Evaluation & Extension Of Meta- Approaches- FPA, PSO, BAT, CSA On Benchmark Functions

Deepak Garg Pardeep Kumar

Abstract: Metaheuristics have been superior methods for solving optimization problems as compared to general heuristics methods in polynomial time with less probability of getting stuck at local optimal values. Originally, Metaheuristics are inspired by nature by mimicking evolution or food-finding or mating strategies of biotic beings in nature. These algorithms do not provide a guarantee to give a global solution but some of these algorithms provide a high probability to give a nearby global solution. Through a deep literature review, it has been found out that there is a huge gap between practical expectations and theoretical algorithmic steps. Thus, this paper fills this gap and practically reviewed various famous metaheuristic algorithms proposed by the famous researcher Xin She Yang et. Al. including their Source of Mimication, input & output parameters, functioning & application areas and then compared them theoretically and practically. These algorithms have been compared on few non-convex multimodal benchmark functions corresponding to Average fitness and Minimum Fitness of population in each iteration. Furthermore, On behalf of these comparisons, made some proposal to improve and extend the performance of BAT and Cuckoo search algorithms which can be very helpful to solve NP-Hard Problems. These experiments have been evaluated using MATLAB 7.8.0 R2009a.

Keywords: Cuckoo Search, Flower Pollination Algorithm, Global Optima, Heuristics, Local Optima, Meta-heuristic, NP-Hard, Optimization Algorithm.

I. INTRODUCTION

Metaheuristics term was first introduced by F. Glover et. Al. in a research paper in the year 1986 [1]. Since then many algorithms have been proposed and evaluated under metaheuristics algorithm. The main idea of this was inspired by nature to solve NP-hard & NP-complete problems in polynomial time. Before the arrival of metaheuristics, there were traditional optimization algorithm to solve these problems. But traditional approaches lack in terms of generality and premature convergence. Generality means that traditional approaches were problem-specific i.e. not applicable to all problems without structural change. While premature convergence causes easily stucking at local optima, a famous example of this is stuck in the case of hill climbing at a local peak [2-5]. Talking about characteristics of metaheuristic algorithms it is inspired by nature, has stochastic nature, not problem-specific and last but not least it provides a good mixture of intensification and diversification. So, one of the advantages is not getting easily stuck at local optima due to all the above features as compared to other traditional heuristicsBut one of the biggest disadvantages is hidden in its one of the advantages of metaheuristic i.e. generality. The reason is in providing generality there is a need of too much parameter setting requirement. Due to these parameter settings comparisons of algorithms becomes difficult and moreover, it causes difficulty in choosing parameter values for solving a problem efficiently.

- *Deepak Garg is currently pursuing Ph.D degree program in the Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India, PH-9467210641. E-mail: erdeepakgarg21@gmail.com*
- *Pardeep Kumar is working as an Associate Professor in the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, India, E-mail: pmittal@kuk.ac.in*

The second disadvantage of metaheuristics was the lack of ideal random walk which was resolved by levy flight introduction and still lots of research is going in this direction [2-5].

1.1 Our Contribution

This paper reviewed various metaheuristic algorithms proposed by the famous senior researcher and scientist Xin She Yang et. Al. including algorithm's Source of Mimication, input & output parameters, functioning & application areas and then compared them theoretically and practically. These algorithms have been compared on few non convex multimodal benchmark functions corresponding to Average fitness and Minimum Fitness of population in each iteration including convergence rate of algorithms. Furthermore, On the behalf of these comparisons made some proposal to improve performance of BAT and Cuckoo Search which can be very helpful to solve NP-Hard Problems. These experiments have been evaluated using MATLAB 7.8.0 R2009a.

1.2 Organization

Organization or structure of the rest paper is as—section 2 describes the literature survey of articles by famous senior research scientist Xin. She. Yang et. Al. and Section 3 describes a few important metaheuristic algorithms developed by him. Section 4 describes the theoretic comparison among these on the behalf of several factors. Section 5 represents experimental settings and experimental results comparison. In section 6 few proposals as the extension has been made. Finally sect. 8 comprised conclusion & future scope of the proposed work.

II. LITERATURE SURVEY

Xin She yang is a famous senior research scientist who has developed many heuristic approaches to solve optimization problems. Out of all the approaches we have deeply surveyed original meta-heuristic algorithms that he has developed so far. Mainly he developed firefly algorithm in 2008, Cuckoo search in 2009, bat algorithm in 2010 and flower pollination algorithm in 2012 chronologically.[2-5] In 2008, Xin-She Yang et. Al. introduced the FFA (Firefly Algorithm) [6-7]. This was inspired from the glowing or

flashing nature of fireflies for mating and for prey. Each firefly has some brightness which is corresponding to fitness value in real-world problems. Phenomena is that every low brightness firefly gets attracted to high brightness fly and moves towards it, in doing so brightness of firefly increases while distance between them decreases. Light absorption or atmosphere and randomness plays salient role in brightness updation. Aim is to find position of highest brighten firefly out of whole population and its value to deal with real world problem with the help of this algorithm. In 2009, Xin-She Yang et. Al. Introduced CSA (Cuckoo Search Algorithm) [8-10]. It was motivated from the brutal breeding nature of bird species Cuckoo. The cuckoo bird picks a host bird nest then mimics host bird egg in color, texture, weight, etc and lay cuckoo egg in the host nest. Out of these two eggs the hatched egg is considered as more fit then un-hatched egg and hatched child bird drops un-hatched egg. When host bird comes then he may or may not discover alien egg with a discovery probability. If he disovered alien egg then host bird leave that nest and create new nest with new egg. As replacement or survival of fittest policy best nest is chosen out of left nest and new nest and these steps goes no upto some time or iterations with aim to find position and value of best eggs. This algorithm involves levy flight in formation of cuckoo bird. In 2010, Xin-She Yang et. Al. developed the Bat Algorithm [11] It is motivated from the micro-bat methods for finding food and navigation. When prey of obstacle is far then micro bats generate loud or high sound with low pulse rate (frequencies) which are echoed back specifying distance or direction. On the basis of this bats regenerates a varied level of sound with varied frequency with random velocity. On the basis of this generation and receiving of sound bats adjust its position, frequency, loudness. High loudness and low pulse rate depicts object is far or exploration while low loudness and high pulse rate depicts nearby object or exploitation. This process continues up to convergence criterion. In 2012, Xin-She Yang introduced FPA (Flower Pollination Algorithm) [12]. It is encouraged from the pollination procedure of flowers. Pollination (passing of pollen grains) can be occurred due to living organisms like flies, insects termed as biotic pollination (or pollination in same plant termed as self-pollination) which generally occurred in local area thus treated as local pollination or exploitation. While pollination due to non-living things like air, water is termed as abiotic pollination which generally occurred in wide area thus also termed as global pollination or exploration. Xin She yang used these local and global pollination movement controlled by switching probability. For simulation in real world it has been assumed that each plant has one flower which have only one pollen grain. Objective is to find best flower or best pollen grain for the process of pollination.

III. METAHEURISTIC ALGORITHMS

It has been observed through vast search that in many published papers only theoretical steps has been mentioned corresponding to algorithm which causes gap between theory and practical regarding algorithm [2-16]. Algorithms were explained in too general way that it does not meet practical expectation due to missing of specific details of each and every steps. Thus, in this section every textual step of algorithms (like firefly, bat, cuckoo search and flower pollination) has been explained through dummy codes corresponding to Matlab. These all algorithms have been shown for two dimension problems which can be extended as per need. All algorithms return two values first is the average fitness of population for current iteration and second is optimum or minimum fitness value found in that population for that iteration.

N_iter: Maximum iterations
Sol: Population Array of individuals
N: No. of individuals
Fun(X,Y): Objective or fitness function
LB: Lower Bound of X and Y
UB: Upper Bound of X and Y
Xbest: Array of <X, Y> values on which got minimum fitness values in every iteration from 1 to N_iter
fbest: Array of minimum fitness values in every iteration from 1 to N_iter
favg: Array of average fitness value in every iteration from 1 to N_iter.

3.1 Firefly Algorithm

Input:(N,N_iter,Sol, LB, UB,alpha,gamma, delta)
Output:[Xbest,fbest,favg]
Where, meaning and values of few notations are given below [6-7]:
alpha: For Randomness 0 to 1 (highly random)
Sol: Populations of all individuals as solution
gamma=1 (Light Absorption coefficient due to air or atmosphere)
delta=0.97 (For Reduction in Randomness (as in Simulated annealing))
beta=exp(-gamma*r2) (beta represents attractiveness between firefly i and j with distance r)
Movement: x (i)=x (i). * (1-beta) + x (j).*beta + alpha.*(rand-0.5);
Here Fun(x,y) is treated as maximization problem.
STEP 1: Evaluate Fitness

```
for i=1:Pop.Size,
 Fitness(i)=Fun(Sol(i,:));
End
xn=Sol(:,1);
yn=Sol(:,2);
range=[LB UB LB UB];
Step 2: Start Iterations
for t=1:N_iter,
% 2.1 Sort fireflies for comparison according to brightness
[Lightn,Pos]=sort(Fitness);
xn=xn(Pos); yn=yn(Pos);
xtemp=xn;ytemp=yn;  Lighttemp=Lightn;%temp variables.

% 2.2 Update firefly postions by Moving towards better fireflies (Main step)
[xn,yn]=ffalgo_move(xn,yn,Lightn,xtemp
ytemp,Lighttemp,alpha,gamma,range);

% 2.3 As iteration increases reduce the randomness
alpha=alpha*delta;

% 2.4 Update population
Sol=[xn,yn];
% 2.5 Again Evaluate Fitness to get minimum and average
for i=1:Pop.Size,
 Fitness(i)=Fun(Sol(i,:));
End
```

```
% 2.6 Update minimum & average
fmin,I]=min(Fitness);
xbest(t,:)=Sol(i,:);
fbest(t,:)=fmin;
favg(t,:)=mean(Fitness);
end
```

Where, ffa_move means firefly movement corresponding to every other brighter firefly, this is the main step of firefly algorithm which is explained below:

Step 2.2 Move all fireflies to the better locations (Main step)

```
function[xn,yn]=ffalgo_move(xn,yn,Lightn,xtemp,ytemp,Light
temp,alpha,gamma,range)
ni=size(yn,2);%comparison for firefly
nj=size(ytemp,2); %comparison to firefly
for i=1:ni,
for j=1:nj,
% i. if Brighter and more attractive
if Lightn(i)<Lighttemp(j),
% Find Cartesian distance between 2 fireflies
r=sqrt((xn(i)-xtemp(j))^2+(yn(i)-ytemp(j))^2);

% Update attractiveness beta=exp(-gamma*r^2)
beta0=1;
beta=beta0*exp(- gamma*r.^2);

% Movement of firefly i towards j
xn(i)=xn(i).*(1-beta)+xtemp(j).*beta +alpha.*(rand-0.5);
yn(i)=yn(i).*(1-beta)+ytemp(j).*beta +alpha.*(rand-0.5);
end
end % termination for jth firefly
end % termination for ith firefly
% ii. Keep in bounds
[xn,yn]=Bounds(xn,yn,range);
End
```

3.2 Cuckoo Search Algorithm

Input:(N, N_iter, Sol, LB, UB, pa)
Output:[Xbest,fbest,favg]
Where, meaning and values of few notations are given below [8-10]:
Pa= 0.25 (Discovery Probability)
Gamma(x)=factorial(x-1)
Beta=3/2 (by default)

```
Step 1: Evaluate Fitness of all eggs
for i=1:Pop.Size,
 Fitness(i)=Fun(Sol(i,:));
End
xn=Sol(:,1);
yn=Sol(:,2);
range=[LB UB LB UB];

Step 2: Find best egg in current iteration
[fmin,pos]=min(Fitness);
best_nest=Sol(pos,:);
Stemp=Sol;

Step 3: Start Iterations
for t=1:N_iter,
% 3.1: cuckoo egg generation for every nest using levy flight
(global random walk)
new_nest=generate_cuckoos(sol,best_nest,LB,UB)
;
% 3.2 Evaluate cuckoo egg & compare it with host egg to
keep better one
[fnew,best,sol,Fitness]=get_best_nest(sol,new_nest,Fitness
);
% 3.2 on discovery replace whole nest by local random walk
new_nest=Discovery_nests(sol,LB,UB,pa) ;
% 3.4 Evaluate egg after discovery & compare with egg
before discovery and keep better one on the basis of
survival of fittest theory
[fnew,best,sol,Fitness]=get_best_nest(sol,new_nest,Fitness)
% 3.5 Update minimum & average
if fnew<fmin,
fmin=fnew;
best_nest=best;
end
                    xbest(t,:)=best_nest;
fbest(t,:)=fmin;
          favg(t,:)=mean(fitness);
end
```

Where, get_cuckoos generates cuckoo egg by levy flight using Mantegna algorithm while empty_nests function replace complete nest after discovery of that nest by host bird.

```
Step 3.1: cuckoo egg generation for every nest using levy
flight (global random walk)
Function_nest=generate_cuckoos(sol,best,LB,UB)
beta=3/2;
for j=1:n,
s=sol(j,:);
% i. Levy flights by Mantegna's algorithm
Sigma=(gamma(1 + beta)*sin( pi * beta / 2 ) / ( gamma((1 +
beta ) / 2)* beta *2 ^(( beta – 1 ) / 2))) ^(1 / beta);
u=randn(size(s))*Sigma;  v=randn(size(s));
% ii. Global random walk
step= u . / abs(v). ^ ( 1 / beta); stepsize=0.01 * step
. * ( s - best); s= s + stepsize . * randn(size(s));
% iii. Bound invalid positions
nest(j,:)=bounds(s,LB,UB);
end
end

Step 3.2 on discovery replace whole nest by local random
walk
Function new_nest=Discovery_nests( sol, LB,UB,pa)
% i. Select whole nest rather than an egg
n=size(sol,1);
% ii. Alien nest Discovery rather than alien egg discovery
K=rand(size(sol,1))>pa; (promotes local random walk)
% iii. for choosing first 2 random nests
rr=randperm(n);
% iv. local random walk
stepsize=rand*(sol(rr(1),:)-sol( rr(2),:));
new_nest=sol;
for j=1:size(new_nest,1)
new_nest(j,:)=sol(j,:)+stepsize.*K(j);
s=new_nest(j,:);
% iv. Bound invalid positions
new_nest(j,:)= bounds(s,Lb,Ub);
end
end
```

3.3 BAT Algorithm

Input:(N, N_iter, Sol, LB, UB, A, r)

Output:[Xbest,fbest,favg]
Where, meaning and values of few notations are given below[11]:
A= 0.5 (Sound Loudness)
r=0.5 (Sound Pulse rate)
Freqmin=0 (Min. Frequency)
Freqmax=2 (Max. Frequency)

Step 1: Randomizing frequency, the velocity of each bat.
Freq=zeros(N,1); %related to Frequency
vel=zeros(N,dimension); %related to Velocities
Step 2: Evaluate Fitness
for i=1:N,
Fitness(i) = Fun(Sol(i,:));
end
Step 3: Find best position of bat for food
[fmin,pos]=min(Fitness);
best_bat=Sol(pos,:);
Stemp=Sol;
Step 4: Start Iterations
for t=1:N_iter,
% 4.1 Traverse & modify for all the bats
for i=1:N,
% 4.1.1 update bats corresponding to frequency, best and velocity Freq(i)=Freqmin+(Freqmin-Freqmax)*rand; vel(i,:)=vel(i,:)+(Sol(i,:)-best_bat)*Freq(i); Stemp(i,:)=Sol(i,:)+vel(i,:);
% 4.1.2 if pulse rate is less than random, select local solution around best one
if rand>r %r is pulse rate which causes exploitation
% In below 0.001 bounds size of step for walk
Stemp(i,:)=best_bat+0.001*randn(1,d);
End
% 4.1.3 Evaluate fitness of new bat
Fnewbat=Fun(Stemp(i,:));
% 4.1.4 comparison for better bat with low loudness
if (Fnewbat<=Fitness(i)) & (rand<A)
Sol(i,:)=Stemp(i,:);
Fitness(i)=Fnewbat;
End
% 4.1.5 Update the position and value of best bat
if Fnewbat<=fmin,
best_bat=Stemp(i,:);
fmin=Fnewbat;
end
end
% 4.2 update minimum and average of whole population
xbest(t,:)=best_bat;
fbest(t,:)=fmin;
favg(t,:)=mean(Fitness);
end
3.4 Flower Pollination Algorithm
Input:(N, N_iter, Sol, LB, UB, p)
Output:[Xbest,fbest,favg]
Where, meaning and values of few notations are given below[12]:
p= 0.8 (Switching Probability)
Biotic pollination: Global
Abiotic Pollination: local
Step 1: Evaluate Fitness of all pollengrains
for i=1:Pop.Size,
Fitness(i) = Fun(Sol(i,:));
end

Step 2: Find best pollen grain
[fmin,pos]=min(Fitness);
best_pollen=Sol(pos,:);
Stemp=Sol;

Step 3: Traverse & update for all pollen grains
for t=1:N_iter,
for i=1:N,
if rand>p
% 3.1 Global Pollination due to abiotic or cross pollination
Lglobal=Levy(Beta);//Levy Flight for step size
dS=Lglobal.*(Sol(i,:)-best_pollen);
Stemp(i,:)=Sol(i,:)+dS;
else
% 3.2 Local Pollination due to biotic or self pollination
epsilon=rand(1);
% in below generate integral random out of N for neighboring pollen grains to work in local pollination
Neigh=randperm(N);
Stemp(i,:)=Stemp(i,:)-epsilon* (Sol(Neigh(1),:)-Sol(Neigh(2),:));
End
Fnew=Fun(Stemp(i,:));
if (Fnew<=Fitness(i)),
Sol(i,:)=Stemp(i,:);
Fitness(i)=Fnew;
End
if Fnew<=fmin,
best_pollen=Stemp(i,:) ;
fmin=Fnew ;
end
end //Pop. Updated
% 3.3 Update min and average
xbest(t,:)=best_pollen;
fbest(t,:)=fmin;
favg(t,:)=mean(Fitness);
end

4. COMPARISON OF VARIOUS META-HEURISTIC ALGORITHMS ACCORDING TO SURVEY

Algorithms→ Attributes ↓	*Firefly*	*Cuckoo*	*Bat*	*Flower Pollination*
1. Year of Introduction	*In 2008*	*In 2009*	*In 2010*	*In 2012*
2. Global or Local or both in each individual	*Partially global*	*Both*	*Both*	*Either global or local not both*
3. Swarm based or not	*Yes*	*Yes*	*No*	*Yes*
4. Origin	*Flashing behavior of fireflies for food and mating*	*Breeding behavior of cuckoo bird*	*Echolocation for food and for navigation*	*Pollination in flower*
5. Main input Parameters	*Light absorption, randomness reduction*	*Discovery Probability*	*Loudness, pulse rate, frequency*	*Switching probability*
6. Levy Flight	*No*	*yes*	*no*	*yes*
7.Application Area	*TSP, Vertex cover, Artificial Intelligence*	*Load Scheduling, Machin*	*Fuzzy Logic, Image processing*	*Colouring Graph, Approxi-mization*

e Learning	

5. EXPERIMENTAL SETTINGS & COMPARISON RESULTS

Evaluation and comparisons of Firefly, Cuckoo, BAT & Flower Pollination algorithm have been done on 4 benchmark functions (tabulated in table 2) in MATLAB on the same initial population. As from the algorithms specified in section 3 we can get 3 outputs which are the best position, Fmin and Favg on each iteration but for the sake of convergence property, the performance of algorithms has been compared corresponding to Favg of whole population in each iteration.

.5.1 Initial Parameters

Dim= 2 (Dimension of the problem X, Y)
N=50 (population size)
N_iter: 200 (Maximum Iterations)
A=0.5 (loudness)
r=0.5 (pulserate)
Pa=0.25 (discovery Probability)
P=0.8 (Swithcing Probability)

5.2 Benchmark Functions

For experiments five benchmark functions have been used, all of these are non-convex having multimodality in nature with aim to minimize objective value [17]. These are tabulated in table 1.

Table 1: *Benchmark Functions with Value and Bounds on dimension [17]*

Name	*Function*	*Value*	*LB& UB*
Ackley N.4	$exp(-0.2) * sqrt(X^2 + Y^2) + 3 * (cos(2 * X) + sin(2 * Y))$	*-4.590101633799*	*-35 to 35*
Ackley N.3	$-200 * exp(-0.02 * sqrt((X^2) + (Y^2))) + 5 * exp(cos(3 * X) + sin(3 * Y))$	*- 195.6290282*	*-32 to 32*
Bird function	$sin(X) * exp((1 - cos(Y))^2) + cos(Y) * exp((1 - sin(X))^2) + (X - Y)^2$	*-106.764537*	*-2pi to 2pi*
Holder Table Function	$-abs(sin(X) * cos(Y) * exp(abs(1 - (sqrt(X^2 + Y^2) / pi))))$	*-19.2085*	*-10 to 10*

5.3 Comparisons among the performance of algorithms in terms of Favg

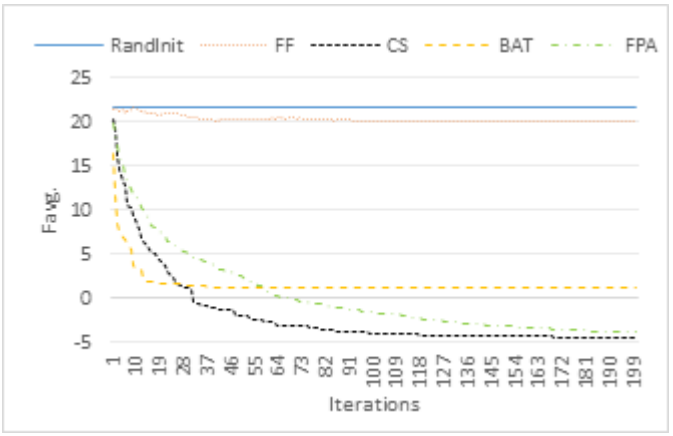

Fig. 1*: Comparison on Ackley N.4 Function*

Fig. 2: *Comparison on Ackley N.3 Function*

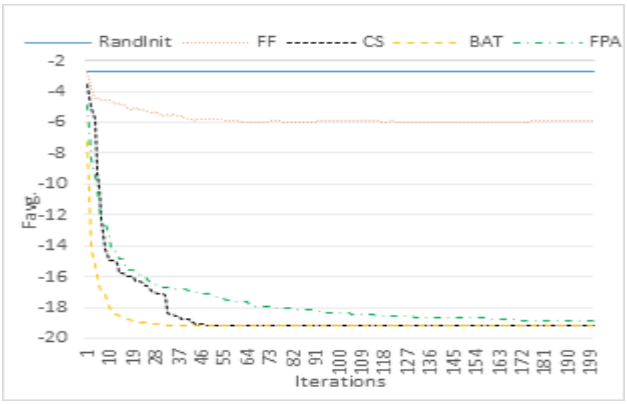

Fig. 3*: Comparison on Holder Table Function*

Table 2: *Convergence characteristics of algorithms afterwards two hundred iterations*

Name	*RandInit*	*FF*	*CS*	*BAT*	*FPA*	*Expected*
Ackley N.4	*21.628*	*20.302*	*-2.023*	*1.561*	*0.257*	*-4.590*
Ackley N.3	*-115.559*	*-117.91*	*-190.60*	*-186.589*	*-182.706*	*-195.629*
Bird function	*27.069*	*-67.966*	*-84.995*	*-103.086*	*-77.955*	*-106.764*

Holder Table	*-2.673*	*-5.742*	*-18.360*	*-18.954*	*-17.578*	*-19.208*

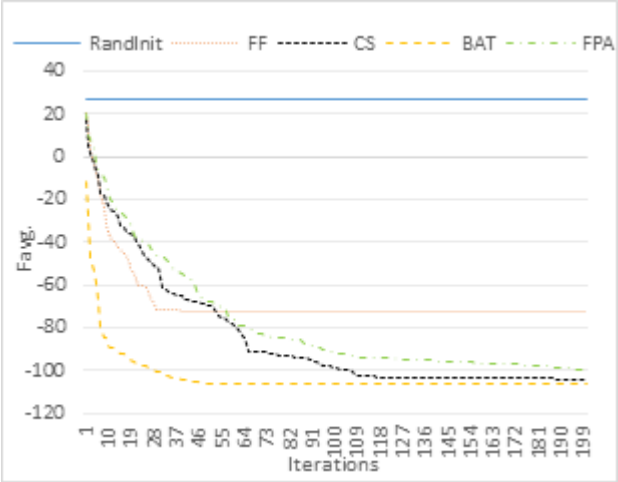

Fig. 4: *Comparison on Bird Function*

Table 3: *Convergence order among algorithm corresponding to table 2*

Function Name	*Convergence rate*
Ackley N.4	*CS, FPA, BAT, FF, RI*
Ackley N.3	*CS, BAT, FPA, FF, RI*
Bird function	*BAT, CS, FPA, FF, RI*
Holder Table	*BAT, CS, FPA, FF, RI*

From the figure 1-4, it has been clear that CS, BAT, FPA has the tendency to reach global optima values during said iterations While table 2 tabulates about convergence characteristics of these algorithms by calculating average fitness of whole population during these iterations. On behalf of this convergence order table, 3 depicts the rate of convergence regarding algorithms. BAT comes first in order in some cases while in some cases CS comes first.

6. PROPOSED WORK

One of the reasons of the bad performance of BAT as compared to CS in some benchmark function cases (as tabulated in table 2 & 3) is because it has three constant input parameters to set which is only one in case of CS. So, to get applicable on all problems, the setting of input parameters is a big headache. Furthermore, CS uses Levy's flight. While the performance of CS lacks in Bird function and holder table function (as tabulated in table 2 & 3) is due to self adaptivity, or constant step size and discovery probability. These weaknesses in BAT and CS can be removed by the following changes:

6.1 Changes in BAT:

i. Self-Adaptive Loudness

Loudness promotes exploration so according to evolution theory it should be high in starting and decreases gradually.

Ainit=0.5 & $A(t) = \frac{tmax-t}{tmax} \times Ainit$ (1)

ii. Self-Adaptive Pulse rate

Pulse rate promotes exploitation so according to evolution theory it should be low in starting and increases gradually.

r_init=0.5 & $r(t) = \frac{t}{tmax} \times rinit$ (2)

6.2 Changes in CS:

Self-Adaptive Discovery Probability

Discovery probability in CS balances exploration and exploitation. But it was constant in all iteration which was 0.25. So, needs to be adaptive (High in starting and low in end).

Painit=0.25 & $Pa(t) = \frac{tmax-t}{tmax} \times Painit$ (3)

Furthermore, Gaussian and Cauchy walk can also make good changes in performance as comparative to use of Mantegna walk only.

7. CONCLUSION

In this paper, the authors reviewed all the famous algorithms of Xin she Yang not only theoretically but practically too, explaining all the steps in Matlab coding as a sample with full documentation in section 3. On behalf of this, the comparison has been made among firefly, bat, cuckoo search, flower pollination algorithm corresponding to the origin, parameters, working, applications, levy flight, swarm nature. Then to validate results, algorithms have been compared on a few standard benchmark functions. Convergence rate has been studied and draw conclusions on behalf of this. Furthermore, after working theoretically and practically with these algorithms few major proposals have been made which helps in overcoming the performance gap of algorithms (which can be very helpful to solve NP-Hard Problems). Experiments were done using MATLAB 7.8.0 R2009a.

8 REFERENCES:

[1] Glover, F.: Future Paths For Integer Programming and Links to Artifical Intelligence. In: Comput. & Ops. Re, Pergamon Journals, vol. 13(5), pp. 533-549 (1986).

[2] Holland, J. H.: Adaption in natural and artificial systems. In: M. A. Harbor, Ed. The University of Michigan Press (1975).

[3] Yang, X. S.: Nature-inspired optimization algorithms. In: First Ed. Oxford: Elsevier (2014).

[4] Yang, X. S.: Nature-Inspired Metaheuristic Algorithms. In: Second ed., Luniver Press (2008).

[5] Garg, D. and Kumar, P.: A Survey on Metaheuristic Approaches and its Evaluation for Load Balancing in Cloud Computing. In: Springer Nature CCIS, Advanced Informatics for Computing Research ICAICR, vol. 955, pp. 585-599 (2018).

[6] Yang X. S. and He, X.: Firefly Algorithm: Recent Advances and Applications. In: Int. J. Swarm Intelligence, vol. 1(1), pp. 36–50 (2013).

[7] Yang, X. S.: Firefly Algorithm, Stochastic Test Functions and Design Optimization. In: Int. J. Bio-Inspired Computation, vol. 2(2), pp. 78–84 (2010).

[8] Yang, X. S. and Deb, S.: Cuckoo Search via L´evy Flights. In: IEEE Conference Publication World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 210–214 (2009).

[9] Yang, X. S. and Deb, S.: Engineering Optimisation by Cuckoo Search. In: Int. J. Mathematical Modelling and Numerical Optimization, vol. 1 (4), pp. 330–343 (2010).
[10] Yang, X. S. and Deb, S.: Cuckoo search: recent advances and applications. In: Neural Computing and Applications, Springer, vol. 24 (1), pp. 169–174 (2014).
[11] Yang, X. S.: Bat algorithm: literature review and applications. In: Int. J. Bio-Inspired Computation, vol. 5 (3), pp. 141–149 (2013).
[12] Yang, X. S. and Karamanoglu, M.: Multi-objective Flower Algorithm for Optimization. In: International Conference on Computational Science, Elsevier Science, pp. 861–868 (2013).
[13] Garg, D., Garg, P.: Basis Path Testing Using SGA & HGA With ExLB Fitness Function. In: Elsevier, Procedia Computer Science, Vol. 70(1), pp. 593-602 (2015).
[14] Shakya, A. ku., Garg, D., Nayak, P. Ch.: Hybrid Live VM Migration: An Efficient Live VM Migration Approach in Cloud Computing. In: Springer Nature CCIS, Advanced Informatics for Computing Research ICAICR, Vol. 955(1), pp. 600-611(2018).
[15] Nayak, P. Ch., Garg, D., Shakya, A. Ku., Saini, P.: A research paper of existing Live VM Migration and a Hybrid VM Migration approach in Cloud Computing. In: IEEE 2nd International Conference on Trends in Electronics and Informatics (ICOEI 2018), pp. 721-726 (2018).
[16] Harkawat, A., Kumari, S., Pharkya, P., Garg, D.: Load Balancing Task Scheduling Based on Variants of Genetic Algorithm: Review Paper. In: Springer Nature CCIS, ICICCT, 750(1), pp.318-325 (2017).
[17] Dieterich, J. M. and Hartke, B.: Empirical review of standard benchmark functions using evolutionary glabal optimization. In: Appl. Math., vol. 3(10), pp. 1552–1564 (2012).
[18] Garg, D. and Kumar, P.: Evaluation and Improvement of Load Balancing Using Proposed Cuckoo Search in CloudSim In: Springer Nature CCIS, Advanced Informatics for Computing Research ICAICR, vol. 955, pp. 343-358 (2019).