On the Randomized Firefly Algorithm

Iztok Fister, Xin-She Yang, Janez Brest and Iztok Fister Jr.

Abstract The firefly algorithm is a stochastic meta-heuristic that incorporates randomness into a search process. Essentially, the randomness is useful when determining the next point in the search space and therefore has a crucial impact when exploring the new solution. In this chapter, an extensive comparison is made between various probability distributions that can be used for randomizing the firefly algorithm, e.g., Uniform, Gaussian, Lévi flights, Chaotic maps, and the Random sampling in turbulent fractal cloud. In line with this, variously randomized firefly algorithms were developed and extensive experiments conducted on a well-known suite of functions. The results of these experiments show that the efficiency of a distributions largely depends on the type of a problem to be solved.

Keywords Chaos · Firefly algorithm · Randomization · Random sampling in turbulent fractal cloud · Swarm intelligence

I. Fister (✉) · J. Brest · I. Fister Jr.
Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia
e-mail: iztok.fister@uni-mb.si

J. Brest
e-mail: janez.brest@uni-mb.si

I. Fister Jr.
e-mail: iztok.fister@guest.arnes.si

X.-S. Yang
School of Science and Technology, Middlesex University, London, UK
e-mail: x.yang@mdx.ac.uk

X.-S. Yang (ed.), *Cuckoo Search and Firefly Algorithm*,
Studies in Computational Intelligence 516, DOI: 10.1007/978-3-319-02141-6_2,

1 Introduction

Automatic problem solving with a digital computer has been the eternal quest of researchers in mathematics, computer science and engineering. The majority of complex problems (also NP-hard problems [1]) cannot be solved using exact methods by enumerating all the possible solutions and searching for the best solution (minimum or maximum value of objective function). Therefore, several algorithms have been emerged that solve problems in some smarter (also heuristic) ways. Nowadays, designers of the more successful algorithms draw their inspirations from Nature. For instance, the collective behavior of social insects like ants, termites, bees and wasps, or some animal societies like flocks of bird or schools of fish have inspired computer scientists to design intelligent multi-agent systems [2].

For millions of years many biological systems have solved complex problems by sharing information with group members [3]. These biological systems are usually very complex. They consists of particles (agents) that are definitely more complex than molecules and atoms, and are capable of performing autonomous actions within an environment. On the other hand, a group of particles is capable of intelligent behavior which is appropriate for solving complex problems in Nature. Therefore, it is no coincidence that these biological systems have also inspired computer scientists to imitate their intelligent behavior for solving complex problems in mathematics, physics, engineering, etc. Moreover, interest in researching various biological systems has increased recently. These various biological systems have been influenced by swarm intelligence (SI) that can be viewed as an artificial intelligence (AI) discipline concerned with the designing of intelligent systems.

It seems that the first use of the term 'swarm intelligence' was probably by Beni and Wang [4] in 1989 in the context of a cellular robotic system. Nowadays, this term also extends to the field of optimization, where techniques based on swarm intelligence have been applied successfully. Examples of notable swarm intelligence optimization techniques are ant colony optimization [5], particle swarm optimization [6], and artificial bees colony (ABC) [7, 8]. Today, the more promising swarm intelligence optimization techniques include the firefly algorithm (FA) [9–14], the cuckoo search [15], and the bat algorithm [16, 17].

Stochastic optimization searches for optimal solutions by involving randomness in some constructive way [18]. In contrast, if optimization methods provide the same results when doing the same things, these methods are said to be deterministic [19]. If the deterministic system behaves unpredictably, it arrives at a phenomenon of chaos [19]. As a result, randomness in SI algorithms plays a huge role because this phenomenon affects the exploration and exploitation in search process [20]. These companions of stochastic global search represent the two cornerstones of problem solving, i.e., exploration refers to moves for discovering entirely new regions of a search space, while exploitation refers to moves that focus searching the vicinity of promising, known solutions to be found during the search process. Both components are also referred to as intensification and diversification in another terminology [21]. However, these refer to medium- to long- term strategies based on the usage of mem-

ory (Tabu search), while exploration and exploitation refer to short-term strategies tied to randomness [20].

Essentially, randomness is used in SI algorithms in order to explore new points by moving the particles towards the search space. In line with this, several random distributions can be helpful. For example, uniform distribution generates each point of the search space using the same probability. On the other hand, Gaussian distribution is biased towards the observed solution, that is that the smaller modifications occur more often than larger ones [22]. On the other hand, the appropriate distribution depends on the problem to be solved, more precisely, on a fitness landscape that maps each position in the search space into fitness value. When the fitness landscape is flat, uniform distribution is more preferable for the stochastic search process, whilst in rougher fitness landscapes Gaussian distribution should be more appropriate.

This chapter deals with an FA algorithm that uses different randomization methods, like Uniform, Gaussian, Lévy flights, Chaotic maps, and the Random sampling in turbulent fractal cloud. Whilst the observed randomization methods are well-known and widely used (e.g., uniform, Gaussian, Lévi flights, and chaotic maps), the random sampling in turbulent fractal cloud is taken from astronomy and, as we know, it is the first time used for the optimization purposes. Here, two well-known chaotic maps are also taken into account, i.e., Kent and Logistic.

The goal of our experimental work is to show how the different randomized methods influence the results of the modified FA. In line with this, a function optimization problem was solved by this algorithm. A suite of ten well-known functions were defined, as defined in the literature.

The structure of the rest of the chapter is as follows: Sect. 2 describes a background information of various randomization methods. In Sect. 3, the randomized firefly algorithms are presented. Experiments and results are illustrated in Sect. 4. The chapter is finished summarizing our work and the directions for further development are outlined.

2 Background Information

This section describes the background information about generating random variates and computing their probability distributions, as follows:

- Uniform,
- Normal or Gaussian,
- Lévy flights,
- Chaotic maps,
- Random sampling in turbulent fractal cloud.

Continuous random number distributions [23] are defined by a probability density function $p(x)$, such that the probability of x occurring within the infinitesimal range x to $x + dx$ is $p \cdot dx$. The cumulative distribution function for the lower tail $P(x)$ is defined as the integral

$$P(x) = \int_{-\infty}^{x} p(x) \cdot dx, \tag{1}$$

which gives the probability of a variate taking a value of less than x. The cumulative distribution function for the upper tail $Q(x)$ is defined as the integral

$$Q(x) = \int_{x}^{+\infty} p(x) \cdot dx, \tag{2}$$

which provides the probability of a value greater than x.

The upper and lower cumulative distribution functions are related by $P(x) + Q(x) = 1$ and satisfy the following limitations $0 \leq P(x) \leq 1$ and $0 \leq Q(x) \leq 1$. In the remainder of this section, these randomized methods are presented in more detail.

2.1 Uniform Distribution

Uniform continuous distribution has the density function, as follows:

$$p(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Note that each possible value of the uniform distributed random variable is within optional interval $[a, b]$, on which the probability of each sub-interval is proportional to its length. If $a \leq u < v \leq b$ then the following relation holds:

$$P(u < x < v) = \frac{v-u}{b-a}. \tag{4}$$

Normally, the uniform distribution is obtained by a call to the random number generator [23]. Note that the discrete variate functions always return a value of type unsigned which on most platforms means a random value from the interval $[0, 2^{32} - 1]$. In order to obtain the random generated value within the interval [0, 1], the following mapping is used:

$$r = ((double)rand()/((double)(\text{RAND\_MAX}) + (double)(1))), \tag{5}$$

where r is the generated random number, the function $rand()$ is a call of the random number generator, and the RAND_MAX is the maximal number of the random value $(2^{32} - 1)$.

2.2 Normal or Gaussian Distribution

Normal or Gaussian distribution is defined with the following density function:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-a}{\sigma})^2}. \tag{6}$$

The distribution depends on parameters $a \in \mathbb{R}$ and $\sigma > 0$. This distribution is denoted as $\mathrm{N}(a, \sigma)$. The standardized normal distribution is obtained, when the distribution has an average value of zero with standard deviation of one, i.e., $\mathrm{N}(0, 1)$. In this case, the density function is simply defined as:

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \tag{7}$$

The Gaussian distribution has the property that approximately $2/3$ of the samples drawn lie within one standard deviation [22]. That is, the most of the modifications made on the virtual particle will be small, whilst there is a non-zero probability of generating very large modifications, because the tail of distribution never reaches zero [22].

2.3 Lévy Flights

In reality, resources are distributed non-uniformly in Nature. This means, that the behavior of a typical forager needing to find these resources as fast as possible does not obey Gaussian distribution. In order to simulate foragers search strategies, Lévy flights is closer to their behavior [24]. It belongs to a special class of α-stable distributions defined by a Fourier transform [23]:

$$p(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-itx - |ct|^\alpha}. \tag{8}$$

The α-stable means that it exhibits the same probability density distributions for each randomly generated variable. This density function has two parameters: scale $c \in \mathbb{R}$ and exponent $\alpha \in [0, 2]$. A main characteristic of this distribution is that it has an infinite variance.

Interestingly, for $\alpha = 1$ the density function reduces to the Cauchy distribution, whilst for $\alpha = 2$ it is a Gaussian distribution with $\sigma = \sqrt{2c}$. For $\alpha < 1$ the tails of the distribution become extremely wide [24]. The more appropriate setting of this parameter for optimization is therefore $\alpha \in (1.0, 2.0)$, where the distribution is non-Gaussian with no variance (as in the case of Lévy flights) or with no mean, variance or higher moments defined (as in the case of Cauchy distribution). Essentially, the difference between non-Gaussian and Gaussian distributions is that the tail of the

distribution by the former is wider than by the latter. This means, the probability of generating very large modifications is much higher by Lévy flights than by Gaussian distribution.

2.4 Chaotic Maps

Chaos is a phenomenon encountered in science and mathematics wherein a deterministic (rule-based) system behaves unpredictably [19]. Let us assume a Logistic equation defined as a map:

$$x_{n+1} = r x_n (1 - x_n), \tag{9}$$

where $x_n \in (0, 1)$ and parameter r is a parameter. A generated sequence of numbers by iterating a Logistic map (also orbit) with $r = 4$ are *chaotic*. That is, it posses the following propositions [19]:

- the dynamic rule of generating the sequence of numbers is deterministic,
- the orbits are aperiodic (they never repeat),
- the orbits are bounded (they stay between upper and lower limits, normally, within the interval [0, 1]),
- the sequence has sensitive dependence on the initial condition (also SDIC).

Similar behavior can also be observed by the Kent map [25]. The Kent map is one of the more studied chaotic maps that has been used to generate pseudo-random numbers in many applications, like secure encryption. It is defined as follows:

$$x(n+1) = \begin{cases} \frac{x(n)}{m}, & 0 < x(n) \leq m, \\ \frac{(1-x(n))}{1-m}, & m < x(n) < 1, \end{cases} \tag{10}$$

where $0 < m < 1$. Hence, if $x(0) \in (0, 1)$, for all $n \geq 1, x(n) \in [0, 1]$. In accordance with the propositions in [25], $m = 0.7$ was used in our experiments.

2.5 Random Sampling in Turbulent Fractal Cloud

Stars formation begins with a random sampling of mass in a fractal cloud [26]. This random sampling is performed by the initial mass function (IMF) and represents a basis for a new sampling method named as random sampling in turbulent fractal cloud (RSiTFC).

The method can be described as follows. Let us consider a fractal cloud that is divided into a hierarchical structure consisting of several levels with a certain number of cloud pieces containing a certain number of sub-pieces. Then, a sampling method randomly samples a cloud piece from any level. The sampled pieces at the top of this

hierarchical structure are denser than the pieces at the bottom. When the cloud piece is chosen the initial mass of that piece is identified and the piece representing the formed star is removed from the hierarchy. This process is repeated until all of the cloud is chosen [26]. This mentioned method is formally illustrated in Algorithm 1.

The algorithm RSiTFC consists of six parameters: scaling factor L, number of levels H, number of sub-pieces for each piece N, fractal cloud pieces x, level h, and piece number i. The scaling factor is determined as $S = L^{-h}$ when calculated from the fractal dimension expressed as $D = \frac{\log N}{\log L}$. The number of levels H determines the depth of the hierarchical structure. The number of pieces increases with level h according to Poisson distribution $P_N(h) = N^h e^{-N}/h!$. The length of fractal cloud pieces x is limited by N^H and consists of elements representing the initial mass of the star to be formed. Level h denotes the current hierarchical level, whilst i determines the star to be formed.

Algorithm 1 RSiTFC(L, H, N, x, h, i)

Input: L scaling factor, H number of levels, N number of sub-pieces
Input: $*x$ fractal cloud, h current level, $*i$ piece number

```
if(i == 0)
  x = new double [N^H];
  for(j = 0; j < N^h; j++)
    x[*i] = 2 * (U(0, 1) − 0.5)/L^h + 0.5;
    *i = *i + 1;
  end for
else
  for(j = 0; j < N^h; j++)
    x[*i] = x[*i] + 2 * (U(0, 1) − 0.5)/L^h;
    *i = *i + 1;
  end for
end if
if(h < H)
  return RSiTFC(L, H, N, x, h + 1, i);
end if
return x;
```

For example, let $N = 2$ be a constant number of sub-pieces for each piece. Then, there is one cloud at the top level $h = 0$, with two pieces inside this cloud at $h = 1$, etc. For $H = 4$, the total number of pieces is expressed as $1 + 2 + 4 + 8 + 16 = 31$ [26].

3 Randomized Firefly Algorithms

Fireflies (Coleoptera: Lampyridae) are well known for bioluminescent signaling, which is used for species recognition and mate choosing [27]. Bioluminiscence that comprises a complicated set of chemical reactions is not always a sexual signal only but also warns off potential predators. In the remainder of the chapter, an original FA

algorithm is described that captures the bioluminiscent behavior of fireflies within the fitness function. Further, this algorithm is then modified with various randomized methods.

3.1 Original Firefly Algorithm

The light-intensity I of the flashing firefly decreases as the distance from source r decreases in terms of $I \propto 1/r^2$. Additionally, air absorption causes the light to become weaker and weaker as the distance from the source increases. This flashing light represented the inspiration for developing the FA algorithm by Yang [9] in 2008. Here, the light-intensity is proportional to the objective function of the problem being optimized (i.e., $I(\mathbf{s}) \propto f(\mathbf{s})$, where $\mathbf{s} = S(\mathbf{x})$ represent a candidate solution).

In order to formulate the FA, some flashing characteristics of fireflies were idealized, as follows:

- All fireflies are unisex.
- Their attractiveness is proportional to their light intensity.
- The light intensity of a firefly is affected or determined by the landscape of the objective function.

Note that light-intensity I and attractiveness are in some way synonymous. While the intensity I is referred to as an absolute measurement of emitted light by firefly, the attractiveness is a relative measurement of the light that should be seen in the eyes of beholders and judged by the other fireflies [9]. The light intensity I varies with distance r is expressed by the following equation

$$I(r) = I_0 e^{-\gamma r^2}, \tag{11}$$

where I_0 denotes the intensity of the light at the source, and γ is a fixed light absorption coefficient. Similarly, the attractiveness β that also depends on the distance r is calculated according to the following generalized equation

$$\beta(r) = \beta_0 e^{-\gamma r^2}. \tag{12}$$

The distance between two fireflies i and j is represented as the Euclidian distance

$$r_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\| = \sqrt{\sum_{k=1}^{D} s_{ik} - s_{jk}}, \tag{13}$$

where s_{ik} is the k-th element of the i-th firefly position within the search-space, and D denotes the dimensionality of a problem. Each firefly i moves to another more attractive firefly j, as follows

$$\mathbf{s}_i = \mathbf{s}_i + \beta_0 e^{-\gamma r_{ij}^2}(\mathbf{s}_j - \mathbf{s}_i) + \alpha \cdot N_i(0, 1). \tag{14}$$

Equation (14) consists of three terms. The first term determines the position of the i-th firefly. The second term refers to the attractiveness, while the third term is connected with the randomized move of the i-th firefly within the search-space. This term consists of the randomized parameter α, and the random numbers $N_i(0, 1)$ drawn from a Gaussian distribution. The scheme of FA is sketched in Algorithm 1. The FA algorithm (Algorithm 2) runs on the population of fireflies $P^{(t)}$ that are

Algorithm 2 Original FA algorithm

1: $t = 0$; $\mathbf{s}^* = \emptyset$; $\gamma = 1.0$; // initialize: gen.counter, best solution, attractiveness
2: $\mathbf{P}^{(t)}$=InitFA(); // initialize the firefly population $\mathbf{s_i^{(0)}} \in \mathbf{P}^{(0)}$
3: **while** $t \leq$ MAX_GEN **do**
4: $\alpha^{(t)}$ = AlphaNew(); // determine a new value of α
5: EvaluateFA($\mathbf{P}^{(t)}$, $f(\mathbf{s})$); // evaluate $\mathbf{s}_i^{(t)}$ according to $f(\mathbf{s}_i)$
6: OrderFA($\mathbf{P}^{(t)}$, $f(\mathbf{s})$); // sort $\mathbf{P}_i^{(t)}$ according to $f(\mathbf{s}_i)$
7: s^* = FindTheBestFA($\mathbf{P}^{(t)}$, $f(\mathbf{s})$); // determine the best solution $\mathbf{s}^*$
8: $\mathbf{P}^{(t+1)}$=MoveFA($\mathbf{P}^{(t)}$); // vary attractiveness according Eq. (14)
9: $t = t + 1$;
10: **end while**
11: return $\mathbf{s}^*$, $f(\mathbf{s})$; // post process

represented as real-valued vectors $\mathbf{s}^{(t)}{}_i = s_{i0}^{(t)}, \ldots s_{in}^{(t)}$, where $i = 1 \ldots NP$ and NP denotes the number of fireflies in population $P^{(t)}$ at generation t. Note that each firefly $\mathbf{s}_i^{(t)}$ is of dimension D. The population of fireflies is initialized randomly (function InitFA) according to equation

$$s_{ij}^{(0)} = (ub_i - lb_i) \cdot \text{rand}(0, 1) + lb_i, \tag{15}$$

where ub_i and lb_i denote the upper and lower bounds, respectively. The main loop of the firefly search process that is controlled by the maximum number of generations *MAX_GEN* consists of the following functions. Firstly, the new values for the randomization parameter α is calculated according to the following equation (function AlphaNew):

$$\begin{aligned} \Delta &= 1 - 10^{-4}/0.9^{1/MAX\_GEN}, \\ \alpha^{(t+1)} &= 1 - \Delta \cdot \alpha^{(t)}, \end{aligned} \tag{16}$$

where Δ determines the step size of changing the parameter $\alpha^{(t+1)}$. Note that this parameter monotony descends with the increasing of generation counter t. Secondly, the new solution $\mathbf{s}_i^{(t)}$ is evaluated according to a fitness function $f(\mathbf{s}^{(t)})$, where $\mathbf{s}_i^{(t)} = S(\mathbf{x}_i^{(t)})$ (function EvaluateFA). Thirdly, solutions $\mathbf{s}_i^{(t)}$ for $i = 1 \ldots NP$ were ordered with respect to the fitness function $f(\mathbf{s}_i^{(t)})$ ascending, where $\mathbf{s}_i^{(t)} = S(\mathbf{x}_i^{(t)})$ (function

OrderFA). Fourthly, the best solution the best solution $\mathbf{s}^* = \mathbf{s}_0^{(t)}$ is determined in the population $P^{(t)}$ (function FindTheBestFA). Finally, the virtual fireflies are moved (function MoveFA) towards the search space according to the attractiveness of their neighbors' solution (Eq. 14).

In the remainder of this paper, the randomized FA is discussed in more detail.

3.2 Variants of the Randomized Firefly Algorithm

Randomized FA (RFA) is based on the original FA that is upgraded using the mentioned randomized methods. In place of the original Eq. (14), the modified equation is used by RFA, as follows:

$$\mathbf{s}_i = \mathbf{s}_i + \beta_0 e^{-\gamma r_{ij}^2}(\mathbf{s}_j - \mathbf{s}_i) + \alpha \cdot R_i, \tag{17}$$

where R_i denotes one of the randomized methods presented in Table 1.
As can be seen from Table 1, six randomized methods are used in the RFA algorithm that also differ according to the interval of generated random values. The former returns the random values in an interval [0, 1], like Uniform distribution, and both chaotic maps, whilst the latter within an interval $(-\infty, +\infty)$, like Gaussian, Lévy flights, and RSiTFC. When the random value is generated within the interval [0, 1], this value is extended to the interval $[-1, 1]$ using a formula $r_i = 2(r_i - 0.5)$. However, the generated solution value s_{ij} is verified to lie within the valid interval $s_{ij} \in [lb_j, ub_j]$, in both cases. Interestingly, some implementations of random generators can be found in Standard C-library (as a standard random generator for generating uniformly distributed random values), another in the GNU Scientific Library [23] (as random generators for generating the Gaussian and Lévy flights distributed random values), and the rest were developed from scratch (as chaotic maps and RSiTFC). According to the used randomized method, six different variants of the RFA algorithm are developed (as UFA, NFA, LFA, CFA1, CFA2, and FFA).

Table 1 Variants of the RFA algorithm

Randomization method	Random generator	Implementation	RFA variant
Uniform distributed	$U_i(0, 1)$	Standard C-library	UFA
Gaussian distributed	$N_i(-\infty, +\infty)$	GSL-library	NFA
Lévy flights	$L_i(-\infty, +\infty)$	GSL-library	LFA
Kent chaotic map	$C_i^K(0, 1)$	From scratch	CFA1
Logistic chaotic map	$C_i^L(0, 1)$	From scratch	CFA2
RSiTFC	$F_i(-\infty, +\infty)$	From scratch	FFA

4 Experiments and Results

An aim of our experimental work was to show that the developed randomized methods has a substantial (if not significant) impact on the results of the RFA algorithm. In line with this, six variants of the RFA algorithm (UFA, NGA, LFA, CFA1, CFA2, and FFA) were compared on a suite of ten well-known functions taken from publications. In the remainder of this chapter, the test suite is presented, then an experiment setup is defined that is followed by a description of a PC configuration, on which the experiments were performed. Finally, the results of the experiments are presented, in detail.

4.1 Test Suite

The test suite consisted of ten functions which were selected from two references. The first five functions were taken from Karaboga's paper [7], in which the ABC algorithm was introduced, while the last five were from the paper of Yang [28] that proposed a set of optimization functions suitable for testing the newly-developed algorithms.

The functions within the test suite can be divided into *unimodal* and *multimodal*. The multimodal functions have two or more local optima. The function is *separable*, when the set of variables can be rewritten as a sum of the function of just one variable. The separable and multimodal functions are more difficult to solve. The more complex functions are those that have an exponential number of local optima randomly distributed within the search space. The definitions and characteristics of functions constituting the test suite, can be summarized as follows:

- Griewangk's function:

$$f_1(\mathbf{s}) = -\prod_{i=1}^{D} \cos\left(\frac{s_i}{\sqrt{i}}\right) + \sum_{i=1}^{D} \frac{s_i^2}{4000} + 1, \tag{18}$$

  where $s_i \in [-600, 600]$. The function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$. It is highly multimodal, when the number of variables is higher than 30.
- Rastrigin's function:

$$f_2(\mathbf{s}) = D * 10 + \sum_{i=1}^{D} (s_i^2 - 10\cos(2\pi s_i)), \tag{19}$$

  where $s_i \in [-15, 15]$. The function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$ and is also highly multimodal.
- Rosenbrock's function:

$$f_3(\mathbf{s}) = \sum_{i=1}^{D-1} 100\,(s_{i+1} - s_i^2)^2 + (s_i - 1)^2, \tag{20}$$

where $s_i \in [-15, 15]$ and whose global minimum $f^* = 0$ is at $\mathbf{s}^* = (1, 1, \ldots, 1)$. This function, also known as the 'banana function' has several local optima. Gradient-based algorithms are especially difficult to converge to the global optima by optimizing this function.

- Ackley's function:

$$f_4(\mathbf{s}) = \sum_{i=1}^{D-1} (20 + e - 20e^{-0.2\sqrt{0.5(s_{i+1}^2 + s_i^2)}} - e^{0.5(\cos(2\pi s_{i+1}) + \cos(2\pi s_i))}), \tag{21}$$

where $s_i \in [-32.768, 32.768]$. The function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$ and it is highly multimodal.

- Schwefel's function:

$$f_5(\mathbf{s}) = 418.9829 * D - \sum_{i=1}^{D} s_i \sin(\sqrt{|s_i|}), \tag{22}$$

where $s_i \in [-500, 500]$. The Schwefel's function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (1, 1, \ldots, 1)$ and is highly multimodal.

- De Jong's sphere function:

$$f_6(\mathbf{s}) = \sum_{i=1}^{D} s_i^2, \tag{23}$$

where $s_i \in [-600, 600]$ and whose global minimum $f^* = 0$ is at $\mathbf{s}^* = (0, 0, \ldots, 0)$. The function is unimodal and convex.

- Easom's function:

$$f_7(\mathbf{s}) = -(-1)^D \left(\prod_{i=1}^{D} \cos^2(s_i) \right) \exp\left[-\sum_{i=1}^{D} (s_i - \pi)^2 \right], \tag{24}$$

where $s_i = [-2\pi, 2\pi]$. The function has several local minimum and the global minimum $f^* = -1$ at $\mathbf{s}^* = (\pi, \pi, \ldots, \pi)$.

- Michalewicz's function:

$$f_8(\mathbf{s}) = -\sum_{i=1}^{D} \sin(s_i) \left[\sin\left(\frac{i s_i^2}{\pi} \right) \right]^{2 \cdot 10}, \tag{25}$$

where $s_i = [0, \pi]$. The function has the global minimum $f^* = -1.8013$ at $\mathbf{s}^* = (2.20319, 1.57049)$ within two-dimensional parameter space. In general, it has several local optima.

- Xin-She Yang's function:

$$f_9(\mathbf{s}) = \left(\sum_{i=1}^{D} |s_i|\right) \exp\left[-\sum_{i=1}^{D} \sin(s_i^2)\right], \tag{26}$$

where $s_i = [-2\pi, 2\pi]$. The function is not smooth because it has several local optima and the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$.

- Zakharov's function:

$$f_{10}(\mathbf{s}) = \sum_{i=1}^{D} s_i^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i s_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i s_i\right)^4, \tag{27}$$

where $s_i = [-5, 10]$. The function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$ with no local optima.

The lower and upper bounds of the design variables determine intervals that limit the size of the search space. The wider this interval, the wider the search space. Note that the intervals were selected so that the search space was wider than those proposed in the standard literature. Another difficulty was represented by the dimensions of the functions. Typically, the higher the dimensional function, the more difficult to optimize. Note that functions with dimensions $D = 10$, $D = 30$ and $D = 50$ were employed in our experiments.

4.2 Experimental Setup

The characteristics of RFA algorithms are illustrated in Table 2, from which it can be seen that the specific FA parameters were set according to the propositions in [9], i.e., $\alpha = 0.1$, $\beta = 0.2$, and $\gamma = 0.9$. The population size was set as $NP = 100$, because extensive experiments have shown that this value represents the good balance between exploration and exploitation within the FA search process.

The number of fitness function evaluations depends on a dimension of problem D and was limited to $MAX\_FEs = 1,000 \cdot D$. However, the number of generations used as termination condition in RFA can simply be expressed as $MAX\_GEN = MAX\_FES/NP$. Because all the algorithms have stochastic natures they were run 25 times. The results from these algorithms were accomplished according to five standard measures, as follows: the *Best*, the *Worst*, the *Mean*, the *StDev*, and the *Median* values.

Table 2 Characteristics of the RFA algorithms

Parameter	Designation	Setting
Randomized parameter	α	0.1
Attractiveness	β	0.2
Light absorption	γ	0.9
Population size	NP	100
Maximum number of evaluations	MAX_FEs	$1,000 \cdot D$
Maximum number of generations	MAX_GEN	MAX_FEs/NP
Maximum number of runs	MAX_RUN	25

4.3 PC Configuration

All runs were made on a HP Compaq using the following configurations:

1. Processor—Intel Core i7-2600 3.4 (3.8) GHz
2. RAM—4GB DDR3
3. Operating system—Linux Mint 12

All versions of the tested algorithms were implemented within the Eclipse Indigo CDT framework.

4.4 Results

The following experiments were conducted, in order to show how the various randomized methods influenced the results of the RFA algorithms:

- analyzing the characteristics of the various randomization methods,
- verifying the impact of these randomizing methods on the results of the RFA algorithms,
- comparing the results of the RFA algorithms with other well-known metaheuristics, like BA, DE, and ABC.

The results of the mentioned experiments are observed in the remainder of this chapter.

4.4.1 Characteristics of the Various Randomized Methods

In this experiment, characteristics of the randomized methods, such as:

- Uniform distributed,
- Gaussian distributed,
- Lévy flights,

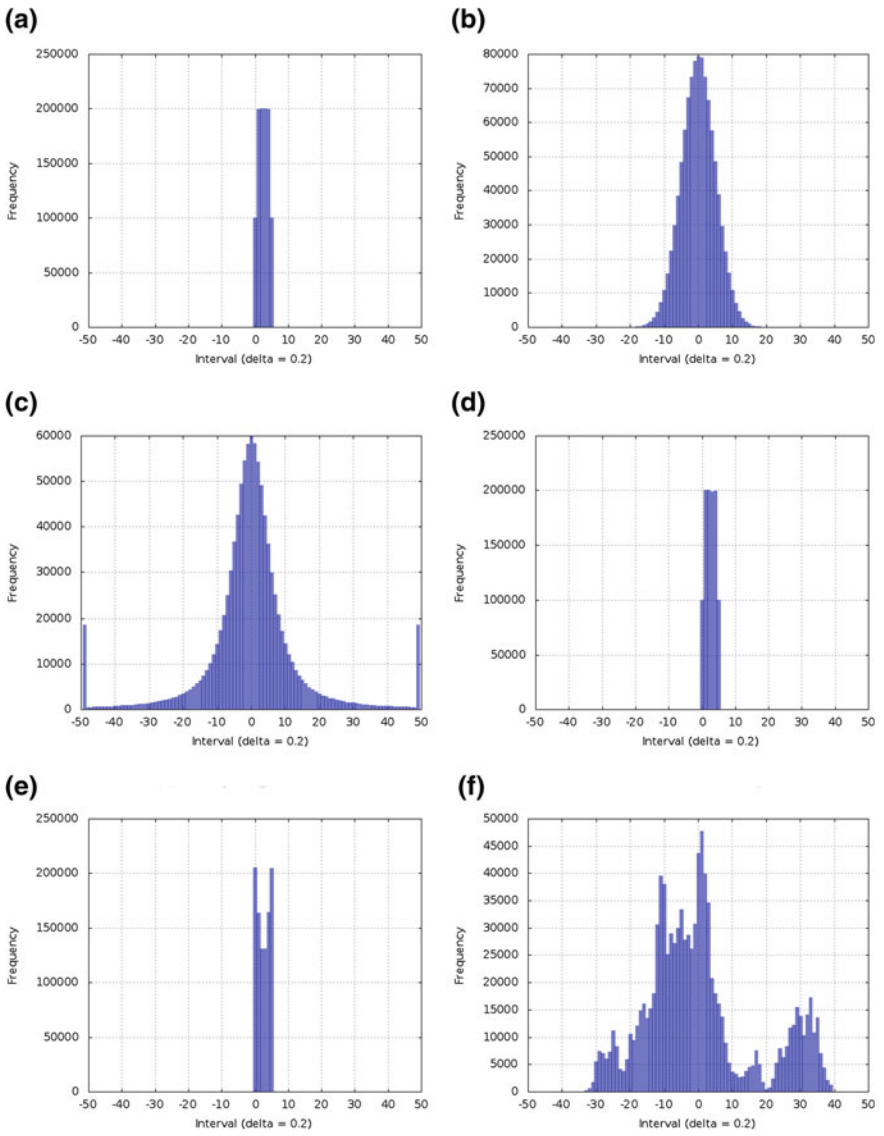

Fig. 1 Results of the various randomized methods

- Kent chaotic maps,
- Logistic chaotic map, and
- Random sampling in turbulent fractal cloud

are taken into account. The results are illustrated in Fig. 1 that is divided into six diagrams. Each of these presents a specific randomized method.

The results diagrams were obtained as follows. A million random numbers were generated for each method. Then, these were represented in a histogram plot that is a natural graphical representation of the distribution of random values. Each histogram consists of intervals coated on the x-axis denoting the value of the statistic variable, and their frequencies coated on the y-axis. Indeed, the range of real values $[-9.9, 9.9]$ is divided into 101 intervals each of width 0.2. The interval zero comprises the range $[-0.1, 0.1]$, whilst intervals -50 and 50 capture values < -9.9 and > 9.9, respectively. However, the total sum of the frequencies is one million.

The following characteristics can be summarized from the presented plots:

1. At first glance, a Kent chaotic map is similar to the uniform distribution, but the frequencies of the intervals slightly differ between each other by the former. On the other hand, Logistic chaotic map is inversion of both previously mentioned, because the border intervals 0 and 5 exhibit higher frequencies than the inner.
2. Gaussian distribution is more compact than Lévy flights, because the latter enables the generating the random numbers that are outside the intervals -50 and 50. This phenomenon can be seen in Fig. 1c as peeks in the first and last intervals.
3. The RSiTFC plot exhibits more peaks. This behavior might be useful by the optimization of multi-modal problems.

In summary, three randomized methods generates random numbers into interval [0, 1], whilst the other three into an interval that is much wider than mentioned. This finding complies with Table 1.

4.4.2 Impact of Randomized Methods on the Results of the RFA Algorithms

In this experiment, an impact of various randomized methods on the results of the RFA algorithms was verified. Therefore, the six variants of the RFA algorithms, i.e., UFA, NFA, LFA, CFA1, CFA2, and FFA were applied to the test suite as defined in Sect. 4.1. Indeed, the experimental setup was employed as presented in Sect. 4.2. Although the functions with dimensions $D = 10$, $D = 30$, and $D = 50$ were optimized, only those results optimizing the functions with dimension $D = 30$ are presented in Table 3, because of the limitation of the chapter's length. The best results are bold in this table.

As can be seen from Table 3, the FFA variant of RFA achieved the best results by optimizing functions f_1, f_2, f_4, f_6, and f_7, the LFA by functions f_5, f_9, and f_{10}, whilst the UFA outperformed the other algorithms by optimizing the function f_3 and the CFA1 by f_8. Indeed, the functions f_1, f_2, and f_4 are highly multi-modal.

The Friedman test was conducted in order to estimate the quality of the results. The Friedman test [29, 30] compares the average ranks of the algorithms. A null-hypothesis states that two algorithms are equivalent and, therefore, their ranks should be equal. If the null-hypothesis is rejected, i.e., the performance of the algorithms is statistically different, the Bonferroni-Dunn test [31] is performed that calculates the critical difference between the average ranks of those two algorithms. When the statistical difference is higher than the critical difference, the algorithms

Table 3 Detailed results of the RFA algorithms (D = 30)

Function	Measure	UFA	NFA	LFA	CFA1	CFA2	FFA
f_1	Mean	6.65E−001	1.08E+000	1.05E+000	6.55E−001	8.89E−001	**3.09E−001**
	Stdev	6.40E−001	1.07E+000	1.05E+000	6.96E−001	8.48E−001	3.32E−001
f_2	Mean	2.44E+002	3.51E+002	4.35E+002	2.30E+002	3.01E+002	**2.18E+002**
	Stdev	2.35E+002	3.53E+002	4.42E+002	2.31E+002	2.94E+002	2.16E+002
f_3	Mean	**1.12E+002**	2.25E+004	2.00E+005	3.95E+002	3.14E+002	5.89E+002
	Stdev	1.01E+002	1.72E+004	2.39E+005	3.71E+002	2.35E+002	5.88E+002
f_4	Mean	2.11E+001	2.05E+001	2.02E+001	2.10E+001	2.10E+001	**1.45E+001**
	Stdev	2.11E+001	2.05E+001	2.02E+001	2.11E+001	2.10E+001	1.43E+001
f_5	Mean	6.78E+003	1.46E+003	**2.55E+002**	7.17E+003	5.44E+003	8.64E+003
	Stdev	6.75E+003	−1.39E+003	−3.37E+002	7.12E+003	5.41E+003	8.45E+003
f_6	Mean	5.19E+000	2.64E+002	9.92E−001	3.67E+000	7.54E+000	**3.61E−001**
	Stdev	5.14E+000	2.63E+002	6.69E−001	3.09E+000	7.20E+000	3.61E−001
f_7	Mean	−3.81E−030	−6.58E−069	−8.56E−014	−1.81E−036	−4.07E−039	**−1.29E−078**
	Stdev	−3.73E−030	−4.94E−069	−9.68E−018	−1.38E−036	−2.59E−039	−4.00E−084
f_8	Mean	−5.15E+000	−1.20E+001	−7.60E+000	**−7.73E+000**	−4.82E+000	−2.60E+000
	Stdev	−5.35E+000	−1.23E+001	−8.24E+000	−7.94E+000	−4.76E+000	−2.63E+000
f_9	Mean	1.70E−004	6.72E−005	**9.11E−006**	1.15E−004	9.21E−005	4.69E−003
	Stdev	4.72E−005	3.13E−005	2.37E−010	5.59E−005	7.81E−005	4.09E−003
f_{10}	Mean	1.32E+004	2.27E+002	**1.12E+002**	6.09E+002	1.41E+003	3.12E+005
	Stdev	1.32E+004	2.28E+002	8.40E+001	6.09E+002	1.41E+003	3.12E+005

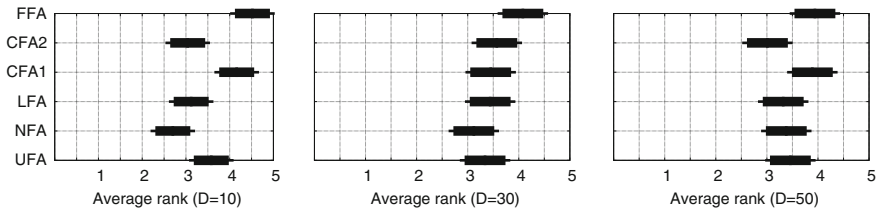

Fig. 2 Results of the Friedman non-parametric test on different variants of RFA algorithms

are significantly different. The equation for the calculation of critical difference can be found in [31].

Friedman tests were performed using a significance level 0.05. The results of the Friedman non-parametric test are presented in Fig. 2 being divided into three diagrams that show the ranks and confidence intervals (critical differences) for the algorithms under consideration. The diagrams are organized according to the dimensions of functions. Two algorithms are significantly different if their intervals in Fig. 2 do not overlap.

The first diagram in Fig. 1 shows that the FFA and KFA variants of RFA significantly outperform the results of all other variants of RFA (i.e., NFA, LFA, CFA2), except UFA, according to dimension $D = 10$. However, the results became insignificant when these were compared in regard to the dimensions $D = 30$ and $D = 50$. In fact, the results of FFA and KFA variants of RFA still remained substantially better, but this difference was not significant.

In summary, we can conclude that the selection of the randomized method has a great impact on the results of RFA. Moreover, in some cases the selection of the more appropriate randomized method can even significantly improve the results of the original FA (Gaussian NFA). Indeed, the best results were observed by the RSiTFC and Kent chaotic map.

4.4.3 Comparative Study

In this experiment, the original FA algorithm (NFA) was compared with other well-known algorithms as bat algorithm (BA), differential evolution (DE), and artificial bees colony (ABC) as well as the FFA algorithm that was exhibited as the most promising variant of the developed RFA algorithms.

The specific BA parameters were set as follows: the loudness $A_0 = 0.5$, the pulse rate $r_0 = 0.5$, minimum frequency $Q_{max} = 0.0$, and maximum frequency $Q_{max} = 0.1$. The DE parameters were configured as follows: the amplification factor of the difference vector $F = 0.9$, and the crossover control parameter $CR = 0.5$. The percentage of onlooker bees for the ABC algorithm was 50 % of the colony,

Table 4 Comparing algorithms (D = 10)

Function	Measure	NFA	FFA	BA	DE	ABC
f_1	Mean	7.19E−001	**6.03E−002**	8.99E+000	2.62E+000	6.26E−001
	Stdev	7.00E−001	6.25E−002	6.44E+000	4.32E−001	1.99E−001
f_2	Mean	6.59E+001	4.29E+001	1.46E+002	9.01E+001	**1.24E+001**
	Stdev	6.99E+001	4.46E+001	7.44E+001	9.10E+000	4.36E+000
f_3	Mean	5.51E+005	**4.08E+001**	8.91E+004	1.53E+004	2.05E+002
	Stdev	9.62E+004	3.36E+001	1.45E+005	8.94E+003	2.62E+002
f_4	Mean	2.02E+001	9.73E+000	1.02E+001	8.55E+000	**4.08E+000**
	Stdev	2.02E+001	9.72E+000	3.23E+000	9.78E−001	9.03E−001
f_5	Mean	1.37E+003	4.00E+003	2.19E+003	1.08E+003	**5.99E+002**
	Stdev	1.38E+004	4.01E+003	2.58E+002	1.26E+002	1.25E+002
f_6	Mean	8.02E+001	**5.88E−002**	2.38E+004	6.90E+003	1.23E+001
	Stdev	8.03E+001	5.66E−002	1.75E+004	2.05E+003	1.68E+001
f_7	Mean	−3.89E−018	**−5.15E−034**	−6.46E−002	−3.96E−001	−1.13E−002
	Stdev	−3.88E−018	−4.11E−034	2.18E−001	1.40E−001	5.67E−002
f_8	Mean	−4.79E+000	**−6.37E−001**	−5.99E+000	−6.69E+000	−8.60E+000
	Stdev	−5.63E+000	−6.38E−001	1.04E+000	3.24E−001	2.89E−001
f_9	Mean	3.49E−002	1.32E−001	1.39E−003	1.92E−003	**6.10E−004**
	Stdev	2.15E−002	1.30E−001	6.95E−004	1.31E−004	5.96E−005
f_{10}	Mean	6.98E+001	4.99E+003	**2.05E+001**	2.93E+001	2.38E+001
	Stdev	6.96E+001	4.99E+003	2.03E+001	8.29E+000	7.80E+000

the employed bees represented another 50 % of the colony, whilst one scout bee was generated in each generation (i.e., *limits* = 100, when the population size is $NP = 100$).

The results of comparing the mentioned algorithms by optimizing the functions with dimension $D = 10$, are presented in Table 4. Again, only one instance of data is illustrated in the table, although the experiments were conducted on all three observed dimensions. The best results of the algorithms are written in bold.

As can be seen from Table 4, the FFA algorithm outperformed the results of the other algorithms when solving the functions f_1, f_3, f_6, f_7, and f_8. Again the majority of these functions are highly multi-modal. The ABC algorithm was the best by solving the functions f_2, f_4, f_5, and f_9, whilst the BA algorithm excellently solved the function f_{10}.

Also here, the Friedman tests using the significance level 0.05 were conducted according to all the observed dimensions of the functions. The results of these tests are presented in Fig. 3, which is divided into three diagrams.

The first diagram illustrates the results of the Friedman test that observes the results obtained by optimizing the functions with dimensions $D = 10$. It can be shown from this diagram that ABC and FFA outperformed the results of all other algorithms in test (i.e., NFA, BA, and DE) significantly. Furthermore, the ABC outperformed the results of the same algorithms when also optimizing the functions with dimension $D = 30$, whilst the FFA algorithm was substantially better than those on the same

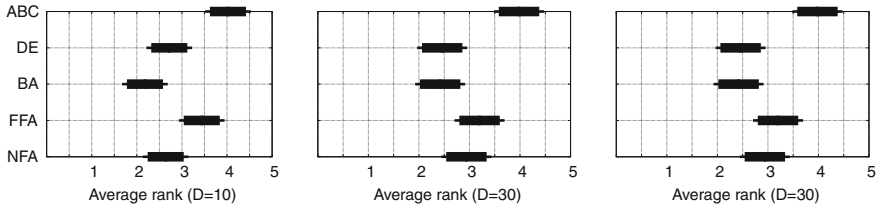

Fig. 3 Results of the Friedman non-parametric test on suite of test algorithms

instances. Finally, on the functions with dimension $D = 50$, the ABC achieved significantly better results than BA and DE. Here, both firefly algorithms exhibited good results.

In summary, the FFA variant of RFA outperforms the results of the original FA algorithm significantly by optimizing the test functions with dimension $D = 10$, and substantially by optimizing the test functions with dimensions $D = 30$ and $D = 50$. Indeed, the ABC algorithm outperforms significantly all the other experiments in tests except FFA. In general, the results of experiments have been shown that the Gaussian distribution method is appropriately selected by the original FA algorithm.

5 Conclusion

In this chapter, an extensive comparison of various probability distributions is performed that can be used to randomize the firefly algorithm, e.g., uniform, Gaussian, Lévi flights, chaos maps and the random sampling in turbulent fractal cloud. In line with this, various firefly algorithms with various randomized methods were developed and extensive experiments were conducted on well-known suite of functions.

The goal of the experiments were threefold. Firstly, the mentioned randomized methods were analyzed. Secondly, an impact of randomized methods on the results of the RFA algorithms were verified. Finally, the results of the original FA algorithm and FFA variant of RFA were compared with the other well-known algorithms like ABC, BA, and DE.

In summary, the selection of an appropriate randomized method has a great impact on the results of RFA. Moreover, this selection depends on the nature of the problem to be solved. On the other hand, selecting the appropriate randomized method can improve the results of the original FA significantly.

In the future, further experiments should be performed with the random sampling in turbulent fractal cloud that exhibits the excellent results especially by optimizing the multi-modal functions.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co, New York (1979)
2. Blum, C., Li, X.: Swarm intelligence in optimization. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence: Introduction and Applications, pp. 43–86. Springer, Heidelberg (2008)
3. Beekman, M., Sword, G.A., Simpson, S.J.: Biological foundations of swarm intelligence. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence: Introduction and Applications, pp. 3–41. Springer, Berlin (2008)
4. Beni, G., Wang, J.: Swarm intelligence in cellular robotic systems. Proceedings of NATO Advanced Workshop on Robots and Biological Systems, pp. 26–30. Tuscany, Italy (1989)
5. Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 11–32. McGraw Hill, London (1999)
6. Kennedy, J., Eberhart, R.C.: The particle swarm optimization: social adaptation in information processing. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 379–387. McGraw Hill, London (1999)
7. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**, 459–471 (2007)
8. Fister, I., Fister, I. Jr., Brest, J., Žumer, V.: Memetic artificial bee colony algorithm for large-scale global optimization. In: IEEE Congress on Evolutionary Computation, Brisbane, Australia, pp. 3038–3045. IEEE Publications (2012)
9. Yang, X.-S.: Firefly algorithm. In: Yang, X.-S. (ed.) Nature-Inspired Metaheuristic Algorithms, pp. 79–90. Wiley Online, Library (2008)
10. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Stochastic Algorithms: Foundations and Applications, pp. 169–178. Springer, Berlin (2009)
11. Fister, I. Jr., Yang, X.-S., Fister, I., Brest, J.: Memetic firefly algorithm for combinatorial optimization. In: Filipič, B., Šilc, J. (eds.) Bioinspired optimization methods and their applications : proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications—BIOMA 2012, pp. 75–86. Jožef Stefan Institute (2012)
12. Gandomi, A.H., Yang, X.-S., Talatahari, S., Alavi, A.H.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013)
13. Fister, I., Yang, X.-S., Brest, J., Fister Jr, I.: Memetic self-adaptive firefly algorithm. In: Yang, X.-S., Xiao, R.Z.C., Gandomi, A.H., Karamanoglu, M. (eds.) Swarm Intelligence and Bio-Inspired Computation: Theory and Applications, pp. 73–102. Elsevier, Amsterdam (2013)
14. Fister, I., Fister Jr., I., Yang, X.-S., Brest, J.: A comprehensive review of firefly algorithms. Swarm and Evolutionary Computation (2013). Available via ScienceDirect. http://www.sciencedirect.com/science/article/pii/S2210650213000461. Cited 03 Jul 2013
15. Yang, X.-S., Deb, S.: Cuckoo search via Levy flights. In: World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), pp. 210–214. IEEE Publications (2009)
16. Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: Cruz, C., Gonzlez, J.R., Krasnogor, N., Pelta, D.A., Terrazas, G. (eds.) Nature Inspired Cooperative Strategies for Optimization (NISCO 2010), vol. 284, pp. 65–74. Springer, Berlin (2010)
17. Fister Jr, I., Fister, D., Yang, X.-S.: A Hybrid bat algorithm. Electrotech. Rev. **80**, 1–7 (2013)
18. Hoos, H.H., Stützle, T.: Stochastic local search: Foundations and applications. Morgan Kaufmann, San Francisco (2004)
19. Feldman, D.P.: Chaos and Fractals: An Elementary Introduction. Oxford University Press, Oxford (2012)
20. Črepinšek, M., Mernik, M., Liu, S.H.: Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees. Int. J. Innovative Comput. Appl. **3**, 11–19 (2011)
21. Hertz, A., Taillard, E., de Werra, D.: Tabu search. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 121–136. Princeton University Press, New Jersey (2003)
22. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2003)

23. Galassi, D., et al.: GNU Scientific Library: Reference Manual, Edn. 1.15. Network Theory Ltd, Bristol (2011)
24. Jamil, M.: Zepernick: Lévy flights and global optimization. In: Yang, X.-S., Xiao, R.Z.C., Gandomi, A.H., Karamanoglu, M. (eds.) Swarm Intelligence and Bio-Inspired Computation: Theory and Applications, pp. 49–72. Elsevier, Amsterdam (2013)
25. Zhou, Q., Li, L., Chen, Z.-Q., Zhao, J.-X.: Implementation of LT codes based on chaos. Chin. Phys. B **17**(10), 3609–3615 (2008)
26. Elmegreen, B.G.: The initial stellar mass function from random sampling in a turbulent fractal cloud. Astrophys. J. **486**, 944–954 (1997)
27. Long, S.M., Lewis, S., Jean-Louis, L., Ramos, G., Richmond, J., Jakob, E.M.: Firefly flashing and jumping spider predation. Anim. Behav. **83**, 81–86 (2012)
28. Yang, X.-S.: Appendix A: Test Problems in Optimization. In: Yang, X.-S. (ed.) Engineering Optimization, pp. 261–266. John Wiley and Sons, Inc., New York (2010)
29. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**, 675–701 (1937)
30. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. An. Math. Stat. **11**, 86–92 (1940)
31. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)

http://www.springer.com/978-3-319-02140-9

Cuckoo Search and Firefly Algorithm
Theory and Applications
Yang, X.-S. (Ed.)
2014, XI, 360 p. 100 illus., 5 illus. in color., Hardcover
ISBN: 978-3-319-02140-9