13

Firefly Algorithm: Variants and Applications

Xin-She Yang

School of Science and Technology
Middlesex University, London, United Kingdom

CONTENTS

13.1 Introduction

The original firefly algorithm (FA) was first developed by Xin-She Yang in late 2007 and early 2008 [1], based on the flashing characteristics of tropical fireflies. Since then, the FA has been applied to a wide range of applications [2], and it has also been extended to multiobjective optimization [3].

In addition, FA has been extended to its corresponding discrete variants so as to solve discrete and combinatorial optimization [4]. Modifications have been carried out to solve a navigation problem [5], and some randomly attracted neighborhood search has also been introduced [6]. Some qualitative and quantitative analysis about the FA may partly explain why FA works well in practice [7].

For more challenging problems such as protein structure prediction, FA was also attempted for lattice models [8]. Scheduling problems on grid computing has been solved using FA [9], and self-adaptive decision-making for robot swarms has been carried out [10]. Furthermore, FA-based feature selection has been used for network intrusion detection [11].

Though randomization has been used for most algorithms, chaos can have some advantages over standard randomization. For example, a fractional calculus-based FA has been developed to solve parameter estimation problems [12], and a chaos-based firefly algorithm can be used to solve large-sized steel dome design problems [13], and other optimization problems [14]. A set of compact FA variants has been developed with various modifications and enhancements [15].

Representations of solution vectors are usually in real numbers. An alternative is to use other representations such as quaternion [16], which shows some improved efficiency for certain types of problems.

Other interesting applications of FA include optimization of modular granular neural networks [17], resource-constrained project scheduling [18], feature selection [19], and clustering [20]. For a relatively comprehensive review on the firefly algorithm and its application, please refer to the edited book [21].

This chapter focuses on the variants of FA and their applications. Let us briefly review the standard FA first.

13.2 Firefly algorithm

A solution vector $\boldsymbol{x}$ to an optimization problem can be considered as the position of a firefly in a D-dimensional space with D independent variables. That is

$$\boldsymbol{x} = [x_1,\ x_2,\ x_3,\ ...,\ x_D], \tag{13.1}$$

where we have used a row vector that can be changed into a column vector by a transpose (T).

13.2.1 Standard FA

The main equation for the firefly algorithm is to calculate the new position vector $\boldsymbol{x}_i^{t+1}$ at iteration t such that

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha \boldsymbol{\epsilon}_i^t, \tag{13.2}$$

which has three terms on the right-hand side. The first term is the current position $\boldsymbol{x}_i^t$, while the second term is the attraction term where β_0 is the attractiveness at zero distance and γ is an absorption coefficient. The third term is a perturbation term with a scaling factor α and the perturbation is

carried out by drawing a random vector $\boldsymbol{\epsilon}_i^t$ from a normal distribution with a zero mean and unity variance.

The Cartesian distance r_{ij} is defined as

$$r_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2 = \sqrt{\sum_{k=1}^{D} \left(x_{i,k} - x_{j,k}\right)^2}, \tag{13.3}$$

where $x_{i,k}$ means the kth component or variable of the solution vector $\boldsymbol{x}_i$.

For the parameter values, we usually use $\beta_0 = 1$ in most applications. However, γ should be linked to the scale L of the problem under consideration. So we often set

$$\gamma = \frac{1}{L^2}. \tag{13.4}$$

For example, if a variable varies from -5 to $+5$, its scale is $L = 10$, so we can use $\gamma = 1/10^2 = 0.01$. If there is no prior knowledge of the scale of the problem, $\gamma = 0.01$ to 1 can be used as an initial guess.

On the other hand, the scaling factor α controls the step size of the movements of the fireflies. Ideally, α should be gradually reduced. In most applications, we can use

$$\alpha = \alpha_0 \theta^t, \tag{13.5}$$

where $0 < \theta < 1$ is a constant, and α_0 is the initial value of α. For example, we can use $\theta = 0.9$ to 0.99 and $\alpha_0 = 1$ [1,21].

13.2.2 Special cases of FA

Though there is only a single update equation (13.2) in FA, it still has much richer dynamics, and a few other algorithms can somehow be considered as special cases of FA.

- In the case of $\gamma = 0$, the attractiveness coefficient becomes a constant β_0. This means that all fireflies are visible and can be seen by other fireflies. Eq. (13.2) becomes

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha \boldsymbol{\epsilon}_t^t. \tag{13.6}$$

  If we impose further that $\alpha = 0$ (no additional randomization), the above equation becomes a key equation in differential evolution (DE) if we now interpret β_0 as the learning or mutation factor F in DE. On the other hand, if we replace $\boldsymbol{x}_j^t$ in Eq. (13.6) by the best solution $\boldsymbol{g}^*$ found so far, it becomes the accelerated particle swarm optimization (APSO) [1].

- In the case of $\gamma \to +\infty$, the attraction term becomes zero, which means that its contribution becomes zero. If we impose $\alpha \neq 0$, the FA equation becomes a random walk, which is essentially the main mechanism in simulated annealing (SA).

This implies that DE, APSO, and SA algorithms can be considered as special cases of the firefly algorithm, and thus it is no surprise that FA can be very effective.

The above discussions also provide possible routes for improving the FA by modifying certain components or tuning parameters. In the rest of this chapter, we will briefly highlight a few such variants.

13.3 Variants of firefly algorithm

There are quite a few variants of the FA and many different modifications. It is not our intention to review most of them in this chapter. Instead, we just highlight a few so as to see the ideas of how different variants have been developed to suit different tasks.

13.3.1 Discrete FA

The standard FA is originally for optimization with continuous variables. For discrete and combinatorial optimization problems, the variables are discrete. Thus, we have to convert the standard FA into a discrete version. One way to discretize a continuous variable x is to use the sigmoidal or logistic function

$$S(x) = \frac{1}{1 + e^{-x}}, \tag{13.7}$$

which is an S-shaped function. It essentially converts a continuous variable x into a binary variable S when $|x|$ is large. When $x \to +\infty$, $S \to 1$. When $x \to -\infty$, $S \to 0$. However, this is not easy to implement in practice. So a random number $r \in [0, 1]$ is usually generated and used as a conditional switch. That is, if $S(x) > r$, $u = 1$, otherwise $u = 0$, which gives a binary variable $u \in \{0, 1\}$.

Once we have a binary variable $u \in \{0, 1\}$, we can convert it to binary variables with other discrete values. For example, we can use $y = 2u - 1 \in \{-1, +1\}$.

It is worth pointing out that a useful property of $S(x)$ is that its derivative can be computed by multiplication

$$\frac{dS}{dt} = -\frac{1}{(1 + e^{-x})^2}(-e^{-x}) = \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}}$$
$$= \frac{1}{1 + e^{-x}} \cdot \frac{[(1 + e^{-x}) - 1]}{1 + e^{-x}} = \frac{1}{1 + e^{-x}} \cdot [1 - \frac{1}{1 + e^{-x}}] = S(1 - S). \tag{13.8}$$

Another way for discretization is to use round-up operations to get integer values. For example, we can use

$$y = \lfloor x \rfloor, \tag{13.9}$$

to convert x to integer y.

Alternatively, we can convert a continuous variable into m discrete integers by using a mod function

$$y = \lfloor x + k \rfloor \mod m, \tag{13.10}$$

where k and $m > 0$ are integers.

Sometimes, a so-called random key can be used to generate a set of discrete values for nodal numbers (as in the travelling salesman problem) and job numbers as in job shop scheduling problems. For example, with a random key

$$x = [0.91,\ 1.1,\ 0.14,\ 0.09,\ 0.77, -0.23, 0.69], \tag{13.11}$$

which can be converted to

$$J = [6,\ 7,\ 3,\ 2,\ 5,\ 1,\ 4]. \tag{13.12}$$

This is done by ranking the real number vector x first, and then transforming them into labels of ranks. In some applications, such continuous numbers are drawn from uniformly distributed numbers in [0,1]. For example, we have

$$\begin{pmatrix} \text{Real numbers} & 0.65 & 0.25 & 0.37 & 0.04 & 0.89 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \text{Random keys} & 4 & 2 & 3 & 1 & 5 \end{pmatrix}. \tag{13.13}$$

Such random-keys-based approaches have been applied in many applications such as the travelling salesman problem (TSP), vehicle routing problems [4] and scheduling problems [9, 21].

13.3.2 Chaos-based FA

One way to potentially enhance the performance of algorithms, including FA, is to use chaotic maps to replace the fixed values of algorithmic parameters. In the case of FA with chaos [14], they used 12 different chaotic maps such as the Chebyshev map, Gauss map and Logistic map to replace β and γ. For example, the Chebyshev map is based on Chebyshev polynomials and can be written as

$$u_{k+1} = \cos(k \cos^{-1}(u_k)), \tag{13.14}$$

which generates a chaotic sequence in the range of [-1, +1] (for $n = 1, 2, 3, ...$). The Logistic map is often written as

$$u_{k+1} = \lambda u_k (1 - u_k), \tag{13.15}$$

which gives a chaotic sequence in (0,1) for $\lambda = 4$ when $u_0 \neq 0$.

Extensive simulations carried out by Gandomi et al. [14] indicate that the best map for this purpose is the Gauss map

$$u_{k+1} = \frac{1}{u_k} - \left\lfloor \frac{1}{u_k} \right\rfloor, \quad u_k \neq 0, \tag{13.16}$$

which essentially takes the mod operation to generate a chaotic sequence. The main reason for this enhancement is probably that chaotic maps tend to increase the mixing ability of different solutions/fireflies in the population, and thus increase the mobility and the probability of finding the global optimality of complex objective functions.

Other chaos-based improvement has been been investigated with promising results [13]. Similar enhancement can be also achieved by quantum-based approaches where the parameter values are usually replaced by the probability derived from quantum mechanics.

13.3.3 Randomly attracted FA with varying steps

Another type of variants may focus more on the variations of parameters related to iteration counter t. For example, in the randomly attracted FA variant, Wang et al. [6] used a formula for varying the attractiveness $\beta = \beta_0 e^{-\gamma r_{ij}^2}$ as

$$\beta = [\beta_{\min} + (\beta_{\max} - \beta_{\min}) e^{-\gamma r_{ij}^2}] \frac{t}{G_{\max}}, \tag{13.17}$$

where $\beta_{\max}$ and $\beta_{\min}$ are the maximum and minimum of β, respectively, though $\beta_{\max} = 0.9$ and $\beta_{\min} = 0.3$ were used in their paper. Here, $G_{\max}$ is the maximum number of iterations or generations.

In addition, the full attraction model was used in the standard FA. That is, all the fireflies will be attracted to the best firefly. The advantage of this approach is that it can lead to very good convergence; however, if the attraction is too strong, premature convergence can occur under certain conditions. Thus, there is a need to reduce strong attraction. One modification is to use a random attraction approach as outlined by Wang et al. [6]. The main idea is that each better firefly can only be attracted to another firefly that is randomly selected. This is further enhanced by neighbourhood search in terms of one local operator and two global search operators. This FA variant showed a very good improvement in performance.

13.3.4 FA via Lévy flights

Another way of increasing the mobility of fireflies is to use other probability distributions. In the original FA, the random number vector $\boldsymbol{\epsilon}_t^t$ in the perturbation term is drawn from either a uniform distribution or a Gaussian distribution. FA via Lévy flights [1, 21] uses Lévy flights to replace the Gaussian distribution. Lévy flights are a random walk with step sizes s drawn from a Lévy distribution in the approximate power-law form

$$L(s) \sim \frac{1}{s^{1+\lambda}}, \quad 0 < \lambda \leq 2. \tag{13.18}$$

It is worth pointing out that this form is an approximation, because the Lévy distribution should strictly be defined in terms of an integral form, which makes it difficult to draw random numbers [1].

The advantage of Lévy flights is that such flights consist of a fraction of large step sizes in addition to many local steps. This leads to a phenomenon, called super-diffusion. As a result, the diversity of the firefly population is enhanced and the solutions generated can be sufficient far away from any local optima. This will ultimately increase the probability of finding the true global solution to the optimization problem under consideration [21].

13.3.5 FA with quaternion representation

In almost all metaheuristic algorithms (including FA), representations in terms of real numbers are used for formulating problems and representing the values of design variables. Such representations have rigorous mathematical foundations. In many applications such as electrical engineering, complex numbers (i.e., $a+bi$) are often used. In addition, there are other mathematical representations such as quaternions and octonions, which are the extensions of complex numbers. Such representations can have certain advantages. For example, in computational geometry, quaternion-based representations can be useful for manipulating 3D geometrical shapes.

Along this line, a quaternion-based FA variant was developed by Fister et al. [16]. The main idea was to use quaternions

$$q = a + bi + cj + dk, \tag{13.19}$$

where a, b, c, d are real numbers, while i, j, k satisfy Hamiltonian permutation conditions

$$ij = k, \quad jk = i, \quad ki = j, \tag{13.20}$$

$$ji = -k, \quad kj = -i, \quad ik = -j, \tag{13.21}$$

and

$$i^2 = j^2 = k^2 = -1. \tag{13.22}$$

Such quanternions obey standard quaternion algebra. For example, the norm of q is defined as

$$||q|| = \sqrt{a^2 + b^2 + c^2 + d^2}. \tag{13.23}$$

Though it seems that such representations may increase the computational efforts, the performance can indeed improve without any big increase in computation costs, as shown in the work by Fister et al. [16].

13.3.6 Multi-objective FA

The standard FA was initially designed for solving single objective optimization problems. However, most real-world problems are intrinsically multi-objective optimization. Thus, FA has been extended to solve multi-objective optimization by Yang in 2013 [3]. For example, even a simple bi-objective optimization problem such as Schaffer's min-min test function

$$f_1(x) = x^2, \quad f_2(x) = (x-2)^2, \quad -1000 \leq x \leq 1000, \tag{13.24}$$

requires some modifications to algorithms for single objective optimization. This bi-objective optimization problem can have many solutions (in fact, an infinite number of solutions) forming a so-called Pareto front.

One simple approach is to use a weighted sum to combine the two functions into a single objective

$$f(x) = af_1 + bf_2, \quad a + b = 1, \quad a, b \in [0, 1]. \tag{13.25}$$

Clearly, the optimal solution of $f(x)$ will depend on a (or $b = 1 - a$). As a varies from 0 to 1 (thus b from 1 to 0), a Pareto front can be traced. Thus, for a multi-objective optimization problem with m objective functions

$$\text{Minimize} \quad f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^D, \tag{13.26}$$

we can convert it into a single objective problem

$$\text{Minimize} \quad F(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^D, \tag{13.27}$$

where

$$F(\boldsymbol{x}) = \sum_{i=1}^{m} w_i f_i(\boldsymbol{x}), \quad \sum_{i=1}^{m} w_i = 1, \quad \forall w_i \in [0, 1]. \tag{13.28}$$

Once this step is done properly, we can use the standard FA to solve it [3].

Numerical experiments show that the multi-objective firefly algorithm (MOFA) can be effective, in comparison with other algorithms such as multi-objective differential evolution (MODE), non-dominated sorting genetic algorithm (NSGA) and vector-evaluated genetic algorithm (VEGA) [3].

Though this approach is simple to implement and can be effective, it has some limitations because this approach is valid only if the Pareto front is convex. This is true for the above bi-objective problem and many problems, but many applications can have non-convex Pareto fronts. Therefore, some other approaches such as non-dominated sorting and ϵ-constraint methods should be used, in combination with the FA variants.

13.3.7 Other variants of FA

There are many other variants of FA and interested readers can refer to more advanced literature [2, 12, 15, 21]. The intention here is not to provide a comprehensive review, but to highlight ways to improve and modify the standard FA so as to inspire further research for designing more and better variants.

It is worth pointing out that the above ways of modifying the original FA to design different FA variants can also be used and extended to modify other algorithms such as the cuckoo search (CS), the bat algorithm (BA) and the flower pollination algorithm (FPA) as well as other algorithms such as particle swarm optimization (PSO).

In the rest of this chapter, we will review some applications and then conclude with some discussions.

13.4 Applications of FA and its variants

The applications of FA and its variants are very diverse, and it is again not our intention to review even a good fraction of them. Instead, we only highlight a few areas:

- Design optimization: A main class of application of the FA is design optimization in engineering and industry, such as pressure-vessel design, beam design and structure design [2, 3, 13]. Such design problems can be multimodal with multiple optimal solutions. One of the advantages of FA is that the overall swarm can subdivide into multiple subswarms automatically, which makes it naturally suitable for solving multimodal problems [21].

- Travelling salesman problem: Both the travelling salesman problem and vehicle routing problems are hard problems, and there are no efficient methods for solving such large-scale problems. Thus, some metaheuristics and approximations are needed. Preliminary studies [4] suggested that FA can obtain good results for such problems. Similarly, navigation problems for unmanned vehicles and path planning are also challenging problems, and recent studies show that FA can also be effective in this area [5].

- Scheduling: Scheduling is another class of combinatorial problems, which is not only difficult to solve, but also requires discretizing algorithms properly. For example, task scheduling for grid computing has been successfully carried out by FA [9]. Resource-constrained project scheduling has also been carried out by FA [18].

- Protein folding: The so-called protein folding problem is an NP-complete problem, and there are no efficient methods for tackling such problems. Due to its importance in computational biology and pharmaceutical applications, researchers have attempted various methods, including metaheuristic algorithms. For example, in a study by Maher et al. [8], they showed that the firefly-based approach can speed up the simulations.

- Feature selection and clustering: Feature selection is widely used in many applications. For example, detection of network intrusion can be carried out by FA [11], while a binary FA has been applied for return-cost-based feature selection application [19]. A study on clustering and satellite imaging has shown that FA can obtain the best results with the least computing efforts, among the 14 different algorithms and approaches [20].

- Machine learning: Machine learning has become popular in recent years. Though there are a class of specialized techniques for such applications, there are many problems and challenging issues that require alternative approaches. For example, optimization of granular neural networks has been carried out by FA with promising results, while a multi-objective approach

based on the firefly algorithm and ant colony optimization has been successfully used to carry out self-adaptive decision making for a swarm of robots without central control [10].

There are many other applications and the literature is rapidly expanding. Interested readers can refer to recent journal articles in these areas.

13.5 Conclusion

The firefly algorithm is simple, versatile and yet effective in solving many optimization problems in applications. This chapter introduces the basic form of the standard firefly algorithm and then highlights different ways of improving it and designing new variants. Applications have been briefly reviewed with different categories.

Despite the good performance of the FA and many other algorithms, it still lacks mathematical understanding why such metaheuristic algorithms can work well. Thus, some rigorous mathematical analyses in addition to extensive simulations are greatly needed to gain in-depth understanding of the metaheuristic algorithms.

In addition, different variants can have different advantages and also disadvantages. It would be useful to explore the further possibility of designing new variants by other approaches such as hybridization with other algorithms including traditional well-tested optimization algorithms.

Furthermore, most of these applications are small and moderate scales in the sense that the number of variables is typically a few dozens or a few hundred at most. Real-world applications can have thousands or even millions of design variables. As computers get faster and cheaper, a wide range of tools emerge, including grid computing and cloud computing. Thus, it would be useful to extend and explore how to use the FA and its variants to solve large-scale real-world applications.

References

1. X.S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK, 2008.
2. I. Fister, I. Fister Jr., X.S. Yang, J. Brest, "A comprehensive review of firefly algorithms". *Swarm and Evolutionary Computation*, 13(1):34-46, 2013.

3. X.S. Yang, "Multiobjective firefly algorithm for continuous optimization", *Engineering with Computer*, 29(2): 175-184, 2013.
4. E. Osaba, X.S. Yang, F. Diaz, E. Onieva, A.D. Masegosa, A. Perallos, "A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy", *Soft Computing*, 21(18): 5295-5308, 2017.
5. Y. Ma, Y. Zhao, L. Wu, Y. He, X.S. Yang, "Navigability analysis of magnetic map with projecting pursuit-based selection method by using firefly algorithm", *Neurocomputing*, 159(1): 288-297, 2015.
6. H. Wang, Z. Cui, H. Sun, S. Rahnamayan, X.S. Yang, "Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism", *Soft Computing*, 21(18): 5325-5339, 2017.
7. X.S. Yang and X. S. He, "Why the firefly algorithm works", in: *Nature-Inspired Algorithms and Applied Optimization* (Edited by X.S. Yang), Springer, pp. 245-259, 2018.
8. B. Maher, A. Albrecht, M. Loomes, X.S. Yang, K. Steinhöfel, "A firefly-inspired method for protein structure prediction in lattice models", *Biomolecules*, 4(1): 56-75, 2014.
9. A. Yousif, A.H. Abdullah, S.M. Nor, A. Abdelaziz, "Scheduling jobs on grid computing using firefly algorithm", *J. Theor. Appl. Inform. Technol.*, 33(2): 155-164, 2011.
10. N. Palmieri, X.S. Yang, F. De Rango, A.F. Santamaria, "Self-adaptive decision-making mechanisms to balance the execution of multiple tasks for a multi-robots team", *Neurocomputing*, 306(1): 17-36, 2018.
11. B. Selvakumar, K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection", *Computers & Security*, 81(1): 148-155, 2019.
12. Y. Mousavi, A. Alfi, "Fractional calculus-based firefly algorithm applied to parameter estimation of chaotic systems", *Chaos, Solitons & Fractals*, 114(1): 202-215, 2018.
13. A. Kaveh, S. M. Javadi, "Chaos-based firefly algorithms for optimization of cyclically large-size braced steel domes with multiple frequency constraints", *Computers & Structures*, 214(1): 28-39, 2019.
14. A.H. Gandomi, X.S. Yang, S. Talatahari, A. H. Alavi, "Firefly algorithm with chaos", *Commun. Nonlinear Sci. Numer. Simulation*, 18(1): 89-98, 2013.
15. L. Tighzert, C. Fonlupt, B. Mendil, "A set of new compact firefly algorithms", *Swarm and Evolutionary Computation*, 40 (1): 92-115, 2018.

16. I. Fister, X.S. Yang, J. Brest, I. Fister Jr., "Modified firefly algorithm using quaternion representation", *Expert Systems with Applications*, 40(18): 7220-7230, 2013.

17. D. Sánchez, P. Melin, O. Castillo, "Optimization of modular granular neural networks using a firefly algorithm for human recognition", *Engineering Applications of Artificial Intelligence*, 64(1): 172-186, 2017.

18. T. Kassandra, Rojaili, D. Suharono, "Resource-constrained project scheduling problem using firefly algorithm", *Procedia Computer Science*, 135: 534-543, 2018.

19. Y. Zhang, X.F. Song, D.W. Gong, "A return-cost-based binary firefly algorithm for feature selection", *Information Sciences*, 418-419: 561-574, 2017.

20. J. Senthilnath, S.N. Omkar, V. Mani, "Clustering using firefly algorithm: performance comparison", *Swarm and Evolutionary Computation*, 1(3): 164-171, 2011.

21. X.S. Yang, *Cuckoo Search and Firefly Algorithm: Theory and Applications*, Studies in Computational Intelligence, vol. 516, Springer, Heidelberg, 2014.