Optimization Methods - Mid-Term Project 2

Implement of Meta-Heuristic Algorithms: Differential Evolution, Particle Swarm Optimization, And Firefly 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: differential evolution (DE), particle swarm optimization (PSO), and firefly algorithm (FA). We set up the numerical experiments to compare 3 algorithms with different numbers of points. We found that FA performed well with only several points but take much higher time cost with more points. While DE and PSO had excellent performance in terms of precision and efficiency with more points (i.e., $N \geq 200$). K-means can be utilized to make the initialized points distribute more uniformly. All codes of implements and experiment procedures are available on GitHub: https://github.com/jayenliao/Optimization-DE-PSO-FA.

Keywords

Global optimization, meta-heuristic algorithms, differential evolution (DE), particle swarm optimization (PSO), firefly algorithm (FA)

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, differential evolution (DE), particle swarm optimization (PSO), and firefly algorithm (FA) were found to be powerful and efficient.

1.1 Differential Evolution (DE)

DE was developed by Storn and Price (1997). DE with a potential parallel structure is a non-gradient-based, evolutionary computation algorithm. DE uses the difference of randomly sampled pairs of object vectors to guide the mutation process which makes it relatively new when compared to other algorithms. Similar to genetic algorithm (GA), a randomly generated initial population is created. For each individual three other individuals are selected in random. A new vector is created by adding a weighted (mutation factor) difference of two individual to the other. Cross over or recombination is one of the main operators for GA but it is complementary in DE. When the entire individuals are processed by this way then fitness is evaluated. If the fitness value of the new individual is better than that of old individual, then update the old individual with the new one. This process is repeated until maximum number of generations or convergence is reached.

1.2 Particle Swarm Optimization (PSO)

PSO is based on the social behavior of birds (Brest et al., 2006). In PSO, initially a random population is created. Each individual known as particle is assigned a velocity and a small social network. For all particles, fitness or objective function values are evaluated. Unlike GA, PSO does not have crossover or mutation, but the personal optimal for each individual, global optimal in the complete population and neighborhood optimal found by the neighbors of each individual are saved to update velocity and position for each individual. This process is repeated until either the convergence is reached or maximum generations.

1.3 Firefly algorithm (FA)

FA proposed by Xin-She Yang (2008) is also a biologically inspired meta-heuristic optimization algorithm. FA is based on the communication behavior of tropical fireflies and the idealized behavior of the flashing patterns, using the following three idealized rules to build the mathematical model of the algorithm (Yang, 2009; Wang et al., 2016; Marichelvam et al., 2014; Marichelvam & Geetha):

1. All fireflies are unisex so that any firefly would be attracted to other fireflies.
2. Attractiveness is proportional to their brightness. Hence, for any two flashing fireflies, the less bright one would move towards the brighter one. The attractiveness is proportional to the brightness and thus they both decrease as their distance increases.
3. The brightness of a firefly is affected or determined by the landscape of the objective function. As a result, for a maximization problem, the brightness can simply be proportional to the value of the objective function.

The brightness of the firefly can always be assumed to be determined by the encoded objective function landscape. The variation of light intensity and formulate the change of the attractiveness should be defined, that is, they are hyper-parameters. As we know that in nature the light intensity decreases with the distance from its source and the media will absorb the light, so in our simulation we suppose the light intensity *I* varies with the distance r and light absorption parameter γ exponentially and monotonically (Yang, 2010). That is, $I = I_0 e^{-\gamma r^2}$, where I_0 is the original light intensity at the source (i.e., $r = 0$) and γ is the light absorption coefficient. From the idealized rules we known that in our simulation we suppose the attractiveness of firefly is proportional to the light intensity *I*. So we can define the firefly's light attractive coefficient β in the similar way as the light intensity coefficient *I*. That is, $\beta = \beta_0 e^{-\gamma r^2}$, where β_0 is the original light attractiveness at $r = 0$. The Cartesian distance is used to measure the distance between any two fireflies *i* and *j* at x_i:

$$r_{ij} = ||x_i - x_j||_2 = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}\text{, where } d \text{ is the}$$

number of dimensions. The amount of movement of firefly *i* to another more attractive (brighter) firefly *j* is determined by $x_i = x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha \epsilon_i$, where the first term is the current location of firefly *i*, the second term is due to the attraction, while the third term is randomization with the vector of random variables ε_i being drawn from different distributions such as the Uniform distribution, Gaussian distribution and Lévy flight. In the third term, α is a scaling parameter that controls the step size and it should be linked with the interests of the problems.

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 numbers of points setting.

2 Methods

2.1 Problem Statement

Consider a typical model frequently used in pharmaceutical studies, is the 3-compartmental model with mean response given by

$\eta(x;\theta) = \theta_3[e^{-\theta_2 x} - e^{-\theta_1 x}]$, where $x > 0$. Define

$$f^T(x,\theta^0) = [\frac{\partial\eta(x,\theta)}{\partial\theta_1}, \frac{\partial\eta(x,\theta)}{\partial\theta_2}, \frac{\partial\eta(x,\theta)}{\partial\theta_3}]|_{\theta^0}.$$

$$\text{Here } \begin{cases} \frac{\partial\eta(x,\theta)}{\partial\theta_1} = x\theta_3 e^{-\theta_1 x} \\ \frac{\partial\eta(x,\theta)}{\partial\theta_2} = x\theta_3 e^{-\theta_2 x} \\ \frac{\partial\eta(x,\theta)}{\partial\theta_3} = e^{-\theta_2 x} - e^{-\theta_1 x} \end{cases}.$$

Suppose $\xi = \begin{bmatrix} x_1 & x_2 & x_3 \\ w_1 & w_2 & w_3 \end{bmatrix}$ with $w_1 + \ldots + w_n = 1$.

Define $M(\xi,\theta^0) = \sum_i w_i f(x_i,\theta^0) f^T(x_i,\theta^0)$. The goal is to $\max_{\xi} |M(\xi,\theta^0)|$ with the following settings:

1. $\theta^0 = (0.05884, 4.298, 21.8)$
2. $\xi = \begin{bmatrix} x_1 & x_2 & x_3 \\ w_1 & w_2 & w_3 \end{bmatrix}$ and $[x_1 \quad x_2 \quad x_3] \in \Xi = [0, 30]^3$. Here we can set $0 \le x_1 \le x_2 \le x_3 \le 30$.
3. The optimal point is $\xi = \begin{bmatrix} 0.2288 & 1.3886 & 18.4168 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$.

2.2 Hyper-parameters Settings

All hyper-parameters in 3 algorithms were set as TABLE 1 shows. For the comparison of number of setting points, we changed *N* into different numbers.

TABLE 1: Hyper-parameters Setting

Global / Algorithm	Hyper-parameter	Value
Global	Random seed	4028
	No. of initialized points	100
	No. of runs	50
	θ^0	(0.05884, 4.298, 21.8)
DE	F	0.2
PSO	velocityP	1.0
	velocityPT	0.01
	α	1.0
	β	0.3
	γ	0.001
FA	β_0	1.0
	γ	0.5
	α	1.0
	α_2	0.001
	α_P	0.99
	λ	1.0

TABLE 2: Required Packages

Package	Version	Use
argparse	-	Arguments definition
NumPy	1.16.3	Matrix operation
CuPy	1.1.5	Matrix operation on GPU
matplotlib	3.1.3	Visualization of experiments results
os	-	Create output folders
random	-	Random initialization
threading	-	Thread-based parallelism
time	-	Timing
datetime	-	Get date and time info

2.4 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 as TABLE 2.

3 Numerical experiments and results

We ran each algorithm for 50 times with different numbers of points. For all parts with randomization, the random seed were set as 4028, the last four digits of author's student ID number.

Numerical results of running 3 algorithms for 50 times with different numbers of points are presented on TABLE 3. Fig 1 to Fig 4 are convergence plots of three algorithms with different numbers of points: 10, 50, 100, and 200. Fig 5 to Fig 7 are the convergence plots of DE, PSO, and FA with different numbers of points, respectively. Fig 8 to Fig 10 are plots of the smallest global best values of DE, PSO, and FA with different numbers of points, respectively. Fig 11 to Fig 13 are plots of the association between the number of points and the time cost of DE, PSO, and FA, respectively.

As TABLE 3, Fig 1, and Fig 2 show, FA had a much better performance than DE and PSO in term of both precision and efficiency when the number of points is relatively small (i.e., $N < 100$). Both of DE and PSO converged at very high values early. However, when the number of points increased, PSO took advantages instead. The time cost of FA even became too unacceptable to be run when $N > 300$ (see TABLE 3, Fig 3, and Fig 4). In general, PSO had the best performance in terms of high precision and low time cost.

As Fig 5 to Fig 10 demonstrate, the global best value was very sensitive to the number of points. As the number of points larger than 200, both DE and PSO had acceptable performance. FA even needed 25 points to reach the global best value, which was very close to the true value, -1617.5895. However, FA converged much slower than DE and PSO did. Generally speaking, FA needed 30 runs to converges, making its time cost become more expensive.

It is strange that FA had a much higher global vest value with 100 points. It might result from the non-uniformly distributed initialized points. The possible solution can be utilizing *K*-means when initialing points since centroids of *K* groups distributed more uniformly.

TABLE 3: Results of Experiment 2 (50 runs)

# Points	The global best value			Time cost for 50 runs (s)		
	DE	PSO	FA	DE	PSO	FA
10	-0.04	-3.75	**-19.45**	0.78	**0.43**	3.94
20	-67.91	-24.55	**-880.10**	1.57	**0.84**	15.80
30	-67.91	-181.12	**-1597.92**	2.34	**1.29**	35.66
40	169.99	-344.52	**-1582.34**	3.07	**1.74**	62.86
50	-169.99	-226.64	**-1596.27**	3.59	**2.10**	97.70
100	-644.06	**-1610.76**	-392.11	7.60	**4.15**	388.03
200	**-1468.20**	-1291.40	-1171.71	14.49	**8.67**	1514.08
300	-1300.99	-1168.50	**-1587.10**	21.49	**12.98**	3404.98
400	-1454.69	**-1612.97**	-	28.42	**17.15**	-
500	-1320.28	**-1525.84**	-	35.58	**21.81**	-
750	-1095.50	**-1616.17**	-	54.44	**33.18**	-
1000	-1555.89	**-1616.82**	-	76.10	**44.55**	-

Note: Bold number means the best performance among three/two algorithms.

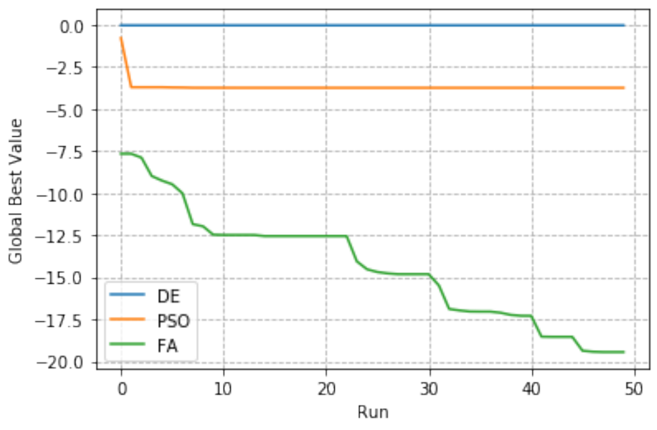

Fig 1: Convergence Plot of Three Algorithms with 10 Points

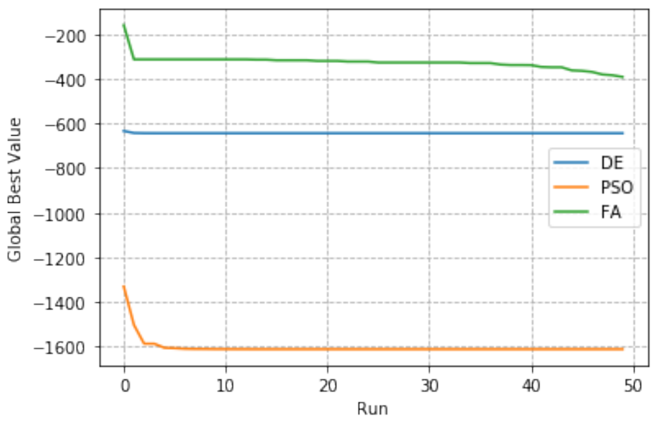

Fig 3: Convergence Plot of Three Algorithms with 100 Points

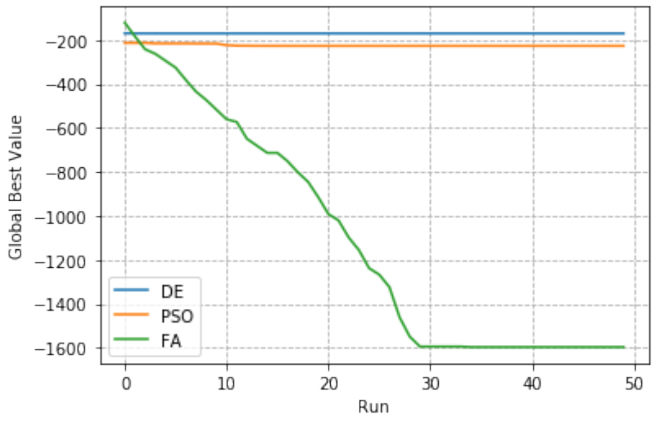

Fig 2: Convergence Plot of Three Algorithms with 50 Points

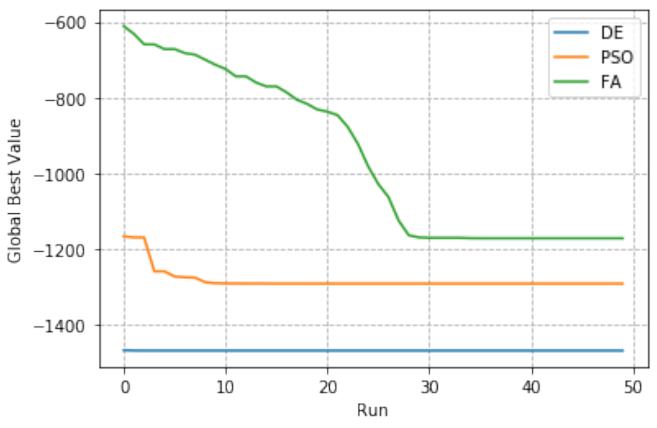

Fig 4: Convergence Plot of Three Algorithms with 200 Points

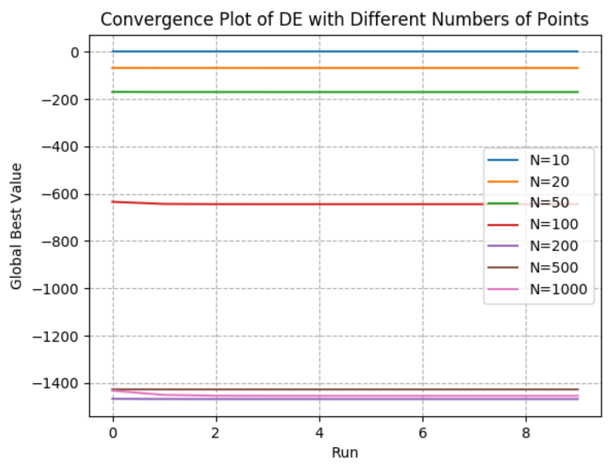

Fig 5: Convergence Plot of DE with Different Numbers of Points

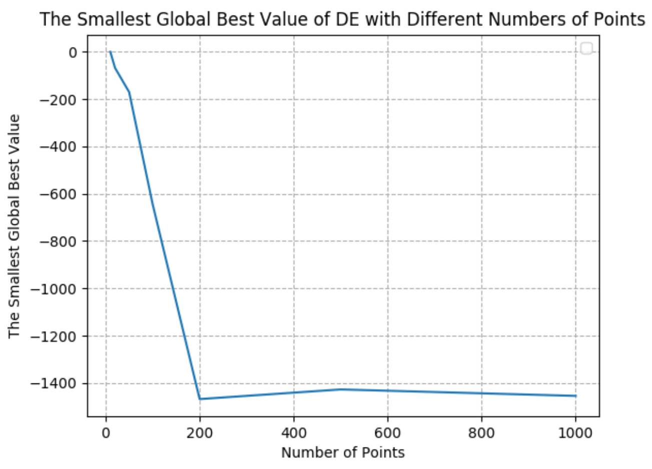

Fig 8: Plot of The Smallest Global Best Values of DE with different numbers of points

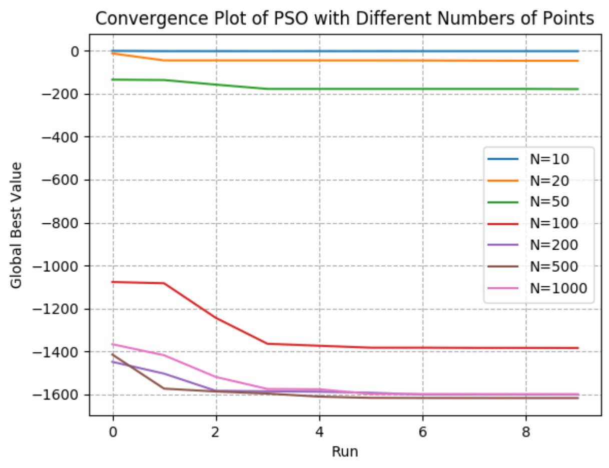

Fig 6: Convergence Plot of PSO with Different Numbers of Points

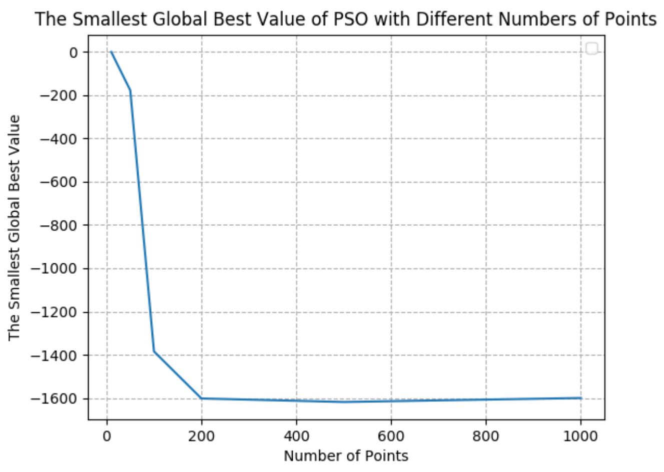

Fig 9: Plot of The Smallest Global Best Values of PSO with different numbers of points

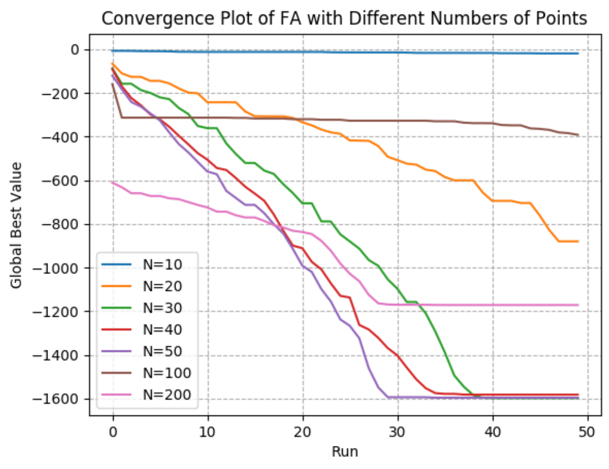

Fig 7: Convergence Plot of FA with Different Numbers of Points

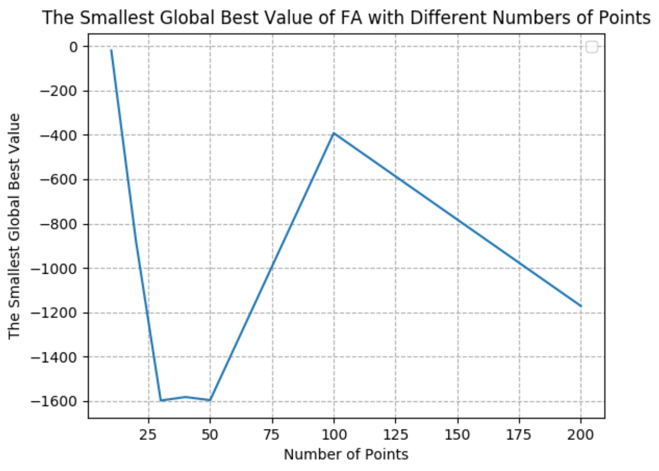

Fig 10: Plot of The Smallest Global Best Values of FA with different numbers of points

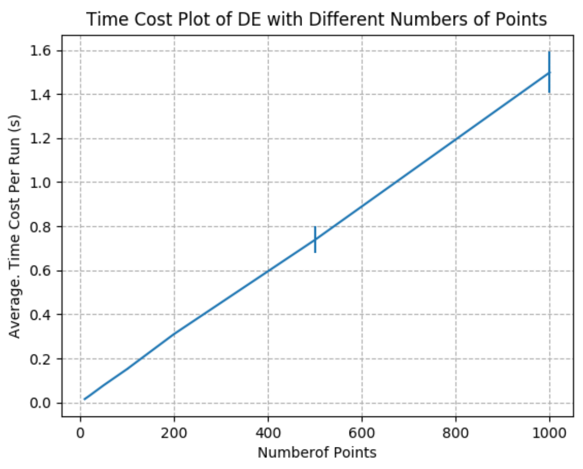

Fig 11: Plot of The Association between The Number of Points And The Time Cost of DE

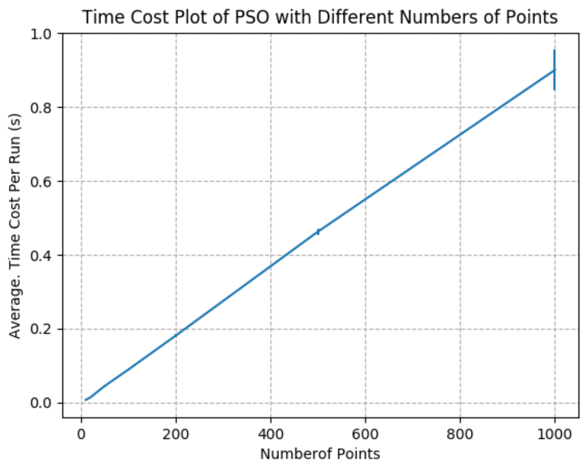

Fig 12: Plot of The Association between The Number of Points And The Time Cost of PSO

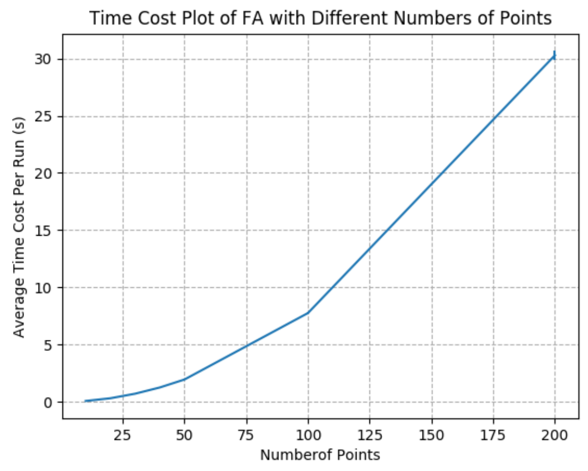

Fig 13: Plot of The Association between The Number of Points And The Time Cost of FA

Curves in Fig 11 and Fig 12 are close to straight lines, clearly tell us that the time complexity of both DE and PSO are $\mathcal{O}(n)$. Fig 13 shows that the time complexity of FA may be more expensive than $\mathcal{O}(n)$.

4 Conclusion

We implemented three well-known meta-heuristic algorithms: DE, PSO, and FA. We found that FA could have a promising performance with only several points but take much higher time cost with more points. On the other hand, DE and PSO could have excellent performance in terms of precision and efficiency with more points (i.e., $N \geq 200$). Compared to DE, PSO had the lower time cost. In the future, we can utilize algorithm of K-means when initialing points to make the initialized points distribute more uniformly.

REFERENCE

1. Reiner Horst, Pardalos Panos M. Eds. Handbook of global optimization. Vol. 2. Springer Science & Business Media. 2013.
2. Kavousi-Fard A., Samet H., Marzbani F. A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting. Expert systems with applications. vol. 41, no. 13, pp. 6047–6056, 2014. doi: 10.1016/j.eswa.2014.03.053
3. Su C. T., Lee C. S. Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution. IEEE Trans. Power Delivery. vol. 18, no.3, pp.1022–1027, 2003. doi: 10. 1109/tpwrd.2003.813641
4. Marichelvam M. K., Geetha M. A hybrid discrete firefly algorithm to solve flow shop scheduling problems to minimize total flow time. International Journal of Bio-Inspired Computation. In press.
5. Yang X. S. Firefly algorithm, Lévy flights and global optimization. Research and development in intelligent systems XXVI. Springer London. 2010, pp. 209–218.
6. Price K., Storn R. M., Lampinen J.A. Differential evolution: a practical approach to global optimization. Springer Science & Business Media. 2006.
7. Storn R., Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization. vol. 11, no. 4, pp. 341–359, 1997.
8. Brest J., S Greiner, Bosˇković B., Mernik M., Zumer V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evolutionary Compu- tation. vol. 10, no. 6, pp. 646–657 2006. doi: 10.1109/tevc.2006.872133
9. Yang X. S. Firefly Algorithm, Nature-inspired meta-heuristic algorithms. Luniver Press. 2010, pp. 79– 90.
10. Yang X. S. Firefly algorithms for multimodal optimization. Stochastic algorithms: foundations and applications. Springer Berlin Heidelberg. 2009, pp. 169–178.
11. Wang H., Cui Z., Sun H., Rahnamayan S., Yang X. Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism. Soft Computing. 2016: 1–15. doi: 10. 1007/s00500-016-2116-z