Optimization Methods - Mid-Term Project 3

Implement of Meta-Heuristic Algorithms: Cuckoo Search, Bat Algorithm, and Flower Pollination Algorithm

Jay Chiehen Liao (廖傑恩)†

Institute of Data Science
National Cheng Kung University
Tainan City, Taiwan
re6094028@gs.ncku.edu.tw

Abstract

This project aimed to implement three well-known meta-heuristic algorithms: cuckoo search (CS), bat algorithm (BA), and flower pollination algorithm (FPA). We found that three algorithms could have a promising performance generally. It might need more runs to be converged when training BA. The time cost of BA was the highest while the differences of time cost among three algorithms were not so large, which might not matter when the number of training runs was not large. We also tuned λ in CS and FPA with the findings of the weakness of larger λ values. All codes of implements and experiment procedures are available on GitHub: https://github.com/jayenliao/Optimization-CS-BA-FPA. A demo video of training process is available on YouTube: https://youtu.be/hlKvODBUyeI.

Keywords

Global optimization, meta-heuristic algorithms, cuckoo search (CS), bat algorithm (BA), flower pollination algorithm (FPA)

1 Introduction

Global optimization is crucially important in many applications but global optimization problems are challenging to be solved because these problems are often highly nonlinear with multiple local optima. Thus, traditional methods such as the gradient-based methods usually struggle to deal with such problems. In recent years, many researchers have proposed some new optimization algorithms (Horst & Panos, 2013; Kavousi-Fard et al., 2014; Su & Lee, 2003).

Nowadays, all modern stochastic algorithms fall under the category of meta-heuristics. It is understood that these algorithms are the only practical solution to obtain global optimal for real world problems which are nonlinear, non differentiable, continuous and real valued (Marichelvam & Geetha; Yang, 2010; Price et al., 2006). Among these algorithms, Cuckoo search (CS) (Yang et al., 2007), bat algorithm (BA) (Yang 2010), and flower pollination algorithm (FPA) have been applied to solve global optimization problems. These algorithms have been widely used to solve unconstrained and constrained problems and their applications. However, few works have been applied to solve minimax and integer programming problems via these algorithms.

1.1 Cuckoo Search (CS)

CS (Yang et al., 2009) is a population-based and nature-inspired meta-heuristic algorithm. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms.

CS uses a balanced combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (x_j^t - x_k^t),$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ϵ is a random number drawn from a uniform distribution, and s is the step size. Here, the notation $\otimes$ means the entry-wise product.

On the other hand, the global random walk is carried out by using Lévy flights, $x_i^{t+1} = x_i^t + \alpha L(s, \lambda)$, where $L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\lambda \pi / 2)}{\pi} \frac{1}{s^{\lambda+1}}$, $s >> s_0 > 0$. Here $\alpha > 0$ is the step size scaling factor, which should be related to the scales of the problem of interest.

1.2 Bat Algorithm (BA)

BA (Yang, 2010) is inspired by the echolocation behavior of micro-bats. It is the first algorithm of its kind to use frequency tuning. Each bat is associated with a velocity v_i^t and a location x_i^t, at iteration t, in a d-dimensional search or solution space. Among all the bats, there exists a current best solution x_*. Therefore, the preceding three rules can be translated into the updating equations for x_i^t and velocities v_i^t:

$$f_i = f_{\min} + \beta(f_{\max} - f_{\min}),$$
$$v_i^t = v_i^{t-1} + f_i(x_i^{t-1} - x_*),$$
$$x_i^t = x_i^{t-1} + v_i^t,$$

where $\beta \in [0,1]$ is a random vector drawn from a uniform distribution.

The loudness and pulse emission rates are regulated by the following equations:

$A_i^{t+1} = \alpha A_i^t$ and $r_i^{t+1} = r_i^0[1 - e^{-\gamma t}]$, where $0 < \alpha < 1$ and $\gamma > 0$ are constants. In essence, here α is similar to the cooling factor of a cooling schedule in simulated annealing.

1.3 Flower Pollination Algorithm (FPA)

FA (Yang, 2012) is inspired by the flower pollination process of flowering plants. It has been extended to multi-objective optimization problems and found to be very efficient (Yang, 2013). For simplicity, we use the following four rules:

1. Biotic and cross-pollination can be considered global pollination process, and pollen-carrying pollinators move in a way that obeys Lévy flights.
2. For local pollination, abiotic pollination and self-pollination are used.
3. Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
4. The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0,1]$, with a slight bias toward local pollination.

To formulate updating formulae, we must convert these rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range. Therefore, Rule 1 and flower constancy can be mathematically represented as $x_i^{t+1} = x_i^t + \gamma L(\lambda)(g_* - x_i^t)$, where x_i^t is the pollen i or solution vector x_i at iteration t, and $\boldsymbol{g}_*$ is the current best solution found among all solutions at the current generation or iteration. Here γ is a scaling factor to control the step size.

Here $L(\lambda)$ is the parameter that corresponds to the strength of the pollination, which essentially is also a step size. Since insects may move over a long distance with various distance steps, we can use a Lévy flight to mimic this characteristic efficiently. That is, we draw $L > 0$ from a Lévy distribution:

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\lambda\pi/2)}{\pi}\frac{1}{s^{\lambda+1}}, \; s >> s_0 > 0.$$

Here $\Gamma(\lambda)$is the standard gamma function, and this distribution is valid for large steps $s > 0$. This step is essentially a global mutation step, which enables us to explore the search space more efficiently.

For the local pollination, both Rules 2 and 3 can be represented as $x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t)$, where x_j^t and x_k^t are pollen from different flowers of the same plant species. This essentially mimics the flower constancy in a limited neighborhood. Mathematically, if x_j^t and x_k^t come from the same species or are selected from the same population, this equivalently becomes a local random walk if we draw ϵ from a uniform distribution in [0,1]. In essence, this is a local mutation and mixing step, which can help to converge in a subspace.

1.4 Objectives

In the presenting project, we aim to implement three algorithms. We would like to conduct numerical experiment with the given setting of the objective function, hyper-parameters, and the optimal points. The followings are purposes of this project:

1. Compare the convergence and the efficiency of three algorithms.
2. Compare the convergence and the efficiency with different dimensions of points.
3. Tune lambda value in training CS and FPA.

2 Methods

2.1 Problem Statement

This project aims to implement CS, BA, and FPA for the following two objective functions in CEC 2014. Two functions are defined as below:

1. Ackley's function:
$$F(x) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi)} + 20 + e$$
2. Shifted and Rotated Weierstrass Function
$$f(x) = \sum_{i=1}^{D}\sum_{k=0}^{20}[0.5^k \cos(2\pi 3^k(x_i + 0.5))] - D'$$
$F(x) = f(0.5x/100)$, where D is the dimension of vector x (i.e., $x \in \mathbb{R}^D$) and $D' = D\sum_{k=0}^{k\max}[0.5^k \cos(\pi 3^k)]$.

In this project, we repeated each algorithm 20 times by independently regenerate the 50 initial points over $[-20, 20]^d$ and $d = 10, 20$. We tried to identify the proper parameters for each algorithm and summarized the results included mean and standard deviations of the best function values based on these 20 independent replications.

2.2 Hyper-parameters Settings

All hyper-parameters in 3 algorithms were set as Table 1 shows. We treated λ as a tuning parameter and tuned the best value when training CS and FPA.

2.3 Implement tools and environment

This project was implemented with Python (version 3.6.9) on a Linux server (Linux Intel Xeon Gold 6138 @2.0GHZ RAM 450G CPU). All required packages are listed on Table 2.

3 Numerical experiments and results

We ran each algorithm for 1,000 runs (epochs) with 50 points with 10 dimensions and 50 points with 20 dimensions. These procedures were replicated for 20 times. For all parts with randomization, the random seed was set as 4028, the last four digits of author's student ID number, for the first replication. The random seed was set as $4028 + 1 = 4029$ for the second replication and so on to avoid obtaining same results from every replication.

All numerical results of running 3 algorithms with various of setting are presented on Table 3. Since these numbers are too small to be compared easily, we visualized them. Fig 1 to Fig 10 are curves of training three algorithms with 10-dimension and 20-dimension points on two objective functions. The curves plots the mean values m_{epoch} among 20 replications for each epoch. We also computed standard deviation std_{epoch} among 20 replications for each epoch. The color-filling area is the range between $m_{epoch} - std_{epoch}$ and $m_{epoch} + std_{epoch}$. Fig 1 to Fig 4 plot training curves of three algorithms together and Fig 5 to Fig 10 plot training curves of two objective functions together. A total of 12 ($3 \times 2 \times 2 = 12$) single training curves of training three algorithms with two numbers of the dimensions of points on two objective functions were presented in APPENDIX.

From Table 3 and Fig 1 to Fig 10, it is obvious that CS and FPA converged much more early than BA did. This phenomenon was more significant when training on objective function 1 with 10-dimension points (Fig 1) and on objective function 2 with 10-dimension points (Fig 3). We can also found that the initial global best values got lower when the objective function changed to 2 from 1, no matter which the algorithm was and no matter which the number of the dimension of points was.

TABLE 1: Hyper-parameters Setting

Global / Algorithm	Hyper-parameter	Defaulted Value [Candidate Values]
Global	Random seed	4028 for the first replication, 4029 for the second one, and so on
	#Points	50
	#Runs	1000
	Objective function	[Ackley's function, Shifted and Rotated Weierstrass Function]
CS/FPA	λ	0.5 [0.5, 1.0, 1.5, 2.0]
	pa	0.25
BA	$f_{\min}$	0
	$f_{\max}$	1
	A_0	1
	R_0	1

TABLE 2: Required Packages

Package	Version	Use
argparse	-	Arguments definition
NumPy	1.16.3	Matrix operation
matplotlib	3.1.3	Visualization of experiments results
Scikit-Learn	0.24.1	Implement of t-Distributed Stochastic Neighbor Embedding
os	-	Create output folders
random	-	Random initialization
math	-	Gamma function
time	-	Timing
datetime	-	Get date and time info

Table 3: Numerical Results of Experiment (50 points, 1000 runs, mean values and std among 20 replications are reported)

Objective Function	#Dimensions of Points	λ for CS/ FPA	The global best value (mean±STD)			Time cost per runs (s) (mean±STD)		
			CS	BA	FPA	CS	BA	FPA
1	10	0.5	$7.17\pm4.57\times10^{-8}$	1.23 ± 0.96	**3.27±3.12** $\times10^{-10}$	0.0026 ± 0.0003	0.0028 ± 0.0001	**0.0015 ± 0.0001**
		1.0	$9.27\pm1.99\times10^{-8}$		**4.87±1.29** $\times10^{-10}$			
		1.5	$9.34\pm2.31\times10^{-8}$		**3.34±3.41** $\times10^{-10}$			
		2.0	**1.68±1.04**		1.99±1.42			
	20	0.5	$9.11\pm3.53\times10^{-8}$	0.62 ± 0.70	**4.27±3.33** $\times10^{-9}$	0.0031 ± 0.0003	0.0037 ± 0.0002	**0.0019 ± 0.0001**
		1.0	$7.73\pm3.29\times10^{-8}$		**4.87±3.59** $\times10^{-9}$			
		1.5	$8.34\pm2.35\times10^{-8}$		**3.45±1.13** $\times10^{-9}$			
		2.0	6.22±1.62		**5.58±1.09**			
2	10	0.5	**9.21±1.51** $\times10^{-8}$	0.26 ± 0.11	$3.91\pm4.53\times10^{-5}$	0.0159 ± 0.0007	0.0246 ± 0.0081	**0.0153 ± 0.0023**
		1.0	**7.99±4.39** $\times10^{-8}$		$9.39\pm3.41\times10^{-5}$			
		1.5	**1.44±9.35** $\times10^{-7}$		$8.98\pm1.01\times10^{-2}$			
		2.0	1.02±0.58		**1.01±0.47**			
	20	0.5	**7.09±1.23** $\times10^{-8}$	0.38 ± 0.10	$6.89\pm1.43\times10^{-8}$	0.0164 ± 0.0013	0.0172 ± 0.0007	**0.0087 ± 0.0004**
		1.0	**6.29±3.39** $\times10^{-8}$		$4.92\pm4.32\times10^{-8}$			
		1.5	**1.03±9.22** $\times10^{-7}$		$1.00\pm3.29\times10^{-7}$			
		2.0	5.59±0.51		**5.03±0.50**			

STD=standard deviation; s=second(s); CS=cuckoo search; BA=bat algorithm; FPA=flower pollination algorithm. Note: Bold number means the best performance among three algorithms.

Generally speaking, time costs of all trainings were quite low. Among three algorithms, the time cost of BA was the highest and that of FPA was the lowest. But the differences were not large. No matter what the algorithm was, it took more time to train on objective function 2 than train on objective function 1. Compared to training with 10-dimension points, time cost got higher when training with 20-dimension points.

To demonstrate that three algorithms converged well enough more directly, we drew scatter plots of 50 initial points and 50 final points after training three algorithms on two objective functions with 10-dimension and 20-dimension points for 1,000 Runs (Fig 11 to Fig 22). To illustrate on the 2-dimension paper, we randomly selected 2 different dimensions from original 10 or 20 dimensions. The indices of selected dimensions were presented on labels of x- and y-axes. We adopted this approach since it was better than dimensionality reduction to illustrate as the experiments showed. In these scatter plots, almost every final point converged to zero in Fig 11, 13, 14, 16, 17, 19, 20, and 22 while most final points only got very close to zero in Fig 12, 15, 18, and 21. We can conclude that three algorithms converged well enough while CS and FPA performed much better than BA did.

In CS and FPA, we tuned λ with candidate values 0.5, 1.0, 1.5, and 2.0. Results of training CS and FPA with different λ were also presented on Table 3. We did not present time cost for each λ since they were similar. A total of 8 (2 algorithms $\times$ 2 objective functions $\times$ 2 numbers of dimensions of points = 8) figures which plotted training curves of different λ values together were presented as Fig 23 to Fig 30. These figures and large global best values of the last run indicated that it might need to take much more runs to be converged when using $\lambda = 2.0$. There were no obvious differences between using $\lambda = 0.5$ and using $\lambda = 1.0$. When using $\lambda = 1.5$, it also converged slowly but at least the global best value can be close to 0. Compared to 10-dimension points, the weakness of using $\lambda = 2.0$ was more obvious when training with 20-dimension points. It may be worthy exploring the change from $\lambda = 1.5$ to $\lambda = 2.0$.

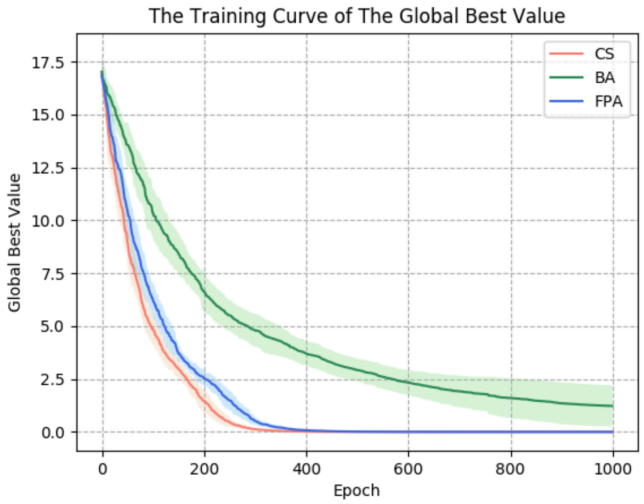

Fig 1: Curves of Training CS, BA, and FPA on Objective Function 1 with 10-dimension Points for 1000 Runs (Epochs)

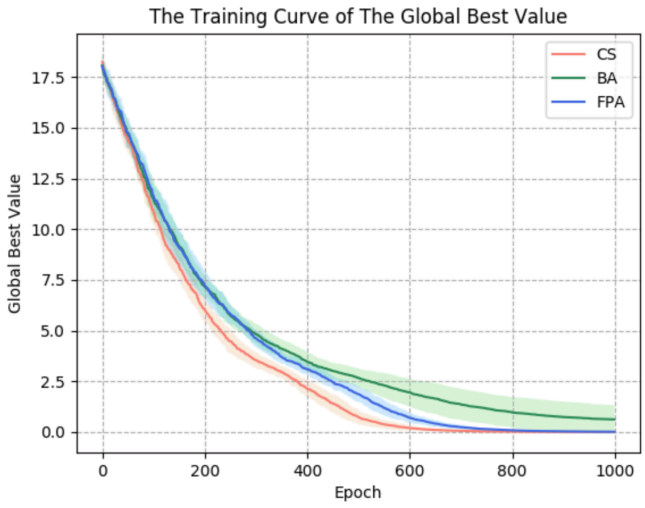

Fig 2: Curves of Training CS, BA, and FPA on Objective Function 1 with 20-dimension Points for 1000 Runs (Epochs)

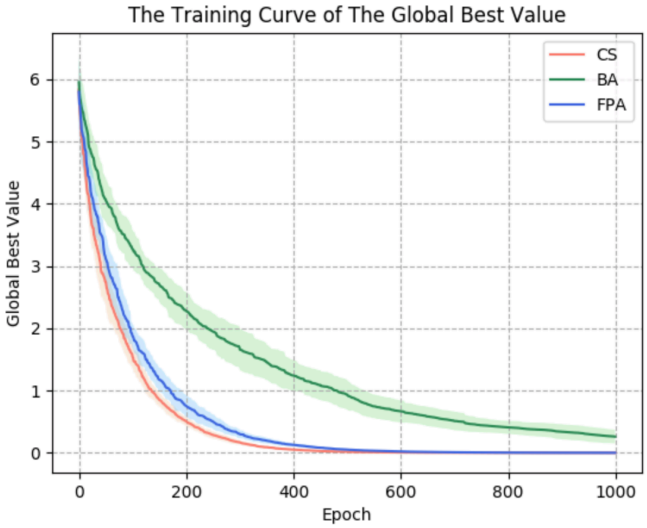

Fig 3: Curves of Training CS, BA, and FPA on Objective Function 2 with 10-dimension Points for 1000 Runs (Epochs)

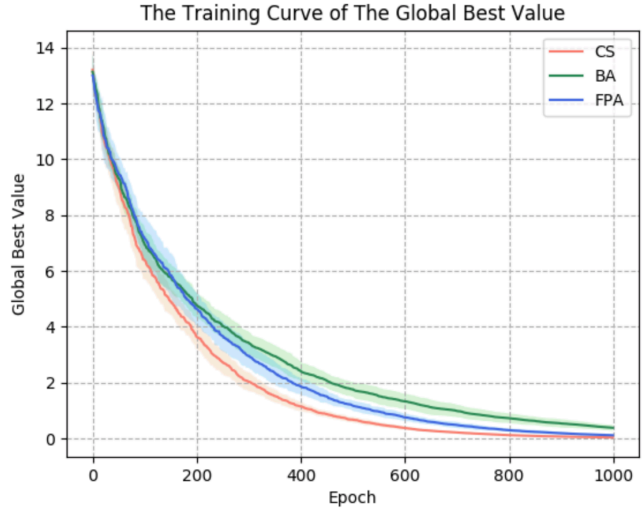

Fig 4: Curves of Training CS, BA, and FPA on Objective Function 2 with 20-dimension Points for 1000 Runs (Epochs)

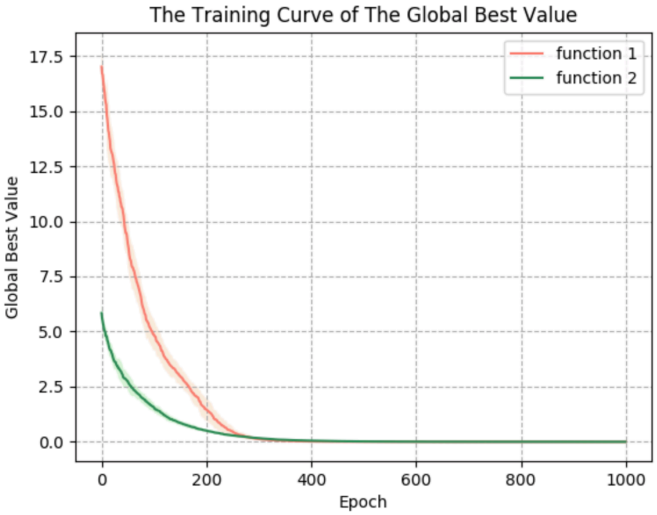

Fig 5: Curves of Training CS on Objective Function 1 and 2 with 10-dimension Points for 1000 Runs (Epochs)

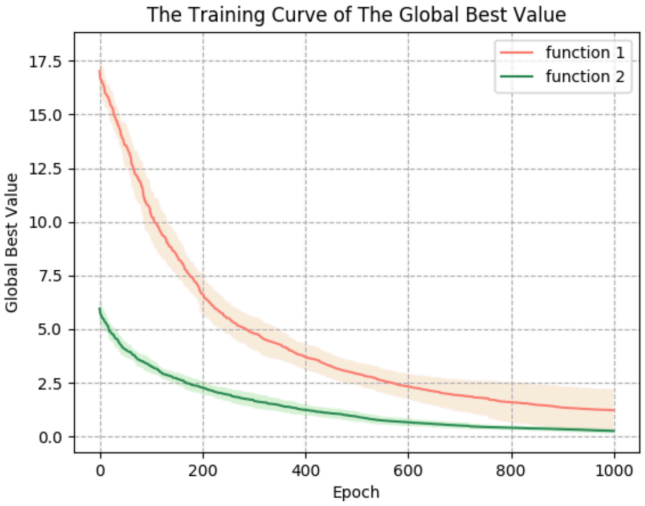

Fig 6: Curves of Training BA on Objective Function 1 and 2 with 10-dimension Points for 1000 Runs (Epochs)

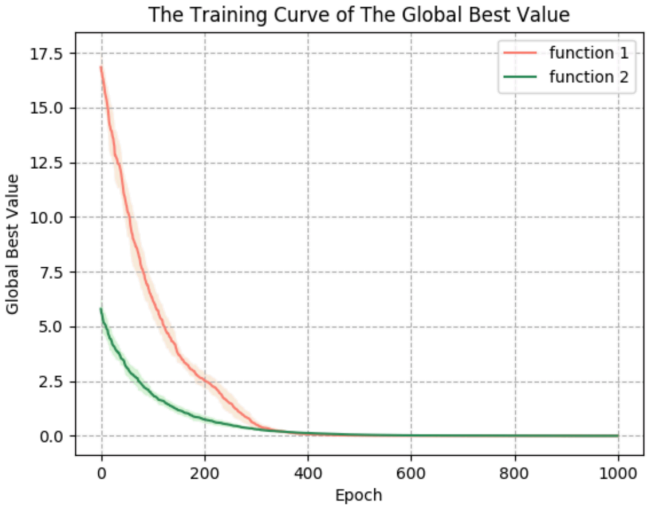

Fig 7: Curves of Training FPA on Objective Function 1 and 2 with 10-dimension Points for 1000 Runs (Epochs)

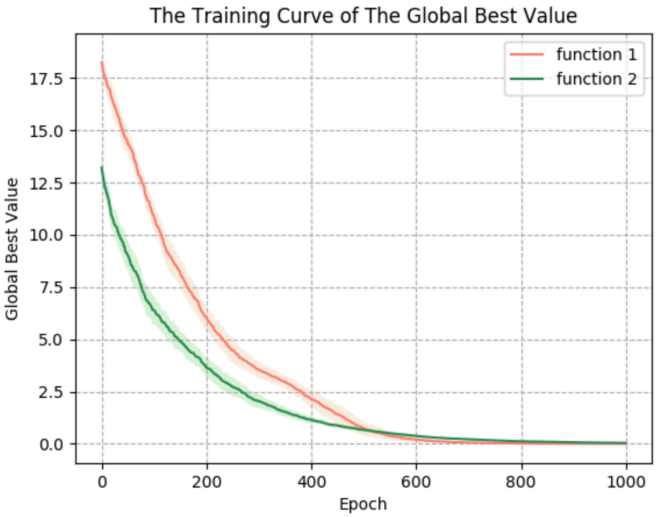

Fig 8: Curves of Training CS on Objective Function 1 and 2 with 20-dimension Points for 1000 Runs (Epochs)

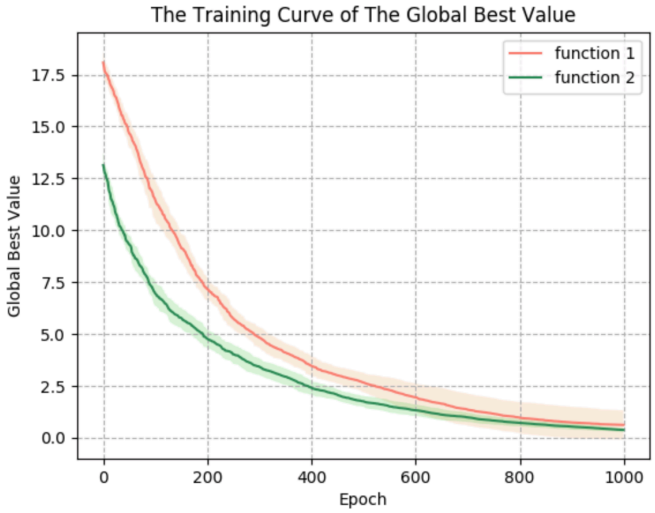

Fig 9: Curves of Training BA on Objective Function 1 and 2 with 20-dimension Points for 1000 Runs (Epochs)

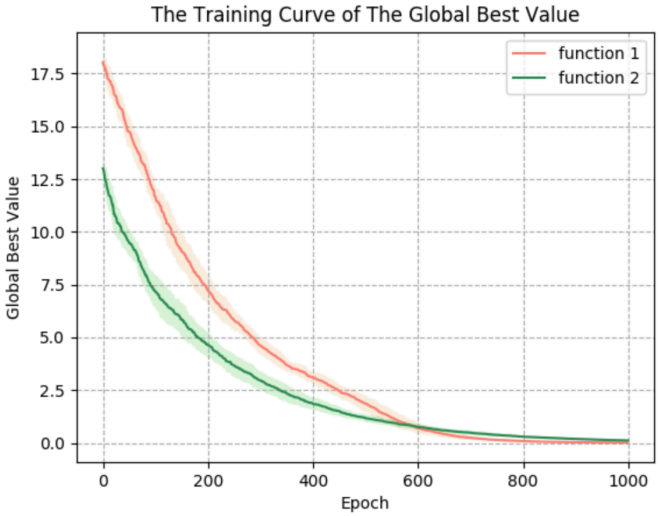

Fig 10: Curves of Training FPA on Objective Function 1 and 2 with 20-dimension Points for 1000 Runs (Epochs)

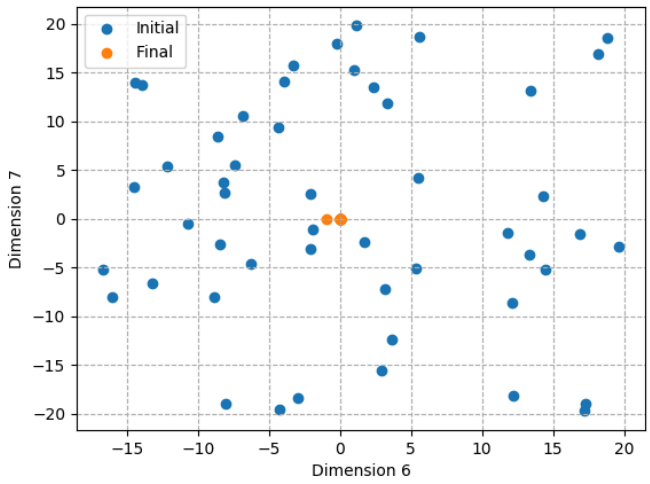

Fig 11: Scatter Plots of 50 Initial and Points after Training CS on Objective Function 1 with 10-dimension Points for 1,000 Runs (Epochs)

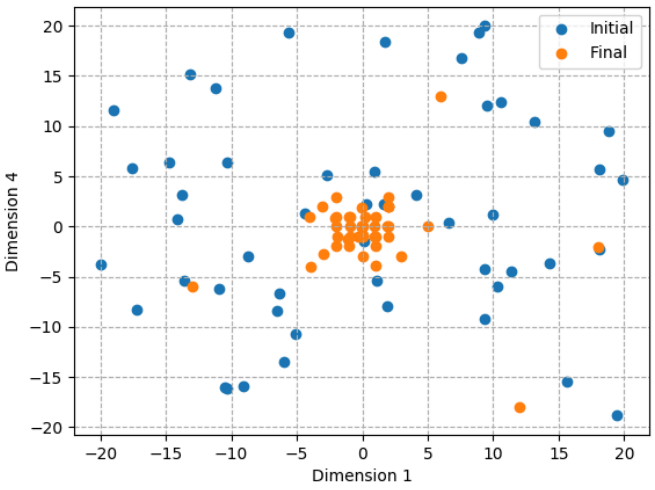

Fig 12: Scatter Plots of 50 Initial and Final Points after Training BA on Objective Function 1 with 10-dimension Points for 1,000 Runs (Epochs)

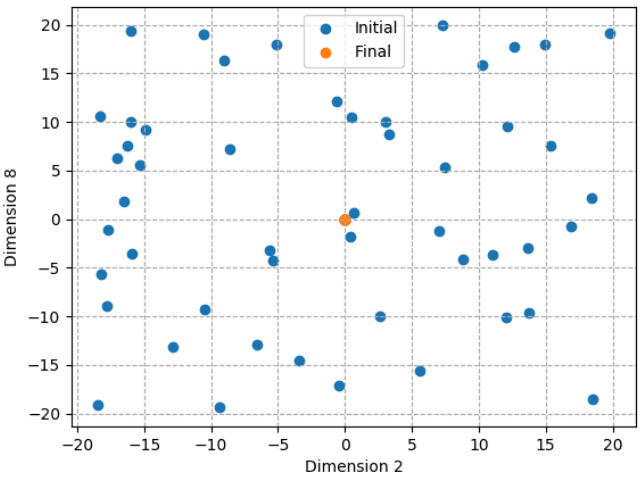

Fig 13: Scatter Plots of 50 Initial and Final Points after Training FPA on Objective Function 1 with 10-dimension Points for 1,000 Runs (Epochs)

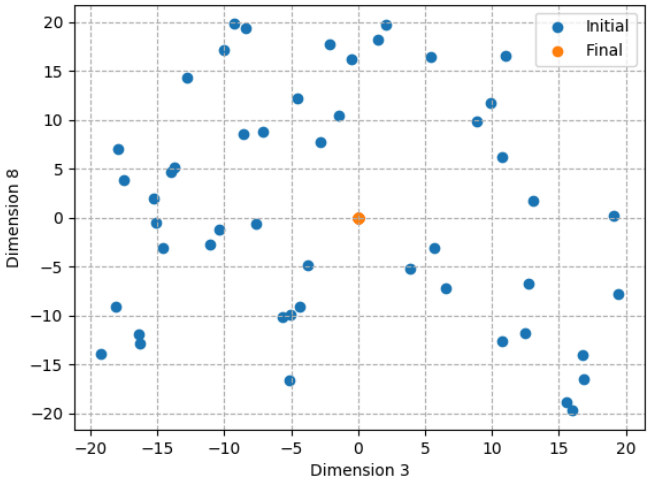

Fig 14: Scatter Plots of 50 Initial and Final Points after Training CS on Objective Function 1 with 20-dimension Points for 1,000 Runs (Epochs)

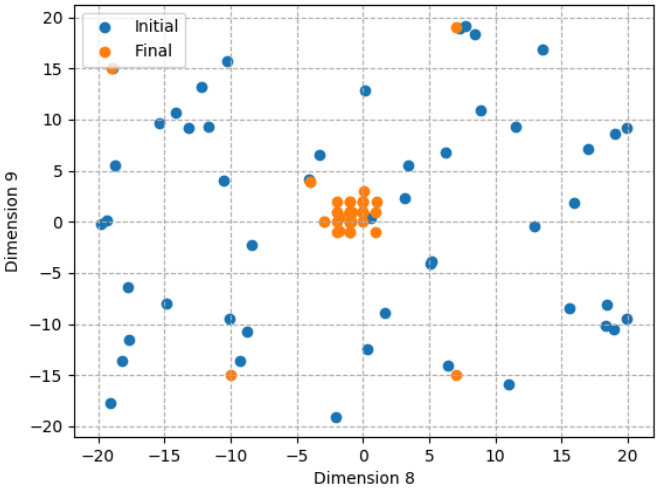

Fig 15: Scatter Plots of 50 Initial and Final Points after Training BA on Objective Function 1 with 20-dimension Points for 1,000 Runs (Epochs)

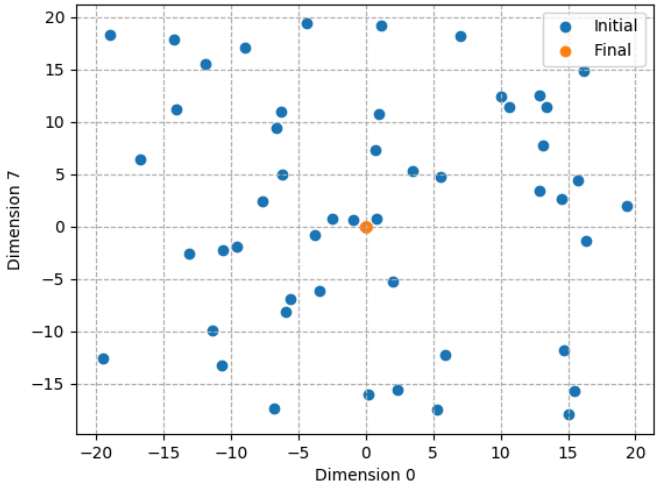

Fig 16: Scatter Plots of 50 Initial and Final Points after Training FPA on Objective Function 1 with 20-dimension Points for 1,000 Runs (Epochs)

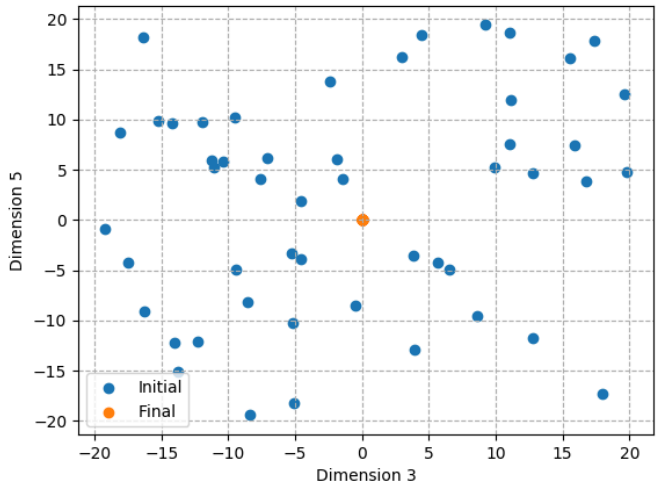

Fig 17: Scatter Plots of 50 Initial and Points after Training CS on Objective Function 2 with 10-dimension Points for 1,000 Runs (Epochs)

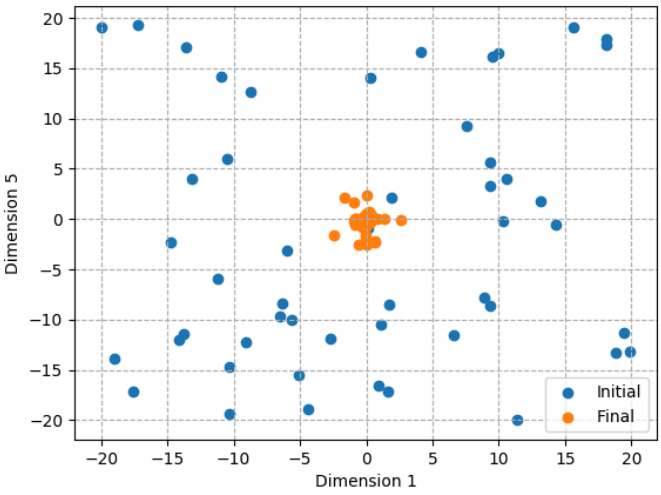

Fig 18: Scatter Plots of 50 Initial and Points after Training BA on Objective Function 2 with 10-dimension Points for 1,000 Runs (Epochs)

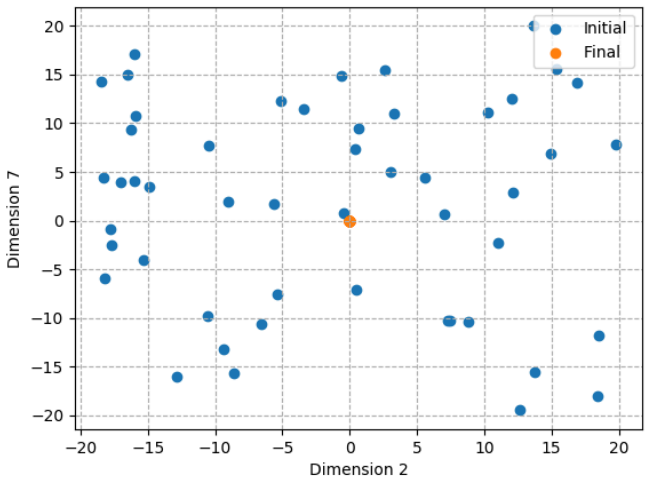

Fig 19: Scatter Plots of 50 Initial and Points after Training FPA on Objective Function 2 with 10-dimension Points for 1,000 Runs (Epochs)

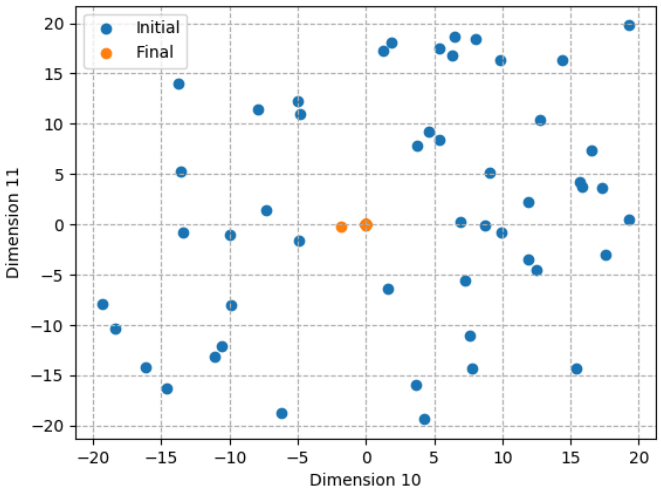

Fig 20: Scatter Plots of 50 Initial and Points after Training CS on Objective Function 2 with 20-dimension Points for 1,000 Runs (Epochs)

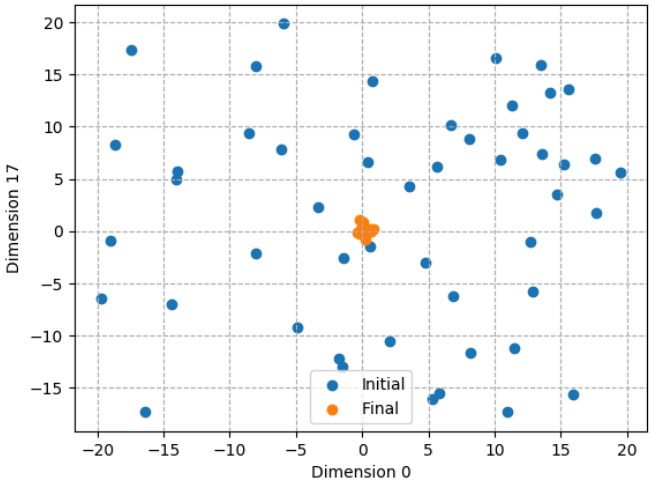

Fig 21: Scatter Plots of 50 Initial and Points after Training BA on Objective Function 2 with 20-dimension Points for 1,000 Runs (Epochs)

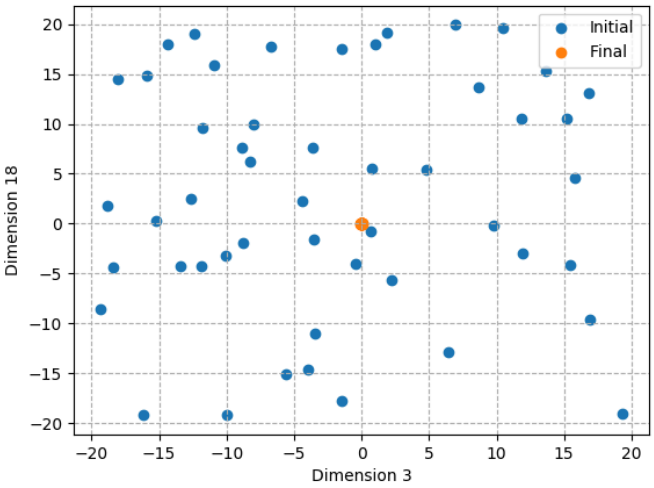

Fig 22: Scatter Plots of 50 Initial and Points after Training FPA on Objective Function 2 with 20-dimension Points for 1,000 Runs (Epochs)

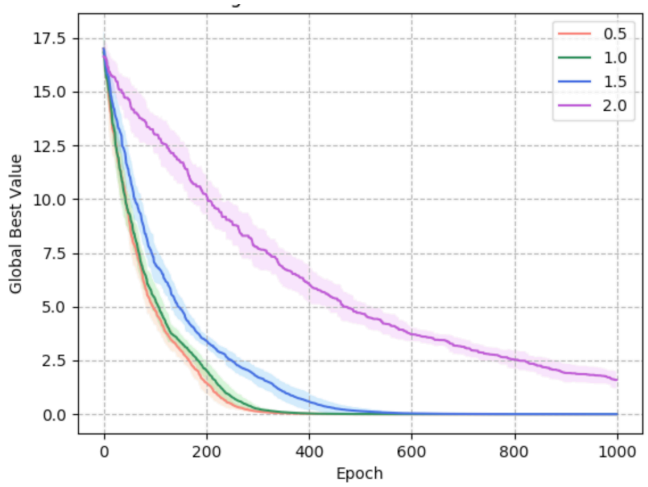

Fig 23: Curves of Training CS on Objective Function 1 with 10-dimension Points with Different λ Values for 1000 Runs (Epochs)

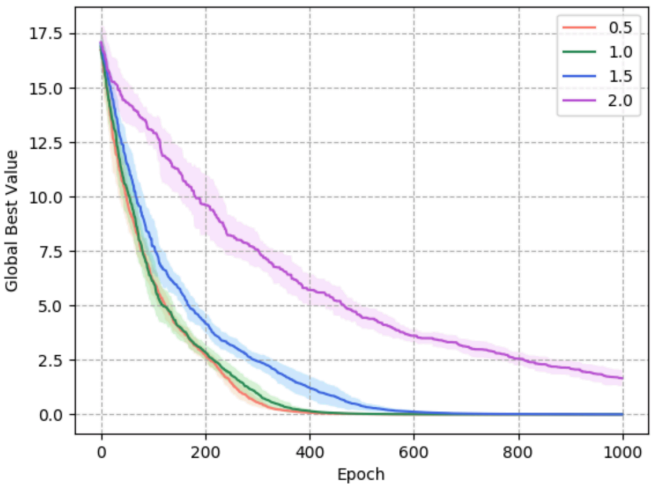

Fig 25: Curves of Training FPA on Objective Function 1 with 10-dimension Points with Different λ Values for 1000 Runs (Epochs)

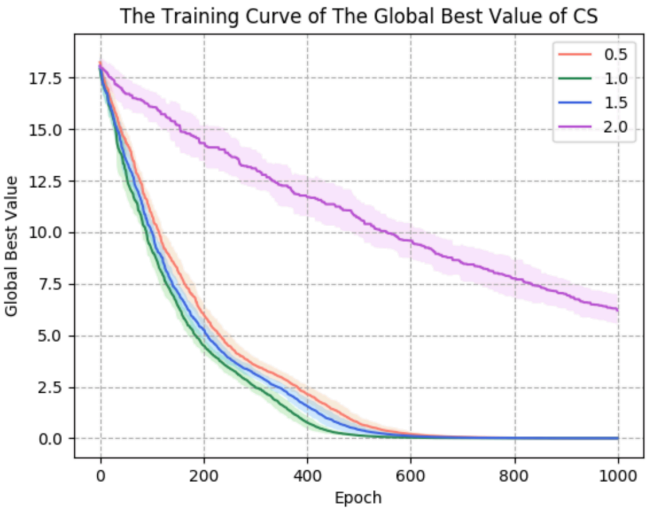

Fig 24: Curves of Training CS on Objective Function 1 with 20-dimension Points with Different λ Values for 1000 Runs (Epochs)

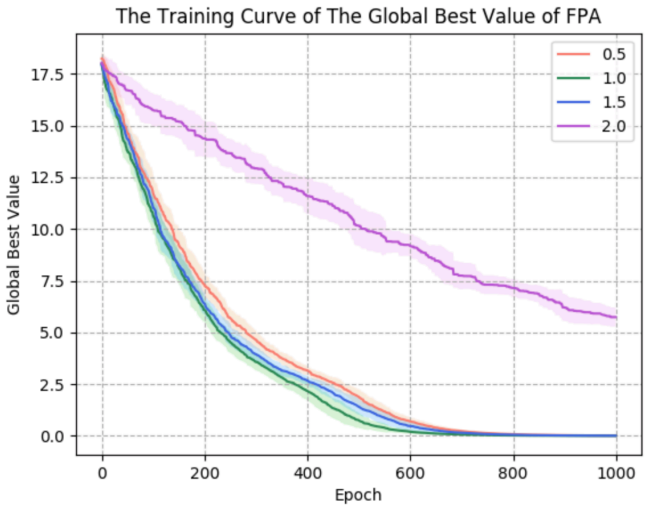

Fig 26: Curves of Training FPA on Objective Function 1 with 20-dimension Points with Different λ Values for 1000 Runs (Epochs)

4 Conclusion

We implemented three well-known meta-heuristic algorithms: CS, BA, and FPA. We found that three algorithms could have a promising performance generally. In term of the convergence, BA performed the worst. It might need more runs to be converged when training BA. Compared to 20-dimension points, the weakness of BA was more obvious when training with 10-dimension points. The time cost of BA was the highest while the differences of time cost among three algorithms were not so large, which might not matter when the number of training runs was not large (i.e., < 2000).

We also tuned λ in CS and FPA with the findings of the weakness of larger λ values. We also found this weakness got more obvious when training with 20-dimension points than raining with 10-dimension points. There were a non-small performance gap between $\lambda = 1.5$ and $\lambda = 2.0$, which can be explored in the future.

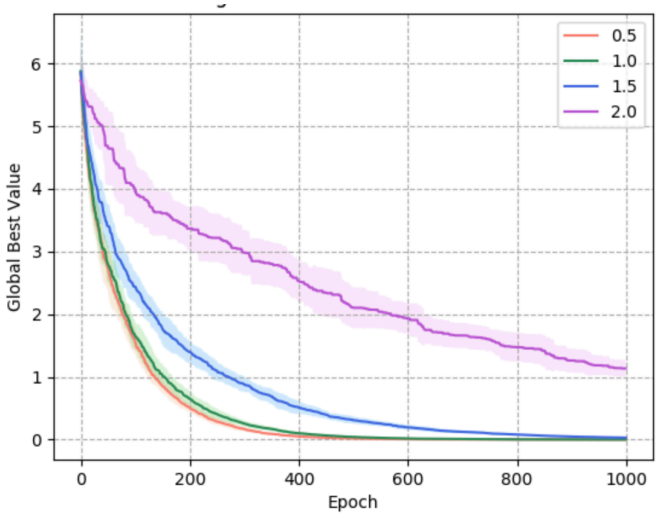

Fig 27: Curves of Training CS on Objective Function 2 with 10-dimension Points with Different λ Values for 1000 Runs (Epochs)

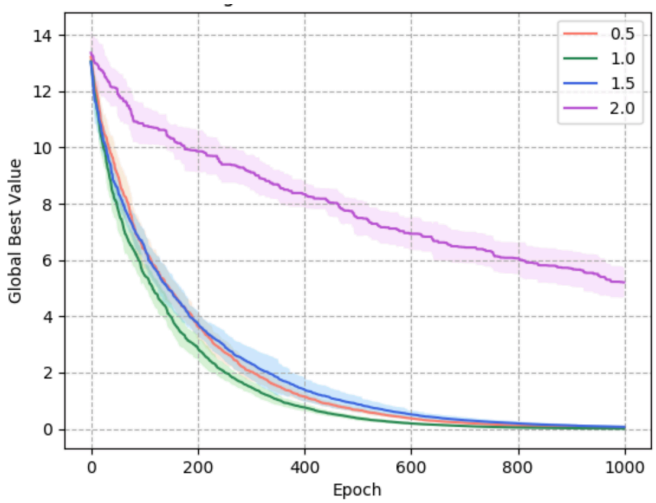

Fig 28: Curves of Training CS on Objective Function 2 with 20-dimension Points with Different λ Values for 1000 Runs (Epochs)

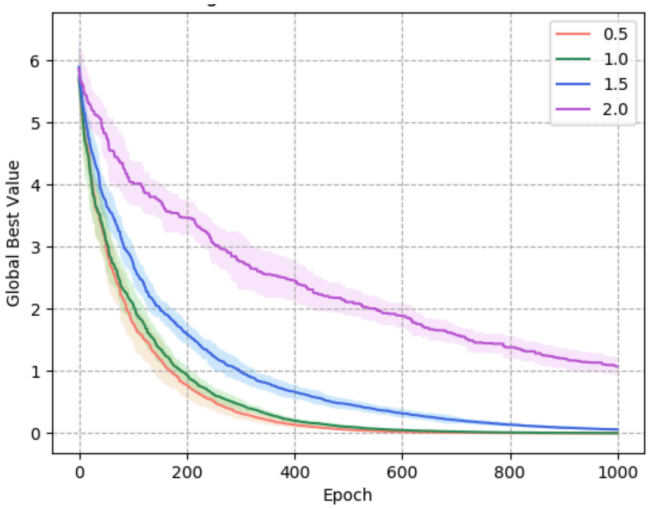

Fig 29: Curves of Training FPA on Objective Function 2 with 10-dimension Points with Different λ Values for 1000 Runs (Epochs)

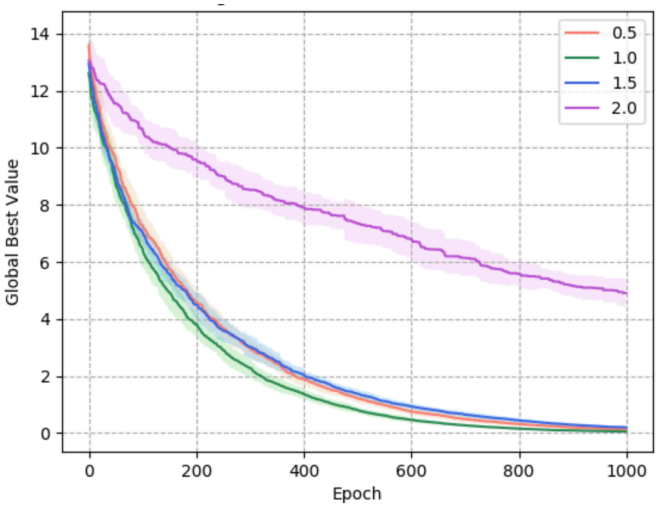

Fig 30: Curves of Training FPA on Objective Function 2 with 20-dimension Points with Different λ Values for 1000 Runs (Epochs)

REFERENCE

1. Reiner Horst, Pardalos Panos M. Eds. Handbook of global optimization. Vol. 2. Springer Science & Business Media. 2013.
2. Yang XS, Deb S. Cuckoo search via Lévy flights. In: Proceedings of world congress on nature & biologically inspired computing (NaBIC 2009). USA: IEEE Publications; 2009. p. 210–14.
3. YangXS,DebS.Engineering optimization by cuckoo search. Int J Math Model Num Optim 2010;1(4):330–43.
4. Yang XS (2010a) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, pp 65–74.
5. Yang XS. Flower pollination algorithm for global optimization. In: Unconventional com- putation and natural computation. Lecture notes in computer science, vol. 7445; 2012. p. 240–49.
6. Yang XS, Karamanoglu M, He XS. Multi-objective flower algorithm for optimization. Procedia Comput Sci 2013;18:861–8.

APPENDIX

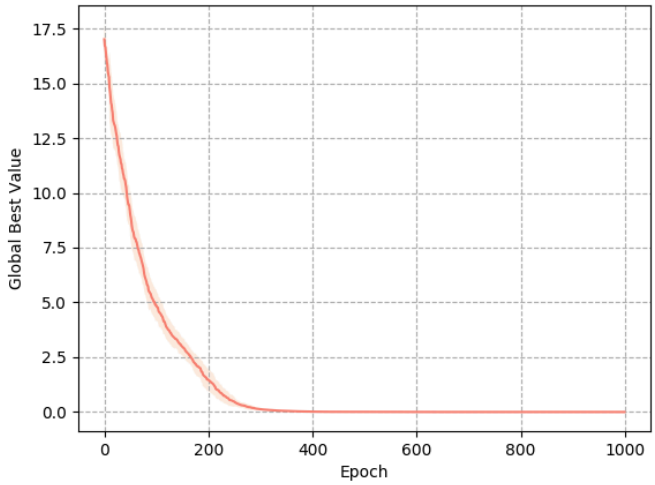

Fig 31: The Curve of Training CS on Objective Function 1 with 10-dimension Points for 1000 Runs (Epochs)

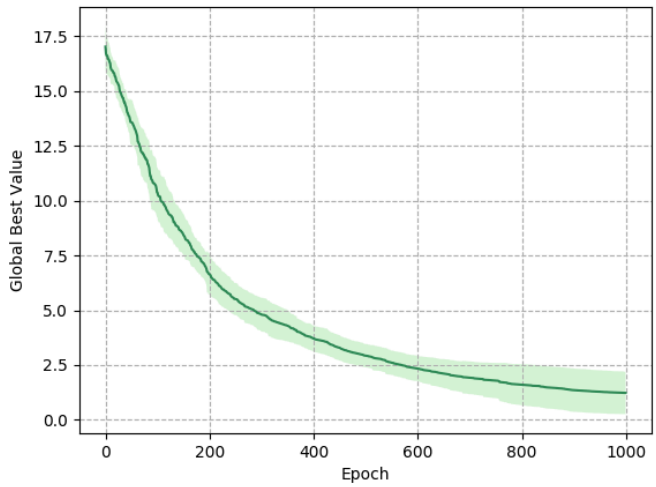

Fig 32: The Curve of Training BA on Objective Function 1 with 10-dimension Points for 1000 Runs (Epochs)

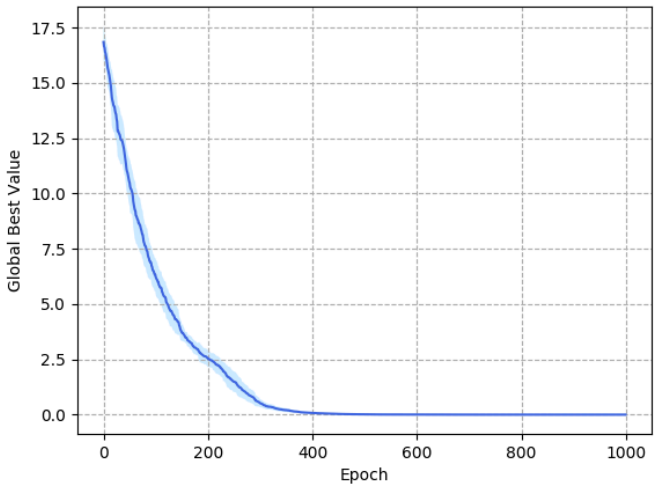

Fig 33: The Curve of Training FPA on Objective Function 1 with 10-dimension Points for 1000 Runs (Epochs)

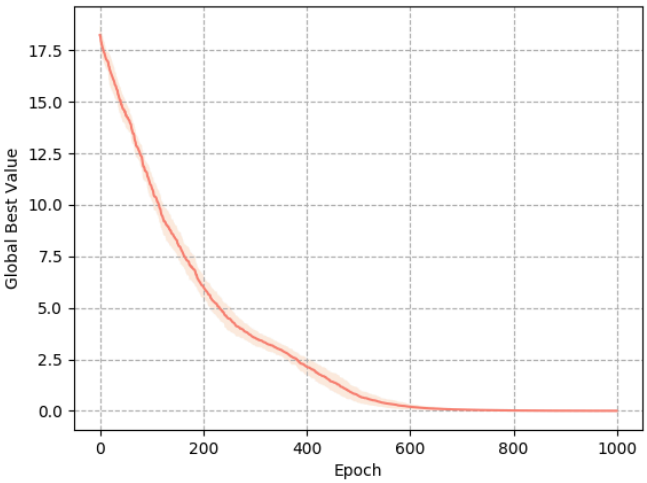

Fig 34: The Curve of Training CS on Objective Function 1 with 20-dimension Points for 1000 Runs (Epochs)

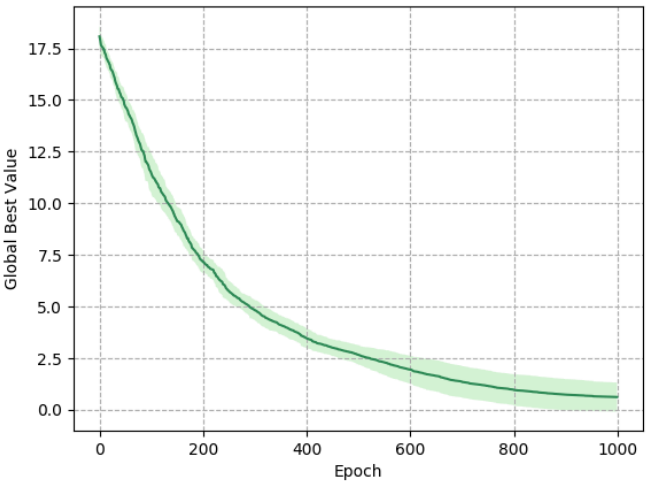

Fig 35: The Curve of Training BA on Objective Function 1 with 20-dimension Points for 1000 Runs (Epochs)

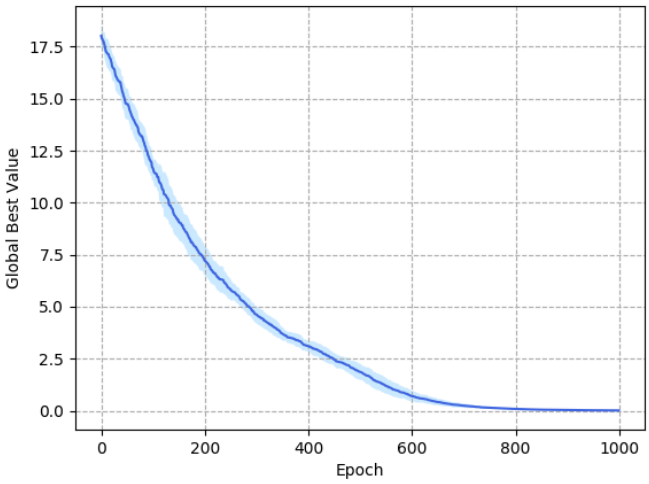

Fig 36: The Curve of Training FPA on Objective Function 1 with 20-dimension Points for 1000 Runs (Epochs)

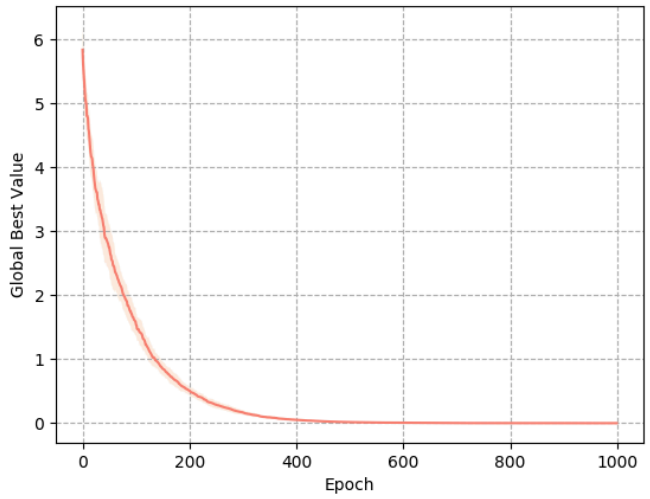

Fig 37: The Curve of Training CS on Objective Function 2 with 10-dimension Points for 1000 Runs (Epochs)

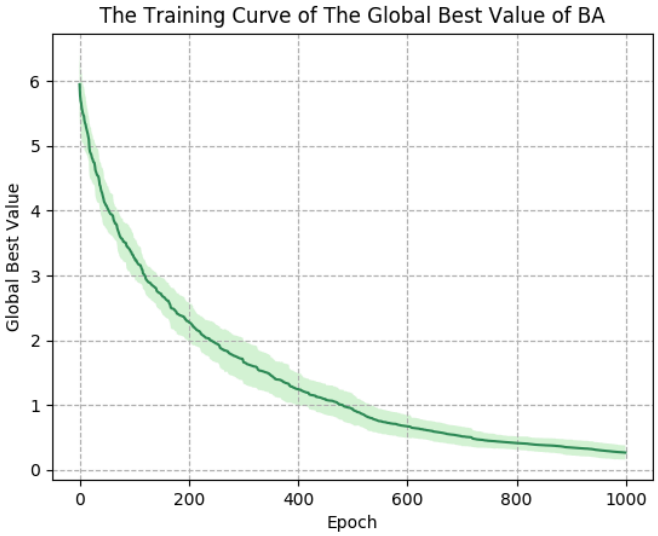

Fig 38: The Curve of Training BA on Objective Function 2 with 10-dimension Points for 1000 Runs (Epochs)

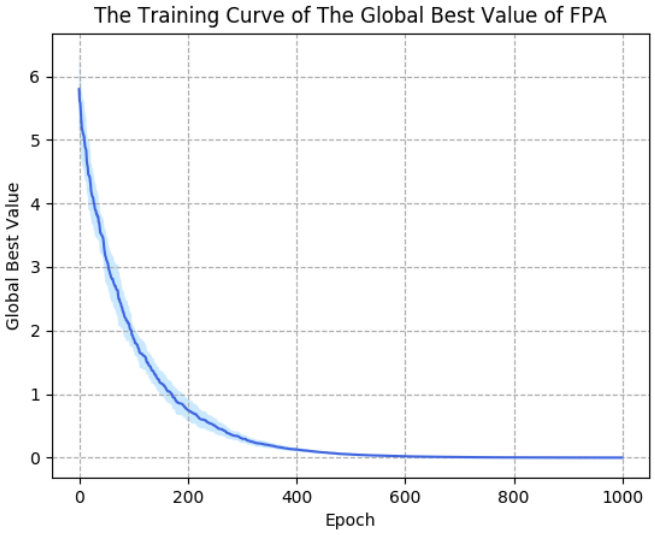

Fig 39: The Curve of Training FPA on Objective Function 2 with 10-dimension Points for 1000 Runs (Epochs)

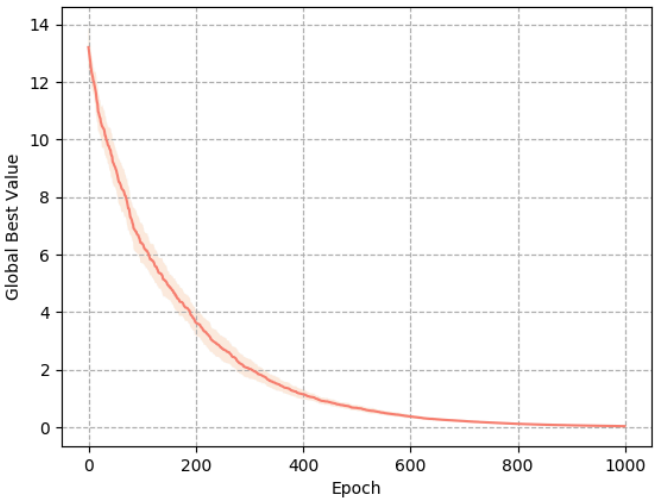

Fig 40: The Curve of Training CS on Objective Function 2 with 20-dimension Points for 1000 Runs (Epochs)

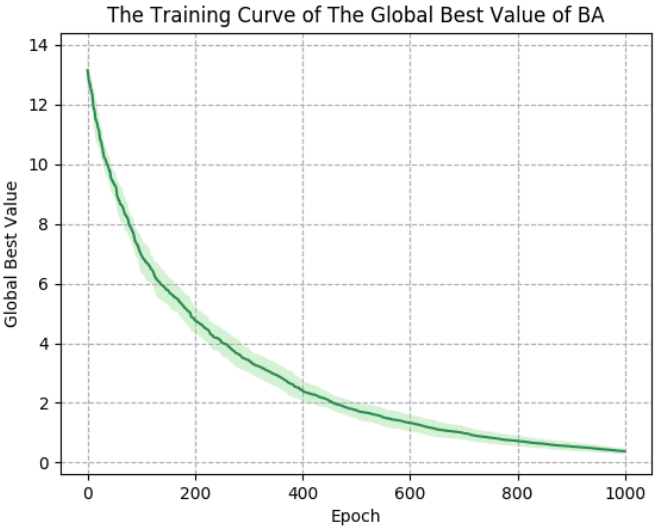

Fig 41: The Curve of Training BA on Objective Function 2 with 20-dimension Points for 1000 Runs (Epochs)

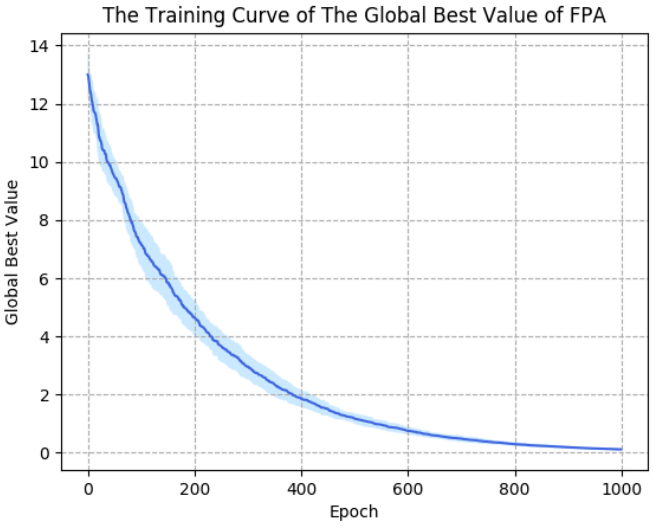

Fig 42: The Curve of Training FPA on Objective Function 2 with 20-dimension Points for 1000 Runs (Epochs)