8: Pseudo code of New_Cuckoo

4. Experiments

In this thesis above discussed algorithms are experimented in various function optimizations and also applied and studied on data clustering as an advancement.

4.1. Optimization functions

Following functions are used in the experiments for optimization.

Table 1: Used functions for optimization

Name	Function	Ref
F1	$f(x) = \prod_{i=1}^{10}(x - 2i)$	[40]
F2	$f(x,y) = 0.5 - \frac{\left\{\sin\sqrt{(x^2+y^2)}\right\}^2 - 0.5}{\left(1.0 + 0.001(x^2+y^2)\right)^2}$	[40]
Sphere	$f(x) = \sum_{i=1}^{d} x_i^2$	[41]
Ackley	$f(x) = -a\, e^{-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)}} + a + e^1$	[41]
Cross-in-tray	$f(x) = -0.0001\left(\left\lvert \sin(x_1)\sin(x_2)\, e^{\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert}\right\rvert + 1\right)^{0.1}$	[41]

Drop wave	$f(x) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5(x_1^2 + x_2^2) + 2}$	[41]
Egg-holder	$f(x) = -(x_2 + 47)\sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right) - x_1 \sin\left(\sqrt{\lvert x_1 - (x_2 + 47)\rvert}\right)$	[41]
Easom	$f(x) = -\cos(x_1)\cos(x_2)\, e^{-(x_1-\pi)^2-(x_2-\pi)^2}$	[41]
Rotated Hyper-Ellipsoid	$f(x) = \sum_{i=1}^{d}\sum_{j=1}^{i} x_j^2$	[41]

4.1.1. Shape of the functions

- **F1** → It is a liner function [40]

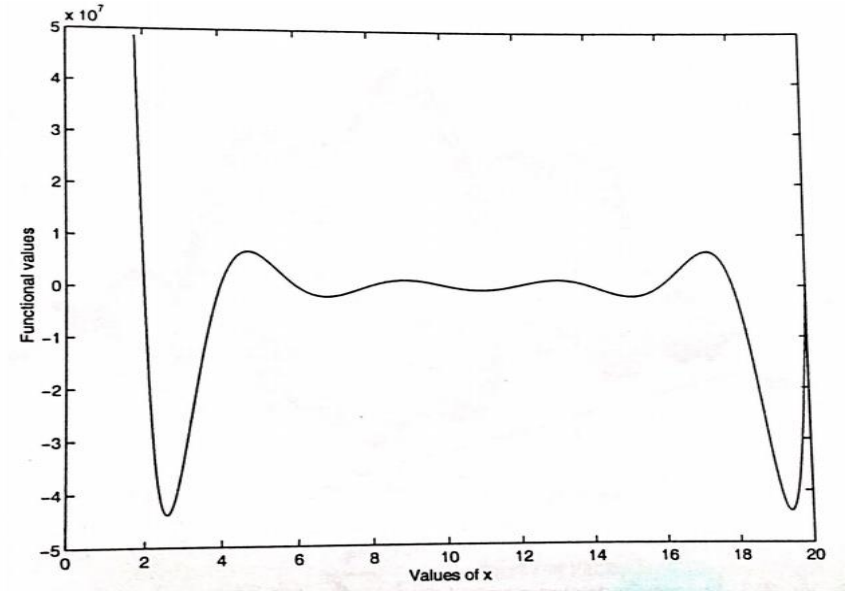

Fig 4.1: nature of F1 function

- **F2 →** It has many local maxima [40]

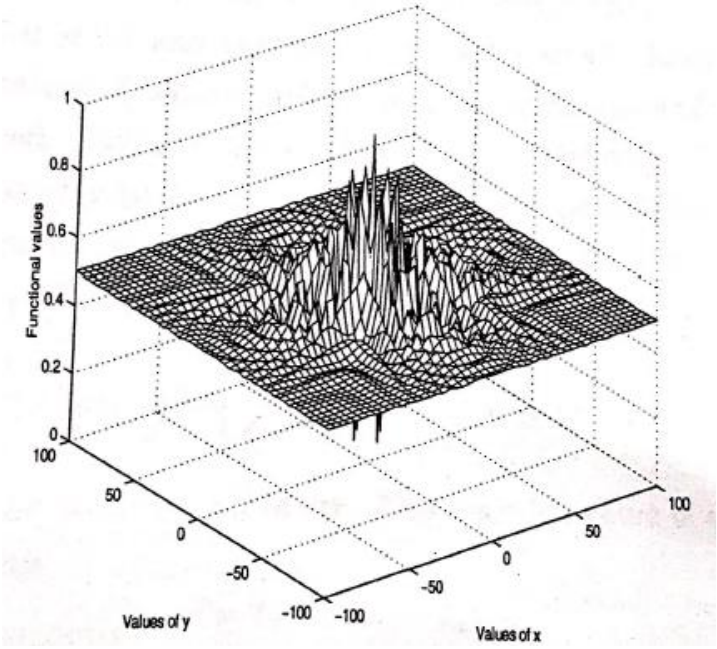

Fig 4.2: nature of F2 function

- **Sphere →** it is a *bowl-shaped* function [41]

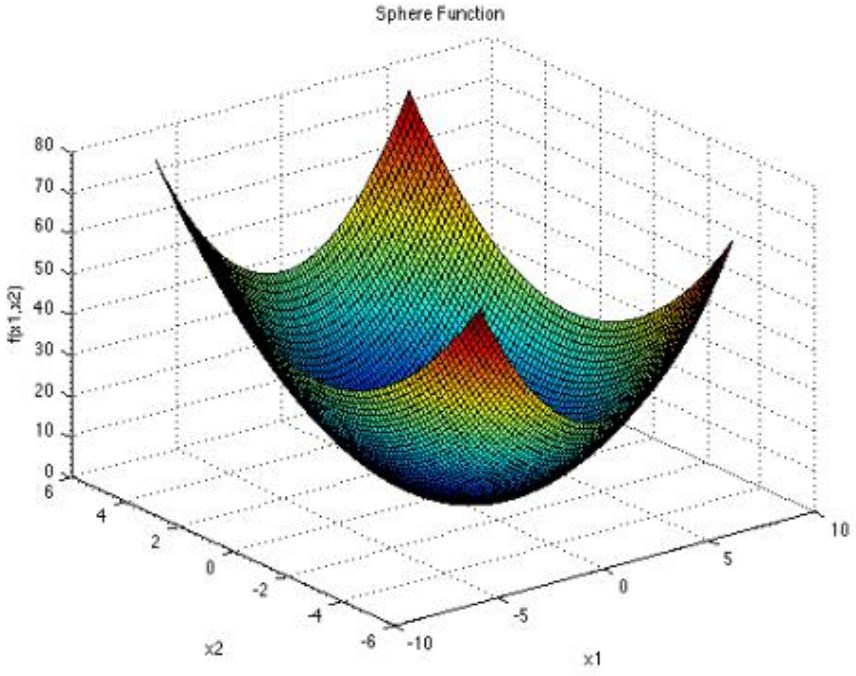

Fig 4.3: nature of sphere function

- **Ackley** → it has *many local minima* function [41]

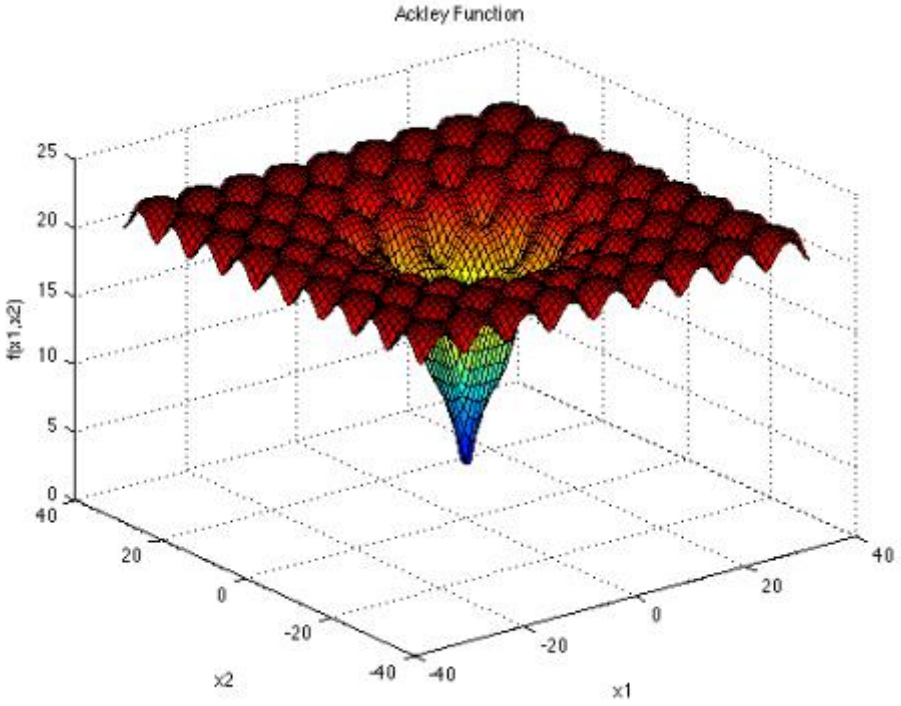

Fig 4.4: nature of Ackley function

- **Cross-in-tray** → it is a *many local minima* function [41]

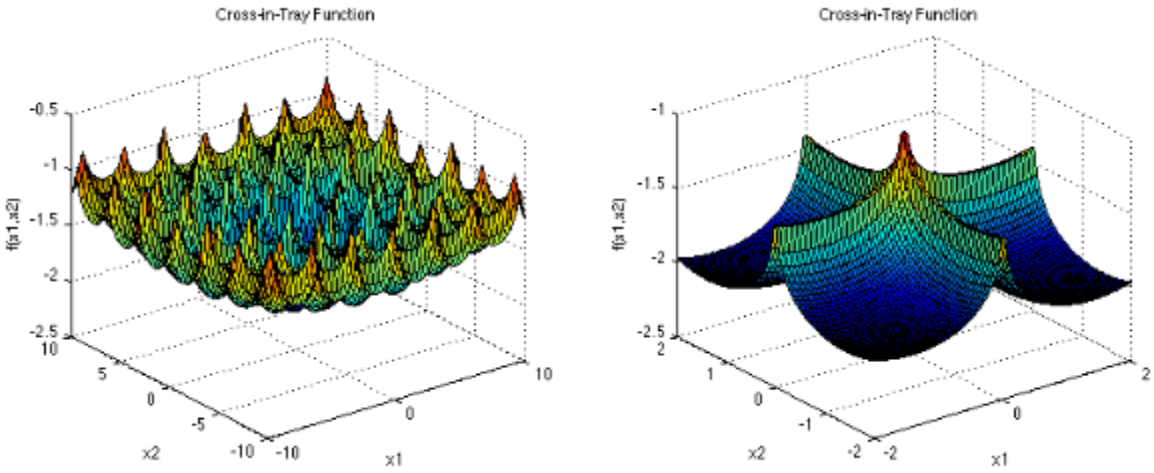

Fig 4.5: nature of Cross-in-tray function

- **Drop-Wave** → it is a *many local minima* function [41]

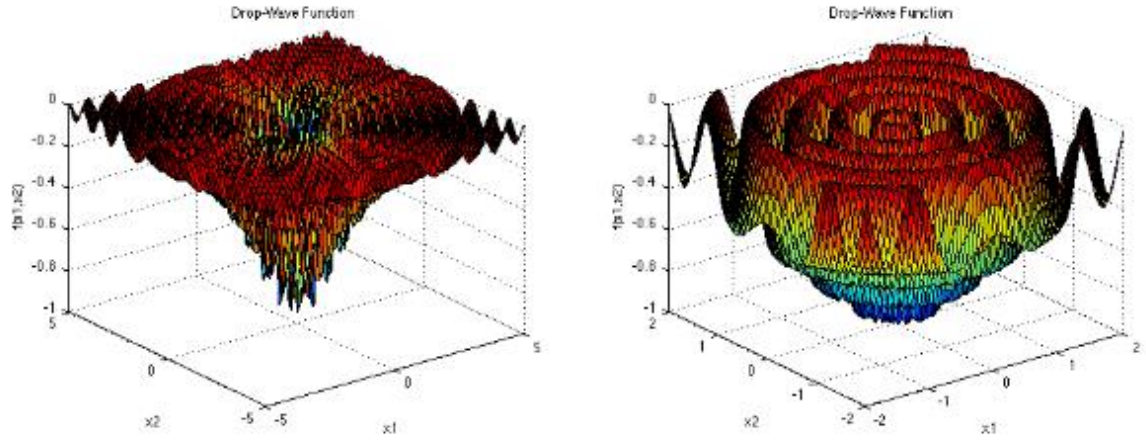

Fig 4.6: nature of Drop-Wave function

- **Egg-holder** → it is a *many local minima* function [41]

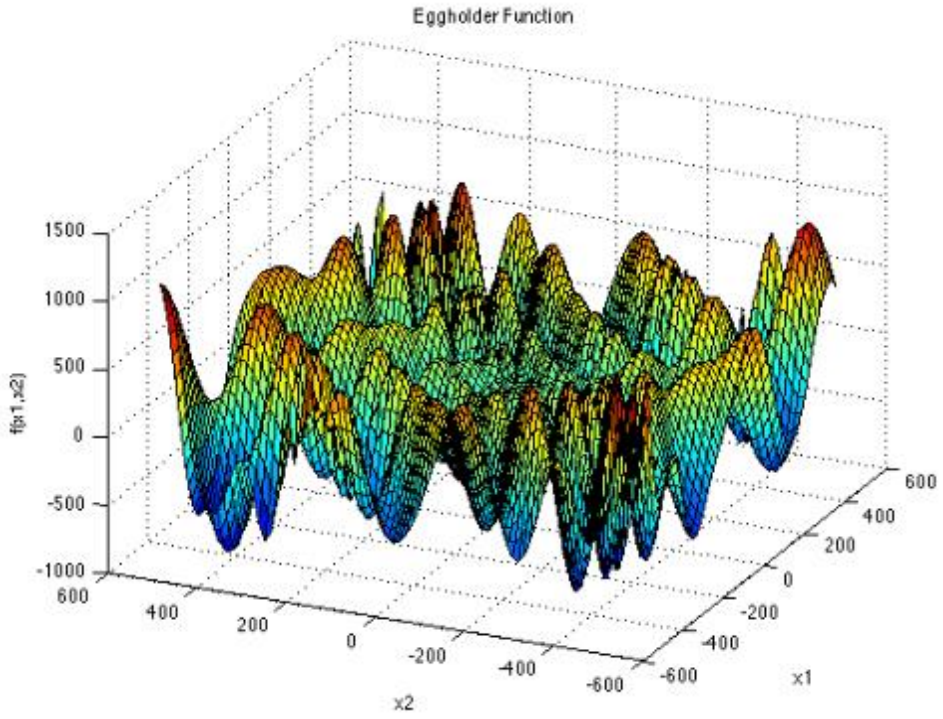

Fig 4.7: nature of Egg-holder function

- **Easom →** It has *steep ridges/drops* [41].

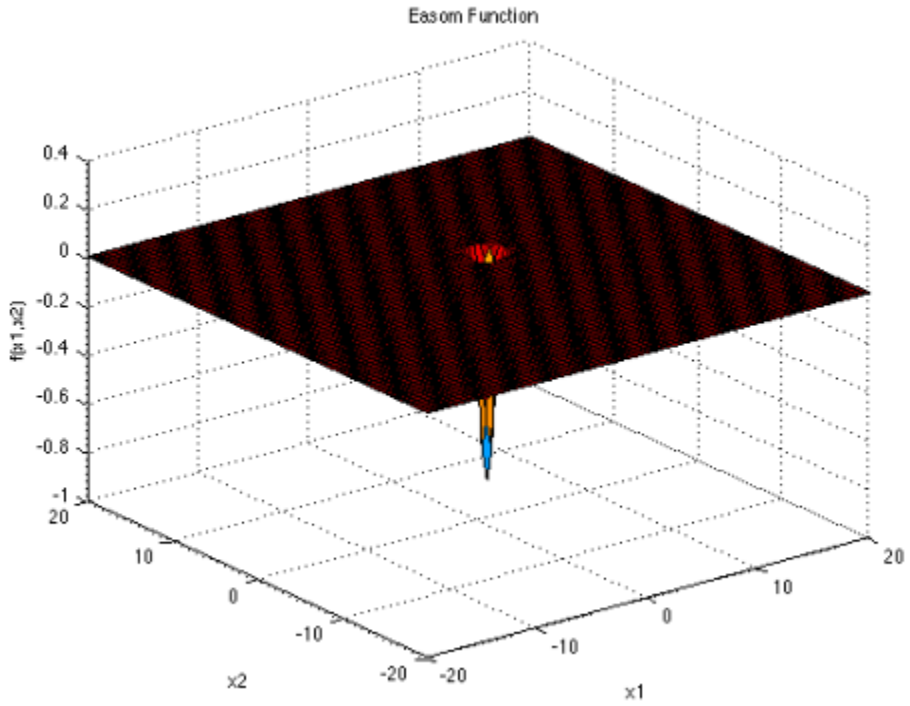

Fig 4.8: nature of Easom function

- **Rotated Hyper-Ellipsoid →** it is a *bowl-shaped* function [41]

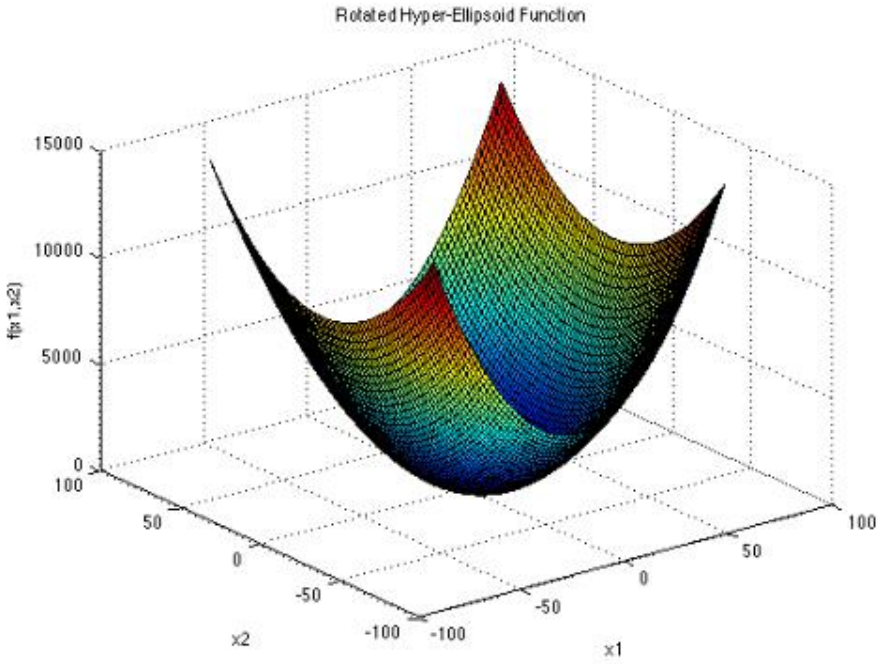

Fig 4.9: nature of Rotated Hyper-Ellipsoid function

4.2. Datasets used

Now for data clustering few common datasets are used.

- **Breast Cancer:**
  - This dataset contains 699 instances, 10 features and number of cluster = 2. Here is no missing value attributes are present. Class names are begin and malignant.

- **Iris dataset:**
  - This dataset contains 150 instances, 4 features and number of cluster =3 where attributes are sepal length, sepal width, petal length, petal width and class name are Iris Setosa, Iris Versicolour and Iris Virginica.

- **Wine dataset:**
  - This dataset contains178 instances, 13 features and number of cluster =3. This dataset contains no missing value attributes.

4.3. Parameter Details

Necessary parameters of each algorithms are defined below

Table 2: Parameter details

Parameter Name	Parameter Value
Bat algorithm (BA):	
Population range	Depends on the optimization function
Frequency	randomly initialized within a range 0 to 100
Pulse rate	randomly initialized within a range 0 to 1
Loudness	randomly initialized within a range 0 to 1
β a numerical constant	randomly chosen within a range 0 to 1
ϵ a numerical constant	randomly chosen within a range -1 to 1
Bee Colony Optimization Algorithm (BCOA)	
Population range	Depends on the optimization function
Number of constructive moves	10
Φ a numerical constant	randomly chosen within a range -1 to 1
Cuckoo Search Algorithm (CSA)	
Population range	Depends on the optimization function
Abandoned frequency	0.25
α a numerical constant	1
λ a numerical constant	randomly chosen within a range 1 to 3
Firefly Algorithm (FA)	
Population range	Depends on the optimization function
α a numerical constant	0.2

β_0 attractiveness of a firefly	1
ϵ a numerical constant	randomly chosen within a range 0 to 1
γ light absorption coefficient	1
* For all the above algorithms Population Size is chosen 20. ** 1000 generation has been used. *** For each experiment 50 simulations are done.	
Cuckoo Search Clustering Algorithm (CSCA)	
Population Size	10
Generation	100
Abandoned frequency	0 to 1 experimented with 0.5
q a numerical constant	0 to 1 experimented with 0.5, 0.8, 0.9

5. Results and Analysis

In this section results are included from all the experiments done and then results are analyzed.

Table 3: Average (over fifty simulations) of optimum fitness, average fitness, time and standard deviation of optimum fitness

Function	Type	Algorithm	Average of optimum fitness	Standard deviation of average optimum fitness	Average of Average fitness	Average time
F1	Maximization	BA	3.67E+09	53901471	2.69E+09	1.321903
		CSA	3.7E+09	18629860	2.69E+09	0.230546
		FA	3.57E+09	7.27E+08	2.03E+08	1.091507
		BCOA	3.72E+09	0.066376	3.71E+09	4.652039
F2	Maximization	BA	0.977549	0.015417	0.76392	1.401583
		CSA	0.972817	0.01543	0.799662	0.240582
		FA	0.977972	0.023941	0.618465	1.040458
		BCOA	0.990181	0.004792	0.914618	4.676088
Sphere	Minimization	BA	34.99429	21.63396	5032.534	1.50059
		CSA	41.65837	26.21916	2602.912	0.252196
		FA	16.95265	27.70662	5974.174	1.137797
		BCOA *	1833.166	1352.86	25915.2	2.41564
Ackley	Minimization	BA	5.062806	1.310949	15.17226	1.621835
		CSA	5.196096	1.473874	12.09984	0.269576
		FA	1.346245	0.010101	17.12504	1.484604
		BCOA *	16.11458	3.78105	21.66998	3.049963
Cross-in-tray	Minimization	BA	-2.06217	0.000443	-1.9712	1.396804
		CSA	-2.06204	0.000543	-1.92271	0.248975

		FA	-2.06261	1.17E-06	-1.65051	1.242404
		BCOA	-2.06255	0.000281	-2.03464	5.000923
Drop Wave	Minimization	BA	-0.94514	0.018546	-0.82015	1.376204
		CSA	-0.94208	0.016835	-0.71537	0.242053
		FA	-0.97218	0.030792	-0.31561	1.221311
		BCOA	-0.97595	0.023245	-0.8176	4.671289
Egg holder	Minimization	BA	-938.166	10.73079	-447.956	1.436716
		CSA	-943.229	10.43927	-677.943	0.240197
		FA	-827.368	94.00024	-269.025	0.999171
		BCOA	-876.614	74.79507	-803.462	4.651792
Easom	Minimization	BA	-0.30036	0.312941	-0.02163	1.405663
		CSA	-0.26423	0.306756	-0.01708	0.14754
		FA	-0.06003	0.237478	-0.003	0.2703
		BCOA	-0.79749	0.17683	9.65E-07	3.64985
Rotated Hyper Ellipsoid	Minimization	BA	25.54424	17.46867	3991.009	1.579205
		CSA	33.21324	25.49343	2351.457	0.263403
		FA	0.000684	0.000766	5249.484	1.297828
		BCOA *	1364.418	1105.968	22602.83	2.824579

NOTE: (*) mark means particular algorithm did not show desired results for the particular optimization function

5.1. Observations

a) In most of the functions *FA, BCOA* shows desired optimum results.
b) In majority *BCOA* shows least deviation, hence stability of *BCOA* is higher among all.
c) Also *CSA* and *FA* shows similar stability in respected cases.
d) *CSA* predominantly fastest among all in every case.
e) *FA* is just behind *CSA* and the second fastest.
f) Considering observation (a), (c), (e) it is to be noted that *FA* is tend to perform well enough with stability and it is fairly faster as well.
g) In terms of reliability *BCOA* performs well enough. In *BCOA* whole population greatly follows the best one.
h) *CSA* is right after it in terms of reliability also stable and fast.

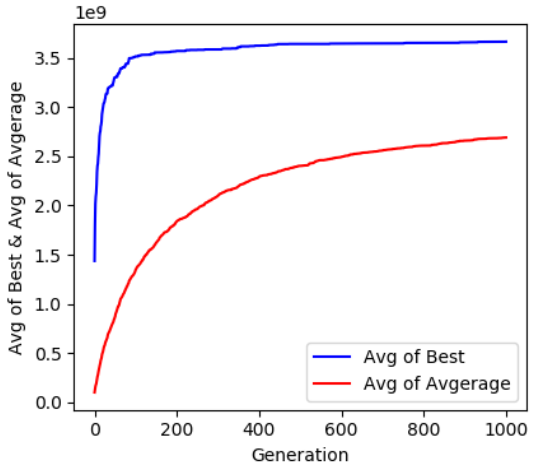

Fig 5.1 (a)

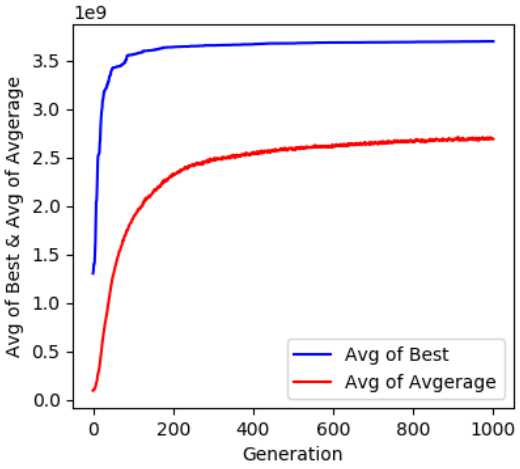

Fig 5.1 (b)

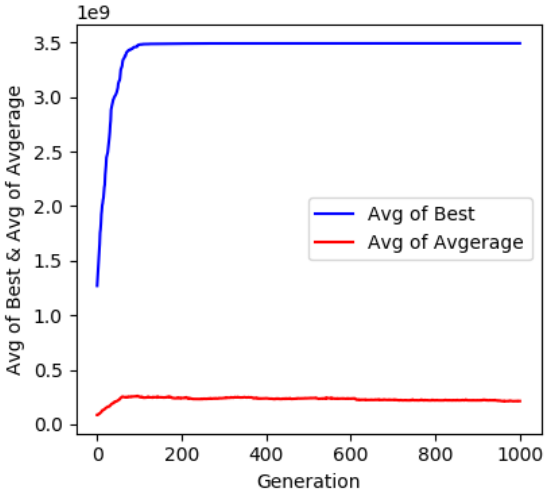

Fig 5.1 (c)

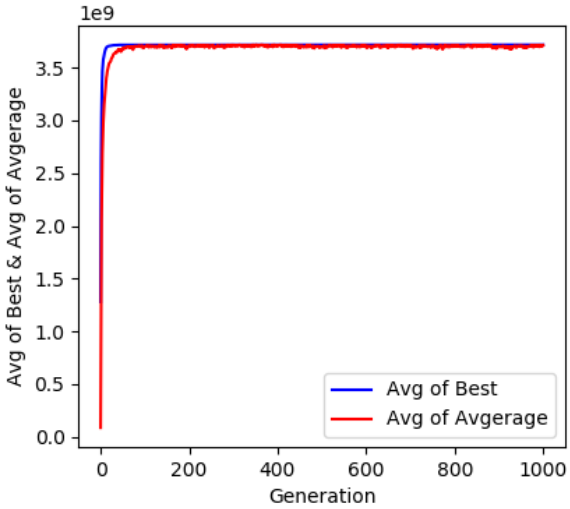

Fig 5.1 (d)

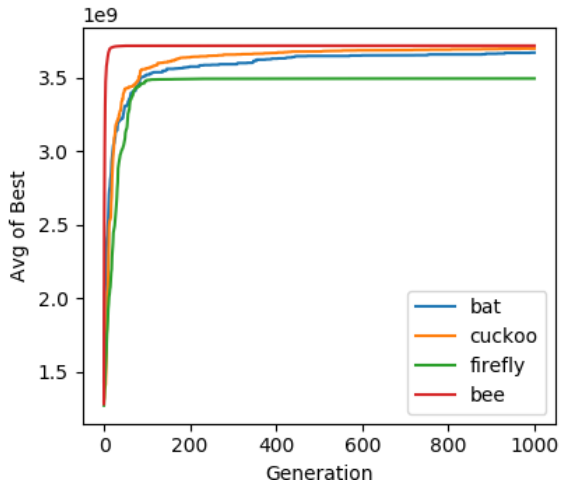

Fig 5.1 (e)

Fig 5.1: Visualization of average optimum fitness and average of average fitness of population of F1 function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation according to BCOA, (e) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.1 (a), (b) shows similar curve of average of average fitness of each generation and also more or less parallel to average optimum fitness value curve. This depicts both the algorithm's decent amount of population follows best solution.
- It is also to be noted the gap between average optimum fitness curve and average of average fitness curve in Fig 5.1 (b) is little less than Fig 5.1 (a) which says *CSA* shows more reliability than *BA* for *F1* function.
- From Fig 5.1 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.1 (d) shows *BCOA* has a very good reliability for *F1* function.
- Fig 5.1 (e) shows *BCOA* has a steep curve and saturated in earlier generation than others. *FA* also saturated earlier than *CSA* and *BA* but did not reach the value as others, whereas rest three attend more or less similar value.

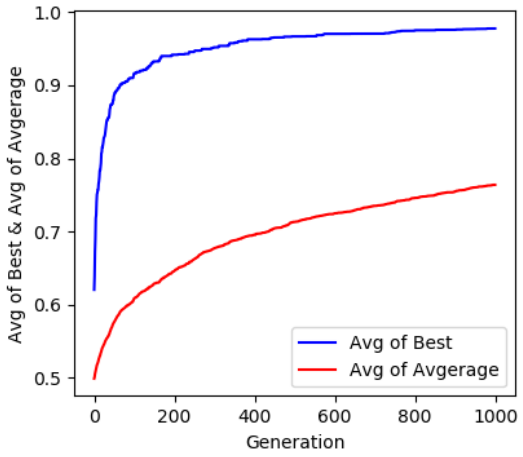

Fig 5.2 (a)

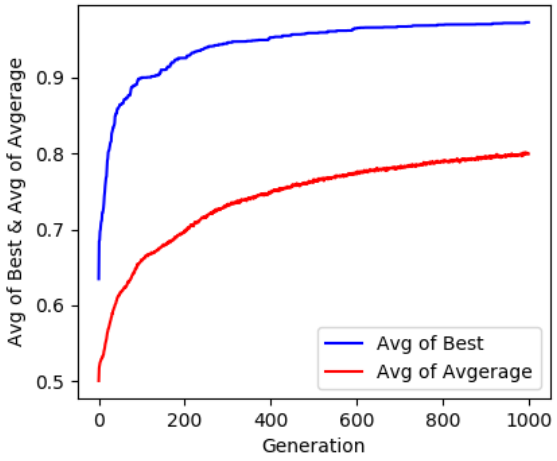

Fig 5.2 (b)

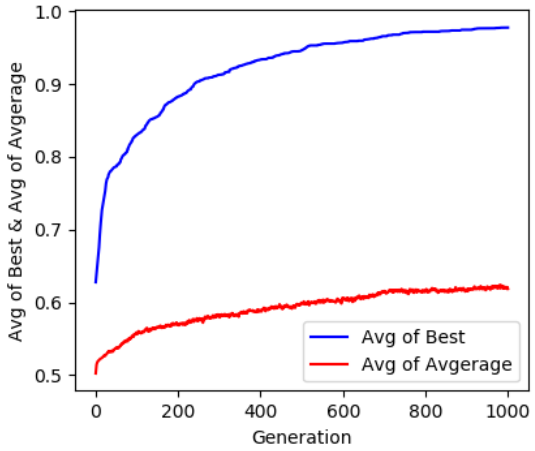

Fig 5.2 (c)

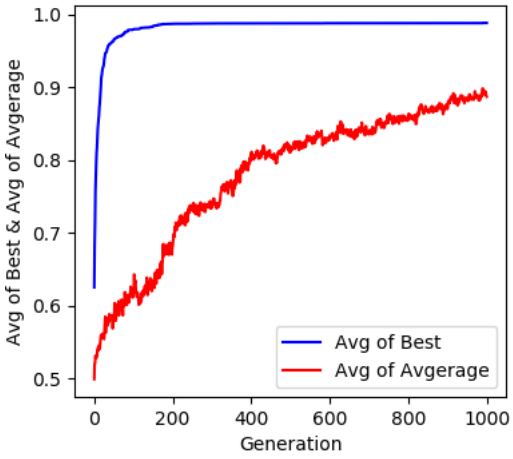

Fig 5.2 (d)

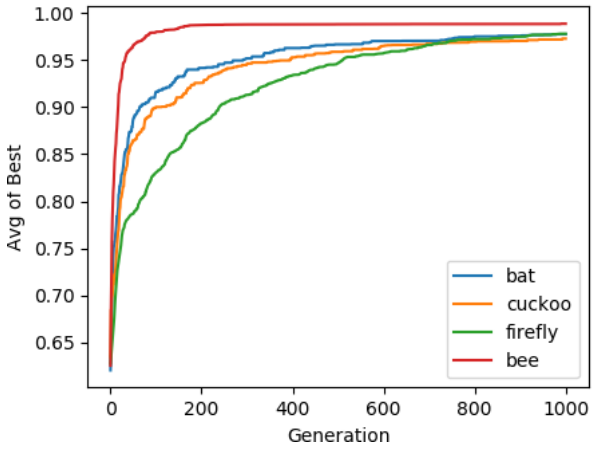

Fig 5.2 (e)

Fig 5.2: Visualization of average optimum fitness and average of average fitness of population of F2 function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation according to BCOA, (e) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.2 (a), (b) shows similar curve of average of average fitness of each generation and also more or less parallel to average optimum fitness value curve. This depicts both the algorithm's decent amount of population follows best solution.
- It is also to be noted the gap between average optimum fitness curve and average of average fitness curve in Fig 5.2 (b) is little less than Fig 5.2 (a) which says *CSA* shows more reliability than *BA* for *F2* function.
- From Fig 5.1 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.1 (d) shows *BCOA* gradually improving average of average fitness for *F2* function. It may be seen more reduced gap between average optimum fitness curve and average of average fitness in later generation which is beyond experiment setup.
- Fig 5.1 (e) shows *BCOA* has a steep curve and saturated in earlier generation than others. *BA* and *CSA* saturated in later generation but interestingly *FA* gradually improved in later generation than *BA* and *CSA.*

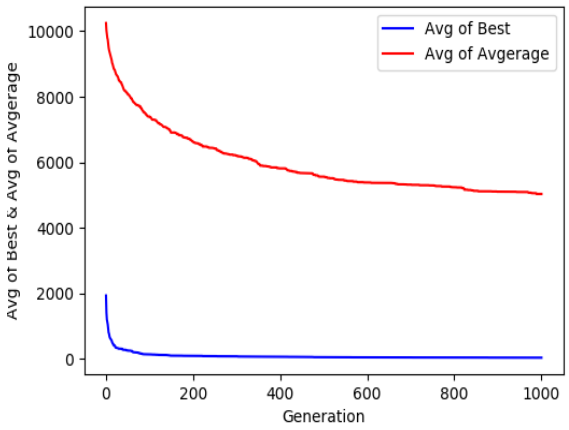

Fig 5.3 (a)

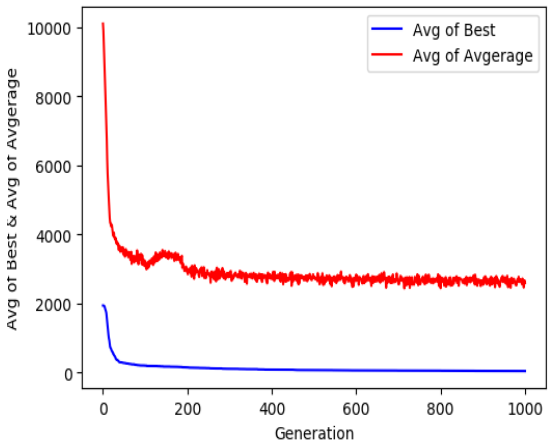

Fig 5.3 (b)

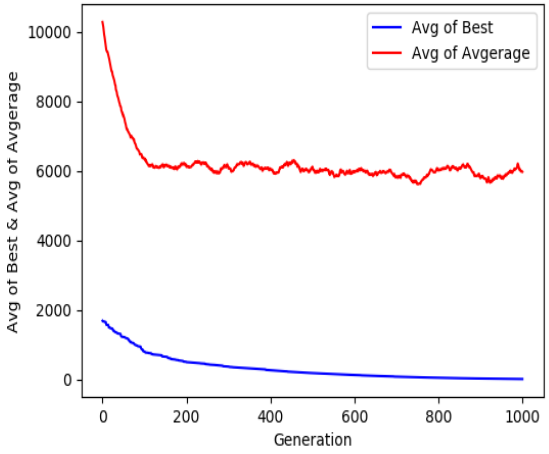

Fig 5.3 (c)

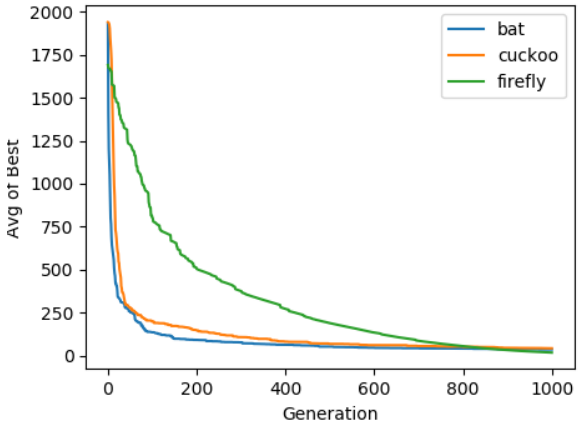

Fig 5.3 (d)

Fig 5.3: Visualization of average optimum fitness and average of average fitness of population of Sphere function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.3 (a), (b) shows similar curve of average of average fitness of each generation and also more or less parallel to average optimum fitness value curve. This depicts both the algorithm's decent amount of population follows best solution.
- It is also to be noted the gap between average optimum fitness curve and average of average fitness curve in Fig 5.3 (b) is little less than Fig 5.3 (a) which says *CSA* shows more reliability than *BA* for *Sphere* function.

- From Fig 5.3 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.3 (d) shows *BA* and *CSA* have a steep curve and saturated in mid rang generations which is earlier than *FA*. Interestingly *FA* gradually improved in later generations than *BA* and *CSA*.

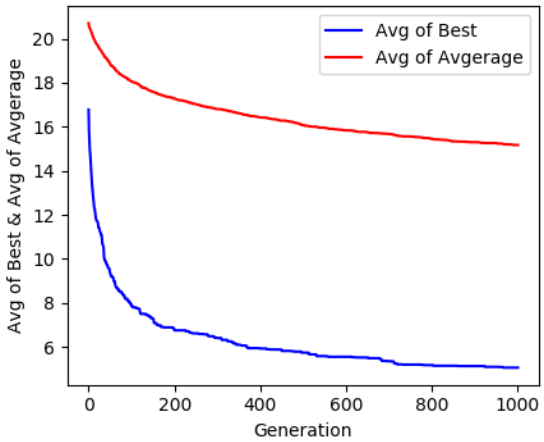

Fig 5.4 (a)

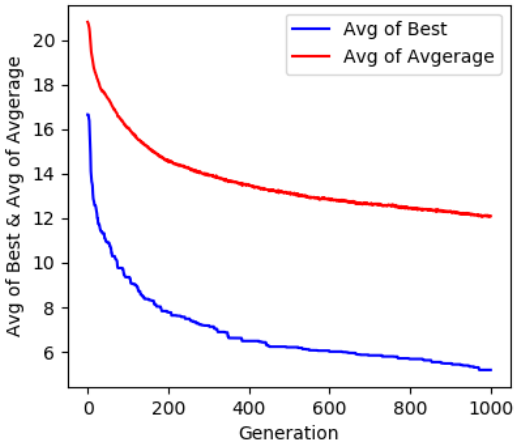

Fig 5.4 (b)

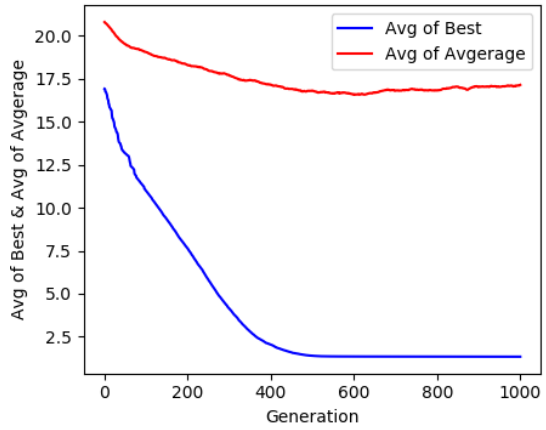

Fig 5.4 (c)

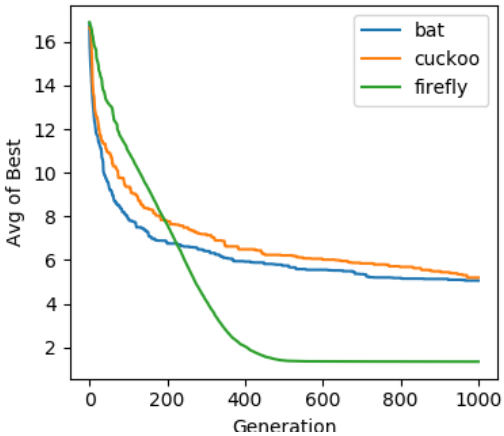

Fig 5.4 (d)

Fig 5.4: Visualization of average optimum fitness and average of average fitness of population of Ackley function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.4 (a), (b) shows similar curve of average of average fitness of each generation and also more or less parallel to average optimum fitness value curve. This depicts both the algorithm's decent amount of population follows best solution.
- It is also to be noted the gap between average optimum fitness curve and average of average fitness curve in Fig 5.4 (b) is little less than Fig 5.4 (a) which says *CSA* shows more reliability than *BA* for *Ackley* function.
- From Fig 5.4 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.3 (d) *FA* gradually improved than *BA* and *CSA* and saturated in mid rang of generation whereas *BA* and *CSA* did not saturate but could not minimize the average optimum fitness value as *FA* in slower rate. They might take more generations to saturate.

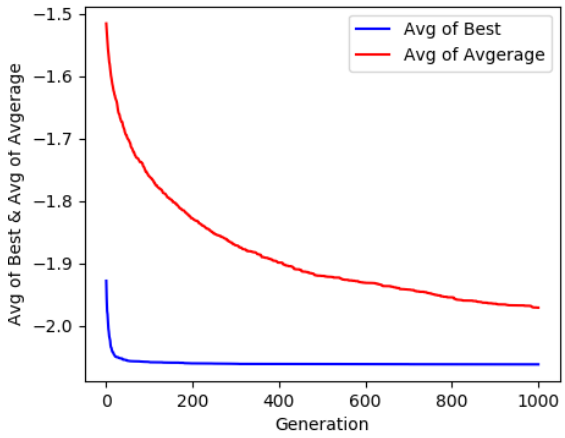

Fig 5.5 (a)

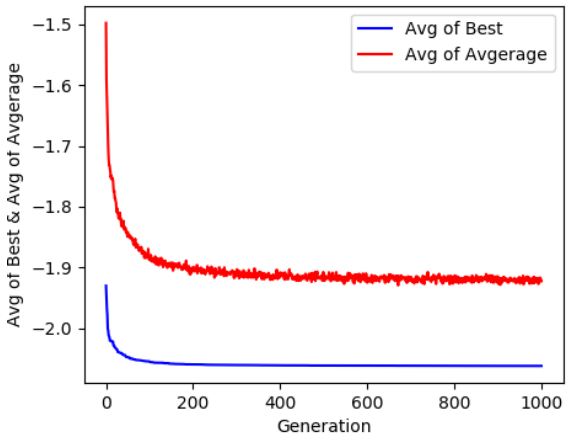

Fig 5.5 (b)

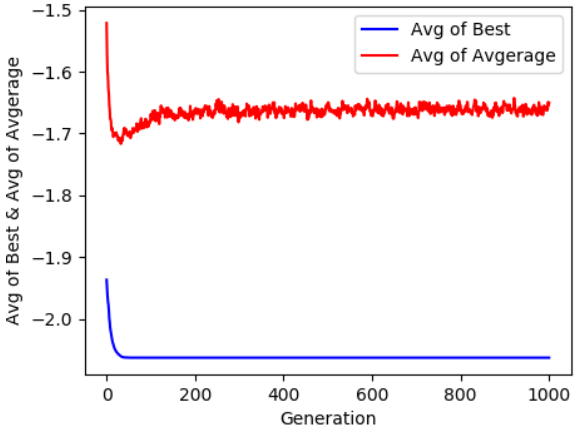

Fig 5.5 (c)

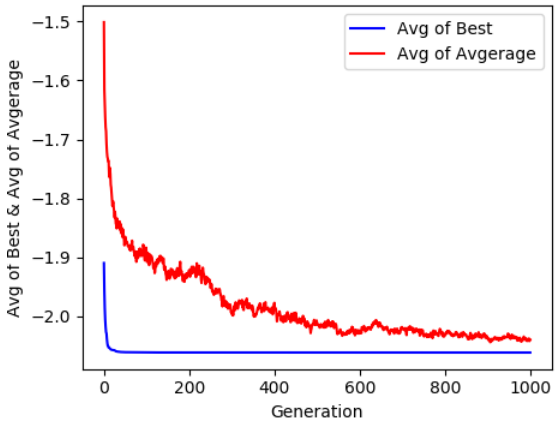

Fig 5.5 (d)

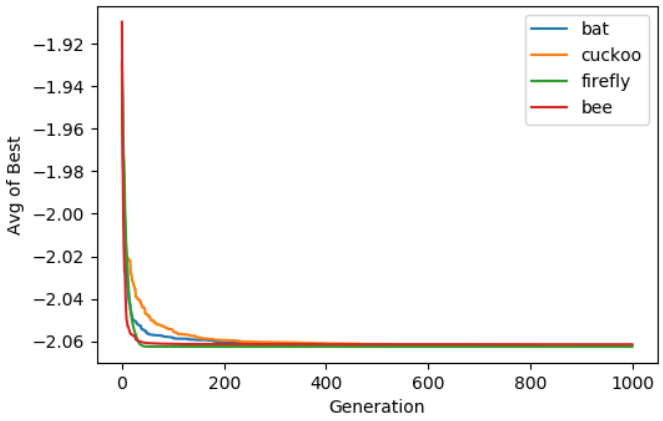

Fig 5.5 (e)

Fig 5.5: Visualization of average optimum fitness and average of average fitness of population of Cross-in-try function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation according to BCOA, (e) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.5 (a), (b), (d) shows desirable curve. Where (a) and (d) shows that average of average fitness is gradually improving. In case of (b) average of average fitness is saturated and continued parallel to average optimum fitness.
- From Fig 5.1 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.5 (e) shows *BCOA* and *FA* has a steep curve and saturated in earlier generation than others. *BA* and *CSA* saturated gradually and met other two.
- It is seen there is continuous small spikes in average of average fitness which indicates that whole population attending a fitness value with in a range per simulation.

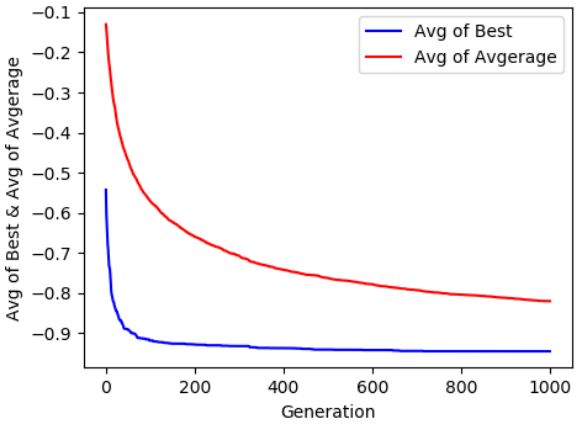

Fig 5.6 (a)

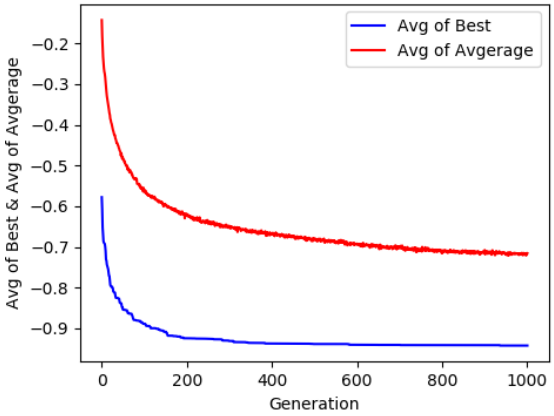

Fig 5.6 (b)

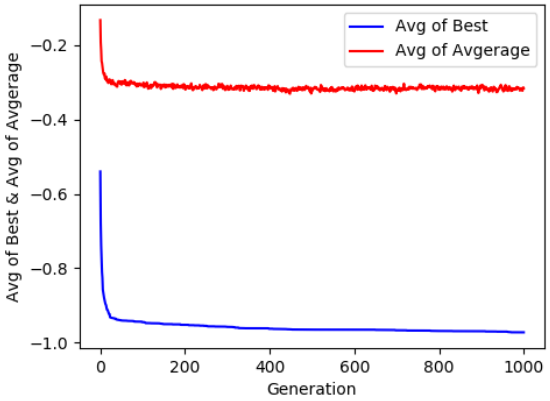

Fig 5.6 (c)

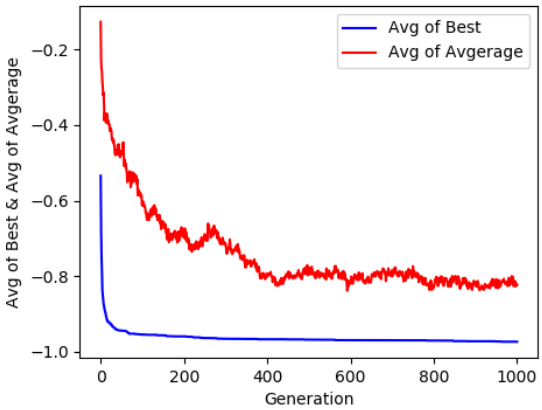

Fig 5.6 (d)

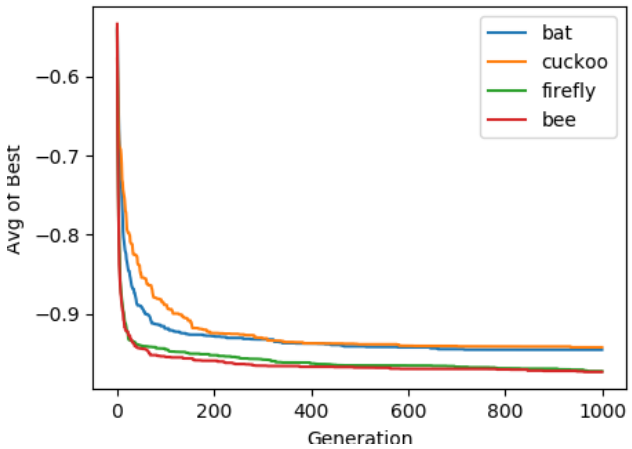

Fig 5.6 (e)

Fig 5.6: Visualization of average optimum fitness and average of average fitness of population of Drop Wave function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation according to BCOA, (e) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.6 (a), (b), (d) shows desirable curve. Where (a) and (b) shows that average of average fitness is gradually improving. In case of (d) it is seen that average of average fitness is improved in early generations and saturates in mid generation and kept saturated in later generations then again improved and saturated including small spikes.
- From Fig 5.6 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.6 (e) shows *BCOA* and *FA* has a steep curve and saturated in earlier generation than others and they both almost attended similar value. *BA* and *CSA* saturated gradually but did not met other two they produced near similar values.
- It is seen there is continuous small spikes in average of average fitness which indicates that whole population attending a fitness value with in a range per simulation.

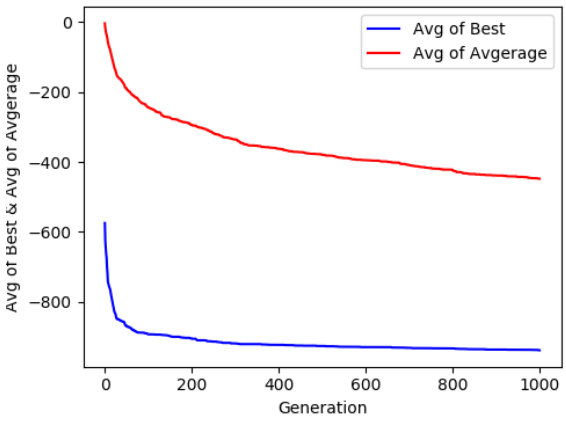

Fig 5.7 (a)

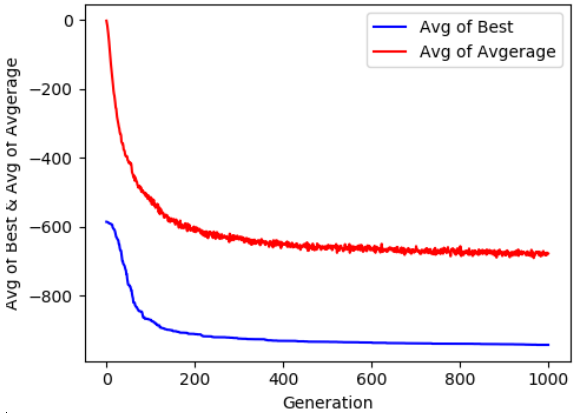

Fig 5.7 (b)

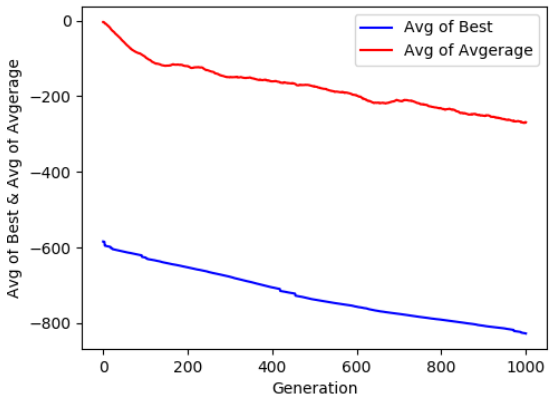

Fig 5.7 (c)

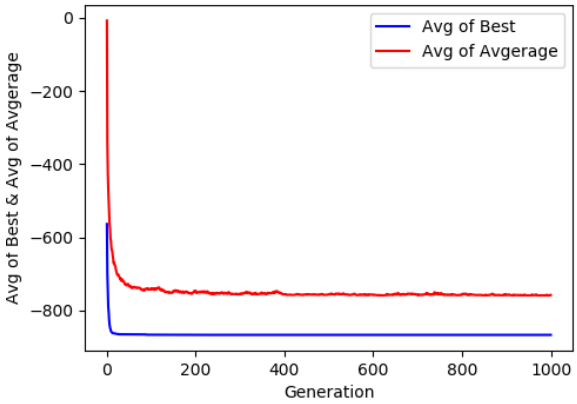

Fig 5.7 (d)

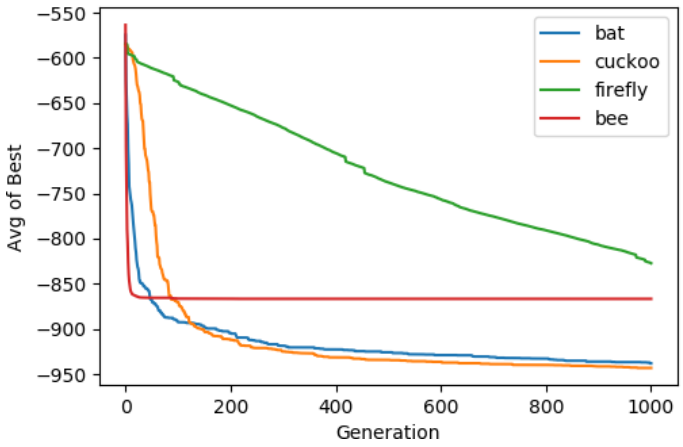

Fig 5.7 (e)

Fig 5.7: Visualization of average optimum fitness and average of average fitness of population of Egg-holder function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation according to BCOA, (e) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.7 (a), (b) shows similar curve of average of average fitness of each generation and also more or less parallel to average optimum fitness value curve. This depicts both the algorithm's decent amount of population follows best solution.
- It is also to be noted the gap between average optimum fitness curve and average of average fitness curve in Fig 5.7 (b) is little less than Fig 5.7 (a) which says *CSA* shows more reliability than *BA* for *Egg-holder* function.
- From Fig 5.7 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.7 (e) shows *BCOA* has a steep curve and saturated in earlier generation than others but did not attend the value as *BA* and *CSA*. *BA* and *CSA* saturated in later generations. *FA* shows almost a liner curve which indicates improvement rate is slow per generation.

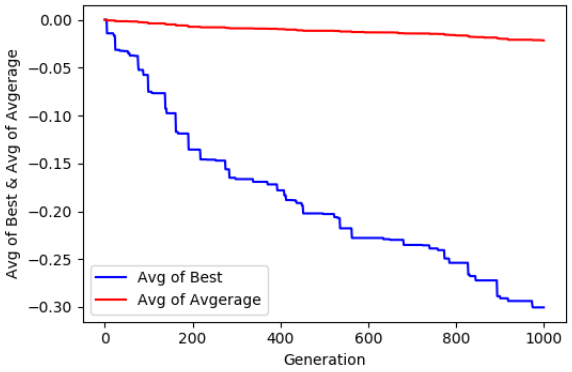

Fig 5.8 (a)

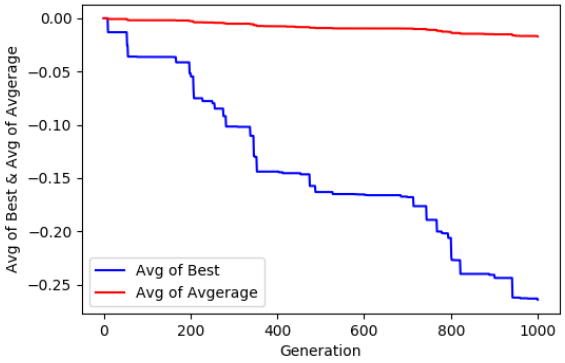

Fig 5.8 (b)

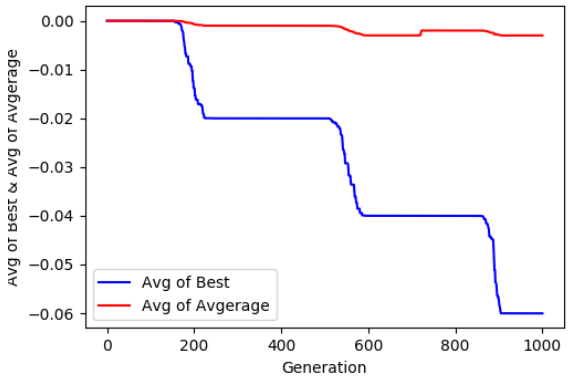

Fig 5.8 (c)

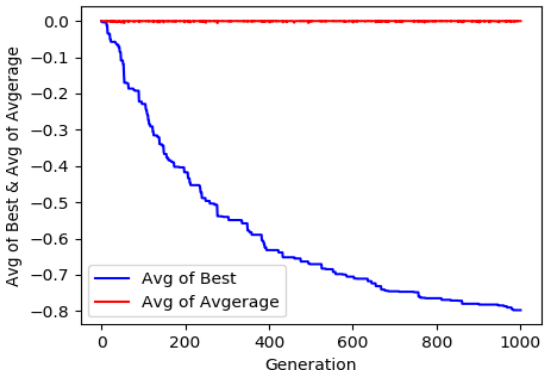

Fig 5.8 (d)

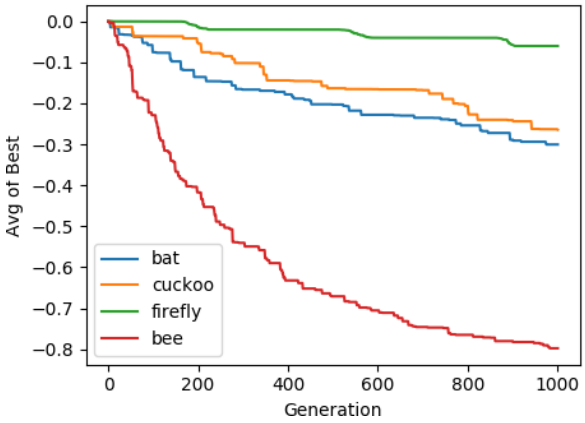

Fig 5.8 (e)

Fig 5.8: Visualization of average optimum fitness and average of average fitness of population of Easom function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation according to BCOA, (e) variation of average optimum fitness value according to all algorithms, with generation

- In above figure Fig 5.8 it is seen that average of average fitness curve is not closer to average of optimum fitness curve, which denotes a less amount of population followed the best one.
- Fig 5.8 (e) shows *BCOA* produced a good optimum value and *BCOA* and *CSA* produced near similar results but *FA* produced poor quality result than others.

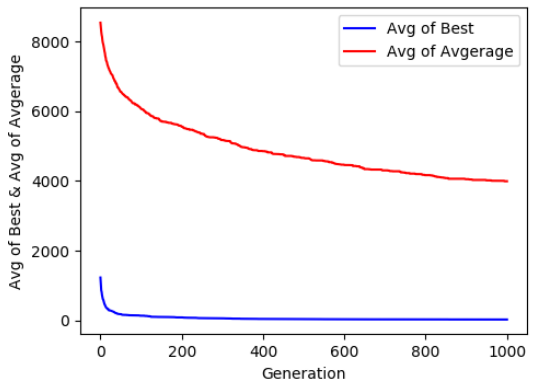

Fig 5.9 (a)

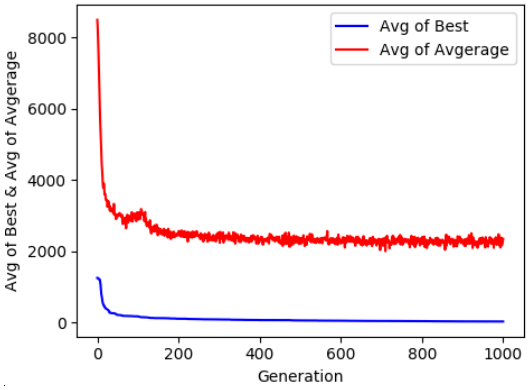

Fig 5.9 (b)

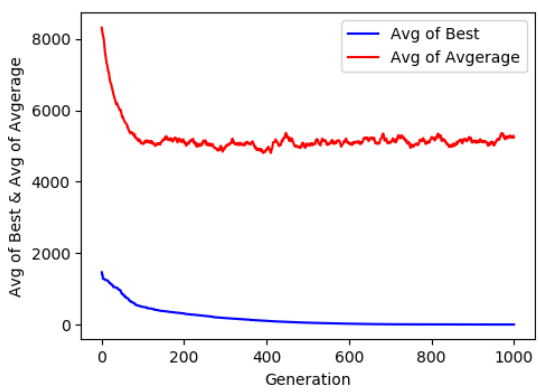

Fig 5.9 (c)

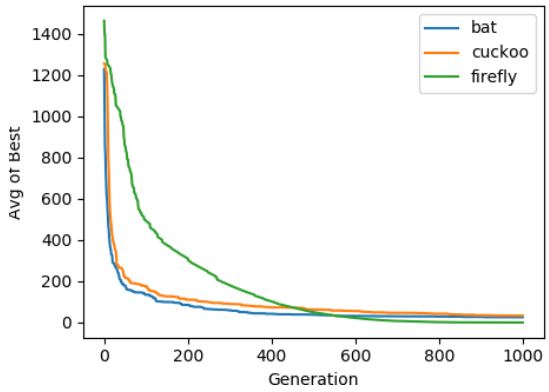

Fig 5.9 (d)

Fig 5.9: Visualization of average optimum fitness and average of average fitness of population of Rotated-hyper-ellipsoid function, (a) variation according to BA, (b) variation according to CSA, (c) variation according to FA, (d) variation of average optimum fitness value according to all algorithms, with generation

- Fig 5.3 (a), (b) shows similar curve of average of average fitness of each generation and also more or less parallel to average optimum fitness value curve. This depicts both the algorithm's decent amount of population follows best solution.
- It is also to be noted the gap between average optimum fitness curve and average of average fitness curve in Fig 5.3 (b) is little less than Fig 5.3 (a) which says *CSA* shows more reliability than *BA* for *Rotated-hyper-ellipsoid* function.
- From Fig 5.3 (c) it is seen that there is an undesirable gap between average optimum fitness and average of average fitness which indicates the *FA* does not show good reliability as others.
- Fig 5.3 (d) shows *BA* and *CSA* have a steep curve and saturated in mid rang generations which is earlier than *FA*. Interestingly *FA* gradually improved in later generations than *BA* and *CSA*.

5.2. Results of clustering

Table 4: Average (over five simulation) of minimum DBI, average DBI and standard deviation of minimum DBI with abandon frequency = 0.5

Dataset	q = 0.5			q = 0.8			q = 0.9		
	DBI	Avg DBI	std	DBI	Avg DBI	std	DBI	Avg DBI	std
Breast Cancer	1.008	1.018	0.006	1.019	1.027	0.057	1.011	1.021	0.014
Iris	1.5738	1.625	0.035	1.531	1.601	0.055	1.526	1.671	0.115
wine	1.48	1.582	0.052	1.493	1.617	0.113	1.537	1.698	0.125

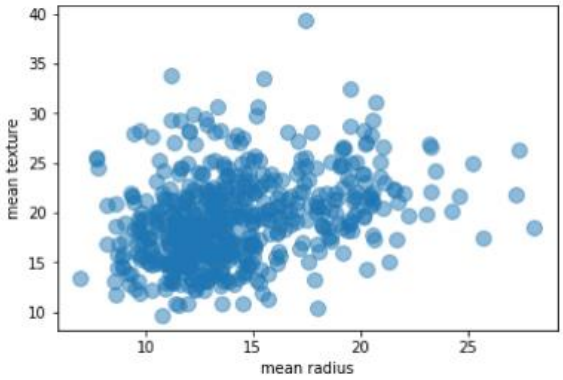

Fig 5.10 (a)

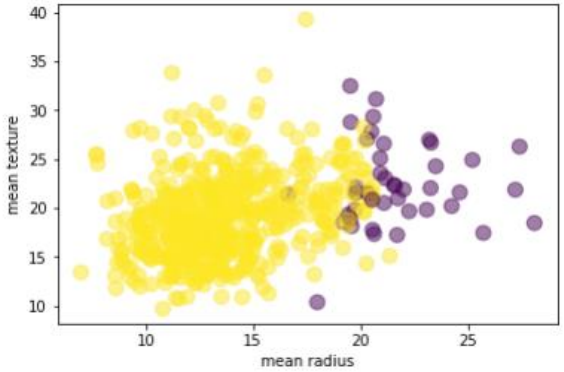

Fig 5.10 (b)

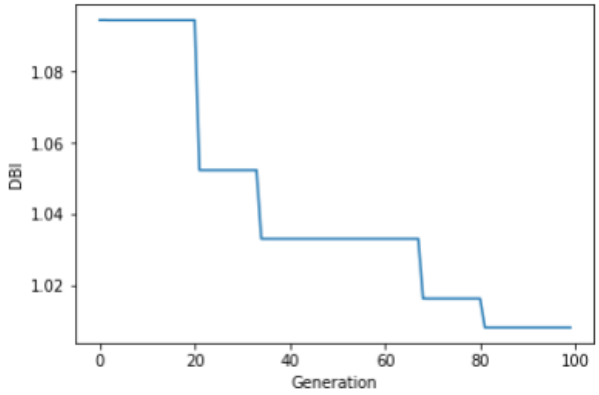

Fig 5.10 (c)

Fig 5.10: Visualization of Breast Cancer data with q = 0.5 (a) actual data without clustering (b) clustered data using CSCA with 10 populations over five simulations with abandon frequency 0.5 (c) variation of DBI with generation.

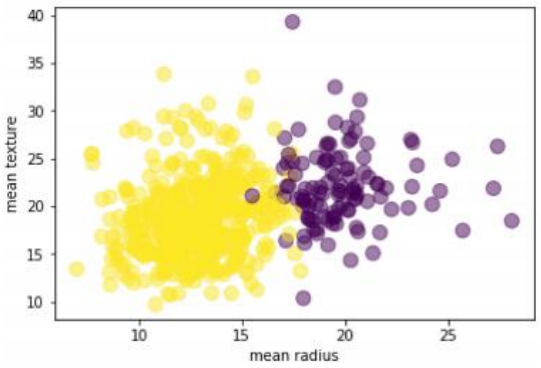

Fig 5.11 (a)

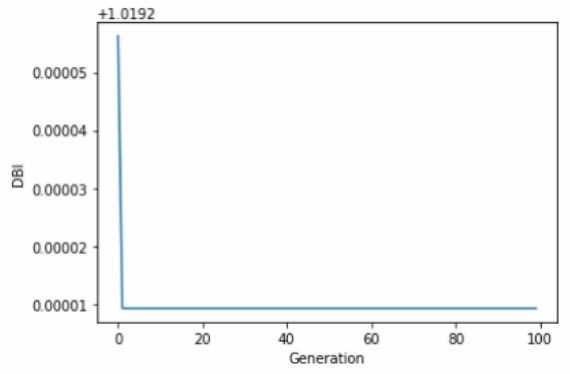

Fig 5.11 (b)

Fig 5.11: Visualization of Breast Cancer data with q = 0.8 (a) clustered data using CSCA with 10 population over five simulations with abandon frequency 0.5 (b) variation of DBI with generation.

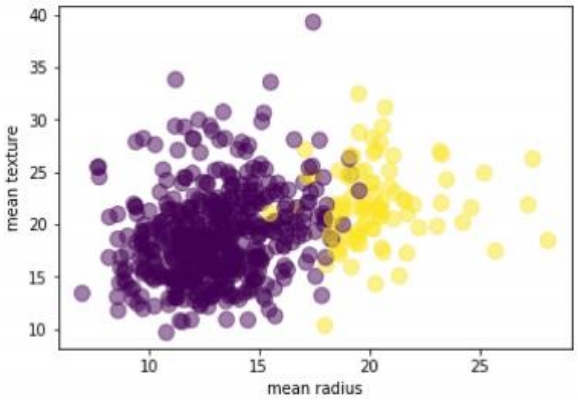

Fig 5.12 (a)

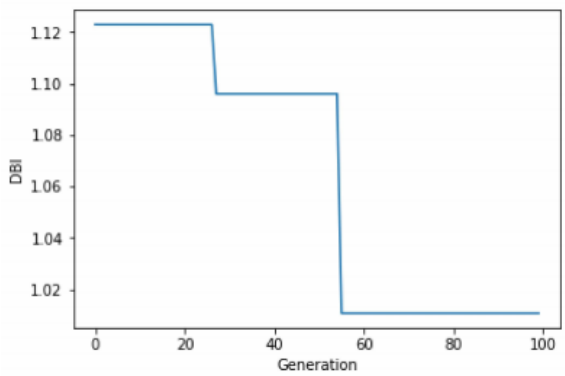

Fig 5.12 (b)

Fig 5.12: Visualization of Breast Cancer data with q = 0.9 (a) clustered data using CSCA with 10 population over five simulations with abandon frequency 0.5 (b) variation of DBI with generation.

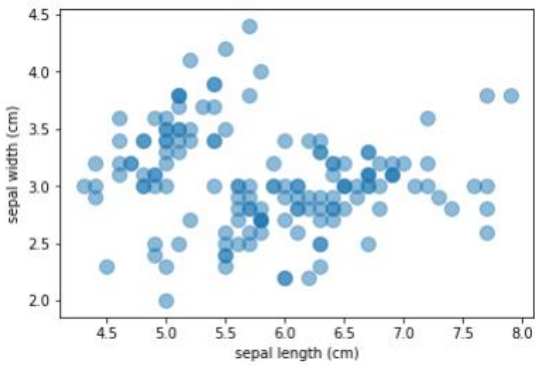

Fig 5.13 (a)

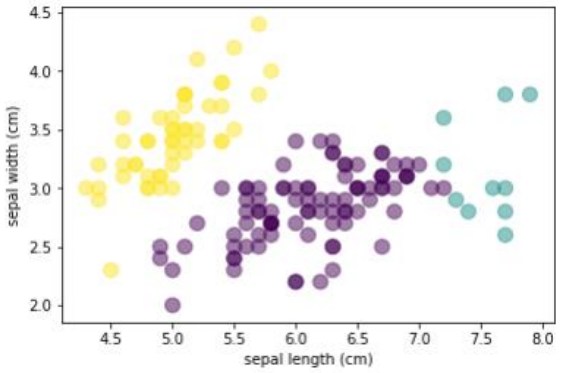

Fig 5.13 (b)

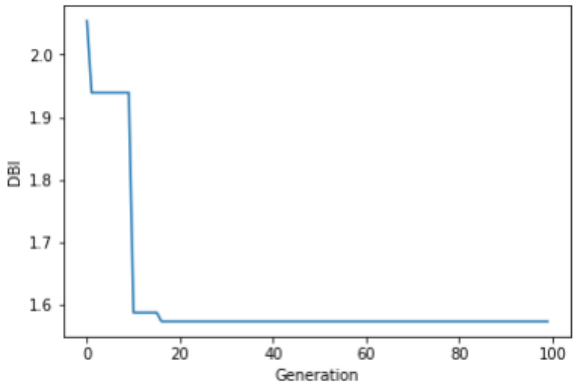

Fig 5.13 (c)

Fig 5.13: Visualization of Iris data with q = 0.5 (a) actual data without clustering (b) clustered data using CSCA with 10 populations over five simulations with abandon frequency 0.5 (c) variation of DBI with generation.

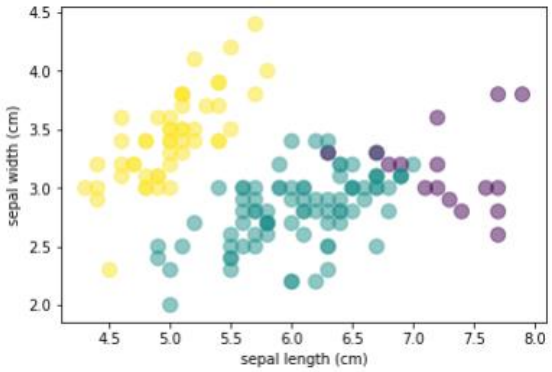

Fig 5.14 (a)

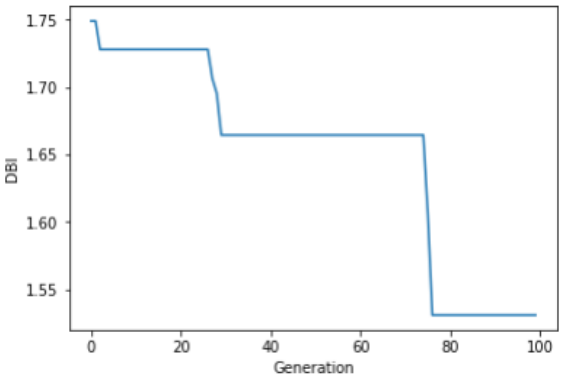

Fig 5.14 (b)

Fig 5.14: Visualization of Iris data with q = 0.8 (a) clustered data using CSCA with 10 population over five simulations with abandon frequency 0.5 (b) variation of DBI with generation.

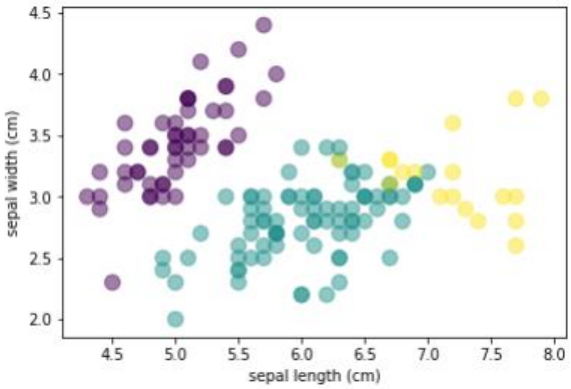

Fig 5.15 (a)

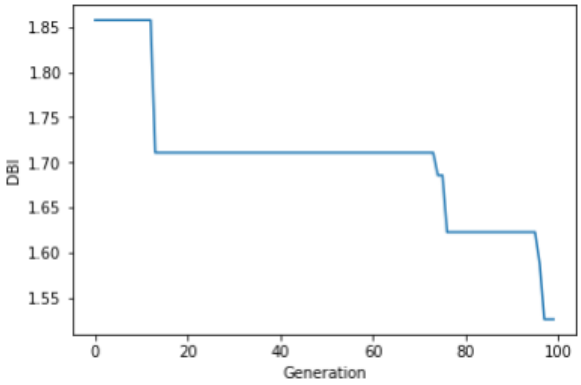

Fig 5.15 (b)

Fig 5.15: Visualization of Iris data with q = 0.9 (a) clustered data using CSCA with 10 population over five simulations with abandon frequency 0.5 (b) variation of DBI with generation.

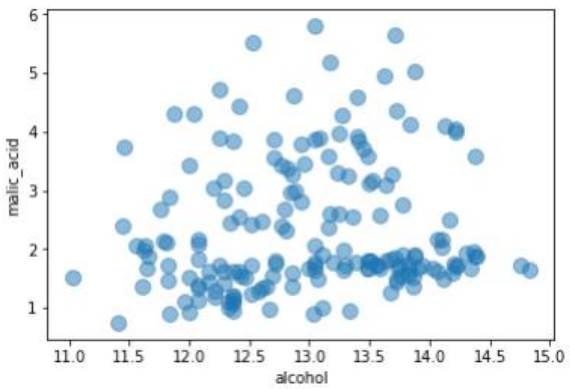

Fig 5.16 (a)

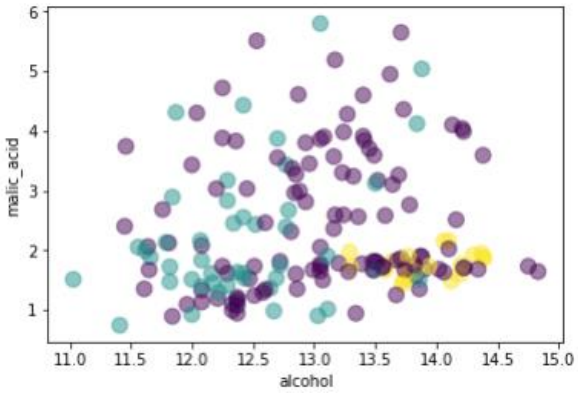

Fig 5.16 (b)

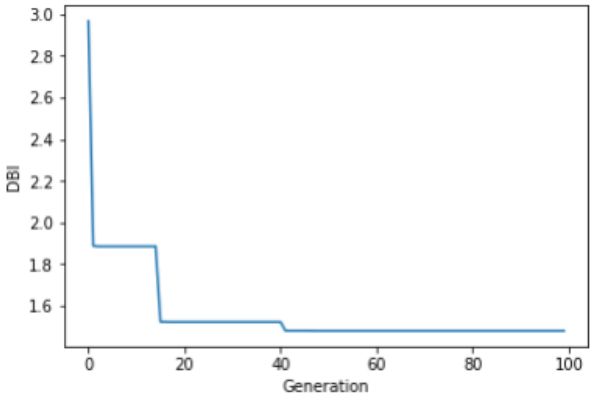

Fig 5.16 (b)

Fig 5.16: Visualization of Wine data with q = 0.5 (a) actual data without clustering (b) clustered data using CSCA with 10 populations over five simulations with abandon frequency 0.5 (c) variation of DBI with generation.

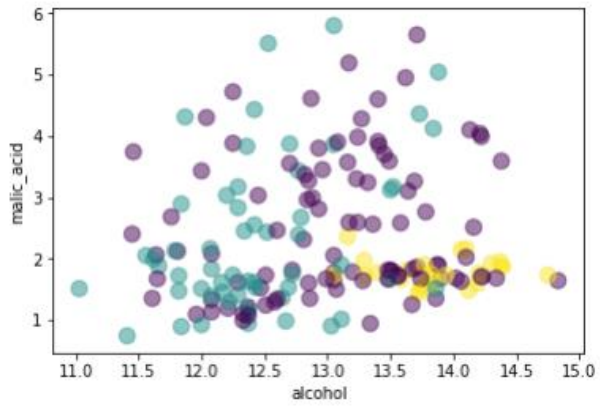

Fig 5.17 (a)

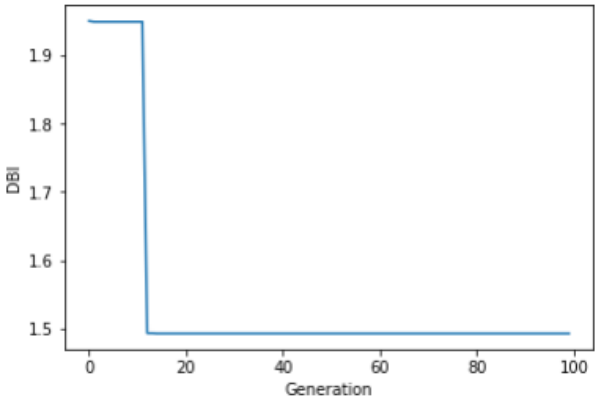

Fig 5.17 (b)

Fig 5.15: Visualization of Iris data with q = 0.8 (a) clustered data using CSCA with 10 population over five simulations with abandon frequency 0.5 (b) variation of DBI with generation.

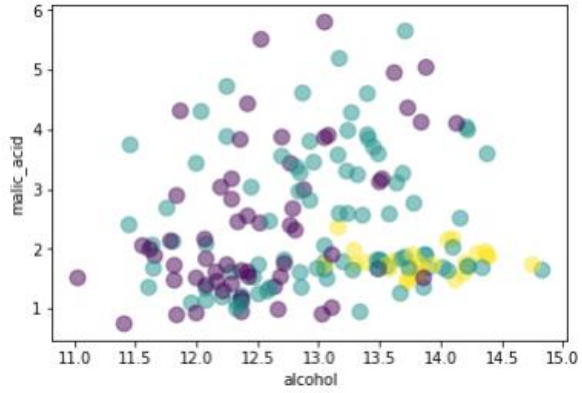

Fig 5.18 (a)

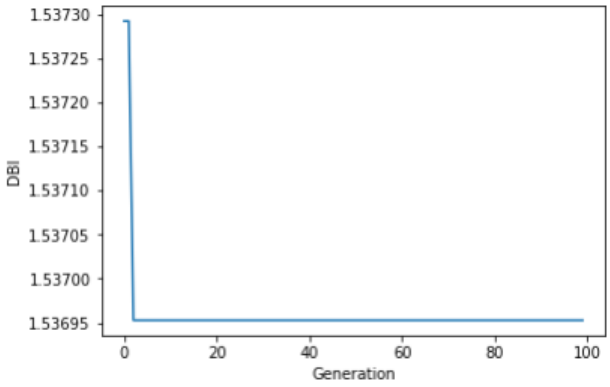

Fig 5.18 (b)

Fig 5.18: Visualization of Wine data with q = 0.9 (a) clustered data using CSCA with 10 population over five simulations with abandon frequency 0.5 (b) variation of DBI with generation.

5.4. Analysis

From above results it has been noted that *CSA* takes the less amount of time among all of them and produces fairly optimum results. Unlike *Cuckoo Search BCOA* takes the longest time per simulation among all but it is noted that *BCOA* is the one to produce most of the accurate results among all and looking towards the *standard deviation* it shows that it is the most stable algorithm among all, though it has been seen that for *bowel shaped* function and also for *Ackley* which has many local minima *BCOA* has failed to produce desired output. Other two Algorithms has shown fairly satisfactory performance. Now in case of clustering *CSA* is incorporated in data clustering. It is seen that due to the variation of parameter *q* results are varying where *q* indicates a fraction constant which determines if a particular *cuckoo* will follow the best one or move randomly. It is noticed clustering quality is proportional to *q.*

6. Conclusion

During the course of study of four bio-inspired algorithms, *Bat Algorithm (BA), Bee Colony Optimization Algorithm (BCOA), Cuckoo Search Algorithm (CSA)* and *Firefly Algorithm (FA)* several observations are noted when they are applied on nine benchmark optimization functions. The results taken during the experiments reveal few characteristics of their performance. From the results it can be concluded that *CSA* is the fastest to optimize the functions. *BCOA* is the slowest one although *BCOA* shows better optimization, stability and reliability among all other. *BA* performance is average among them and *FA* is quick to perform but it has reliability issue.

6.1. Limitations

The main drawbacks noticed during the experiments are *BCOA* is significantly time consuming though it also depends on optimization functions. Also *BCOA* is failed to produce desired results for few of the optimization functions but specifically *Bowl* shaped functions. On the other hand *FA* does not show good reliability meaning the whole population do not follow best one. Only few of the population follows the best one as a result visualization shows a huge gap between average of optimum fitness and average to average fitness of population.

6.2. Future Scope

In future this work can be extended by varying the several parameters and experimenting with several combinations. In this thesis few limitations are noticed which are subject to the future scope of experiment to verify if they persists or not with more benchmark functions with different set of parameter values. Also the usage of different algorithms in clustering are subject to the future scope.

References

[1] Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, Dusan Fister, "A Brief Review of Nature-Inspired Algorithms for Optimization", ELEKTROTEHNISKI VESTNIK 80(3): 1–7, 2013 ENGLISH EDITION

[2] C. M. Bishop F.R.Eng, "Pattern Reorganization and Machine Learning", springer, March, 2008.

[3] N. Zhang, X. Zhang, J. Lu, and T. Yahagi, "Unsupervised neural network to nonlinear blind separation," in NSIP 2005. Abstracts. IEEE-Eurasip Nonlinear Signal and Image Processing, 2005, p. 39.

[4] L. Shen and R. S.- Reconstruction, "Semi-supervised Learning for Multilabel Classification," Cs229.Stanford.Edu.

[5] R. Cordeiro De Amorim and B. Mirkin, "Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering," Pattern Recognitazation ., 2012 , vol. 45, no. 3, pp. 1061–1075.

[6] Garima, H. Gulati, and P. K. Singh, "Clustering techniques in data mining: A comparison," in 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 410–415.

[7] G. Ahalya and H. M. Pandey, "Data clustering approaches survey and analysis," in 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015, pp. 532–537.

[8] Binitha S, S Siva Sathya., "A Survey of Bio inspired Optimization Algorithms", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012

[9] Iago Augusto Carvalho, Daniel G. da Rocha, João Gabriel Rocha Silva, Vinícus da Fonseca Vieira, Carolina Ribeiro Xavier, "Study of Parameter Sensitivity on Bat Algorithm", An Ensemble Similarity Model for Short Text Retrieval (pp.494-508)

[10] Xin-She Yang, “A New Metaheuristic Bat-Inspired Algorithm”, Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).

[11] Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri (Eds.), “Innovations in Swarm Intelligence”, Published on 2009

[12] Dervis Karaboga, Bahriye Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm”, Published online: 13 April 2007 Springer Science+Business Media B.V. 2007

[13] Tatjana DAVIDOVIC´, Dusan TEODOROVIC´, Milica SELMIC´, Yugoslav, “Bee Colony Optimization Part I: The Algorithm Overview”, Journal of Operations Research 25 (2015), Number 1, 33–56 DOI: 10.2298/YJOR131011017D

[14] Xin-She Yang, Suash Deb, “Cuckoo Search via L´evy Flights”, in: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), December 2009, India. IEEE Publications, USA, pp. 210-214 (2009).

[15] Amir Hossein Gandomi, Xin-She Yang, Amir Hossein Alavi, “Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems”, Published online: 29 July 2011 Springer-Verlag London Limited 2011

[16] Sanket Kamat, Asha Gowda Karegowda, “A Brief Survey on Cuckoo Search Applications”, International Journal of Innovative Research in Computer and Communication Engineering Vol.2, Special Issue 2, May 2014

[17] Azam Amin Abshouri, Mohammad Reza Meybodi, Alireza Bakhtiary, “New Firefly Algorithm based On Multi swarm & Learning Automata in Dynamic Environments”, 978-1-4577- 0174-0/11/$26.00 ©2011 IEEE

[18] Xin-She Yang and Xingshi He, (2013). "Firefly Algorithm: Recent Advances and Applications", Int. J. Swarm Intelligence, Vol. 1, No. 1, pp. 36–50. DOI: 10.1504/IJSI.2013.055801

[19] Shubhendu Kumar Sarangi, Rutuparna Panda, Sabnam Priyadarshini , Archana Sarangi, "A New Modified Firefly Algorithm for Function Optimization", International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016

[20] Xiangbo Qi, Sihan Zhu, Hao Zhang, "A Hybrid Firefly Algorithm", 978-1-4673-8979-2/17/$31.00 ©2017 IEEE

[21] Nur Farahlina Johari, Azlan Mohd Zain, Noorfa Haszlinna Mustaffa, Amirmudin Udin, "Firefly Algorithm for Optimization Problem", Applied Mechanics and Materials Vol. 421 (2013) pp 512-517 © (2013) Trans Tech Publications, Switzerland

[22] X. S. Yang, "Nature-Inspired Metaheuristic Algorithms", Luniver Press, UK, (2008).

[23] Apostolopoulos T. and Vlachos A., "Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem, International Journal of Combinatorics", Volume 2011, Article ID 523806.

[24] A. Chatterjee, G. K. Mahanti, and A. Chatterjee, "Design of a fully digital controlled reconfigurable switched beam conconcentric ring array antenna using firefly and particle swarm optimisation algorithm", Progress in Elelectromagnetic Research B., 36, 113-131 (2012).

[25] T. Hassanzadeh, H. Vojodi and A. M. E. Moghadam, "An image segmentation approach based on maximum variance intra-cluster method and firefly algorithm", in: Proc. of 7th Int. Conf. on Natural Computation (ICNC2011), pp. 1817-1821 (2011).

[26] M.-H. Horng, Y.-X. Lee, M.-C. Lee and R.-J. Liou, "Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis", in: Theory and New Applications of Swarm Intelligence (Edited by R. Parpinelli and H. S. Lopes), pp. 115- 132 (2012).

[27] M.-H. Horng, “Vector quantization using the firefly algorithm for image compression, Expert Systems with Applications”, 39, pp. 1078-1091 (2012).

[28] Sayadi M. K., Ramezanian R. and Ghaffari-Nasab N., “A discrete firefly meta-heuristic with local search for make-span minimization in permutation flow shop scheduling problems”, Int. J. of Industrial Engineering Computations, 1, 1–10.

[29] X. S. Yang, Firefly algorithm, “Stochastic test functions and design optimization”, Int. J. BioInspired Computation, 2(2), 78-84 (2010).

[30] H. Banati and M. Bajaj, “Firefly based feature selection approach”, Int. J. Computer Science Issues, 8(2), 473-480 (2011).

[31] Sina K. Azad, Saeid K. Azad, “Optimum Design of Structures Using an Improved Firefly Algorithm”, International Journal of Optimisation in Civil Engineering, 1(2), 327-340(2011).

[32] B. Basu and G. K. Mahanti, “Firefly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna”, Progress in Electromagnetic Research B., 32, 169-190 (2011).

[33] E. A. Calvillo, A. Padilla, J. Muñoz, J. Ponce, and J. T. Fernandez, “Searching research papers using clustering and text mining,” in CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing, 2013, pp. 78–81.

[34] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li, “Automated variable weighting in k-means type clustering,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 5, pp. 657–668, 2005

[35] J. MACQUEEN, “SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS”, Supported by Western Management Science Institute under a grant from the Ford Foundation, and by the Office of Naval Research under Contract No. 233(75), Task No. 047-041.

[36] Kaufman, L. and Rousseeuw, P.J. (1987), "Clustering by means of Medoids, in Statistical Data Analysis Based on the L1 - Norm and Related Methods", edited by Y. Dodge, North-Holland, 405– 416.

[37] David E. Goldberg, "Genetic Algorithms in search, Optimization and Machine Learning", Pearson, Fifteenth Impression.

[38] Y. Endo, S. Ishida, N. Kinoshita, and Y. Hamasuna, "On various types of controlled-sized clustering based on optimization," in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2017, pp. 1–6.

[39] Paul D. McNicholas, "Model-Based Clustering", Journal of Classification 33:331-373 (2016)

[40] C. C. Aggarwal and C. Zhai, "Chapter 4 - A SURVEY OF TEXT CLUSTERING ALGORITHMS," Min. Text Data, pp. 77–128, 2012.

[41] O.A. Mohamed Jafar, R. Sivakumar, "A Study of Bio-inspired Algorithm to Data Clustering using Different Distance Measures", International Journal of Computer Applications (0975 – 8887) Volume 66– No.12, March 2013

[42] Samiksha Goel, Arpita Sharma, Punam Bedi, "Cuckoo Search Clustering Algorithm: A novel strategy of biomimicry", World Congress on Information and Communication Technologies, IEEE, 2011

[43] Susmita Ghosh, "Genetic Algorithms: Some New Operations and Applications To Connectionist Object Extraction"

[44] Virtual Library of Simulation Experiments: Test Functions and Datasets, url: https://www.sfu.ca/~ssurjano/optimization.html