Webology (ISSN: 1735-188X)
Volume 19, Number 2, 2022

# Improvisation Of The Nature-Inspired Meta-Heuristic Algorithm For Optimized Clustering

**Pradeep Kumar D , Sowmya BJ , Ganeshayya S Sidaganti , Anita Kanavalli , S Anusha , Preetham, Shivani Rao , Shilpa Kini**

Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, MSRIT Post, Bangalore 560054

**ABSTRACT**

The Hunger Game Search (HGS) algorithm is very effective when it comes to finding the best answer. This research article proposes a Lévy Flight and Cauchy Flight enhancement to the hunger game search (HGS) method, which makes the HGS quicker and more resilient while avoiding premature convergence. It also helps to promote population variety, which protects against early convergence and improves the capacity to leap out of local optima. This technique aids in achieving a better balance between HGS exploration and exploitation. Rapid convergence and high accuracy characterize the proposed method, and it can effectively eliminate a local optimum. The Usability of HGS was further confirmed by comparing it with a variety of conventional and exceptional algorithms with 28 known optimization functions, as well as the IEEE CEC 2014 benchmark test suite. The usability of this method is further exemplified by its application to a variety of technical problems. In addition, it showed the proposed algorithm with higher identification costs of the simulation research results opposite other algorithms. All findings support the effectiveness of the targeted optimizer and exemplify the superiority of the HSA algorithm.

## 1. INTRODUCTION

The goal of designing algorithms is to solve technical issues more efficiently. As the world becomes more industrialized, engineering issues get more complicated to solve: rising dimensions, variables, time complexity, and space complexity. As a result, efficient solutions for solving comparable issues are necessary. Despite its superior performance when compared to other common approaches, the salp swarm algorithm has drawbacks, such as low accuracy, delayed convergence, and being locked in a local optimum as a result of population variety.

The use of Lévy flight and Cauchy flight is useful to better balance the exploration and exploitation of algorithms, and it is valuable to avoid local optimizations. The HGS algorithm is examined with 28 benchmark functions and the IEEE CEC 2014 benchmark test suite.

The remainder of this paper is composed as follows: Section 2 outlines the original HGS, Levy Flight, and Cauchy flight information. Section 3 demonstrates the concerned experiments' results and qualitative analysis of the aforementioned performance of HGS and compares it with conventional and advanced algorithms on 28 benchmark functions, the IEEE CEC 2014 benchmark test suite, and its application to engineering problems.

## 2. LITERATURE SURVEY

The use of meta-heuristics is widespread for optimization in both scientific and industrial problems. Many meta-heuristic algorithms are used for solving different optimization problems efficiently. From these meta-heuristic algorithms, nature-inspired optimization algorithms are widely used to find better solutions and their best results, few such algorithms are Salp Swarm Algorithm (SSA), Grey Wolf Optimizer, Cuckoo Search algorithm, etc. Various works have investigated the use of Lévy flight in Swarm Intelligence algorithms. We have studied some of these papers and made use of our research in implementing our approach.

Yang and Deb proposed the Cuckoo Search algorithm (CS), based on Lévy flight, in 2009[]. The search rationale is based on the behavior of a cuckoo species, according to their research. This method creates n random solutions, referred to as "nests." Then, using the Lévy distribution, a new solution representing a cuckoo egg is created and compared to the solution of a random nest: if the new solution is better than the old one, the previous solution is replaced. Finally, fresh random solutions replace a proportion of the worst solutions, and a new solution is created using the Lévy distribution, repeating the process until a stopping condition is reached. Yang introduces the algorithm of Lévy Flying Firefly (LFA) 2010 [22], which is modeled by the conduct of fireflies, which attract companions on the flashing lights. In this method, each alternative response is represented by a firefly with a light intensity proportional to its ability. The brightness of a solution is directly proportional to its attractiveness, while its distance from another solution is inversely proportional. According to a selected value of the Lévy distribution, less bright accidents migrate to a brighter luminous vacancy in proportional steps to its functionality. Yang proposed the flowers pollination algorithm (FPA) in 2012 [4], which takes care of each solution as a pollen particle that moves in the search space according to two separate position update rules: local pollination and global pollination. In each phase, one of the two rules is chosen: if local pollination is chosen, the particle moves in a limited neighborhood, and etc. is multiplied by a random number of uniform distribution (0, 1); If the global pollination is chosen, the particle moves towards the best and a random number drawn from the Lévy distribution is multiplied by step-size. The Lévy flight PSO was introduced in 2015 by Hariya [6], which in the same way as the standard PSO with the exception is that the inertial coefficient is determined by

the Lévy Distribution. Each particle has a position and speed at each iteration, and the speed is updated on the basis of the sum of (1) of the current speed multiplied by any number of Lévy distribution (2) of the vector for personal multiplied A random number scanned from a uniform distribution, and (3) a random integer taken from a uniform distribution multiplied by the vector pointing to the global best. In 2016, Sharma et al. proposed the Artificial Bee Colony of Lévy (LFABC), which also uses a Lévy theft to calculate the size of the step for mobile solutions. In this case, however, each solution is a source of food that is handled by three bees groups: (1) Work bees, which modify food sources based on a personal experience and the ability of the new source of food; (2) specific bees, which receive information from the bees of work and seek better solutions based on this information; and (3) scout bees, which replace sources of food abandoned by Rivas.

In 2016, Kalantzis et al. proposed a GPU-based variation of LFA for tackling a restricted optimization issue relating to intensity-modulated radiation therapy treatment planning. In 2016, Majumder and Laha proposed a solution for addressing a 2-machine robotic cell scheduling issue using a discrete version of CS with Lévy flight. In 2017[27], Nguyen and Vo presented a multiobjective version of CS with Lévy flight. In 2018, Tighzert et al. proposed a ‘‘compact" form of LFA, which relies on a probabilistic distribution model rather than a swarm of fireflies.

In 2019[14], Pandey et al. offer two intriguing applications of CS with Lévy flight, in which the method is utilized to optimize power consumption in smartphone screens. In 2019[13], Wang et al. present a system for tackling a specific scheduling problem: the dynamic allocation of berths in seaports, in which Lévy flight is coupled with three ad-hoc local search techniques. The step-sizes created by the method are rounded to integer numbers to address the combinatorial character of the issue, which is a unique feature of this study.

The following research publications were gathered, researched, and evaluated as part of our survey to highlight the benefits and drawbacks, robustness, and weaknesses of various nature-inspired algorithms. The majority of these articles described variants of these swarm intelligence algorithms that had been modified with Levy flight, where the proposed versions of the algorithms helped improve the performance of the original algorithms to solve various optimization problems, such as binary, modifications, hybridizations, chaotic, multi-objective, and parameter-less SSA. HGS has certain similarities to other optimization methods (such as CSA, BA, and FA), such as simplicity, speed in searching, and ease of hybridization. Based on our findings, HGS may increase its exploration pace while retaining its rapid convergence rate by using Levy Flight and Cauchy Flight distributions.

## 3. Design and Implementation

### HUNGER SEARCH ALGORITHM, LEVY FLIGHT, AND CAUCHY FLIGHT

Animals follow sensory knowledge corresponding to several computational principles. These rules not only play as the cornerstone for decision-making and choices when interacting with their surroundings (as part of the environment) but also uphold the expansion of cognitive structures. Hunger is one of the most fascinating homeostatic drivers and causes of animal behavior, decisions, and activities. The animal monarchy has a universal behavioral choice and activity selection, which is the fundamental law of goal-directed behavior observed in nature.

Hunger is a potent incentive for activity, learning, and foraging in any animal, according to neuroscientists, and it serves as a force toward changing the living state to a more stable one. Because social animals naturally cooperate, they may escape predators and investigate food sources such as other animals and plants, boosting their chances of survival. This is how evolution works: stronger animals can find food more easily and have a better chance of survival than weaker animals. This is comparable to a hunger game in nature.

Hunger is a sign of not eating for an extended period of time, and the stronger the hunger, the stronger the need to eat, and the harder the body will have to fight to locate food in a short period of time until it runs out and starvation or death happens. As a result, the game emphasizes logical judgments and sensible species migration.

**Mathematical model:**

The proposed mathematical model, as well as the suggested HGS technique, are discussed in depth in this section. Please bear in mind that our capacity to build a mathematical model based on hunger-driven activities and behavioral choices is limited and that it must be as simple as possible while yet producing the greatest outcomes.

**Approach food:**

During foraging, social animals usually interact with one another, but it's impossible to rule out the chance that a few individuals do not participate. The following game instructions describe the fundamental equation of the HGS algorithm for cooperative communication and foraging behavior of individuals:

$$\overrightarrow{Y(t+1)} = \begin{cases} \text{Game}_1: \overrightarrow{Y(t)} \cdot (1 + \text{random}(1)), & a_1 < 1 \\ \text{Game}_2: \vec{M}_1 \cdot \overrightarrow{Y_b} + \vec{A} \cdot \vec{M}_2 \cdot \left| \overrightarrow{Y_b} - \overrightarrow{Y(t)} \right|, & a_1 > 1,\ a_2 > P \\ \text{Game}_3: \vec{M}_1 \cdot \overrightarrow{Y_b} + \vec{A} \cdot \vec{M}_2 \cdot \left| \overrightarrow{Y_b} - \overrightarrow{Y(t)} \right|, & a_1 > 1,\ a_2 < P \end{cases}$$

Where $\vec{A}$is in the range of [ - a, a];
(3.1)
$\mathbf{a_1}$, $\mathbf{a_2}$ represent two random numbers in the range of [0,1];

t is the number of iterations presently in progress; random(1) is a random integer with a normal distribution.

$\vec{M}_1$ and $\vec{M}_2$are the hunger weights, which are predicated on the fact that hunger-driven signals exist.

$\overrightarrow{Y_b}$represents the location of the best individual in this iteration.

Each individual's location is represented by $\overrightarrow{Y(t)}$;
In the parameter setting experiment, the value of l will be addressed, and it is a parameter that is designed to improve the algorithm.

An agent's hungry and random search for food in their present location is shown by $\overrightarrow{Y(t)} \cdot (1 +random(1))$;

To account for the influence of hunger on the range of activity, $\left|\overrightarrow{Y_b} - \overrightarrow{Y(t)}\right|$models the current individual's range of activity in the present time and multiplies it by $\vec{M}_2$.

Because an individual will stop searching once it is no longer hungry, $\vec{A}$ has been introduced as a ranging controller to limit the range of activity, with $\vec{A}$'s range gradually falling to 0. Depending on $\vec{M}_1 \cdot \overrightarrow{Y_b}$, increasing or reducing the spectrum of activity Once arriving at the food site, the current individual is told by their peers and then seeks food in the current position after the food has been acquired. $\vec{M}_1$ is offered as a misinterpretation of the current condition of reality. The P formula, which is used to regulate variance across all sites, is as follows:

$P = sech(|F(i) – FB|)$
(3.2)
Where $i \in 1,2,\ldots,n$,$F(i)$denotes each individual's fitness value;
BF denotes the best fitness achieved so far in the current iteration process;
A hyperbolic function is sech.
$\left(sech(y) = \frac{2}{e^{y} + e^{-y}}\right)$
The formula for $\vec{A}$ is as follows:
$\vec{A} = 2 \times shrink \times random \times shrink$
(3.3)

$$\text{shrink} = 2 \times \left(1 - \frac{t}{T}\right)$$

(3.4)

where random is a random number in the range of [0,1]; and T is the maximum number of iterations

Based on the rule in Eq. 3.1, the process of searching and logic of HGS in the spaces;

The search directions may be divided into two groups based on the classification of source points.

**Search based on $\vec{Y}$:** The first game instruction depicts a self-sufficient individual who lacks a collaborative attitude, isn't interested in cooperating, and just wants to hunt for food.

**Search based on $\overrightarrow{Y_b}$:** The variables $\vec{A}$, $\vec{M}_1$, and $\vec{M}_2$ are all strongly connected to the second game instruction. The individual's location inside the feature space might be evolved by refining these three criteria. This technique replicates the collaboration of several creatures searching for food.

Individuals can use the rules or principles in Eq. (3.1) to examine possible sites near the optimal solution and locations far away from the perfect solution, ensuring that all areas within the boundaries of solution space are searched to some extent (diversification). The same technique may be applied in a high-dimensional search space.

**Hunger role:**

In this section, a mathematical model is used to emulate the hunger characteristics of individuals in search.

The formula of $\vec{M}_1$ in Eq. (3.1) is as follows:

$$\overrightarrow{M_1(i)} = \{\text{hung}(i) \cdot \frac{N}{\text{SHung}} \times a_4,\ a_3 < 1$$

$$1 a_3 > 1\}$$

(3.5)

The formula of $\vec{M}_2$ in Eq. (3.1) is shown as follows:

$$\overrightarrow{M_2(i)} = (1 - \text{expexp}(-|\text{hung}(i) - \text{SHung}|)) \times a_5 \times 2$$

(3.6)

where hungry denote each individual's hunger and L is the total number of individuals;

SHungry is the summation of all people's hunger sensations, or sum(hungry);

$a_3$, $a_4$ , and $a_5$ are random numbers in the range of [0, 1].

The formula for hung(i) is provided below:

$$\text{Hung(i)} = \begin{cases} 0, & \text{AllFit(i)} == \text{FB} \\ \text{hng(i)} + \text{H}, & \text{AllFit(i)} != \text{FB} \end{cases} \quad (3.7)$$

In the present edition, AllFit(i) maintains track of each individual's fitness. In each cycle, the best individual's hunger was set to zero. For other persons, a new hunger(HN) is introduced based on the prior hunger.

As a result, we may conclude that the equivalent HN of various people will differ.

The following is the formula for HN:

$$\text{HT} = \frac{\text{F(i)} - \text{FB}}{\text{FW} - \text{FB}} \times a_{6,} \times (\text{BU} - \text{BL}) \quad (3.8)$$

$$\text{HN} = \begin{cases} \text{HL} \times (1 + \text{r}), & \text{HT} < \text{HL} \\ \text{HT}, & \text{HT} \geq \text{HL} \end{cases} \quad (3.9)$$

where $a_3$is a random number in the range of [0,1];

F(i) is the fitness value of each individual;

The best fitness attained so far in the current iteration phase is denoted by FB.

FW is for the lowest fitness identified in the current iteration process (so far); BU and BL stand for the upper and lower limits of the feature space, respectively.

A lower constraint, HL constrains the sense of hunger. We modify the upper and lower limits of hunger to enhance the algorithm's performance, with the value of HL being given in the parameter setting experiment.

$\vec{M}_1$ and $\vec{M}_2$are simulated because hunger can have both positive and negative impacts on the spectrum of activities.

The difference between BU and BU is utilized in Eq. (3.8) to indicate the individual’s highest hunger in various situations.

F(i) -FB indicates the quantity of food required for an individual to no longer be hungry; the individual's hunger will fluctuate with each iteration.

In the current method, FW-FD gives an individual's entire foraging capability;

Hunger ratio is denoted by $\frac{F(i) - FB}{FW - FB}$;

$a_6 \times 2$ shows whether environmental variables have a favorable or negative impact on hunger.

While the suggested HGS algorithm may be able to predict social animal commonness, it still has a long way to go. The algorithm can be improved, for example, by mapping it to a real object and incorporating the actual organism's unique features. It can also be enhanced by including additional mechanisms. To enhance scalability, we simplify the method as much as feasible.

Algorithm 1 shows the pseudo-code for the proposed Hunger Games Search. The flowchart is also shown in Figure 3.

**Algorithm**

Initialize the parameters L, T, l, D, SHungry
Initialize the positions of Individuals Yi (i = 1, 2,…. L)
**While** (t ≤ T)
    Calculate the fitness of all Individuals
    Update FB, FW, $\overrightarrow{Y_b}$, BI
    Calculate the Hungry by **Eq**. (3.7)
    Calculate the $\vec{M}_1$ by **Eq**. (3.5)
    Calculate the $\vec{M}_2$ by **Eq**. (3.6)
    **For** each Individual
        Calculate P by **Eq**. (3.2)
        Update A by **Eq**. (3.3)
        Update positions by **Eq**. (3.1)
    **End For**
    t=t+1
**End While**
**Return** FB, $\overrightarrow{Y_b}$

**LEVY FLIGHT**

The Lévy Flight trajectory was initially proposed by Lévy and then improved upon by Benoit Mandelbrot. Lévy flight is a type of random walk whe-re the steps are chosen at random from a Lévy distribution.

Many studies have revealed that when many animals and insects fly, they display the typical features of Lévy flight. Lévy's flying behavior is now being used for optimization and optimum search, and the first findings show that it has a huge amount of potential.

In their flight behavior, several flying creatures show Lévy-like characteristics. Lévy flight is a form of random walk in which the step size probability density function (PDF) is strongly tailed, which means that a particle traveling according to Lévy flight takes an occasional larger step and a lot of smaller steps. The particle moves locally at first, taking numerous little steps, then makes a big movement, and then moves locally again.
The Lévy flight PDF can be defined as follows from a mathematical standpoint:
Distributive Levy,

$$D = \frac{v}{|u|^{\gamma\beta}} ,$$

(3.10)
Where the power-law index is denoted by β and V, U is a random number from Gaussian distribution N(0, 1) and N(0, $\sigma^2$) respectively. Here $\sigma$is the standard deviation given by:

$$\sigma_v = \left[\frac{\tau(1+\beta)\cdot \sin(\pi\beta/2)}{\tau[(1+\beta)/2]\beta\cdot 2^{\beta-1/2}}\right]^{1/\beta}$$

$$\sigma_u = 1 ,$$

(3.11)
where gamma function is denoted by $\Gamma$.

## CAUCHY FLIGHT

The Cauchy distribution (also known as the Lorentz distribution) is a continuous probability distribution with Xo and y parameters. The location parameter is a positive real integer, and a scaling parameter is a number that describes the form of the distribution. A distribution with a large breadth and a high peak has a shape with a lower value in y. The greater y value, on the other hand, indicates a form with a large breadth and a lower apex. The distribution's probability density function is

$$f(y, y_0, x) = \frac{1}{\pi x\left[1 + \left(\frac{y - y_0}{x}\right)^2\right]}$$

(3.12)

The Cauchy distribution's cumulative distribution function is given as:

$$f(y, y_0, x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{y - y_0}{x}\right)$$

(3.13)

The Hunger Game Search algorithm quickly solves the problem of small-dimensional unimodal optimization. Nonetheless, solutions got by HGS while handling multi-modal optimization problems and high-dimensional are not very beneficial. Therefore, this paper recommends an enhanced version of the HGS algorithm with the purpose of improving the exploration, exploitation, convergence of the HGS, and local optimal optima avoidance.

Levy flight is important in the diversification of search agents because it ensures that the algorithm can efficiently investigate the search area and avoid local minima. Recognizing this means that in the HGS algorithm, flying aids in achieving a more favorable exchange between exploitation and exploration. Despite the fact that Lévy's mutant step size makes traversing the search area easier, a superior exploration operator is still required. Instead of the Lévy mutation operator, a Cauchy-based mutation operator is utilized to generate step size in this situation. The Cauchy-based mutation operator is used to generate a Cauchy distributed random number. This random number is then used to produce a new solution in the global search equation.

Thus, the flight is employed to improve the original HGS algorithm, which can be formulated mathematically:

y(t)=Levy(y(t))
(4.1)
y(t)=Cauchy(y(t))
(4.2)

After Eq (3.1), Eq(4.1), and Eq(4.2) are added to improve the algorithm.

Search potency of HGS algorithms is substantially enhanced by flight, and also helps to avoid the local minima in the obtained results. Not only the intensification, which accomplishes the search for best-performing aim solution, a well determining the best performing aim solution, but also the diversification, which provides the assurance that algorithm advances the global ability of HGS is improved in our approach. According to the findings, flight outcomes are beneficial and noteworthy for unimodal and multimodal benchmark functions, among other things. The modified HGS surpasses the original HGS due to its unique characteristics. Various benchmark functions are used to verify the performance of the updated method in dealing with optimization challenges. The HGS's essential phases are explained in Algorithm 2 below, and the HGS flowchart is shown in Fig1.

**ALGORITHM:**
Set the parameters L, T, l, D, and SHungry to their default values.
While (t=T), initialise the locations of Individuals Yi I = 1, 2,.... L).
Calculate each individual’s fitness level.
FB, FW, Yb, and BI should all be updated.

Use Eq. to calculate the Hungry (3.7)
Calculate the M1 using Eq. (3.5) and the M2 using Eq (3.6)
P should be calculated for each individual using Eq (3.2)
Update A by Eq. (3.3)
Eq. should be used to update positions (3.1)
Eq. (3.12) and Eq. (3.13) are used to update location depending on flight technique (3.13)
End For t=t+1
End While
Return FB, Yb

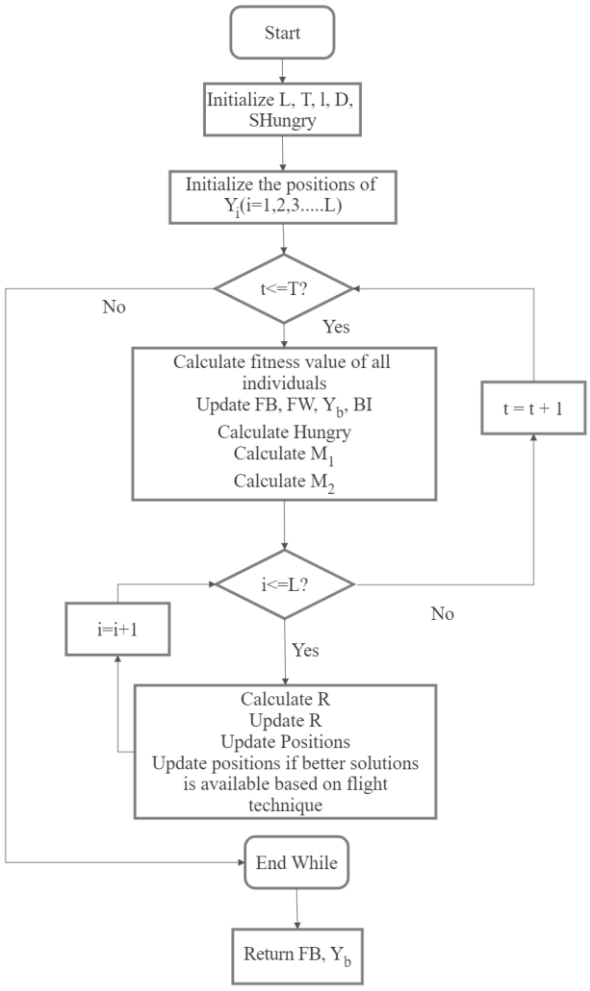



Figure.1: Workflow of HGS Algorithm

We have considered 28 benchmark functions from CEC 2014 to test the algorithm from various aspects. We then employed a Wilcoxon pairwise rank test with 5% significance to verify the results. All tests were carried out on a Windows 10 Home computer with an Intel (R) Core(TM)

i5-8250U CPU running at 1.60GHz and 1.80GHz with 8.00 GB of RAM. For a fair comparison, all algorithms were programmed in the Jupyter notebook. All tests are performed for 51

The statistical significance of the data was validated using the Wilcoxon rank-sum across each algorithm's mean error values. Over 51 runs, the rank-sum ‘p' for two contrasted methods is obtained for each function. As a result, the ‘p' number denotes the likelihood of a difference in mean values of algorithms that occurs only by chance. At a 5% level of confidence, the difference in the errors is judged significant in this study.

**Versus Cauchy HGS**

**1. 10D**

**Table 1.** Algorithms vs Cauchy HGS (10D)

| | Better | Worse | No significance |
|---|---|---|---|
| Original HGS | 0.0 | 2.0 | 23.0 |
| Levy HGS | 0.0 | 0.0 | 25.0 |

**2. 100D**

**Table 2.** Algorithms vs Cauchy HGS (100D)

| | Better | Worse | No significance |
|---|---|---|---|
| Original HGS | 1.0 | 14.0 | 10.0 |
| Levy HGS | 2.0 | 0.0 | 23.0 |

**5.1.2 Versus Levy HGS**

**1. 10D**

**Table 3.** Algorithms vs Levy HGS (10D)

| | Better | Worse | No |
|---|---|---|---|

| | | | significance |
|---|---|---|---|
| Original HGS | 0.0 | 1.0 | 24.0 |
| Levy HGS | 0.0 | 0.0 | 25.0 |

**2. 100D**

**Table 4.** Algorithms vs Levy HGS (100D)

| | Better | Worse | No significance |
|---|---|---|---|
| Original HGS | 0.0 | 14.0 | 11.0 |
| Levy HGS | 0.0 | 2.0 | 23.0 |

### 5.2 Time complexity analysis

The computational time required by any algorithm to perform a specific benchmark function is referred to as time complexity. F18 benchmark function is used to test the time complexity. 200000 iterations are performed to compute the time.

Time is represented in seconds in Table 5. The results indicate that Levy HGS performs best at 10D, 20D, and 100D with the least time and gives competitive results to Cauchy HGS in 30D and 50D. This indicated that Levy HGS outperforms HGS in the case of time complexity.

**Table 5:** Time complexity of algorithms

| | **10D** | **20D** | **30D** | **50D** | **100D** |
|---|---|---|---|---|---|
| **Original HGS** | 160.884305 | 404.603323 | 5306.007253 | 615.148270 | 3657.139789 |
| **Levy HGS** | 143.919007 | 393.860978 | 5352.594429 | 602.848463 | 3558.923320 |

| | | | | | |
|---|---|---|---|---|---|
| **Cauchy HGS** | 760.278522 | 364.762459 | 5328.173065 | 690.149724 | 3571.490297 |

**Functions of the CEC 2014 Benchmark:**

To evaluate the performance of the Levy HGS from a variety of perspectives, 28 benchmark functions were chosen to make use of different characteristics. Unimodal functions, Simple Multimodal Functions, Hybrid Functions, and Composition Functions are the four types of functions. Table 6 summarises these functions. with dimensions, search space boundary, and minimum function value. Unimodal functions help to obtain convergence and exploitation capability of the algorithm by having one global optima. Multimodal functions have many local optima by which it shows the performance of algorithms in exploration tasks by avoiding local optima. The test has been performed with 10000*D maxFES.

**Comparison with unimodal functions:**

The unimodal functions from f1-f3 indicate the convergence rate since it has only global optima and no local optima. Levy HGS outperforms HGS in all these functions and gives competitive results with Cauchy HGS. The mean error of these algorithms proves that levy hgs gives better results in 10D and 100D.

**Comparison with multimodal functions:**

The multimodal functions from f4-f16 indicate the explorations capability of the algorithm. The convergence of Levy HGS is better than HGS for most of the benchmark functions in 10D and 100D. This can be verified with the mean error values where Levy HGS has low error in comparison with other two algorithms.

**Comparison with Composite and Hybrid functions:**

In these complex functions with global optima and many local optima Levy HGS outperforms HGS and gives competitive results with Cauchy HGS. The mean errors of these functions verify these results.

**Table 6:** The IEEE CEC2014 functionalities are described here.

| ID | Function Equations | Dim | Range | $f_{min}$ |
|---|---|---|---|---|
| Uni modal Functions | | | | |
| F1 | Rotated High Conditioned Eliiptic Function | 30 | [-100,100] | 100 |

| | | | | |
|---|---|---|---|---|
| F2 | Rotated Ben Cigar Function | 30 | [-100,100] | 200 |
| F3 | Rotated Discus Function | 30 | [-100,100] | 300 |
| Simple Multimodal Functions | | | | |
| F4 | Shifted Rotated Rosenbrock Function | 30 | [-100,100] | 400 |
| F5 | Shifted Rotated Ackleys Function | 30 | [-100,100] | 500 |
| F6 | Shifted Rotated Weieratrass Function | 30 | [-100,100] | 600 |
| F7 | Shifted Rotated Griewanks Functions | 30 | [-100,100] | 700 |
| F8 | Shifted Rastrigins Function | 30 | [-100,100] | 800 |
| F9 | Shifted and Rotated Rastrigins Function | 30 | [-100,100] | 900 |
| F10 | Shifted Schwefels Function | 30 | [-100,100] | 1000 |
| F11 | Shifted and Rotated Schwefels Function | 30 | [-100,100] | 1100 |
| F12 | Shifted and Rotated Katsuural Function | 30 | [-100,100] | 1200 |
| F13 | Shifted and Rotated Happy Cat Function | 30 | [-100,100] | 1300 |
| F14 | Shifted and Rotated HGBat Function | 30 | [-100,100] | 1400 |
| F15 | Shifted and Rotated Expanded Griewanksplus Rosenbrocks Function | 30 | [-100,100] | 1500 |
| F16 | ShiftedandRotatedExpandedScaffers F6Function | 30 | [-100,100] | 1600 |
| Hybrid Functions | | | | |

| | | | | |
|---|---|---|---|---|
| F17 | Hybrid Function1(N = 3) | 30 | [-100,100] | 1700 |
| F18 | Hybrid Function2(N = 3) | 30 | [-100,100] | 1800 |
| F19 | Hybrid Function3(N = 4) | 30 | [-100,100] | 1900 |
| F20 | Hybrid Function4(N = 4) | 30 | [-100,100] | 2000 |
| F21 | Hybrid Function5(N = 5) | 30 | [-100,100] | 2100 |
| F22 | Hybrid Function6(N = 5) | 30 | [-100,100] | 2200 |
| Composition Functions | | | | |
| F23 | Composition Function1(N = 5) | 30 | [-100,100] | 2300 |
| F24 | Composition Function2(N = 3) | 30 | [-100,100] | 2400 |
| F25 | Composition Function3(N = 3) | 30 | [-100,100] | 2500 |
| F26 | Composition Function4(N = 5) | 30 | [-100,100] | 2600 |
| F27 | Composition Function5(N = 5) | 30 | [-100,100] | 2700 |
| F28 | Composition Function6(N = 5) | 30 | [-100,100] | 2800 |

**10D**

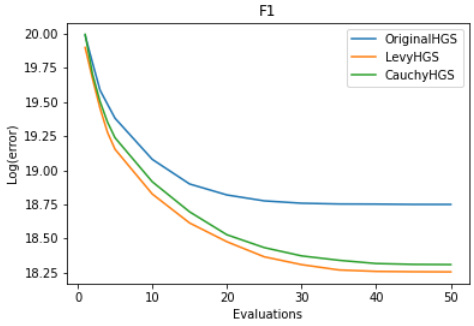



Fig 2. RLD of F1 CEC Function(10D)

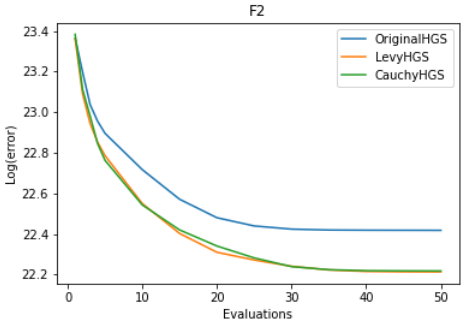



Fig 3. RLD of F2 CEC Function(10D)

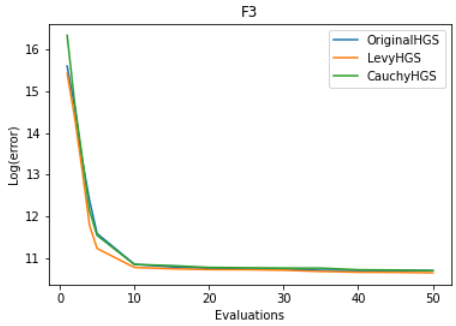



Fig 4. RLD of F3 CEC Function(10D)

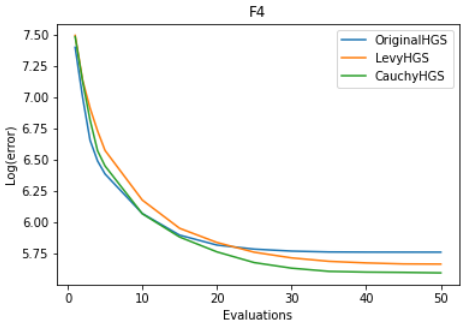



Fig 5. RLD of F4 CEC Function(10D)

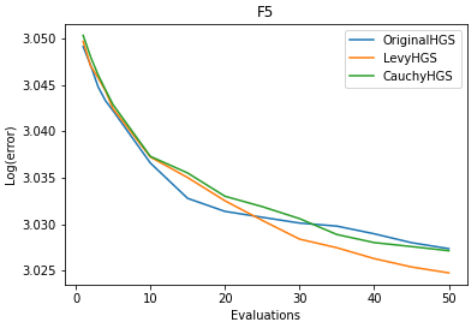



Fig 6. RLD of F5 CEC Function(10D)

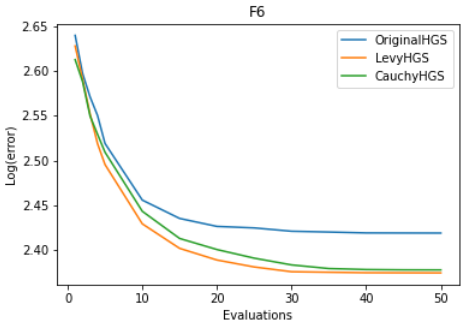



Fig 7. RLD of F6 CEC Function(10D)

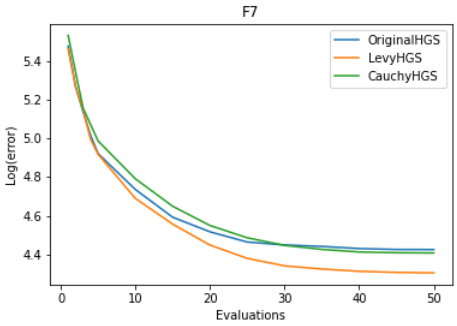



Fig 8. RLD of F7 CEC Function(10D)

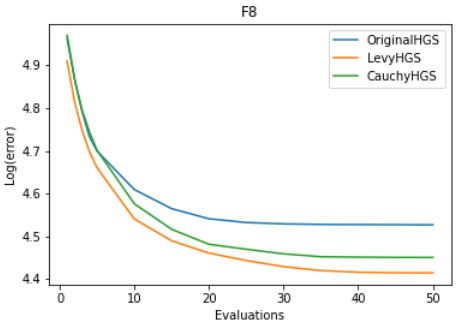



Fig 9. RLD of F8 CEC Function(10D)

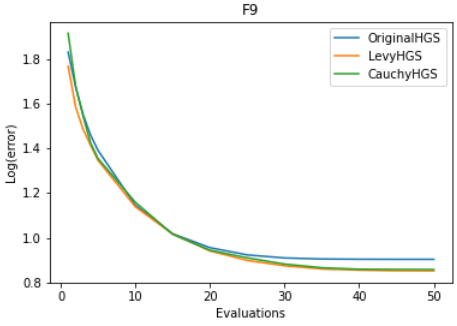



Fig 10. RLD of F9 CEC Function(10D)

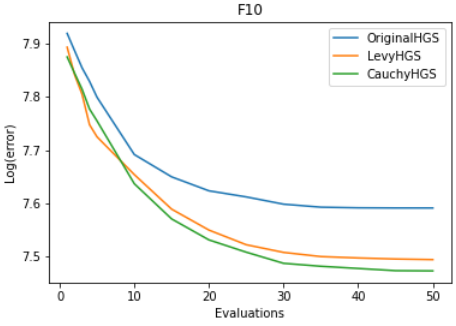



Fig 11. RLD of F10 CEC Function(10D)

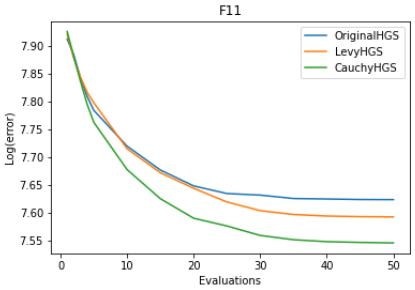



Fig 12. RLD of F11 CEC Function(10D)

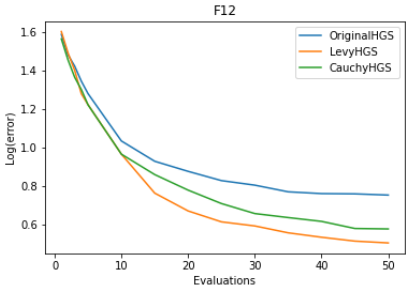



Fig 13. RLD of F12 CEC Function(10D)

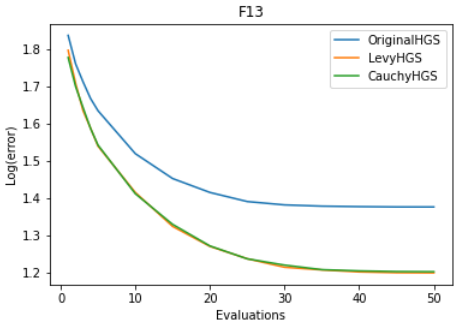



Fig 14. RLD of F13 CEC Function(10D)

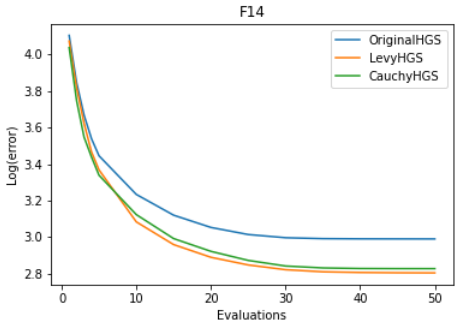



Fig 15. RLD of F14 CEC Function(10D)

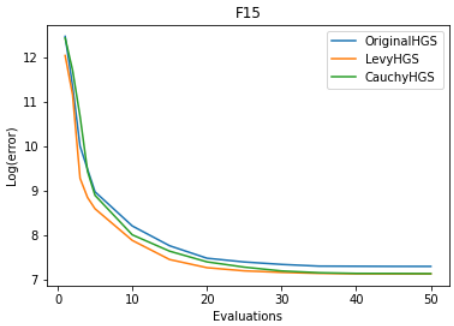



Fig 16. RLD of F15 CEC Function(10D)

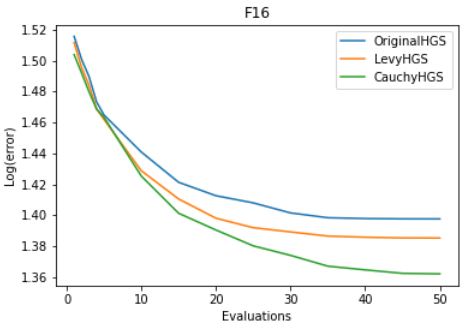



Fig 17. RLD of F16 CEC Function(10D)

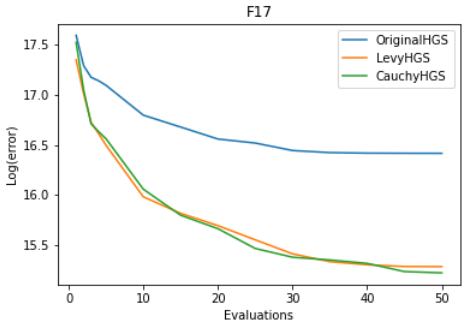



Fig 18. RLD of F17 CEC Function(10D)

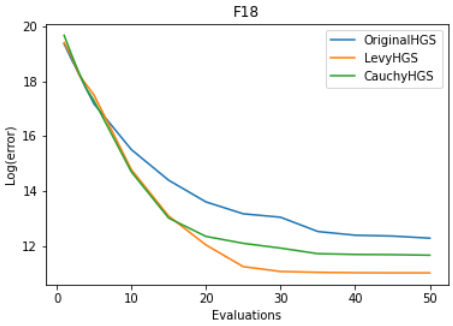



Fig 19. RLD of F18 CEC Function(10D)

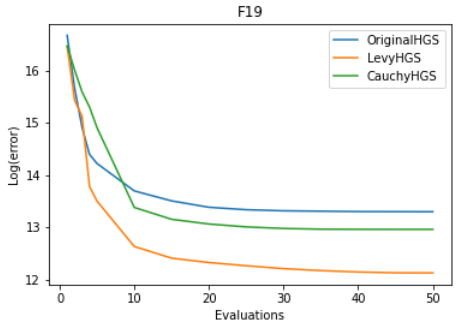



Fig 20. RLD of F19 CEC Function(10D)

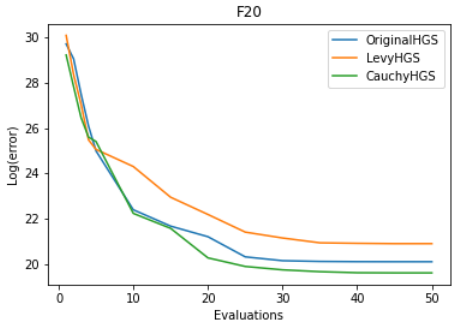



Fig 21. RLD of F20 CEC Function(10D)

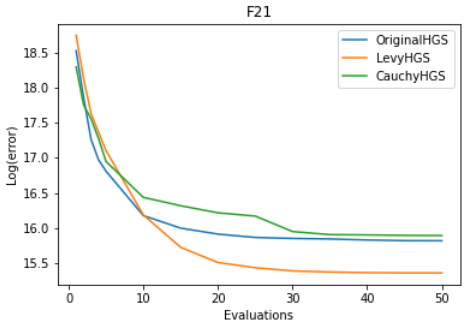



Fig 22. RLD of F21 CEC Function(10D)

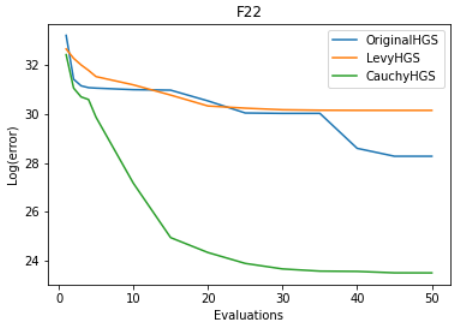



Fig 23. RLD of F22 CEC Function(10D)

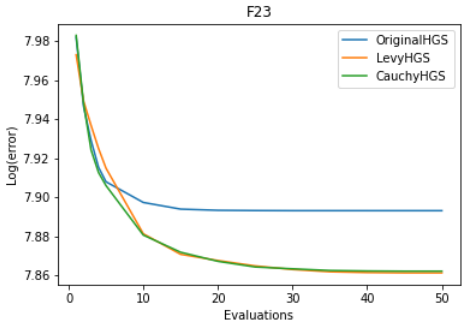



Fig 24. RLD of F23 CEC Function(10D)

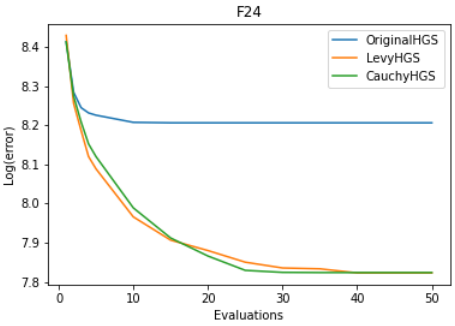



Fig 25. RLD of F24 CEC Function(10D)

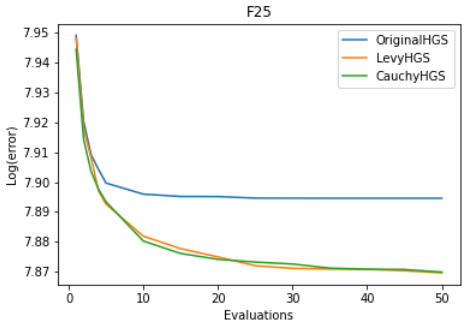



Fig 26. RLD of F25 CEC Function(10D)

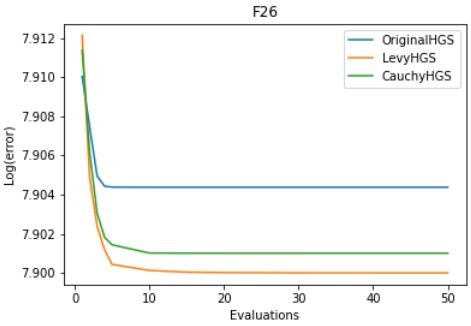
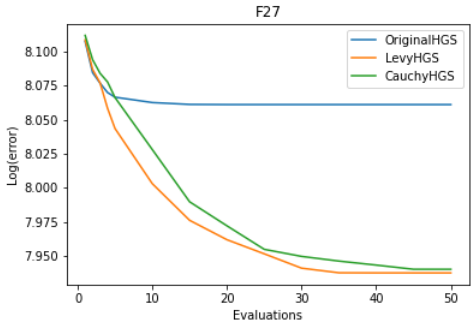



Fig 27. RLD of F26 CEC Function(10D)



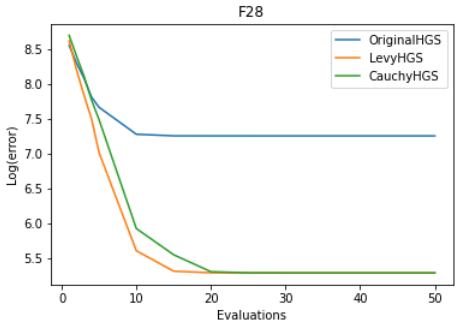
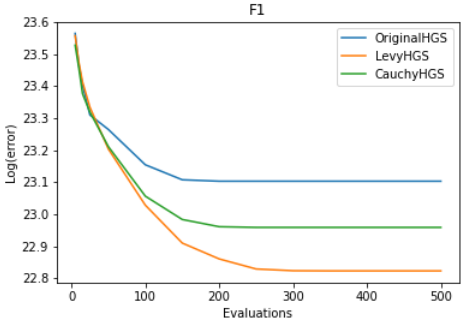




Fig 28. RLD of F27 CEC Function(10D)

Fig 29. RLD of F28 CEC Function(10D)

**100D**



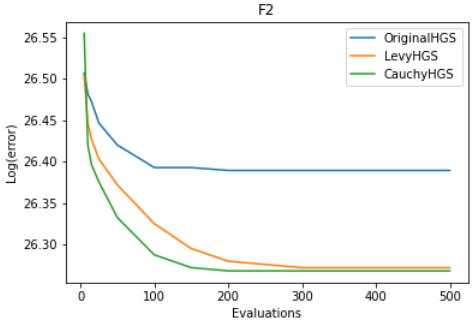
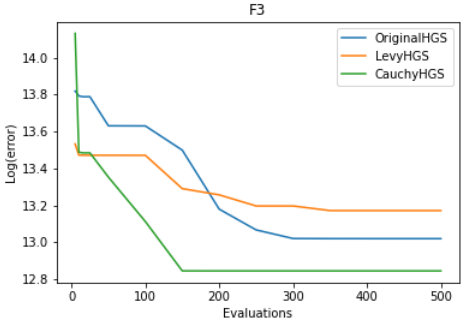




Fig 30. RLD of F1 CEC Function(100D)

Fig 31. RLD of F2 CEC Function(100D)



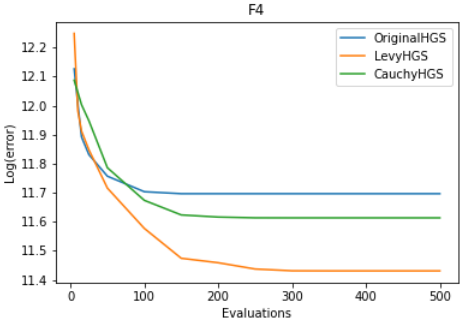




Fig 32. RLD of F3 CEC Function(100D)

Fig 33. RLD of F4 CEC Function(100D)

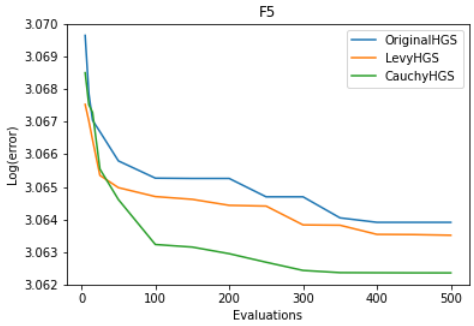



Fig 34. RLD of F5 CEC Function(100D)

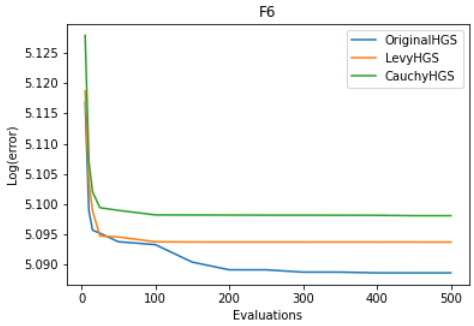



Fig 35. RLD of F6 CEC Function(100D)

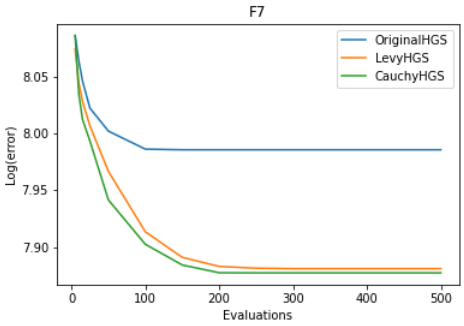



Fig 36. RLD of F7 CEC Function(100D)

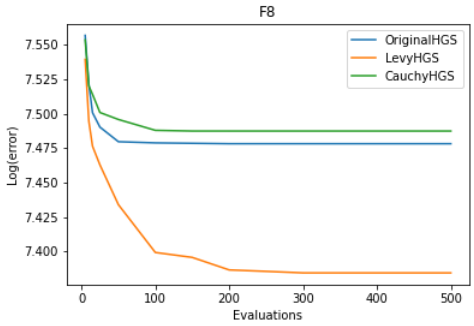



Fig 37. RLD of F2 CEC Function(100D)

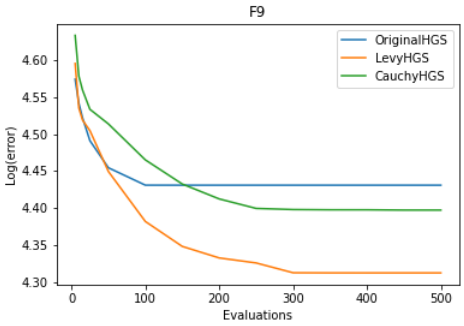



Fig 38. RLD of F9 CEC Function(100D)

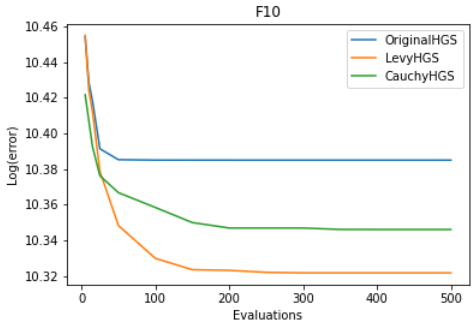



Fig 39. RLD of F10 CEC Function(100D)

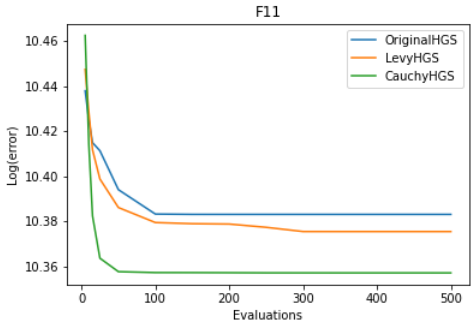



Fig 40. RLD of F11 CEC Function(100D)

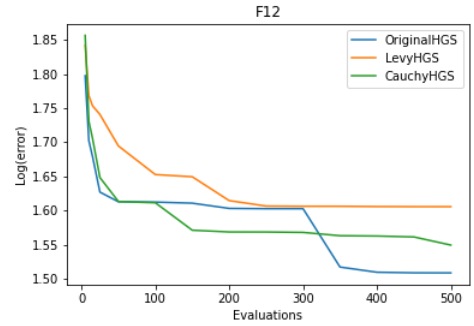



Fig 41. RLD of F12 CEC Function(100D)

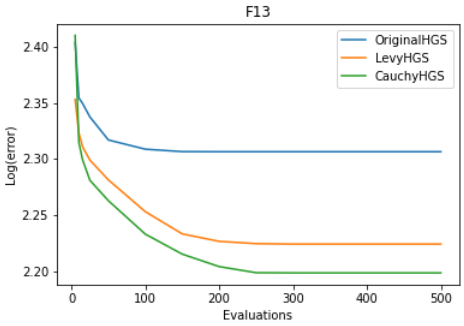



Fig 42. RLD of F13 CEC Function(100D)

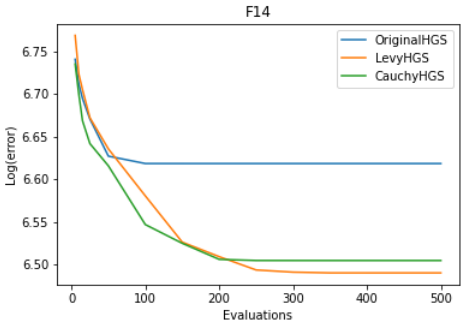



Fig 43. RLD of F14 CEC Function(100D)

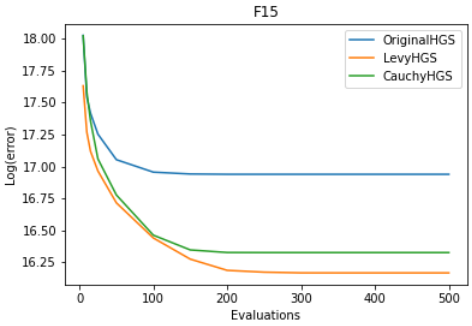



Fig 44. RLD of F15 CEC Function(100D)

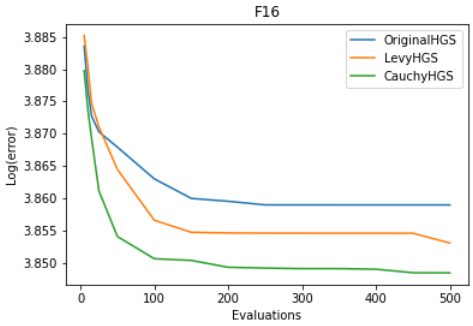



Fig 45. RLD of F2 CEC Function(100D)

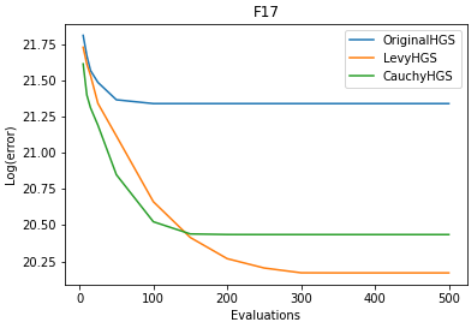



Fig 46. RLD of F17 CEC Function(100D)

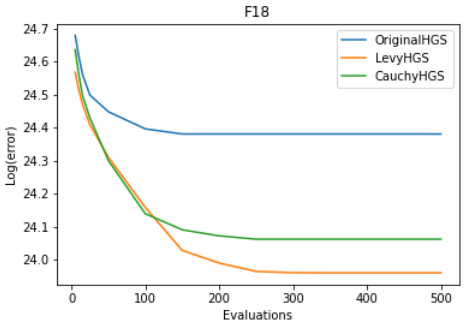



Fig 47. RLD of F18 CEC Function(100D)

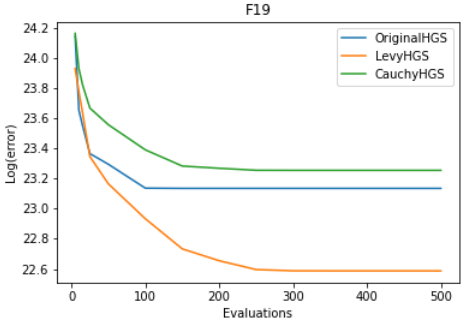



Fig 48. RLD of F19 CEC Function(100D)

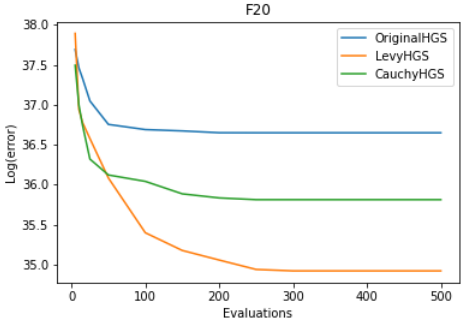



Fig 49. RLD of F20 CEC Function(100D)

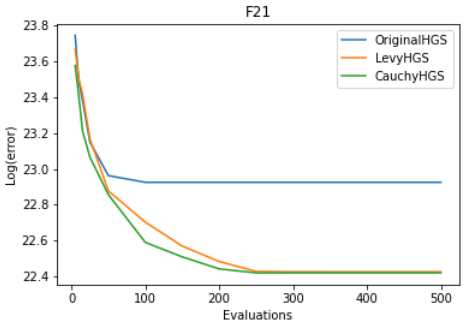



Fig 50. RLD of F21 CEC Function(100D)

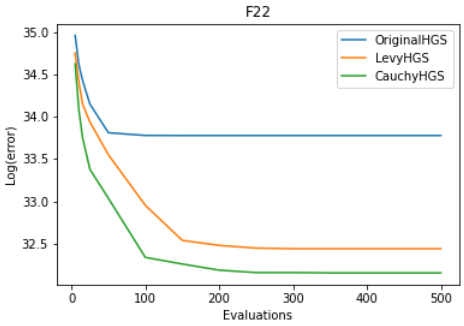



Fig 51. RLD of F22 CEC Function(100D)

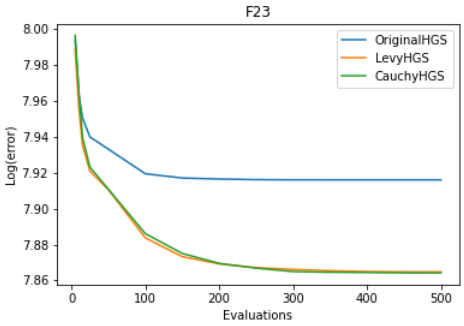



Fig 52. RLD of F23 CEC Function(100D)

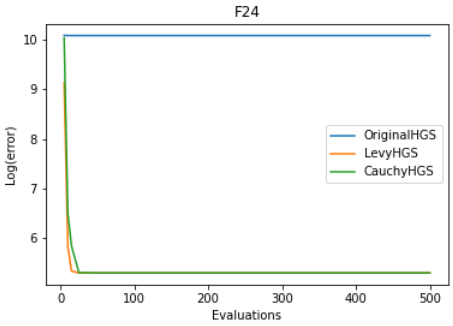



Fig 53. RLD of F24 CEC Function(100D)

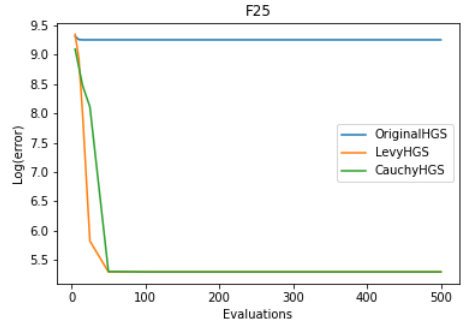



Fig 54. RLD of F1 CEC Function(100D)

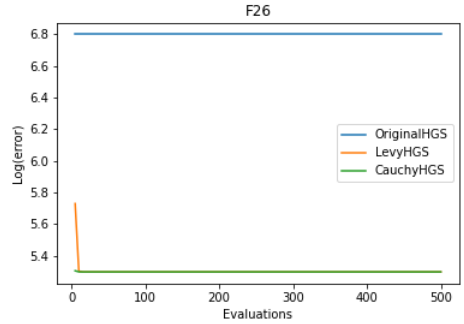



Fig 55. RLD of F26 CEC Function(100D)

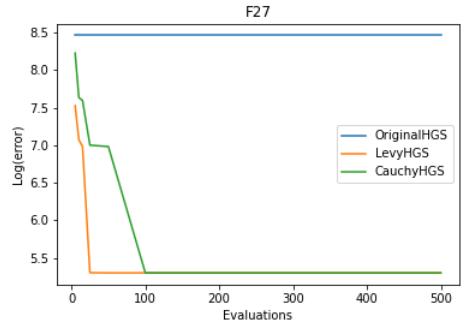



Fig 56. RLD of F27 CEC Function(100D)

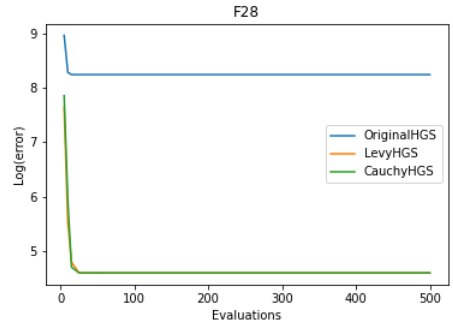



Fig 57. RLD of F28 CEC Function(100D)

## 4. Results

| | Original HGS | Levy HGS | Cauchy HGS |
|---|---|---|---|
| **F1** | 1.39E+08 | 84912335 | 89566438 |
| **F2** | 5.44E+09 | 4.44E+09 | 4.46E+09 |
| **F3** | 43921.1 | 41537.43 | 43819.36 |
| **F4** | 217.8922 | 189.0115 | 169.4616 |
| **F5** | -79.3574 | -79.4112 | -79.362 |
| **F6** | -88.7658 | -89.2552 | -89.2193 |
| **F7** | -16.4323 | -25.902 | -17.8765 |
| **F8** | -7.56976 | -17.4081 | -14.3642 |
| **F9** | -97.5306 | -97.6548 | -97.6409 |
| **F10** | 1881.003 | 1698.06 | 1660.421 |
| **F11** | 1945.561 | 1883.031 | 1792.527 |
| **F12** | -97.8794 | -98.3472 | -98.2223 |
| **F13** | -96.0329 | -96.676 | -96.6657 |
| **F14** | -80.0994 | -83.4721 | -83.0845 |
| **F15** | 1388.257 | 1158.226 | 1164.26 |
| **F16** | -95.8993 | -95.975 | -96.0649 |
| **F17** | 13457854 | 4336160 | 4073758 |
| **F18** | 217497.3 | 61387.67 | 116888.9 |
| **F19** | 599528.4 | 186036.2 | 427767.7 |
| **F20** | 5.42E+08 | 1.2E+09 | 3.31E+08 |
| **F21** | 7382119 | 4660710 | 7952305 |

| | | | |
|---|---|---|---|
| **F22** | 1.91E+12 | 1.25E+13 | 1.59E+10 |
| **F23** | 2578.876 | 2494.916 | 2497.16 |
| **F24** | 3563.189 | 2398.603 | 2400.5 |
| **F25** | 2582.727 | 2516.79 | 2517.479 |
| **F26** | 2609.124 | 2597.28 | 2600 |
| **F27** | 510.7582 | 100.0274 | 100.0017 |
| **F28** | 1321.466 | 100.0003 | 100.0003 |

**Table 7: Mean error 10D**

**Table 8: Mean error 100D**

| | **Original HGS** | **Levy HGS** | **Cauchy HGS** |
|---|---|---|---|
| **F1** | 1.08E+10 | 8.17E+09 | 9.36E+09 |
| **F2** | 2.89E+11 | 2.57E+11 | 2.56E+11 |
| **F3** | 451375 | 525377.3 | 379146.3 |
| **F4** | 120077 | 92051.18 | 110497.8 |
| **F5** | -78.5889 | -78.5973 | -78.622 |
| **F6** | 62.15499 | 62.98325 | 63.70115 |
| **F7** | 2838.006 | 2546.596 | 2536.775 |
| **F8** | 1669.047 | 1510.795 | 1685.394 |
| **F9** | -15.9785 | -25.3797 | -18.7644 |
| **F10** | 32267.52 | 30283.72 | 31032.12 |
| **F11** | 32211.97 | 31966.3 | 31383.47 |
| **F12** | -95.4788 | -95.0187 | -95.291 |
| **F13** | -89.9591 | -90.7529 | -90.9874 |

| | | | |
|---|---|---|---|
| **F14** | 648.8629 | 558.9019 | 568.5063 |
| **F15** | 22740287 | 10518659 | 12328946 |
| **F16** | -52.5822 | -52.8608 | -53.0781 |
| **F17** | 1.86E+09 | 5.75E+08 | 7.5E+08 |
| **F18** | 3.87E+10 | 2.55E+10 | 2.82E+10 |
| **F19** | 1.11E+10 | 6.45E+09 | 1.26E+10 |
| **F20** | 8.25E+15 | 1.46E+15 | 3.57E+15 |
| **F21** | 9.04E+09 | 5.49E+09 | 5.45E+09 |
| **F22** | 4.69E+14 | 1.22E+14 | 9.19E+13 |
| **F23** | 2640.763 | 2504.031 | 2502.592 |
| **F24** | 23786.86 | 100.5 | 100.5 |
| **F25** | 10368.55 | 100 | 100 |
| **F26** | 797.4677 | 100 | 100 |
| **F27** | 4631.064 | 100.0032 | 100.0032 |
| **F28** | 7488.847 | 100.0032 | 100.0032 |

**Comparison with dataset on clustering task:**

Each algorithm is tested upon 24 datasets among which 13 are balanced datasets and 11 are unbalanced datasets. The performance of algorithms is measured based on inter-cluster distance and convergence speed. The results show that Levy HGS works well for 15 datasets and Cauchy HGS for 9 datasets. These algorithms are also tested with K-Means and it is observed that Levy & Cauchy HGS works well for 18 datasets.

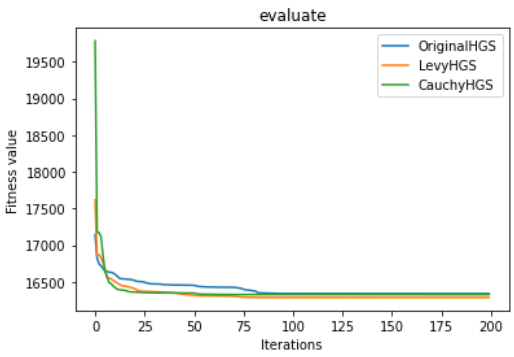



Fig 58. Aniso dataset convergence graph

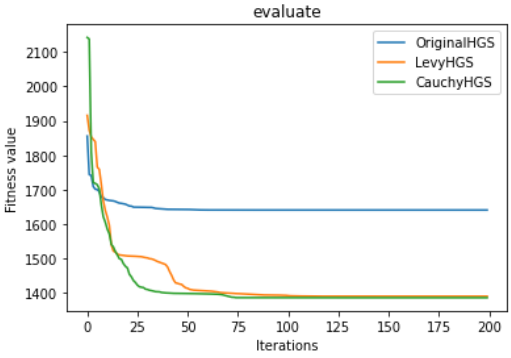



Fig 59. Appendicitis dataset convergence graph

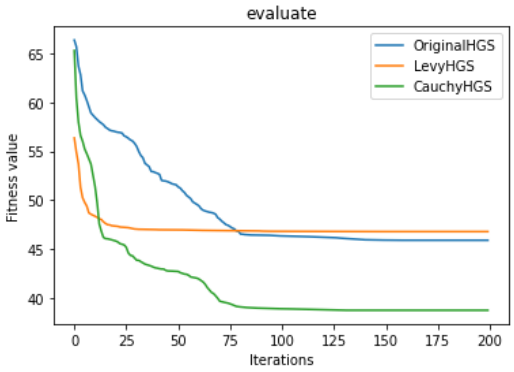



Fig 60. Balance dataset convergence graph

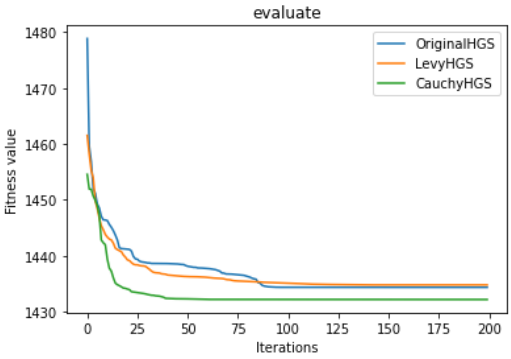



Fig 61. Banknote dataset convergence graph

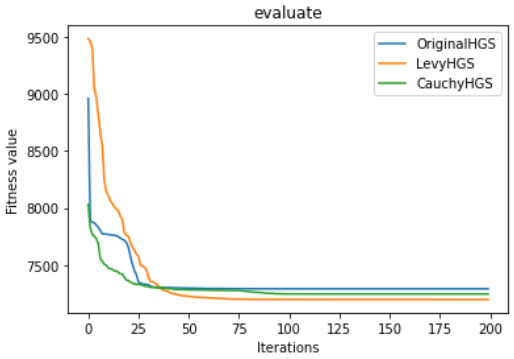



Fig 62. Blobs dataset convergence graph

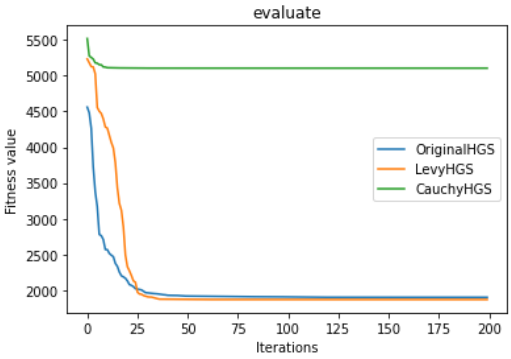



Fig 63. Blood dataset convergence graph

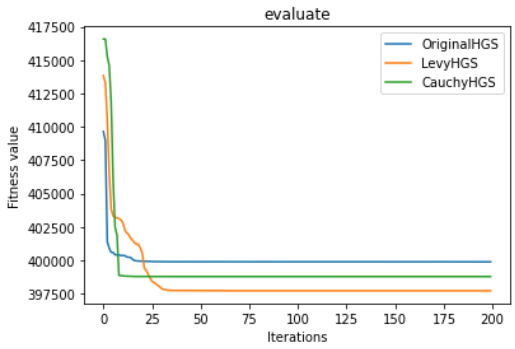



Fig 64. Diagnosis dataset convergence graph

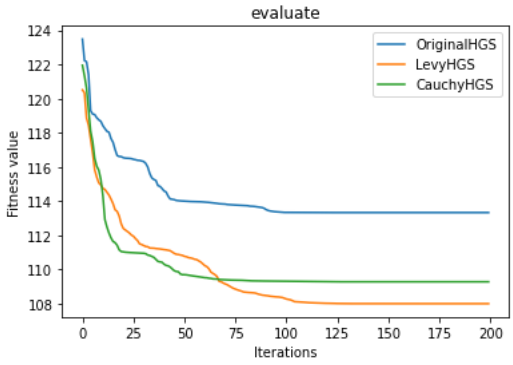



Fig 65. Ecoli dataset convergence graph

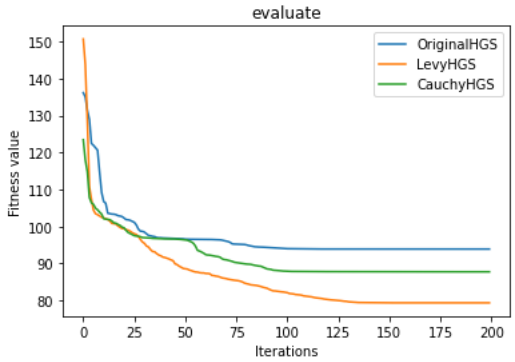



Fig 66. Flame dataset convergence graph

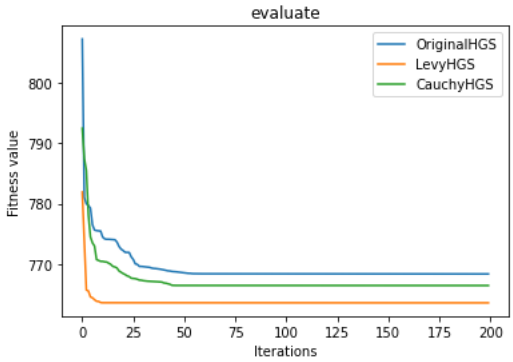



Fig 67. Ionosphere dataset convergence graph

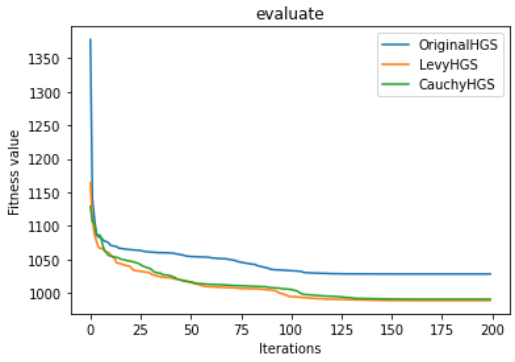



Fig 68. Iris dataset convergence graph

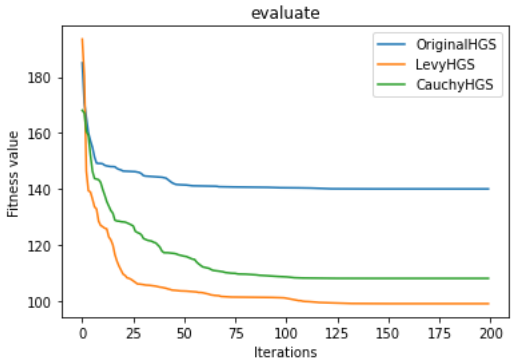



Fig 69. Iris2d dataset convergence graph

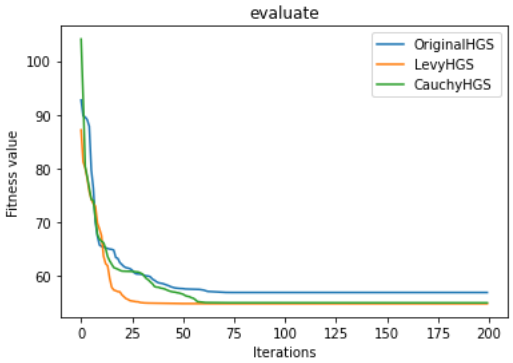



Fig 70. Jain dataset convergence graph

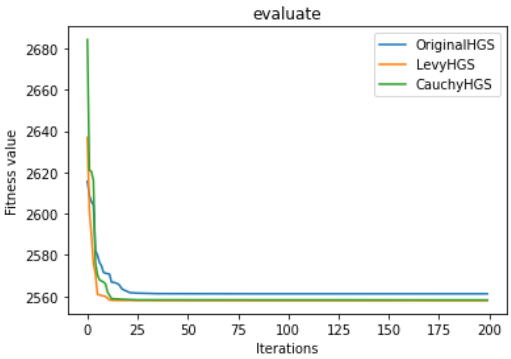



Fig 71. Liver dataset convergence graph

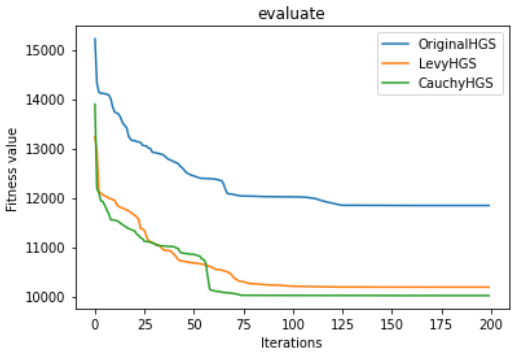



Fig 72. Moons dataset convergence graph

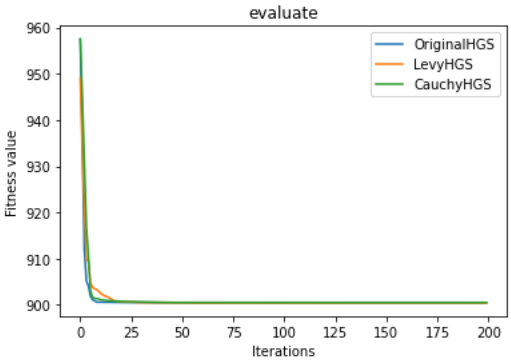



Fig 73. Mouse dataset convergence graph

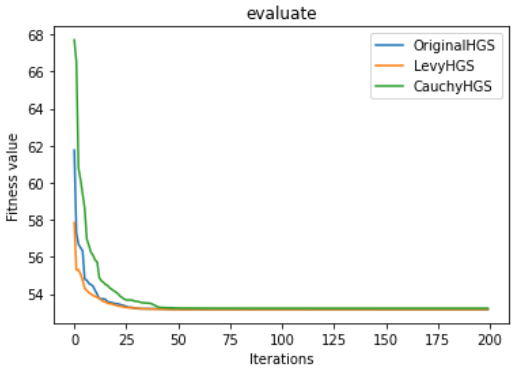



Fig 74. Path-Based dataset convergence graph

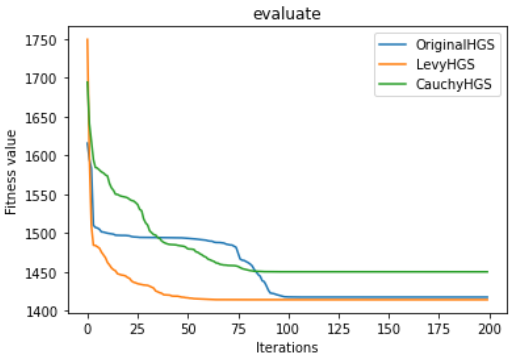



Fig 75. Smiley dataset convergence graph

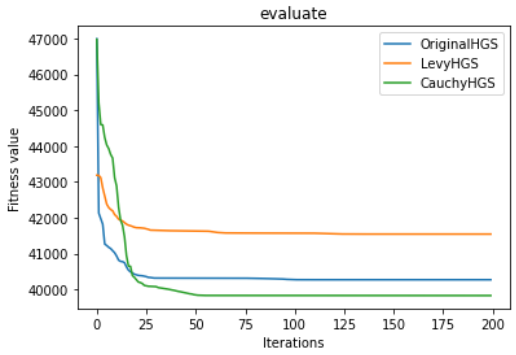



Fig 76. Varied dataset convergence graph

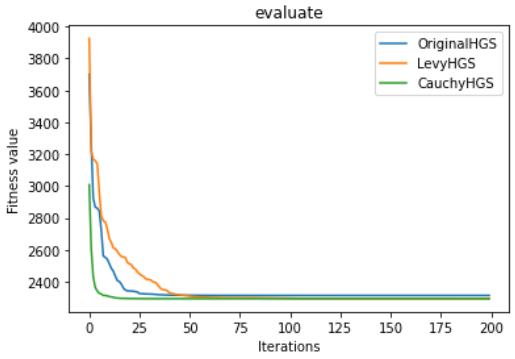



Fig 77. Vary density dataset convergence graph

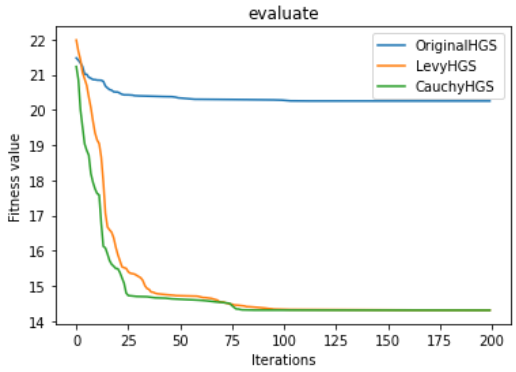



Fig 78. Vertral2 dataset convergence graph

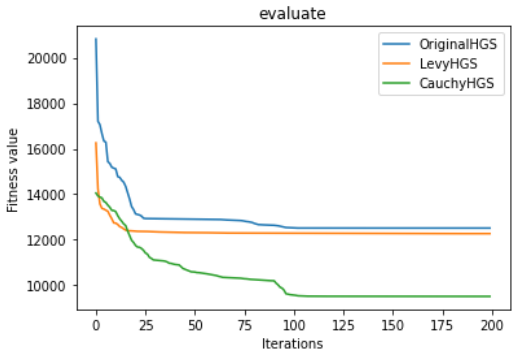



Fig 79. Vertebral3 dataset convergence graph

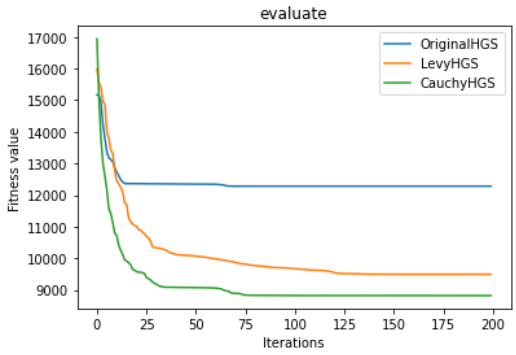



Fig 80. Wdbc dataset convergence graph

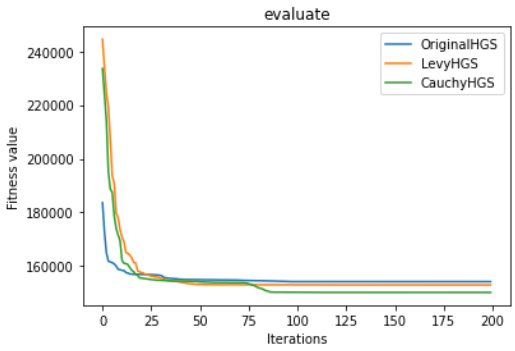



Fig 81. Wine dataset convergence graph

**Table 9: Comparison with K-Means**

| | **Original HGS** | **Levy HGS** | **Cauchy HGS** | **K-means** |
|---|---|---|---|---|
| **Blood** | 3993893.387100 | 397735.992700 | 398790.094800 | 458108.436800 |

| | | | | |
|---|---|---|---|---|
| **Aniso** | 1640.631083 | 1389.922204 | 1385.813822 | 1403.841180 |
| **appendicitis** | 45.932999 | 46.815048 | 38.793834 | 37.771954 |
| **Balance** | 1434.362338 | 1434.787950 | 1432.176341 | 1501.232830 |
| **Banknote** | 7292.287727 | 7198.469472 | 7247.060995 | 7241.516211 |
| **Blobs** | 1900.303935 | 1869.405239 | 5104.612762 | 1870.185169 |
| **Diagnosis** | 113.336836 | 108.007365 | 109.287104 | 109.334315 |
| **Ecoli** | 93.919720 | 79.382201 | 87.771657 | 66.777036 |
| **Flame** | 768.353985 | 763.576414 | 766.441338 | 780.542120 |
| **Ionosphere** | 1028.084022 | 988.487016 | 990.498240 | 794.635445 |
| **Iris** | 140.047592 | 99.023303 | 108.062251 | 97.195772 |
| **Iris2D** | 56.889356 | 54.807600 | 54.964857 | 55.319108 |
| **Jain** | 2561.181964 | 2557.924602 | 2558.179117 | 2608.841432 |
| **Liver** | 11843.768440 | 10188.964790 | 10018.388710 | 10178.859100 |
| **Moons** | 900.495516 | 900.455913 | 900.527324 | 904.741304 |
| **Mouse** | 53.176911 | 53.172753 | 53.236675 | 53.790641 |
| **Pathbased** | 1417.349333 | 1414.052552 | 1449.941309 | 1423.361597 |
| **Smiley** | 40270.206100 | 41543.810330 | 39829.027770 | 41782.423650 |
| **Varied** | 2313.664411 | 2294.586879 | 2293.187261 | 2311.947416 |

| | | | | |
|---|---|---|---|---|
| **Vary-density** | 20.250483 | 14.316154 | 14.310175 | 14.333472 |
| **Vertebral2** | 12513.946220 | 12271.268760 | 9513.769946 | 9082.498449 |
| **Vertebral3** | 12281.086910 | 9494.008187 | 8821.531528 | 7845.149030 |
| **Wdbc** | 154260.304700 | 53034.779800 | 150233.383900 | 152264.678100 |
| **wine** | 16340.873780 | 16286.705680 | 16327.263760 | 18100.762450 |

**Performance analysis:**

In most datasets, statistically, the Levy HGS delivers greater local avoidance and an optimal or near optimum. In the great majority of situations, the performance of the Levy HGS is superior to or at least very equal to that of the standard HGS, and the Levy HGS also outperforms the other algorithms. The causes for the Levy HGS's exceptional and efficient performance are described next. To begin with, utilizing the Levy flight trajectory improves the algorithm's diversity, ensuring its global search capabilities and avoiding local minima. With the addition of Levy flight, the positions are not directly updated but it further uses the levy flight equation to final position value. Levy flight helps in exploration and also inefficient search. Thus the Levy HGS has better local search capability and obtains faster results.

## CONCLUSION AND FUTURE WORKS

This research paper provides an improved version of HGS. We conclude that HGS with Levy flight offers the greatest performance in most situations, and the findings are also similar with K-Means and IEEE CEC 2014 benchmark test suites, using 28 benchmark functions and 24 datasets to evaluate the performance of HGS. In addition, we estimate the findings using the Wilcoxon pair-wise ranking test with a 5% significance level. The findings provided in section 4 show that HGS beats all other algorithms in the majority of test cases, demonstrating that HGS is a feasible and efficient way to solving global optimization issues. However, this algorithm performs better than the original Hunger Game Search algorithm, a meta-heuristic algorithm often being stuck in local minima. By introducing flight, we have reduced this possibility. Nevertheless, there can be some edge cases where the algorithm still gets blocked in local minima.

Furthermore, the results show a higher approximation capacity of HGS in a high-dimensional space. This algorithm applies for clustering tasks to converge to minima faster. The problem in

clustering tasks is the Curse of Dimensionality, our algorithm being a meta-heuristics algorithm that can perform better and converge faster. The future scope would be to introduce techniques like greedy search, mutation and crossover, and ensemble techniques to obtain better results, and testing with realistic engineering optimization problems.

## 7. REFERENCE

[1] Wang SK, Jia HM, Peng XX. Modified salp swarm algorithm-based multilevel thresholding for color image segmentation. Math Biosci Eng. 2019 Oct;17(1) 700-724. doi:10.3934/mbe.2020036. PMID: 31731372.
[2] S. Mirjalili, S. M. Mirjalili, and A. Lewis, ‘‘Grey wolf optimizer,’’ Adv. Eng. Softw., vol. 69, pp. 46–61, Mar. 2014.
[3] Mohammadi-Balani, Abdolkarim & Nayeri, Mahmoud & Azar, Adel & Taghizadeh Yazdi, Mohammad Reza. (2020). Golden Eagle Optimizer: A nature-inspired metaheuristic algorithm. Computers & Industrial Engineering. 152. 107050. 10.1016/j.cie.2020.107050.
[4] X. S. Yang, ‘‘Flower pollination algorithm for global optimization,’’ in Unconventional Computation and Natural Computation (Computer Science). New York, NY, USA: Springer, 2012, pp. 240–249.
[5] X. S. Yang and D. Suash, ‘‘Cuckoo Search via Levy flights,’’ in World Congress Nature Biologically Inspired Computer (NaBIC). New York, NY, USA: IEEE Publication, 2009, pp. 210–214.
[6] J. Kennedy, ‘‘Particle swarm optimization,’’ in Proc. IEEE Int. Conf. Neural Netw., Perth, Australia, vol. 4. Mar. 1995, pp. 1942–1948.
[7] Wang SK, Jia HM, Peng XX. Modified salp swarm algorithm-based multilevel thresholding for color image segmentation. Math Biosci Eng. 2019 Oct;17(1) 700-724. doi:10.3934/mbe.2020036. PMID: 31731372.
[8] L. Zhang, C. Li, Y. Wu, J. Huang, and Z. Cui, "An Improved Salp Swarm Algorithm With Spiral Flight Search for Optimizing Hybrid Active Power Filters’ Parameters," in IEEE Access, vol. 8, pp. 154816-154832, 2020, DOI: 10.1109/ACCESS.2020.3006903.
[9] Rizk-Allah RM, Hassanien AE, Elhoseny M, Gunasekaran M, ‘A new binary salp swarm algorithm: development and application for optimization tasks’ 2018.
[10] Zhang, Yu-Dong & Wang, Shuihua & ji, Genlin. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering. 2015. 1-38. 10.1155/2015/931256.
[11] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm,” J. Glob. Optim., vol. 39, pp. 459-471, Apr. 2007.
[12] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” Theor. Comput. Sci., vol. 344, pp. 243-278, Nov. 2005.

[13] Y. Wan, M. Mao, L. Zhou, Q. Zhang, X. Xi, and C. Zheng, "A novel nature-inspired maximum power point tracking (MPPT) controller based on SSA-GWO algorithm for partially shaded photovoltaic systems," Electronics, vol. 8, no. 6, p. 680, 2019.
[14] H. Gao, Y. Hou, S. Zhang, and M. Diao, "An efficient approximation for Nakagami-m quantile function based on generalized opposition-based quantum salp swarm algorithm," Mathematical Problems in Engineering, vol. 2019, Article ID 8291063, 13 pages, 2019.
[15] Z. Xing and H. Jia, "Multilevel color image segmentation based on GLCM and improved salp swarm algorithm," IEEE Access, vol. 7, pp. 37672–37690, 2019.
[16] Y. Fan, J. Shao, G. Sun, and X. Shao, "Proportional-integral derivative controller design using an advanced Levy-flight salp swarm algorithm for hydraulic systems," Energies, vol. 13, no. 2, p. 459, 2020.
[17] Hossam Faris1, Ibrahim Aljarah1, Mohammed Azmi Al-Betar2, Seyedali Mirjalili3, "Grey wolf optimizer: a review of recent variants and applications" Article in Neural Computing and Applications · July 2018, DOI: 10.1007/s00521-017-3272-5.46
[18] Jie-Sheng Wang1,2 & Shu-Xia Li1, "An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism" 09 May 2019(9:7181), doi.org/10.1038/s41598-019-43546-3
[19] Mohammad H. Nadimi-Shahraki a,b,*, Shokooh Taghian a,b, Seyedali Mirjalili c,d, "An improved grey wolf optimizer for solving engineering problems", Expert Systems With Applications 166 (2021) 113917,16 September 2020,0957-4174
[20] Souvik Dhargupta, Manosij Ghosh, Seyedali Mirjalili, Ram Sarkar, "Selective Opposition based Grey Wolf Optimization", 17 March 2020, doi.org/10.1016/j.eswa.2020.113389, 0957-4174.
[21] Qaddoura, Raneem & Faris, Hossam & Aljarah, Ibrahim & Castillo, Pedro. (2021).
EvoCluster: An Open-Source Nature-Inspired Optimization Clustering Framework. SN Computer Science. 2. 10.1007/s42979-021-00511-0.
[22] Xin-She Yang, "Firefly Algorithm, L´evy Flights and Global Optimization", Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK,10 pages, 5 figures, 2010.
[23] Gaige Wang,[1,2] Lihong Guo,[1] Amir Hossein Gandomi,[3] Lihua Cao,[1] Amir Hossein Alavi,[4] Hong Duan,[5] and Jiang Li, "Lévy-Flight Krill Herd Algorithm", Volume 2013 |Article ID 682073, 2013.
[24] Harish Sharmaa, Jagdish Chand Bansalb, K. V. Aryaa & Xin-She Yangc, "Lévy flight artificial bee colony algorithm", 17 Mar 2015.
[25] Georgia Kalantzisa, Charles Shang, YuLeib, Theodora Leventoria, "Investigations of a GPU-based levy-firefly algorithm for constrained optimization of radiation therapy treatment planning", Volume 26, February 2016, Pages 191-201.
[26] Abhijit Majumdar, Animesh Laha, "Effects of fabric construction and shear thickening fluid on yarn pull-out from high-performance fabrics", December 7, 2015, Research Article.

[27] “Twitter sentiment analysis using hybrid cuckoo search method”, Avinash Chandra Pandey, Dharmveer Singh Rajpoot, Mukesh Saraswat, ©2017 Elsevier Ltd. All rights reserved, dx.doi.org j.ipm.2017.02.004, 2017.