Swarm Optimization
Optimization using Swarm Intelligence

Nisarg Dave, Data Scientist, Michigan Technological University

***Abstract*—This Report Demonstrates the Evolutionary optimization approach using Particle Swarm Optimization and Artificial bee colony algorithm. The PSO is optimization algorithm, which iteratively solves the optimization function. The paper discusses that how to code Particle swarm optimization. Moreover It illustrates the Parameter tuning process for getting optimal parameters. I did test the algorithm on major benchmark functions. Moreover, I tuned the PSO for getting faster convergence. Next I took one more function and tested the same parameters on that function. At last I changed parameters for each benchmark functions and then I tried to optimize each of them separately. Overall I did compare the generalized and personalized parameters for each function. In the end, I developed optimization code using the Artificial Bee colony algorithm and performed the same analysis. The whole procedure and detailed analytics is discussed and presented in this paper.**

***Index Terms*—Particle Swarm optimization,Artificial bee colony,Swarm Intelligence, Evolutionary computation, Evolutionary optimization**

I. Introduction

THIS paper describes the overall process of applying Evolutionary optimization using different algorithms such as Particle swarm optimization and Artificial bee colony. Each phase of algorithm and process is discussed in detail. From the bechmark test to the Fbest's convergence, everything is presented in detailed manner. The performance comparative study with parameter tuning is the most important part of the paper

A. Particle Swarm Optimization

The Optimization is performed by particle swarms. The particles uses velocity and distance to converge in desired direction. PSO searches for the candidate solution Iteratively with the bounded hyperspace.

II. Part I

III. Paricle Swarm Optimization Process and Algorithm

The Particle swarm optimization using Constriction parameter K

A. Initialization

Initializing the n generations of particle using population within the xrange. Initially population parameter is 100 then 100 particles will be initialized.

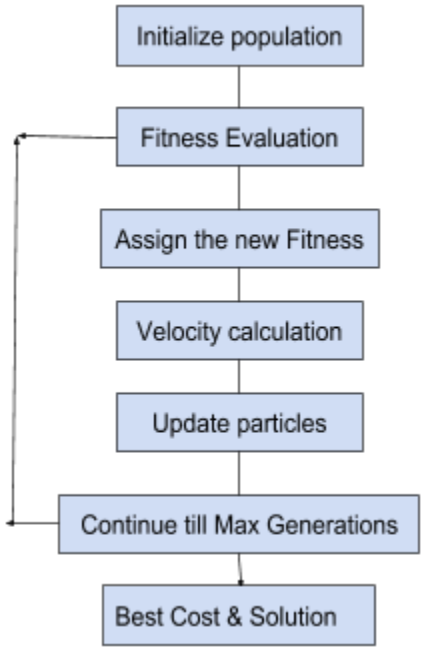

Fig. 1: PSO Generic process

B. Initial Guess

We need to start particle swarms at certain point. Randomly taken point is better but randomly directed point is even better so I took it as,

$$InitialGuess_x = 0.1 * LowerLimit_x + 0.1 * UpperLimit_x$$

C. Directional weights

Three Different kind of weights are considered.

1) Current Direction Weight: Randomly directing particle swarms into current state of directional vector.

2) Current best Weight: Randomly directing particle swarms into Currently found global minimum.

3) Current best Weight: Randomly directing particle swarms towards local minima.

D. Constriction Parameter

$$\phi = globalBestWeight + localBestWeight$$
$$K = 2/(2 - \phi - \sqrt{(\phi^2 - 4 * \phi)})$$

E. Velocity and Inertia Weight

Using the w inertia weight parameter let's get the initial X and V values.

$X1 = w * X0 + (1 - w) * X1$
$X2 = w * X0 + (1 - w) * X2$
$V = X2 - X1$

F. F and X optimum

After getting the K factor and all other parameters now the next step is to calculate the F_Global and X_Global

- Generating random vectors and calculating updated velocity
  $V = K * (currentdirectionWeight * V + globalBestWeight * r1. * ((X_Global * ones(1, m)) - X) + +localBestWeight * r2. * (X_Best - X))$
- Updating the new location and fitness

G. Algorithm

Data: Fitnessfunc and range
Result: F_Best and X_Best
Initialization;
Randomly initialize the particles using the range with tolerance parameter Decode and calculate the function values and transformed values;
while *generationNum is less than n* **do**
  Cycle new $Velocity$ and $distance$;
  get best F and X;
  Cycle through next generation particles;
  if *Elitism* **then**
    $accelerate$ best n particles;
    return elite;
  else
    repeat the process till the best F_Best found;
  end
end

Algorithm 1: Particle Swarm Optimization

IV. Experiment with parameter selection

A. Initial Inertia Weight

The weight at the starting position to decide the initial direction has positive value. I did set that to 0.1 to get the optimal randomly directed start. (lower value has higher hyperspace). Exploration is more important during initial iterations.

B. Current directional Weight

For all the functions we need to sort of exploration in starting so I kept inertial weight low and current directional weight high.
$currentdirectionWeight = 0.9$

C. Global directional Weight

Convergence to the global minima is target so keeping swarm into right direction is achieved by the global directional weight. Setting the global directional weight high is desired.
$globalBestWeight = 0.9$

D. Local directional Weight

The particles needs to explore for better optimization and so keep moving them to current direction or current global minima can lead to inefficient algorithm. So I kept the local directional weight to moderate level,
$localBestWeight = 0.6$

All the parameters were selected based on combinatory result on all 6 benchmarks. I tried certain combinations and certain trial/run type analysis for choosing the right parameters.

V. Parameter tuning and Analysis of Results

Mean and Standard deviation of F_Best		
Benchmark	Mean of F_Best	Standard Deviation of F_Best
Ackley	1.19	0.89
Branin	5.31	6.29
Dejong	0.00692	0.0172
Rosenbrock	2.81	7.62
Rastrigin	1.99	1.568
Sum of Powers	0.018	0.0081

X_Best and X_Worst		
Benchmark	X_Best	X_Worst
Ackley	(0.0103,-0.5245,0.4492)	(0.3778,0.2718,0.5944)
Branin	(3.0044,1.3181)	(0.5104,1.7222)
Dejong	(0.256,0.036,-0.129,0.198,0.0717,-0.0418,-0.0157,-0.2909,0.245,0.143)	(0.0932,0.0501,-0.0862,-0.1,-0.0589,0.0919,-0.0578,0.0664,0.0939,-0.027)
Rosenbrock	(0.5027,0.9703,0.39,0.98,-0.12)	(0.63,0.56,0.44,0.53,0.57)
Rastrigin	(-0.0191,0.00593,0.4311,-0.2847)	(-0.0002,0.015,0.058,0.01)
Sum of Powers	(0.0702,-0.0061,-0.0444,0.0004,0.197)	(0.0032,0.0067,-0.0141,-0.0024,-0.0097)

A. Analysis

As we can see the results are quite good. For the selected set of parameters it works great!

F_Best converges smoothly. We are getting the decreased F value per iteration which is great. The f value is non-increasing. Results for F_Global are also satisfactory. Resulting near the minima with very less variance.

VI. Personalized parameter tuning

In previous section, I did tune the optimal parameters for all the benchmarks. One set of parameters for all of them. Besides of that generalised parameters, let's see the effect of tuning each benchmark separately. So I did the same experiment for each of the benchmark function.

A. Ackley's parameter set

- currentdirectionWeight = 0.8
- globalBestWeight = 0.95
- localBestWeight = 0.5
- StartfromGuess = true
- Weight x0 = 0.9

B. Branin's parameter set

- currentdirectionWeight = 0.65
- globalBestWeight = 0.9
- localBestWeight = 0.75
- StartfromGuess = true
- Weight x0 = 0.9

C. Dejong's parameter set

- currentdirectionWeight = 0.9
- globalBestWeight = 0.9
- localBestWeight = 0.9
- StartfromGuess = true
- Weight x0 = 0.95

D. Rosenbrock's parameter set

- currentdirectionWeight = 0.5
- globalBestWeight = 0.8
- localBestWeight = 0.4
- StartfromGuess = true
- Weight x0 = 0.8

E. Rastrigin's parameter set

- currentdirectionWeight = 0.8
- globalBestWeight = 0.95
- localBestWeight = 0.5
- StartfromGuess = true
- Weight x0 = 0.95

F. Sum of Power parameter set

- currentdirectionWeight = 0.9
- globalBestWeight = 0.8
- localBestWeight = 0.6
- StartfromGuess = true
- Weight x0 = 0.98

VII. testing on Gramacy and Lee's Benchmark

Let's take the new functions that is not previously used in any analysis. Now let's perform the same experiments for generalised and personalised both the parameters. I took the one dimensional Gramacy and Lee's benchmark function.

Input domain: Evaluated in $x \in [0.5, 2.5]$

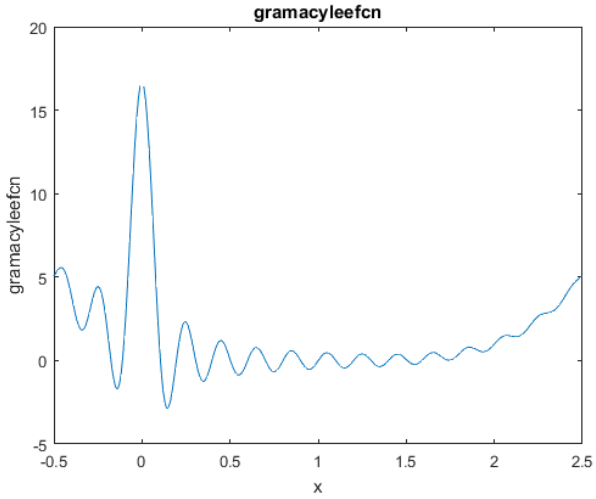

Fig. 2: Gramacy and Lee's Benchmark

A. Code in MATLAB

```
function [y] = grlee(x)
        term1 = sin(10*pi*x) / (2*x);
        term2 = (x-1)^4;
        y = term1 + term2;
end
```

B. Procedure for coding

- Input argument x
- Calculate term1 = sin(10*pi*x) / (2*x); using every x
- In same way calculate term2
- Sum up term1 and term2
- Return y

C. Results

Mean and Standard deviation of F_Best		
Benchmark	Mean of F_Best	Standard Deviation of F_Best
Gramacy and Lee	0.392	0.0000001

X_Best and X_Worst		
Benchmark	X_Best	X_Worst
Gramacy and Lee	(0.407003)	(0.408)

VIII. Part II

IX. Artificial Bee Colony Optimization

The Artificial Bee colony is the optimization problem based on intelligent forging of the honey bee swarm. The algorithm which is inspired by nature to calculate global minima just like particle swarms. The algorithm is part of swarm intelligence and thus similar in nature to PSO.

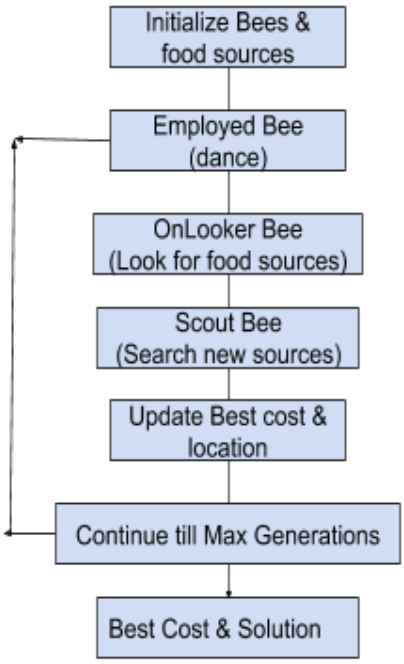

Fig. 3: Artificial Bee Colony Generic process

X. ABC Modelling

The colony has 3 general groups of bees.

- Employed Bees
- Onlookers
- Scouts

A. Critical Modelling constraints

- Each food source has only one artificial employed bees. The employed bee and number of food sources are similar.
- The employed bees go to their food source and then come to dance to this area.
- The abandoned employee bee becomes the scout.
- Onlookers keep track of the dance and chooses the food source according to dances.

B. Process

Initialization Phase

$REPEAT$

1. Employed Bees Phase

2. Onlooker Bees Phase

3. Scout Bees Phase

Result: Memorize the best solution achieved so far

Termination: UNTIL(Cycle=Maximum Cycle Number)

1) Initialization: $bee = repmat(init, total_population, 1)$

2) Scout Phase: Initiate the Food source search by scout bees and Report the location and cost found for the same.

3) Employed Phase: Browsing through all population, reporting results,
$\phi = Acceleration * unifrnd(dimension)$
Reporting location,
$newbee.loc = min(max(bee(i).loc + \phi. * (bee(i).loc - bee(k).loc), xmin), xmax$
Reporting $Cost_Best$,
$newbee.cost = fitnessfunc(newbee.loc)$

4) Onlooker Phase: Find best available food source and calculate fitness,
$F(i) = 1/(1 + bee(i).cost)$
Calculating next selection probability,
$P = F/sum(F)$

5) Result Phase: Getting Global best cost and location,
$globest = bee(i)$
$bestcost(iter) = globest.cost$

Data: Fitnessfunc and range
Result: $Cost_Best$ and X_Best
Initialization;
Randomly initialize the Bees using the range with tolerance parameter Decode and calculate the function values and transformed values;
while *generationNum is less than n* **do**
  Cycle Employee bees: Dance as per nectar amount ;
  Onlookers, look for new food sources;
  Scout bees, search for new sources;
  get $Cost_Best$ and X;
  Cycle through next available best sources;
  if *Elitism* **then**
    $accelerate$ best n locations;
    return elite;
  else
    repeat the process till the best $Cost_Best$ and loc_Bestfound;
  end
end

Algorithm 2: Artificial bees colony

XI. Experiment on Benchmarks

Mean and Standard deviation of F_Best		
Benchmark	Mean of F_Best	Standard Deviation of F_Best
Ackley	0.1081	0.4315
Branin	0.4916	1.59
Dejong	0.0148	0.1265
Rosenbrock	1.21	1.99
Rastrigin	0.81	2.44
Sum of Powers	0.0312	0.1749
Gramacy Lee	0.01	0.5

X_Best and X_Worst		
Benchmark	X_Best	X_Worst
Ackley	(-0.0005,-3.204,-0.0001)	(0.0198,1.6497,-0.2383)
Branin	(2.0304,3.9456)	(5.234,8.684)
Dejong	(-0.0003,-5,0,-5,0,1.322,0.496,0.934,-3.14,2.64)	(-4.72,-4.56,-1.88,0.31,5.93,0.98,0.209,-1.42,0.095,4.92)
Rosenbrock	(-0.5,-1.99,1.11,0.576,0.9898)	(-6.5,1.1,3.121,0.786,2.69)
Rastrigin	(0.0096,2.298,0.0099,-3.006)	(4.2979,2.1865,-2.7918,1.7378)
Sum of Powers	(-0.0003,-0.3929,0.003,-0.8503,-0.5)	(-0.0843,-3.1541,-0.1448,-0.3601,-1.5)
Gramacy Lee	(-0.001,-5.7322;0.1,-5.73)	(0.792,0.798;0.16,1.65)

XII. Conclusion

The Particle swarm optimization performs well generally but after the parameter tuning for each benchmark it works even more better. The Convergence is happening rapidly. When I try to test predefined parameters on new function then also it work well.

Moreover, another swarm intelligence algorithm Artificial bee colony performs even better when we have lesser dimension. For higher dimension the tuning is necessary to get accurate results. The results are quite good compared with PSO because all the benchmark function was tested on the same interval throughout the each step.

References

[1] Particle Swarm Optimization,Chemometrics and Intelligent Laboratory Systems, Volume 149, Part B

[2] Clever Algorithms: Nature-Inspired Programming Recipes By Jason Brownlee PhD

[3] A global best artificial bee colony algorithm for global optimization Journal of Computational and Applied Mathematics,Volume 236, Issue 11

[4] On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing,Volume 8, Issue 1,