AUTISM SPECTRUM DISORDER DETECTION

A COMPARATIVE STUDY OF NATURE INSPIRED ALGORITHMS FOR AUTISM SPECTRUM DISORDER(ASD) DETECTION

By

Vinayak Kukreja and Rajas Deshpande

PROBLEM STATEMENT

- Perform a comparative study of five nature-inspired optimization algorithms for detecting autism in young adults.
- The five algorithms used are -
    - Bat Algorithm
    - Cat Swarm Optimization
    - Cuckoo Optimization Algorithm
    - Firefly Algorithm
    - Grey Wolf Optimization
- The results of all these optimization algorithms is compared with Backpropagation algorithm to find out which algorithm suits the best for Autism detection

APPROACH

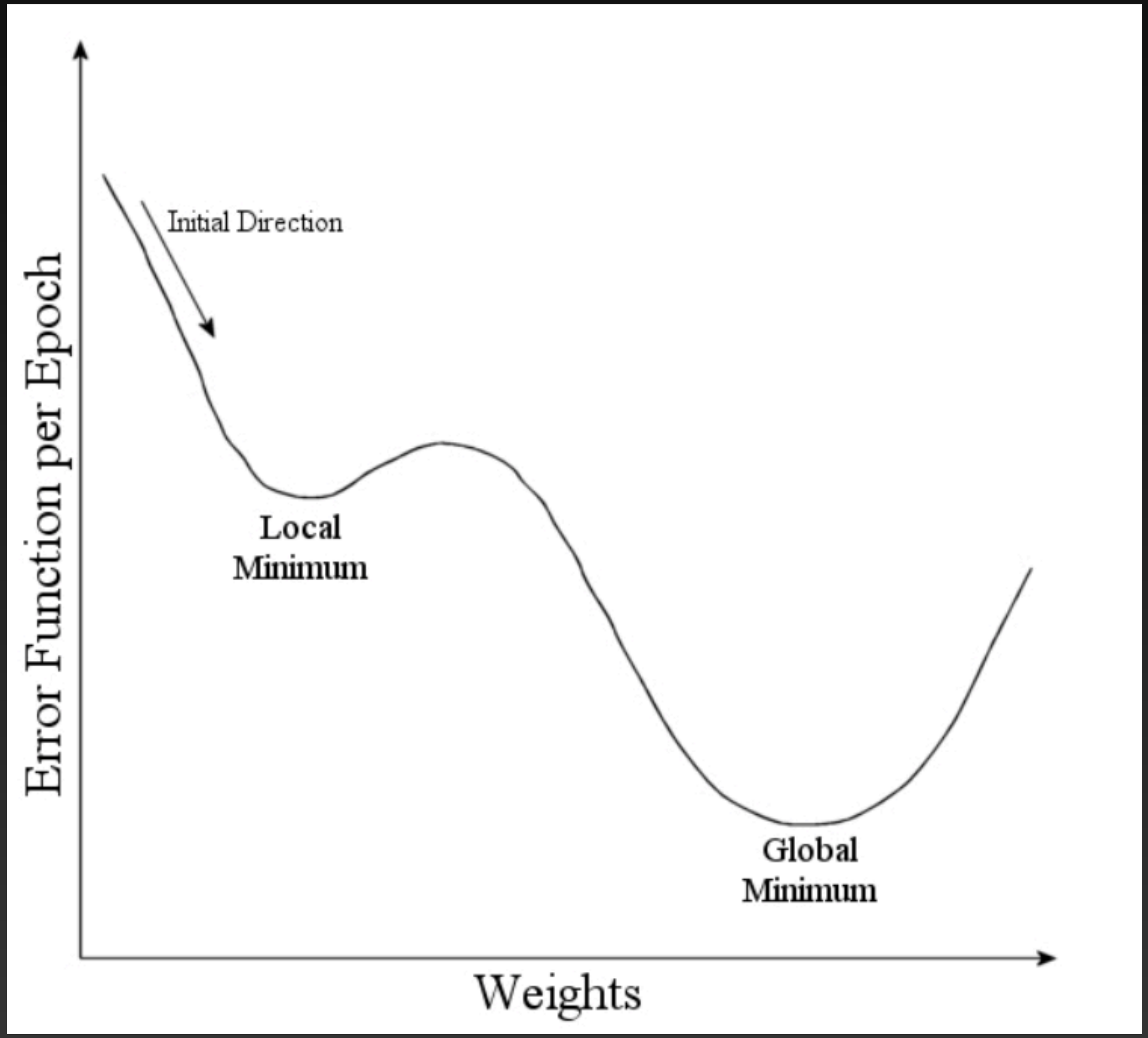
What is Autism Spectrum Disorder?

- Autism Spectrum Disorder (ASD) is a condition related to brain development that impacts how a person perceives and socializes with other, causing problems in social interaction and communication.
- ASD typically begins in early childhood and can show a wide range of symptoms and severity.
- Till now, there is no cure for ASD, but early treatment can make a big difference in the lives of many children.

APPROACH

Why use nature inspired algorithms?

- Back-propagation can be used for such an application but it has a major drawback of getting stuck in a local minima.
- For detecting ASD, the results needs to be accurate and thus the nature-inspired algorithms would help widen the search space and eventually provide the best global solution rather than a local solution.



APPROACH

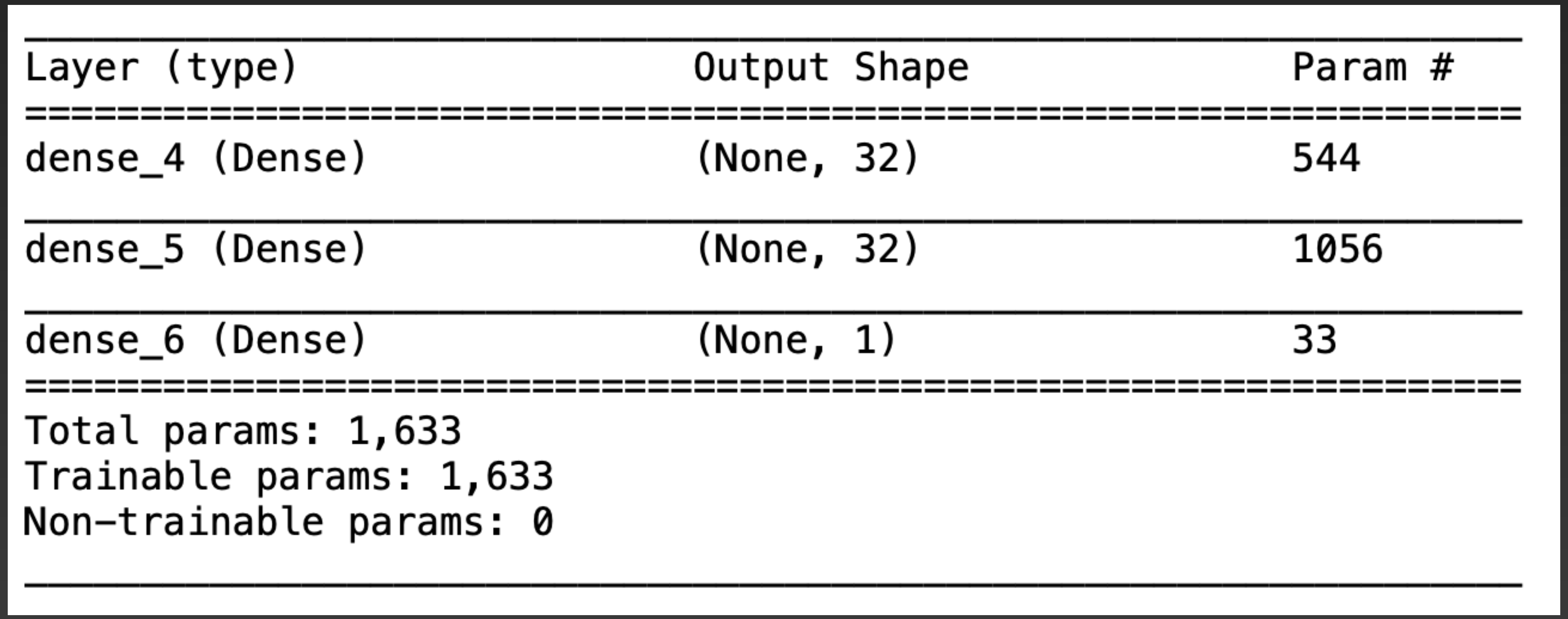
Dataset analysis and preprocessing

- The Autism Spectrum Disorder (ASD) screening dataset has been used for detecting autism in young adults.
- This dataset is labelled and open source and can be obtained from the UCI machine learning repository - http://archive.ics.uci.edu/ml/machine-learning-databases/00426/
- The data has been pre-processed by removing unwanted features and has been converted to numerical format.
- The following features are considered for detection of ASD -
  - A1-A10 Scores : These are some answers given by the people to certain questions
  - Gender : 0 - Female, 1 - Male
  - Jaundice : 0 - False, 1 - True
  - PDD : 0- False, 1 - True
  - Age range
  - ASD Class label
- There are a total of 1100 records and 17 features.

APPROACH

Neural Network Model and Analysis

- A three-layered neural network model has been designed with 32 neurons in the first two layers and 1 neuron in the output layer.
- The output layer will basically provide a predicted label which will be either 0 (does not have ASD) or 1 (has ASD).
- The major goal is to find the best weights and biases for the network so that the model give the highest accuracy.
- All the nature-based algorithms that are going to be explained in the next few slides will provide a best global solution and this solution will be shaped accordingly to match the weights and biases of the neural network model
- Accordingly, the metrics of all the models will be calculated to find out which algorithm is most suited for this application.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 32)                544
_________________________________________________________________
dense_5 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,633
Trainable params: 1,633
Non-trainable params: 0
_________________________________________________________________
```

RESOURCES USED

- We have used a python library called “SwarmPackagePy” which is a library for swarm optimization algorithms.
- This library supports fourteen optimization algorithms which include Grey Wolf Optimization, Bat Algorithm, Firefly Algorithm, Cuckoo Optimization Algorithm and Cat Swarm Optimization Algorithm.
- These are the algorithms that we have used for detection of ASD and compared their efficiency against each other.

IMPLEMENTATION

Nature Inspired Algorithms

- Optimization algorithms based on Swarm Intelligence were developed for simulating intelligent behavior of animals.
- Nature Inspired Algorithms are a sub-category of Computational Intelligence which is a branch of Artificial Intelligence.
- We are here using five nature inspired algorithms that help us detect Autism Spectrum Disorder and we are comparing the results obtained by these algorithms with each other.

GREY WOLF OPTIMIZATION

IMPLEMENTATION

Grey Wolf Optimization

- The Grey Wolf Optimization algorithm mimics the leadership hierarchy and hunting mechanism of gray wolves in nature. Wolves live in a pack. The average pack consists of a family of 5–12 animals. wolves have strict social hierarchy which is represented by the division of a pack into four levels: alpha, beta, delta, and omega.
- Alpha wolves are the leaders of their pack. They are responsible for making decisions, but sometimes alphas can obey to other wolves of the pack.
- Beta wolves help alphas make decisions, every beta is a candidate to become an alpha if an alpha has died or aged. A beta respects an alpha and transfers commands to the pack, ensures discipline among inferior wolves and provides a feedback from the pack to an alpha.
- Delta wolves have to submit to alphas and betas, but they dominate the omega.
- Finally, omega wolves have to obey all other wolves. Sometimes they play a role of caretakers or scapegoats.

IMPLEMENTATION

Grey Wolf Optimization

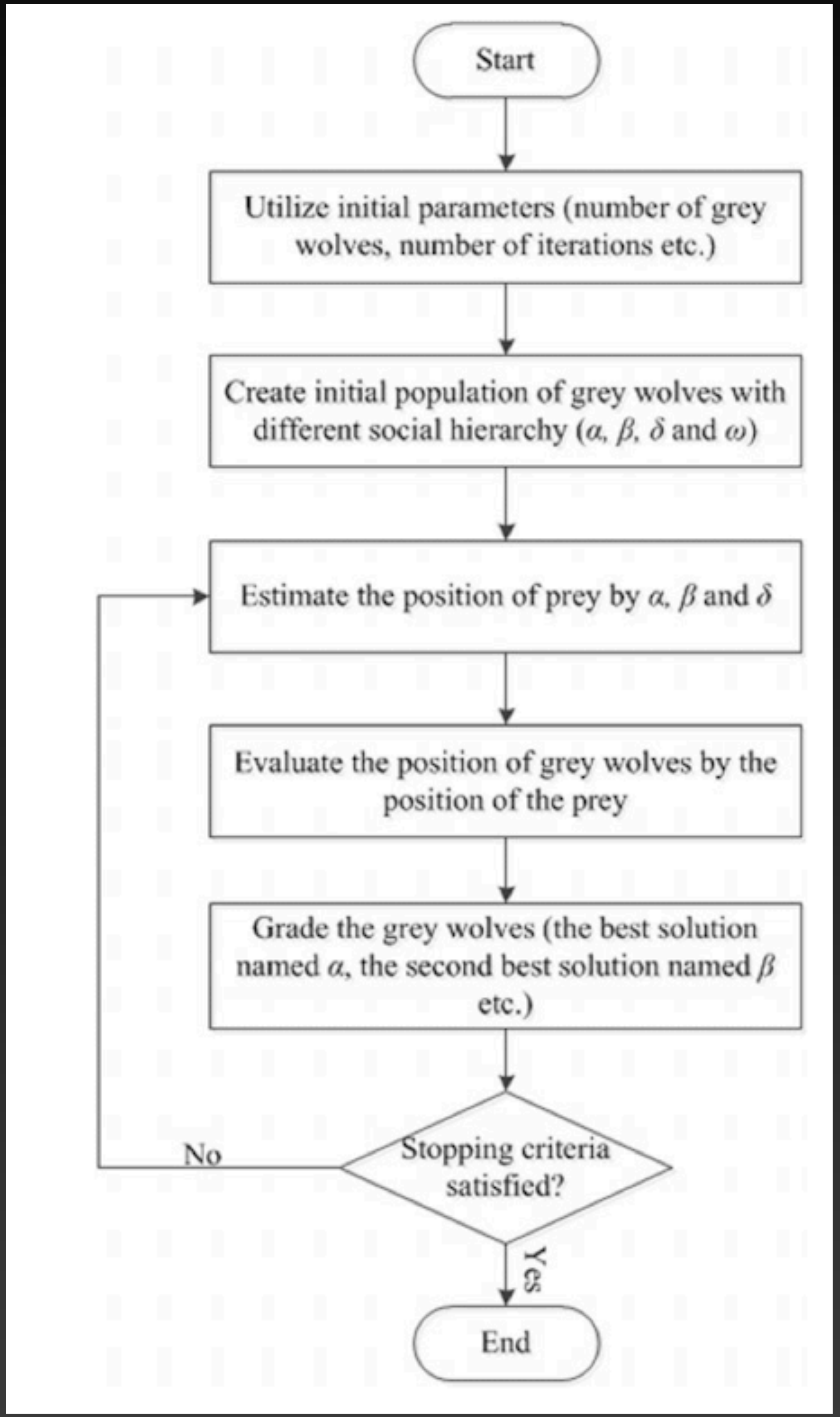
- Gray wolf hunting has three main phases:
  - Tracking, chasing, and approaching the prey
  - Pursuing, encircling, and harassing the prey until it stops moving
  - Attack towards the prey

- Mathematical model:
  - In mathematical model of the social hierarchy of wolves is mapped to the fittest solution.
  - The fittest solution is considered to be the alpha.
  - Beta and delta are the second and third best solutions respectively.
  - The rest of the candidate solutions are assumed to be omega.
  - Alpha, beta and delta lead the hunting (optimization) and omega wolves follow these three wolves.

IMPLEMENTATION

Grey Wolf Optimization

Decision Variable	Grey Wolf
Solution	Position of grey wolf
Old Solution	Old position of grey wolf
New Solution	New position of grey wolf
Best Solution	Position of prey
Fitness Function	Distance between grey wolf and prey
Initial Solution	Initial random position of grey wolf
Process of Generating New Solutions	Hunting operators, i.e., encircling and attacking



Start

Utilize initial parameters (number of grey wolves, number of iterations etc.)

Create initial population of grey wolves with different social hierarchy (α, β, δ and ω)

Estimate the position of prey by α, β and δ

Evaluate the position of grey wolves by the position of the prey

Grade the grey wolves (the best solution named α, the second best solution named β etc.)

Stopping criteria satisfied?

No

Yes

End

IMPLEMENTATION

Grey Wolf Optimization

```
BEGIN
    Initialize randomly the gray wolf population
    Find 1st, 2nd and 3rd best agents ($\alpha, \beta, \delta$)
    Set global best agent to the 1st best agent
    Calculate the fitness of each search agent
    WHILE count < max number of iterations
        FOR each search agent
            Update the position of the current search agent
        END FOR
        Update $\alpha$, $\beta$ and $\delta$
        Calculate the fitness of all search agents
        Update the best search agent, the 2nd best search agent, and the 3rd best search agent
        ADD 1 to count
    END WHILE
    RETURN the best search agent
END
```

BAT ALGORITHM

IMPLEMENTATION

Bat Algorithm

- The Bat Algorithm is based on the bats echolocation ability.
- By using echolocation bats can detect their food and preys and even distinguish between the different kinds of insects in the darkness.
- A bat emits a loud sound and listens to the echo which is created from the sound reflection from the surrounding objects.
- Sounds emitted by a bat are vary in properties and can be used depending on the hunting strategy.
- Each sound impulse lasts from 8 to 10 milliseconds and has constant frequency between 25 and 150 KHz.
- A bat can emit from 10 to 20 of supersonic impulses per second, an impulse lasts from 5 to 20 milliseconds. The number of signals emitted by a bat can be increased during a hunt to 200.

IMPLEMENTATION

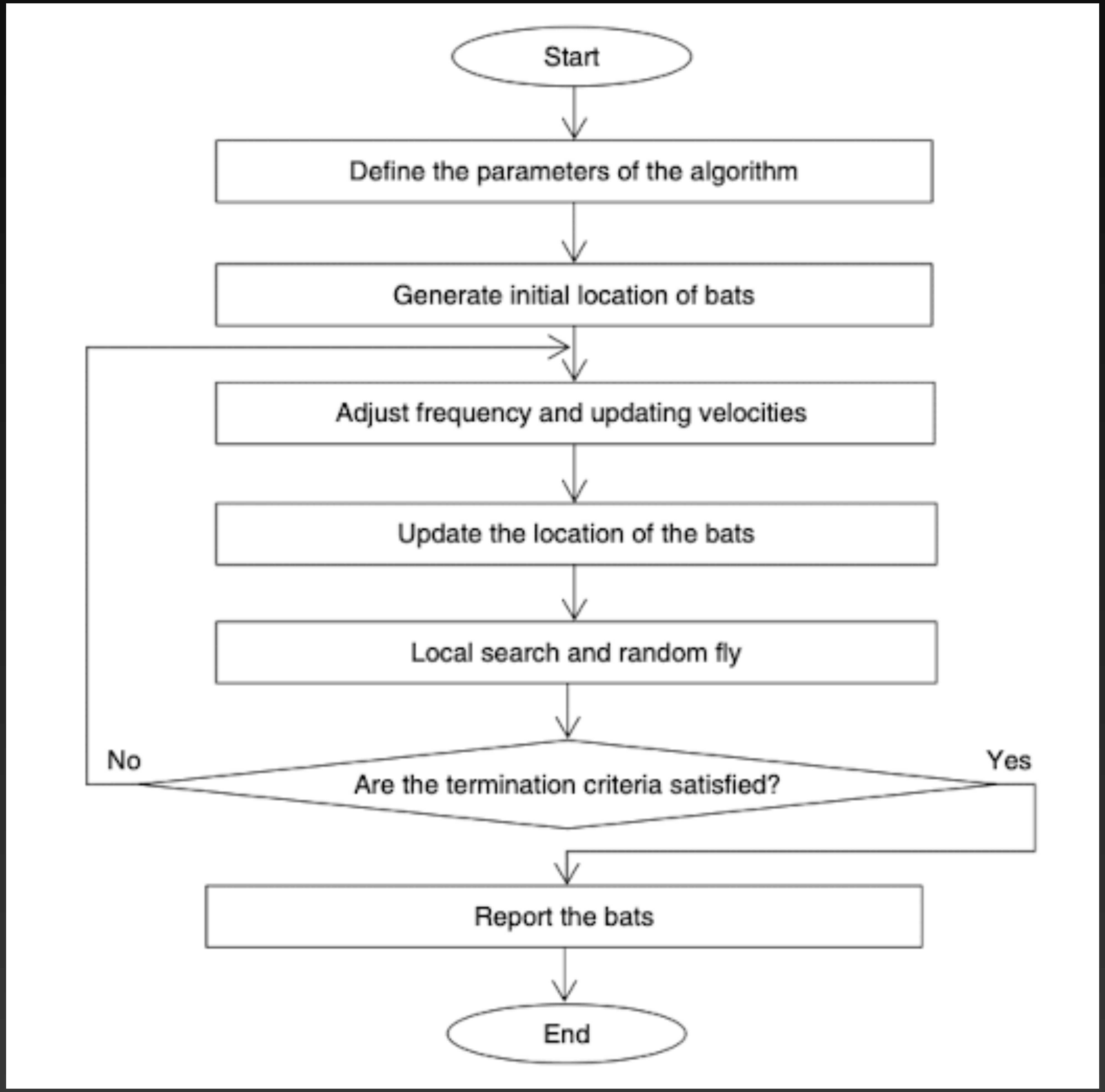
Bat Algorithm

- The Bat Algorithm uses the following principles:
  - A bat uses echolocation for distance estimation and "knows" the difference between the food/prey and an obstacle
  - Bats fly randomly with a velocity of v_i in the position x_i, fixed frequency f_{min}, variable wavelength λ and loudness A_0 for the search of a prey. They can automatically adjust the wave length (or frequency) of emitted sound impulse and level of emission $r \in [0, 1]$ depending on the proximity to the prey.
  - While the loudness can be changed by different means we assume that the loudness vary from big positive value A_0 to minimum constant A_{min}. In addition to these simplified principles, let's use the next approximations: frequency f from the segment $[f_{min}, f_{max}]$ corresponds to the wavelength segment $[\lambda_{min}, \lambda_{max}]$. For instance, a frequency segment [20 KHz, 500 KHz] corresponds to wavelength segment [0.7 mm, 17 mm].
  - The loudness usually decreases once a bat has found its prey, while the pulsation rate increases
  - A new solution is locally generated using random walk once a solution has been selected among the current best solutions. This is the local search part (random fly) of the BA.

IMPLEMENTATION

Bat Algorithm

Decision Variable	Position of bat in any dimension
Solution	Position of bat
Old Solution	Old position of bat
New Solution	New position of bat
Best Solution	Best bat
Fitness Function	Distance with food
Initial Solution	Random Bat
Process of Generating New Solutions	Fly bats


Start
Define the parameters of the algorithm
Generate initial location of bats
Adjust frequency and updating velocities
Update the location of the bats
Local search and random fly
No
Are the termination criteria satisfied?
Yes
Report the bats
End

IMPLEMENTATION

Bat Algorithm

```
BEGIN
Initialize the bat population $x_i$ (i= 1, 2, ..., n) and $v_i$
Define pulse frequency $f_i$ at $x_i$
Initialize pulse rates $r_i$ and the loudness $A_i$
    WHILE count < max number of iterations
        Generate new solutions by adjusting frequency, and updating velocities and locations/solutions
        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 < $A_i$ AND $f(x_i) < f(x^*)$
            Accept the new solutions
            Increase $r_i$ and reduce $A_i$
        END IF
        Rank the bats and find the current best $x^*$
    END WHILE
Post-process results and visualization
```

FIREFLY ALGORITHM

IMPLEMENTATION

Firefly Algorithm

- There are about 2000 firefly species, most of which produce short and rhythmic flashes.
- Usually a particular species exhibits a unique flashing pattern.
- The flashing light is generated by bioluminescence.
- It is considered that the main function of flashes are to attract fireflies of the opposite sex, evade predators and attract potential preys.
- A signal flash can communicate to a predator that a firefly has a bitter taste.

IMPLEMENTATION

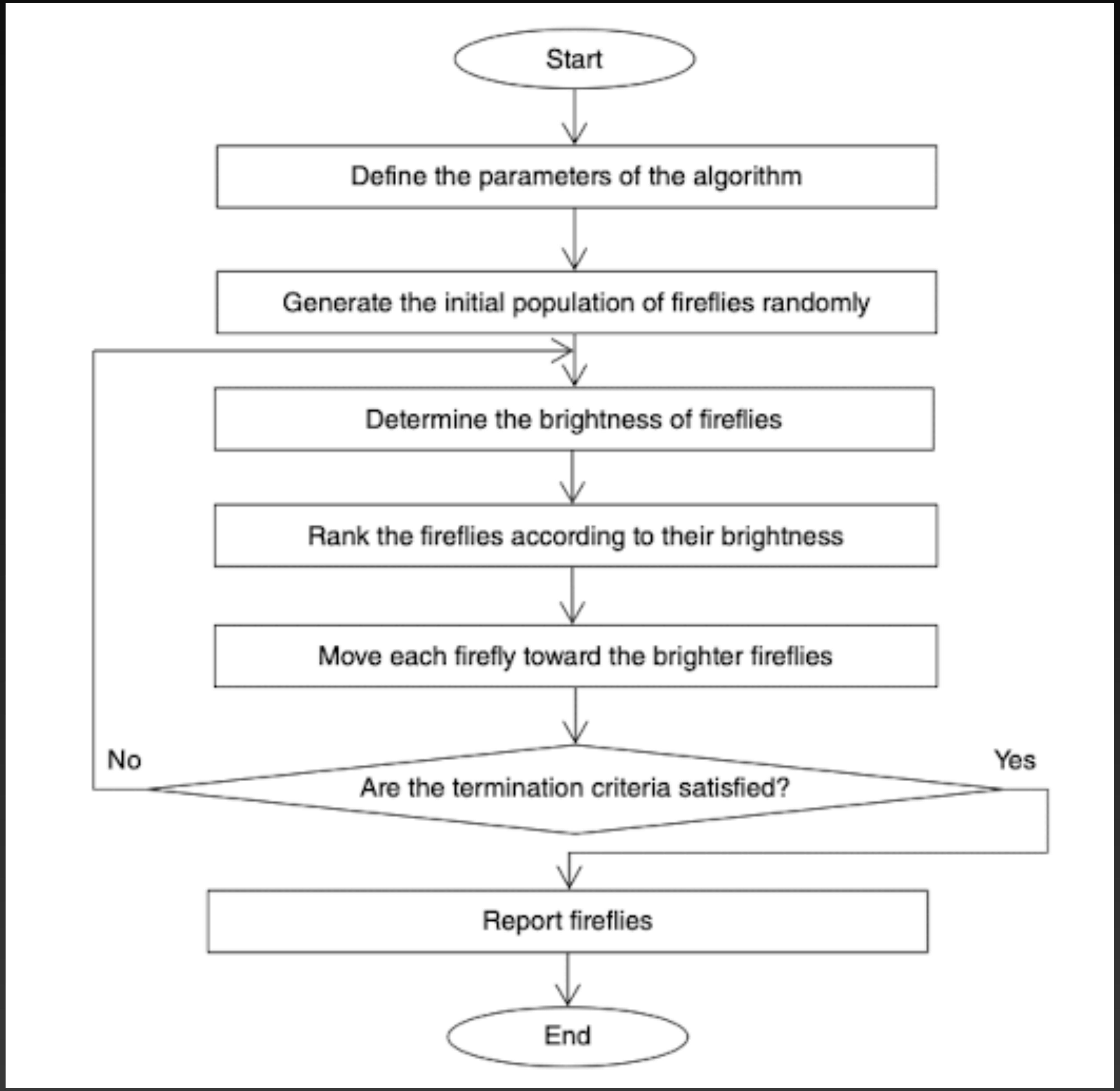
Firefly Algorithm

- The Firefly Algorithm is based on two important things: the change in light intensity and attractiveness. For simplicity, it is assumed that the attractiveness of a firefly is defined by its brightness which is connected with the objective function.
- The algorithm utilizes the following firefly behavior model:
  - All fireflies are able to attract each other independently of their gender;
  - A firefly attractiveness for other individuals is proportional to its brightness.
  - Less attractive fireflies move in the direction of the most attractive one.
  - As the distance between two fireflies increases, the visible brightness of the given firefly for the other decreases.
  - If a firefly sees no firefly that is brighter than itself, it moves randomly.

IMPLEMENTATION

Firefly Algorithm

Decision Variable	Position of firefly in each dimension
Solution	Position of firefly
Old Solution	Old position of firefly
New Solution	New position of firefly
Best Solution	Best firefly
Fitness Function	Brightness
Initial Solution	Random firefly
Process of Generating New Solutions	Movement of firefly


Start
Define the parameters of the algorithm
Generate the initial population of fireflies randomly
Determine the brightness of fireflies
Rank the fireflies according to their brightness
Move each firefly toward the brighter fireflies
Are the termination criteria satisfied?
No
Yes
Report fireflies
End

IMPLEMENTATION

Firefly Algorithm

```
Begin
    Input the parameters of the algorithm and initial data
    Generate M initial possible solutions randomly
    While (the termination criteria are not satisfied)
        Determine fitness value of all solutions
        Sort all solutions according to their fitness values
        For k = 1 to M
            For j = 1 to M
                If $F(X_j)$ is better than $F(X_k)$
                    Move the solution k toward the solution j
                End if
            Next j
        Next k
    End while
    Report all solutions
End
```

CUCKOO OPTIMIZATION ALGORITHM

IMPLEMENTATION

Cuckoo Optimization Algorithm

- The algorithm was inspired by cuckoo bird's lifestyle, egg laying features, and breeding technique.
- This as other evolutionary approaches start off with an initial population and this population consists of mature cuckoos and eggs.
- Cuckoo being a brood parasite i.e. birds that have detached themselves from the challenge of nest making for survival and evading predators, lay their eggs in other specie's nests and hence uses a cunning way to raise their families.
- A mother cuckoo lays eggs in hosts nest and destroys the host's eggs and flies away from the location fast and the caring for the egg is left to the host species. This process takes hardly ten seconds.

IMPLEMENTATION

Cuckoo Optimization Algorithm

- Therefore, they make other’s nests parasitized by their own eggs and mimic the color and the patterns of the existing eggs carefully so that the new eggs in the nest looks like the previous eggs. Each female cuckoo is specialized for a certain host species.
- Some of the birds recognize cuckoos’ eggs and either they throw the eggs out of the nest or completely leave the nest and build a new one.
- Cuckoo eggs hatch earlier than their host’s eggs. In most cases, a cuckoo chick throws the host’s eggs or the host’s chicks out of the nest.
- A cuckoo chick makes the host provide a food suitable to its growth and beg for food again and again.

IMPLEMENTATION

Cuckoo Optimization Algorithm

- COA starts with an initial population (population of cuckoos). These cuckoos have got some eggs that will be laid in other species’ nests. Some of these eggs that look like the host’s eggs are more probable to be raised and turned into cuckoos. Other eggs are detected by the host and are demised.
- The rate of the raised eggs shows the suitability of the area. If there are more eggs to be survived in an area, there is more profits to that area. Thus, the situation in which more eggs are survived will be a parameter for the cuckoos to be optimized.
- Cuckoos search for the best area to maximize their eggs’ life lengths. After hatching and turning into mature cuckoos, they form societies and communities. Each community has its habitat to live. The best habitat of all communities will be the next destination for cuckoos in other groups.

IMPLEMENTATION

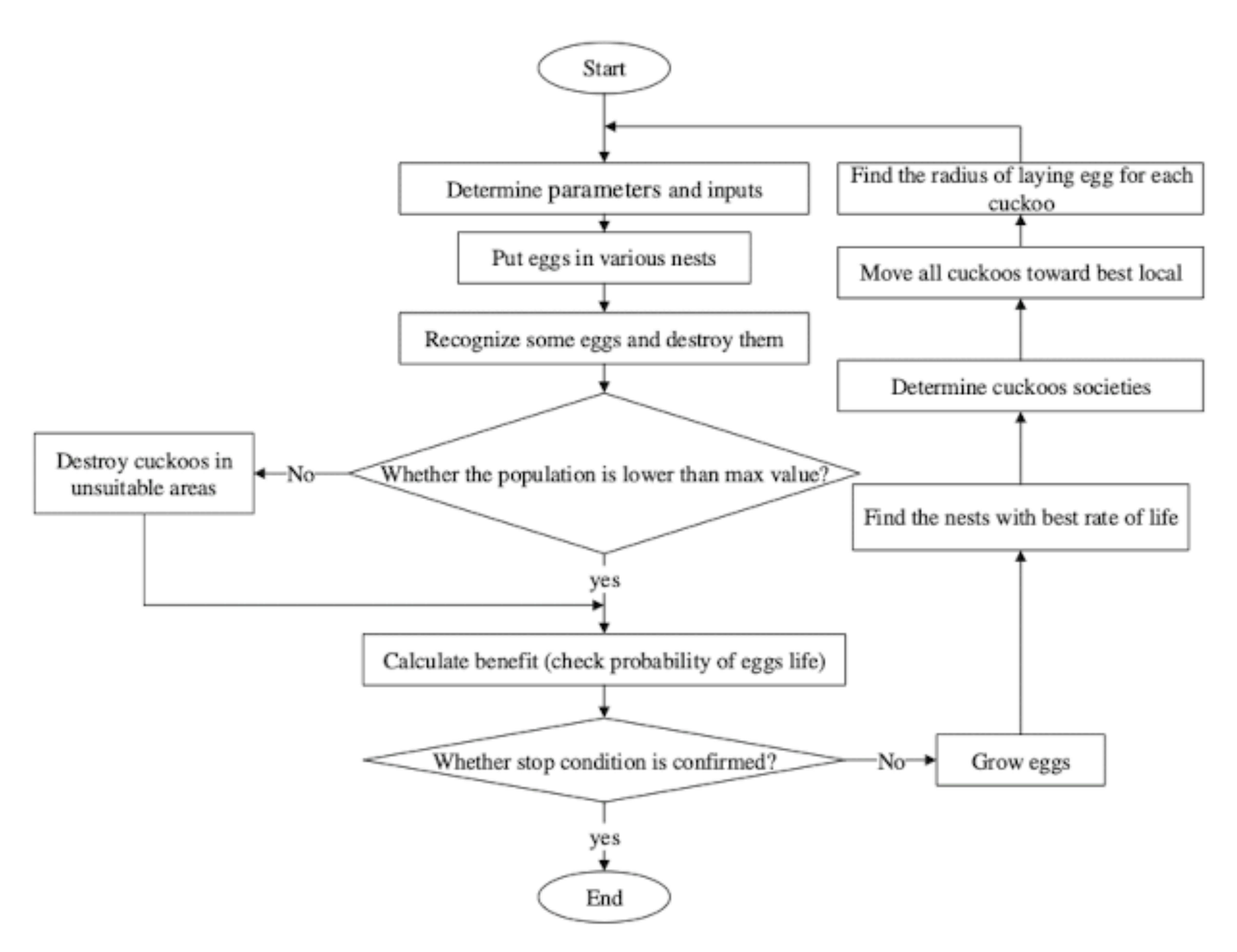
Cuckoo Optimization Algorithm

- All groups immigrate to the best current existing area. Each group will be the resident in an area near the best current existing area. An egg laying radius (ELR) will be calculated regarding the number of eggs each cuckoo lay and its distance from the current optimized area.
- Afterwards, cuckoos start laying eggs randomly in the nests within their egg laying radii. This process continues until reaching the best place for egg laying (a zone with the most profit). This optimized zone is the place in which the maximum number of cuckoos gathers together.

IMPLEMENTATION

Cuckoo Optimization Algorithm

Decision Variable	Cuckoo habitat
Solution	Habitat
Old Solution	Old habitat
New Solution	New habitat
Best Solution	Habitat with best rate of life
Fitness Function	Distance between best habitat and recent habitat
Initial Solution	Random eggs for all cuckoos
Process of Generating New Solutions	Emigration cuckoos toward best area


Start
Determine parameters and inputs
Put eggs in various nests
Recognize some eggs and destroy them
Whether the population is lower than max value?
No
Destroy cuckoos in unsuitable areas
yes
Calculate benefit (check probability of eggs life)
Whether stop condition is confirmed?
No
Grow eggs
yes
End
Find the nests with best rate of life
Determine cuckoos societies
Move all cuckoos toward best local
Find the radius of laying egg for each cuckoo

IMPLEMENTATION

Cuckoo Optimization Algorithm

```
Begin
    Define the number of habitats by generating $X_i$ for i =. 1, 2, 3,...., N
    Determine the upper and lower limits of parameters of the optimization problem ($var_{hi}$, $var_{low}$)
    Consider the maximum number of cuckoos($N_{max}$)
    Specify the maximum and minimum number of eggs($E_{max}$, $E_{min}$)
    Specify the maximum number of iterations($Iter_{max}$)
    While(Stop criterion is not satisfied)
        Define some cuckoos and assign some eggs to each cuckoo
        Calculate the radius of laying eggs for each cuckoo(ELR)
        Consider egg position or population($X_{i,new}$) for each cuckoo in ELR
        If the population is lower than the minimum number of cuckoos($X_{i,new} < N_{max}$)
            Evaluate the fitness function of each egg($F_x$)
        Else
            Destroy cuckoos in unsuitable areas
        End if
    End while
End
```

CAT SWARM OPTIMIZATION

IMPLEMENTATION

Cat Swarm Optimization

- Despite spending most of their time in resting, cats have high alertness and curiosity about their surroundings and moving objects in their environment.
- Compared to the time dedicated to their resting, they spend too little time on chasing preys to conserve their energy.
- Inspired by this hunting pattern, CSO was developed with two modes: “seeking mode” for when cats are resting and “tracing mode” for when they are chasing their prey.
- In CSO, a population of cats are created and randomly distributed in the M-dimensional solution space, with each cat representing a solution. This population is divided into two subgroups.

IMPLEMENTATION

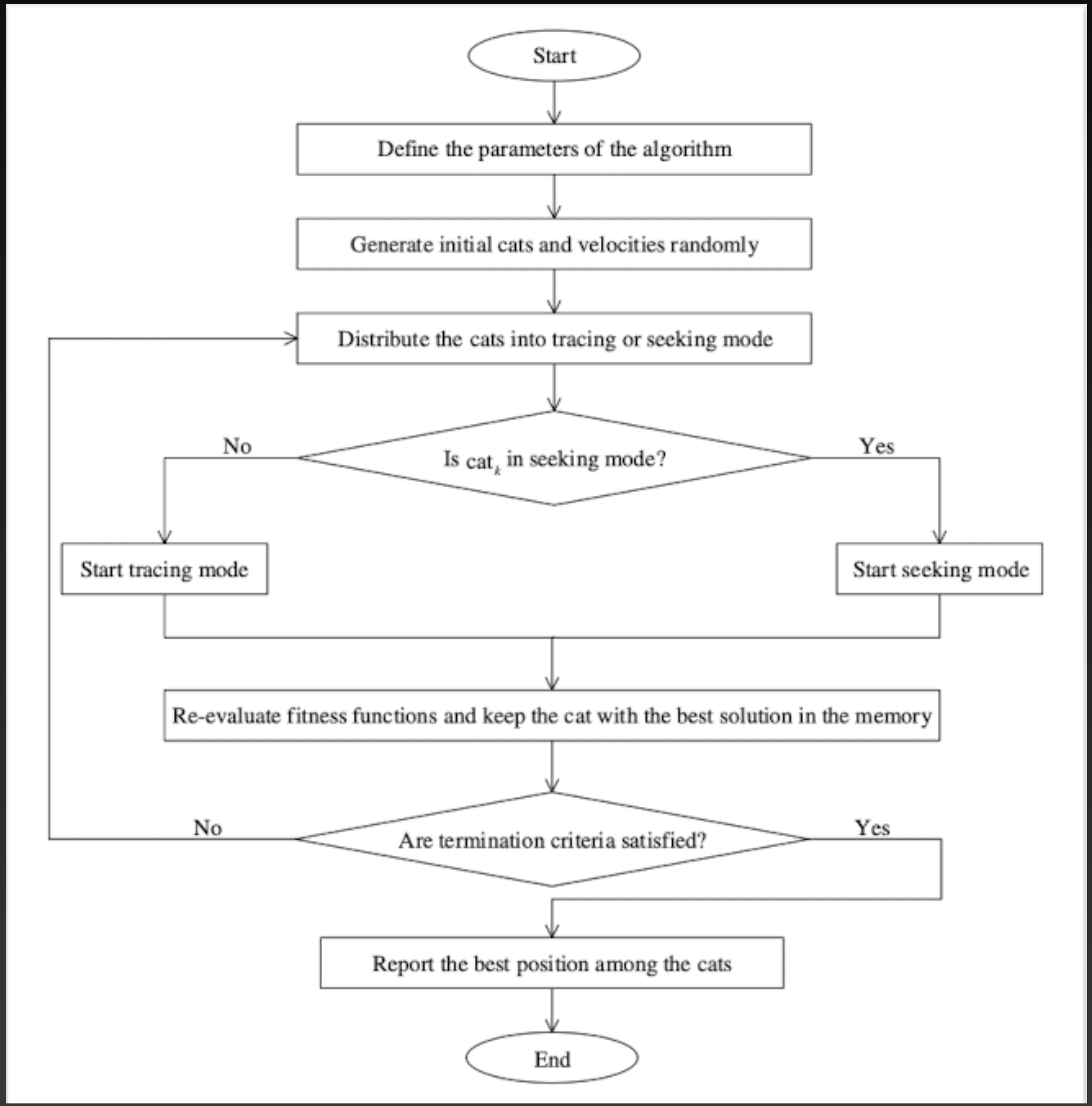
Cat Swarm Optimization

- The cats in the first sub- group are resting and keeping an eye on their surroundings (i.e., seeking mode), while the cats in the second subgroup start moving around and chasing their preys (i.e., tracing mode).
- The mixture of these two modes helps CSO to move toward the global solution in the M-dimensional solution space.
- Since the cats spend too little time in the tracing mode, the number of the cats in the tracing subgroup should be small. This number is defined by using the ***mixture ratio (MR)*** which has a small value.
- After sorting the cats into these two modes, new positions and fitness functions will be available, from which the cat with the best solution will be saved in the memory. These steps are repeated until the stopping criteria are satisfied

IMPLEMENTATION

Cat Swarm Optimization

Decision Variable	Cat’s position in each dimension
Solution	Cat’s Position
Old Solution	Old position of cat
New Solution	New position of cat
Best Solution	Any cat with the best fitness
Fitness Function	Distance between cat and prey
Initial Solution	Random position of cats
Process of Generating New Solutions	Seeking and tracing a prey


Start
Define the parameters of the algorithm
Generate initial cats and velocities randomly
Distribute the cats into tracing or seeking mode
No
Is cat_k in seeking mode?
Yes
Start tracing mode
Start seeking mode
Re-evaluate fitness functions and keep the cat with the best solution in the memory
No
Are termination criteria satisfied?
Yes
Report the best position among the cats
End

IMPLEMENTATION

Cat Swarm Optimization

```
Begin
    Input parameters of the algorithm and the initial data
    Initialize the cat population Xi (i = 1, 2, ... , n), v, and SPC
    While (the stop criterion is not satisfied or I < $I_{max}$)
        Calculate the fitness function values for all cats and sort them
        $X_g$= cat with the best solution
        For =1:N
            If SPC = 1
                Start seeking mode
            Else
                Start tracing mode
            End if
        End for i
    End while
    Post-processing the results and visualization
End
```

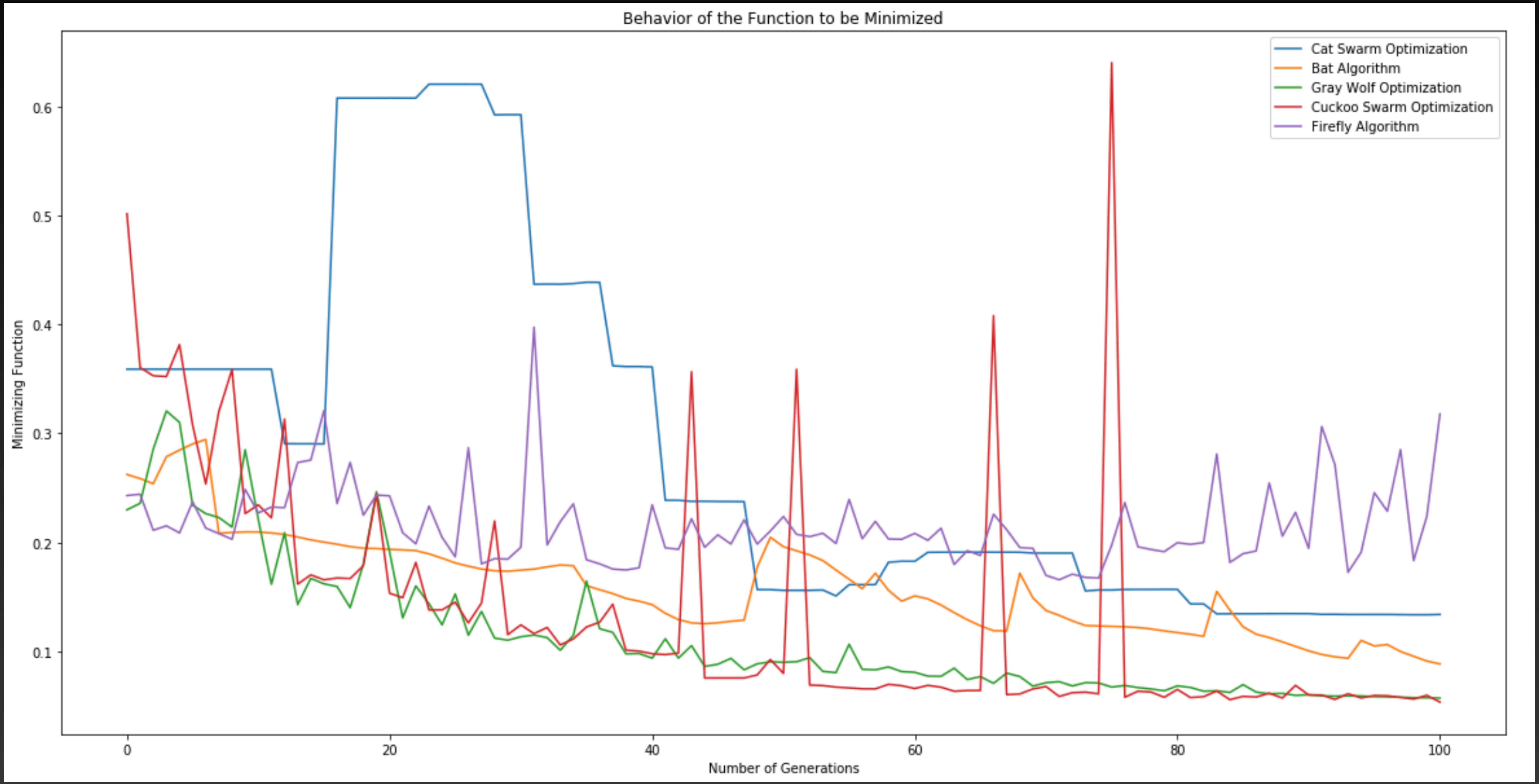
RESULTS

Metrics Analysis

Sr No.	Algorithm	Accuracy	MSE	Precision	Recall	Runtime
1	Bat Algorithm	92.7%	0.07	0.93	0.97	16.14 mins
2	Cat Algorithm	83.4%	0.16	0.87	0.87	6:03 mins
3	Grey Wolf Optimization	94.5%	0.05	0.96	0.95	6:45 mins
4	Cuckoo Search	88.6%	0.11	0.92	0.90	28:08 mins
5	Firefly Algorithm	85.9%	0.14	0.87	0.92	4.5 hrs
6	Particle Swarm	97%	0.03	0.99	0.98	6:03 mins
7	Back Propagation	98%	0.02	0.99	0.98	11:17 mins

RESULTS

Behavior of Minimization Function



CONCLUSION

- Based on the metric results and the behavior of minimization function, it can be concluded that back propagation algorithm was the best fit algorithm for this application.
- It found the best global solution and the model gave an accuracy of almost 98%.
- The best algorithm amongst the nature-inspired algorithms was the grey wolf optimization algorithm which gave an accuracy of almost 94%.

Thank You