Analysis of randomisation methods in swarm intelligence

Iztok Fister Jr.*

University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Smetanova 17, 2000 Maribor, Slovenia
Email: iztok.fister1@um.si
*Corresponding author

Xin-She Yang

Middlesex University,
School of Science and Technology,
London NW4 4BT, UK
Email: x.yang@mdx.ac.uk

Janez Brest, Dušan Fister and Iztok Fister

University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Smetanova 17, 2000 Maribor, Slovenia
Email: janez.brest@um.si
Email: dusan.fister@um.si
Email: iztok.fister@um.si

Abstract: Nowadays, many stochastic metaheuristics have been developed to solve various optimisation problems. The primary characteristics of these heuristics often involve the use of randomness in their search process. Essentially, randomness is useful when determining the next point in the search space and therefore has a crucial impact when exploring new solutions. In this paper, an extensive comparison is made between various probability distributions that can be used for randomising the swarm intelligence algorithms, e.g., uniform, Gaussian, Lévy flights, chaotic maps, and the random sampling in turbulent fractal cloud. These randomisation methods were incorporated into the bat algorithm that is one of the newest member of this domain. In line with this, various variants of bat algorithms randomised with different randomisation methods have been developed and extensive experiments were conducted on a well-known set of 24 BBOB benchmark functions. In addition, the results of randomised bat algorithms were compared with the results of the other well-known algorithms, including the firefly algorithm, differential evolution and artificial bee colony algorithms. The results of these experiments show that the efficiencies of the distributions used during the tests depend on the problem to be solved as well as on the algorithm used.

Keywords: bat algorithm; chaos; fractal; Lévy flights; swarm intelligence.

Reference to this paper should be made as follows: Fister Jr., I., Yang, X-S., Brest, J., Fister, D. and Fister, I. (2015) 'Analysis of randomisation methods in swarm intelligence', *Int. J. Bio-Inspired Computation*, Vol. 7, No. 1, pp.36–49.

Biographical notes: Iztok Fister Jr. received his BSc and MSc from Computer Science in 2011 and 2013, respectively. Currently, he is working towards his PhD. His research activities encompasses swarm intelligence, pervasive computing and programming languages.

Xin-She Yang received his DPhil in Applied Mathematics from University of Oxford. Currently, he is a Reader in Modelling and Simulation at Middlesex University, UK, an Adjunct Professor at Reykjavik University, Iceland, and a Distuiguished Guest Professor at Xi'an Polytechnic University, China. He is also the IEEE CIS Chair for the Task Force on Business Intelligene and Knowledge Management, and the Editor-in-Chief of *International Journal of Mathematical Modelling and Numerical Optimisation* (*IJMMNO*).

Janez Brest received his PhD in Computer Science from the University of Maribor, Maribor, Slovenia in 2000. He has been with the Laboratory for Computer Architecture and Programming Languages, University of Maribor since 1993. He is currently a Full Professor and Head of the Laboratory for Computer Architecture and Programming Languages. His research interests include evolutionary computing, artificial intelligence, and optimisation. His fields of expertise embrace programming languages, and parallel and distributed computing research.

Dušan Fister is a student of third-year of Mechatronics at the University of Maribor. His research activities encompasses GPS solutions, swarm intelligence and operating systems.

Iztok Fister graduated in Computer Science from the University of Ljubljana in 1983. In 2007, he received his PhD from the Faculty of Electrical Engineering and Computer Science, University of Maribor. Since 2010, he has worked as a Teaching Assistant in the Computer Architecture and Languages Laboratory at the same faculty. His research interests include computer architectures, programming languages, operational researches, artificial intelligence, and evolutionary algorithms. He is a member of IEEE.

This paper is extension of some parts from MSc thesis: 'A comprehensive review of bat algorithms and their hybridization' presented at University of Maribor, Slovenia, 2013.

1 Introduction

Decision making and automatic problem-solving play a crucial role in the areas such as computer science, mathematics, engineering and other real-world problems. The most complex problems, so named NP-hard problems (Garey and Johnson, 1979), cannot be solved using the exact methods that search for the best solutions by enumerating all the possible solutions. In order to successfully cope with such problems, some new methods have been developed that solve the problems approximately, but still accurately enough to be used in the real-world applications. In line with this, researchers look for an inspirations for the designs of these algorithms from the nature, e.g., collective behavior of social living insects (e.g., ants, bees, termites, etc.) and social behavior of some animals societies (e.g., birds, fish, dolphins, etc.) (Blum and Li, 2008; Beekman et al., 2008). Consequently, a new area of artificial intelligence called swarm intelligence (SI) has been emerged. This term was probably first used in Beni and Wang (1989). Nowadays, there are many algorithms which live under the swarm intelligence umbrella, like ant colony optimisation (Dorigo and Di Caro, 1999), particle swarm optimisation (Kennedy and Eberhart, 1999; Fister et al., 2014b), and artificial bees colony (ABC) (Karaboga and Basturk, 2007; Fister et al., 2012a). Recently, the more promising SI-based optimisation techniques include the firefly algorithm (FA) (Yang, 2008; Fister et al., 2012b; Gandomi et al., 2013; Fister et al., 2013a,e), the cuckoo search (CS) (Yang and Deb; Fister et al., 2014d, 2013d), and the bat algorithm (BA) (Yang, 2010a; Fister et al., 2013c).

SI-based algorithms incorporate some randomness in some constructive way (Hoos and Stützle, 2004) in order to search across the search space. Therefore, each run of these stochastic algorithms may obtain different results. On the other hand, algorithms that ensure the same results in each run are called also deterministic (Feldman, 2012). Interestingly, although chaos systems are based on the deterministic mathematical equations, such systems can behave stochastically and therefore unpredictable (Feldman, 2012). As a result, chaos-based randomisation methods have increasingly being used recently, especially in SI-based algorithms. Randomness in SI-based algorithms is very important, due to its increase in the exploration ability in the search process (Črepinšek et al., 2011; Hertz et al., 2003). In other words, it helps to discover new points in the search space by moving particles or agents towards different regions in the search space. So far, many different randomisation methods have been developed, as follows:

- uniform distribution
- Gaussian distribution
- Lévy flights
- chaotic maps
- many more.

This paper proposes a modified bat algorithm randomised by different randomisation methods, including the uniform, Lévy flights, chaotic maps, Gaussian and the random sampling in turbulent fractal clouds. The proposed randomised bat algorithm (RBA) incorporates various randomisation methods. Most of these randomisation methods are well-known and widely used, especially the uniform and Gaussian. However, the random sampling in a turbulent fractal cloud is a method from astronomy and was rarely used for optimisation. In Fister et al. (2014a), this method was first applied into the FA algorithm, where it outperformed the same algorithm randomised with other randomisation methods, especially by optimising multi-modal functions.

The main purpose of the experimental work was to show how different randomisation methods influence the results of the proposed RBA algorithm. In line with this, a set of function optimisation problems have been taken as a test-bed for measuring the performance of this algorithm. In order to make experiments as valuable as possible, the BBOB benchmark function suite (Hansen et al., 2013) was employed. Beside comparing the various RBA algorithms, the comparison study was conducted, in which the results of the best three variants of the RBA algorithm as found in the last numerical experiments were compared with the results of the other well-known algorithms, like FA, differential evolution (DE) (Brest et al., 2006; Das and Suganthan, 2011) and ABC.

The structure of the rest of the paper is as follows: Section 2 describes the background information of different randomisation methods. In Section 3, the bat algorithm is

presented. Moreover, a brief review and applications using bat algorithms are presented. Experiments and results are presented in Section 4. Conclusions are drawn and the directions for further development are outlined in the last Section 5.

2 Background information and probability distributions

This section describes the background information about drawing random numbers from a given random variable with a probability distribution, and the probability distributions are as follows (Fister, 2013):

- uniform
- normal or Gaussian
- Lévy flights
- chaotic maps
- random sampling in turbulent fractal cloud.

Continuous random variable distributions (Galassi et al., 2011) 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 for a variable 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 for 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, more details about randomisation methods will be presented.

2.1 Uniform distribution

A uniform continuous distribution has a 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 a uniformly distributed random variable is within the 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, a uniform distribution is obtained by using a pseudo-random number generator (Galassi et al., 2011). It is worth pointing out that on most computer platforms, the random values generated usually vary in the interval $[0, 2^{32} - 1]$. In order to obtain the random generated value within the interval $[0, 1]$, the following mapping is used:

$$r = ((double)rand()/ \\ ((double)(\text{RAND\_MAX}) + 1.0)), \tag{5}$$

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

A normal or Gaussian distribution is defined by the following density function:

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

The distribution depends on parameters $\mu \in \mathbb{R}$ and $\sigma > 0$. This distribution is denoted as $\mathrm{N}(\mu, \sigma)$. The standardised normal distribution is obtained, when the distribution has a zero mean with a standard deviation of one; i.e., $\mathrm{N}(0, 1)$. In this case, the density function can be 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 (Eiben and Smith, 2003). That is, most of the modifications made on the solutions will be small, whilst there is a non-zero (but small) probability of generating very large modifications, because the tail of distribution never reaches zero (Eiben and Smith, 2003).

2.3 Lévy flights

In reality, resources are distributed non-uniformly in Nature. This means that the behaviour of a typical forager, aiming to find these resources as quickly as possible, does not obey a Gaussian distribution. In order to simulate foragers search strategies, Lévy flights is closer to their behaviour (Jamil and Zepernick, 2013). It belongs to a special class of α-stable distributions defined by a Fourier transform (Galassi et al., 2011):

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

The α-stable means that the sum of random variables has the same probability density distributions as that for each random 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, in the case of $\alpha = 1$, the density function reduces to the Cauchy distribution, whilst for $\alpha = 2$, it

becomes a Gaussian distribution with $\sigma = \sqrt{2c}$. For $\alpha < 1$, the tails of the distribution become extremely wide (Yang and Deb; Jamil and Zepernick, 2013). The more appropriate setting of this parameter for optimisation is therefore $\alpha \in (1.0, 2.0)$ (Yang and Deb). Essentially, the difference between non-Gaussian and Gaussian distributions is that the tail of the distribution of the former is wider than that of the latter. This means, the probability of generating very large modifications is much higher by Lévy flights than by a Gaussian distribution.

2.4 Chaotic maps

Chaos is a phenomenon encountered in science and mathematics where a deterministic system behaves unpredictably (Feldman, 2012). A well-know logistic map is

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

where $x_n \in [0, 1]$ and r is a parameter. A generated sequence of numbers by iterating a Logistic map (also orbit) with $r = 4$ exhibits *chaotic* behaviour. That is, it posses the following properties (Feldman, 2012):

- the dynamic rule of generating these numbers is deterministic
- the orbits are aperiodic (never repetitive)
- the orbits are bounded (between upper and lower limits, usually, in the interval $[0, 1]$)
- the sequence is very sensitive to the minute change in the initial condition.

Similar behaviour can also be observed in the Kent map (Zhou et al., 2008). The Kent map is one of the more studied chaotic maps and has been used to generate pseudo-random numbers in many applications such as secure encryption. The Kent map is defined as

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

where $0 < m < 1$. Hence, if $x(0) \in [0, 1]$, for all $n \ge 1$, $x(n) \in [0, 1]$. To be consistent with the propositions in Zhou et al. (2008), $m = 0.7$ will be used in our experiments.

2.5 Random sampling in a turbulent fractal cloud

Stars formation begin with random sampling of mass in a fractal cloud (Elmegreen, 1997). This random sampling is performed by the initial mass function (IMF) and represents a basis for a new sampling method, called the random sampling in a turbulent fractal cloud (RSiTFC).

The basic idea in this 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 a 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 (Elmegreen, 1997). This mentioned method is formally illustrated in Algorithm 1.

The algorithm RSiTFC consists of six parameters: the scaling factor L, the number of levels H, the number of sub-pieces for each piece N, the fractal cloud pieces x, the level h, and the piece number i. The scaling factor is determined by $S = L^{-h}$ 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's 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.

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 $1 + 2 + 4 + 8 + 16 = 31$ (Elmegreen, 1997).

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.
Output: $*x$ fractal cloud.

```
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.
```

3 The original bat algorithm

The bat algorithm (BA) was developed by Yang (2010a). The main inspiration for this algorithm was microbats and their echolocation. BA mimics the natural behaviour of echolocation and thus have created a powerful algorithm which could be applied to almost all areas of optimisation. The pseudo-code of the BA is illustrated in Algorithm 2.

Algorithm 2 Original bat algorithm

Input: Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, MAX_FE.
Output: The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.

```
1: init_bat();
2: eval = evaluate_the_new_population;
3: f_min = find_the_best_solution(x_best); {initialisation}
4: while termination_condition_not_met do
5:    for i = 1 to Np do
6:       y = generate_new_solution(x_i);
7:       if rand(0,1) > r_i then
8:          y = improve_the_best_solution(x_best)
9:       end if{ local search step }
10:      f_new = evaluate_the_new_solution(y);
11:      eval = eval + 1;
12:      if f_new ≤ f_i and N(0,1) < A_i then
13:         x_i = y; f_i = f_new;
14:      end if{ save the best solution conditionally }
15:      f_min=find_the_best_solution(x_best);
16:   end for
17: end while
```

The behaviour of microbats is mimicked by the algorithm updating action and relationship with the fitness function of the problem to be solved. The actions of the original BA algorithm presented in Algorithm 2 can be described as the following steps:

- *initialisation* (lines 1–3): initialising the algorithm parameters, generating the initial population, evaluating this population, and finally, determining the best solution $\mathbf{x}_{best}$ in the population
- *generate_the_new_solution* (line 6): moving the virtual bats in the search space according to mathematical rules of bat echolocation
- *local_search_step* (lines 7–9): improving the best solution using the random walk direct exploitation (RWDE) heuristic
- *evaluate_the_new_solution* (line 10): evaluating the new solution
- *save_the_best_solution_conditionaly* (lines 12–14): saving the new best solution under some probability via the variation of A_i similar to simulated annealing
- *find_the_best_solution* (line 15): finding the current best solution.

These components are denoted in the algorithm either as function names, when the function call is performed in one line, or a component name is designated as a comment between two curly brackets in the last line, when it comprises more lines. Initialisation of the bat population is performed randomly. Generating the new solutions is performed according to the following equations:

$$\begin{aligned} Q_i^{(t)} &= Q_{min} + (Q_{max} - Q_{min})U(0,1), \\ \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^t + (\mathbf{x}_i^t - \mathbf{best})Q_i^{(t)}, \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \end{aligned} \tag{11}$$

where $U(0,1)$ is a random number drawn from a uniform distribution. A RWDE heuristic implemented in the function *improve_the_best_solution* modifies the current best solution according to the equation:

$$\mathbf{x^{(t)}} = \mathbf{best} + \epsilon A_i^{(t)} N(0,1), \tag{12}$$

where $N(0,1)$ denotes the random number drawn from a Gaussian distribution with zero mean and a standard deviation of one. In addition, ϵ is a scaling factor and $A_i^{(t)}$ is the loudness. A local search is launched with the probability of pulse rate r_i. As already stated, the probability of accepting the new best solution in the component *save_the_best_solution_conditionaly* depends on loudness A_i. Actually, the original BA algorithm is controlled by two algorithm parameters: the pulse rate r_i and the loudness A_i. Typically, the rate of pulse emission r_i increases and the loudness A_i decreases when the population draws nearer to the local optimum. Both characteristics imitate natural bats, where the rate of pulse emission increases and the loudness decreases when a bat finds a prey. Mathematically, these characteristics are captured using the following equations:

$$A_i^{(t+1)} = \alpha A_i^{(t)}, \quad r_i^{(t)} = r_i^{(0)}[1 - \exp(-\gamma\epsilon)], \tag{13}$$

where α and γ are constants. Actually, the α parameter plays a similar role as the cooling factor in the simulated annealing algorithm that controls the convergence rate of this algorithm.

Though BA has some similarity to the PSO algorithm (Kennedy and Eberhart, 2001), however, the bat algorithm uses different search and diversity mechanisms towards improving the best solution. In other words, the exploitation is controlled by the parameter **r** and exploration by the parameter **A**. As a result, the BA algorithm tries to explicitly control the exploration and exploitation components within its search process.

3.1 Novel variants of the bat algorithm and current progress

Currently, the bat algorithm is very popular in the research community of optimisation and computational intelligence (Yang and He, 2013). Therefore, the bat algorithm has been used to solve both

- continuous (especially for function optimisation)
- discrete problems (e.g., flow shop scheduling, travelling salesman problem and many problems from mathematics, e.g., graph colouring).

According to the No Free Lunch (NFL) theorem (Wolpert and Macready, 1997), there are no algorithms which should cope well with all problem families, but some algorithms can deal with a specific class of problem very effectively. Therefore, there is still room for improvement for most algorithms. There are two main directions for improving the original algorithm:

- hybridisation
- adaptation.

Hybridisation is usually done by incorporating other algorithm or search component into an existing algorithm. One simple way is to hybridise with other metaheuristic algorithm. On the other hand, adaptation is usually done on algorithm-dependent control parameters. In other words, instead of tuning parameters and using trial-and-error approaches, an algorithm is able to adapt parameters during the run. In the recent years, there have some advancements of the bat algorithm in both domains. According to the adaptation, two interesting BA variants were proposed recently:

- self-adaptive bat algorithm (SABA) – self-adapting mechanisms were taken from self-adaptive differential evolution with acronym jDE
- hybrid self-adaptive bat algorithm (HSABA) – self-adaptive bat algorithm was additionally hybridised with different differential evolution strategies.

Furthermore, if we look into hybridisation more closely, there are much more BA variants. For example, He et al. (2013) proposed a new variant called simulated annealing Gaussian bat algorithm (SAGBA). Since the incorporation of simulated annealing heuristic and Gaussian perturbations into the bat algorithm, the result is very promising with improved search performance and convergence rate. Wang and Guo (2013) used some ingredients of harmony search and created a new hybrid bat algorithm with harmony search. On the other hand, Fister et al. (2013c) hybridised the bat algorithm with differential evolution. In addition, Fister et al. (2013f) linked the hybrid bat algorithm with random forests.

Readers are invited to refer to some other hybrid variants of that bat algorithm (Laamari and Kamel, 2014; Taha et al., 2013).

3.2 Bat algorithm in applications

Though there is a big gap between both theory and practice. We know most nature-inspired algorithm work in practice, but mathematical analysis lacks behind. Fortunately, there was a lot of progress made in incorporating algorithms such as the bat algorithm to solve real-world problems. The recent interesting applications using the bat algorithm can be summarised as follows:

- Multilevel image thresholding: An improved bat algorithm was developed by Alihodzic and Tuba (2014) and this variant was applied to multilevel image thresholding which is a well-known image processing method.
- Topology optimisation in microelectronic applications: Yang et al. (2012) used the bat algorithm for topology optimisation where the main task is to find the best geometric configurations, which leads to use the minimum amount of materials.
- The aircraft landing problem (ALP): ALP is one problem from the NP-hard family, and the main task here is to minimise the total cost of landing deviation from a predefined target time under the condition of safe landing (Xie et al., 2013). In Xie et al. (2013), the bat algorithm was used to deal with this problem effectively.
- Planning path for UCAV: Planning paths for uninhabited combat air vehicle is a very complicated problem with high dimensions. Some variants of the bat algorithm were used to solve this UCAV problem (Wang et al., 2012).
- Fuel management optimisation in reactor core: Kashi et al. (2014) use the bat algorithm for fuel management in reactor core.
- Planning sport training: Fister et al. (2014g) developed a novel solution for sport training plan generation for athletes based on the bat algorithm.

3.3 Variants of the randomised bat algorithm

Our proposed randomised BA (RBA) is based on the original BA by improving randomisation mechanisms in the BA. In place of the original equation (12), the modified equation is used in the RBA, as follows:

$$\mathbf{x^{(t)}} = \mathbf{best} + \epsilon A_i^{(t)} R_i, \tag{14}$$

where R_i denotes one of the randomisation methods presented in Table 1.

Table 1 Variants of the RBA algorithm

Randomisation method	*Random generator*	*Implementation*	*Variant*
Uniform distributed	$U_i(0,1)$	Standard C-library	UBA
Gaussian distributed	$N_i(-\infty,+\infty)$	GSL-library	NBA
Lévy flights	$L_i(-\infty,+\infty)$	GSL-library	LBA
Kent chaotic map	$C_i^K(0,1)$	From scratch	CBA1
Logistic chaotic map	$C_i^L(0,1)$	From scratch	CBA2
RSiTFC	$F_i(-\infty,+\infty)$	From scratch	FBA

As can be seen from Table 1, six randomisation methods are used in the RBA algorithm, which can also differ in terms of the interval of generated random values. The random variables with values in an interval $[0,1]$ are the uniform distribution and chaotic maps, while others have an interval $(-\infty,+\infty)$, like Gaussian, Lévy flights, and RSiTFC. When a random value is generated within the interval $[0,1]$, this value is then extended to the interval $[-1,1]$ using a formula $r_i = 2(r_i - 0.5)$. However, the generated solution value x_i usually lies within the valid interval $x_i \in [lb_i, ub_i]$, which can be transformed from [–1, 1]. Interestingly, some implementations of random generators can be found in the Standard C-library (as a standard

random generator for generating uniformly distributed random values), also in the GNU Scientific Library (GSL) (Galassi et al., 2011) (as random generators for generating the Gaussian (Cai et al., 2014) and Lévy flights distributed random values), and the rest were developed from scratch (e.g., chaotic maps and RSiTFC). According to the different randomisation method used for hybridisation, six different variants of the RBA algorithm are developed, i.e., UBA, NBA, LBA, CBA1, CBA2, and FBA.

4 Experiments and results

Our experimental studies were divided into three test categories (identified by):

- characteristics of the various randomisation methods
- the impact of the employed randomisation methods on the results of the RBA
- the quality of results, when they are compared with the other well-known algorithms.

One of the aims of the first experiment was to find out the characteristics of the randomisation methods used in the paper. In line with this, the random numbers generated by various randomisation methods were aggregated and presented in terms of histograms, where the characteristics of a specific method can easily be identified.

In the second experiment, the impact of the various randomisation method was tested on the results of the RBA algorithms as obtained by optimising the benchmark function suite. The RBA used the probability distribution by using the random generators either as implemented in the standard libraries (e.g., uniform and Gaussian distribution, and Lévy flights) or developed from the scratch (e.g., Kent and Logistic chaotic maps, and RSiTFC method). All variants of the implemented RBA algorithms with corresponding implementations of random generator are illustrated in Table 1.

The third experiment compares the results obtained by the more promising variants of the RBA algorithm, like NBA, LBA and FBA, were compared with the other well-known algorithms, like FA, DE and ABC in order to show how these hybridisation improve the results of the original BA algorithm in general.

The RBA parameters in the experiments were set as follows: the loudness $A_0 = 0.5$, pulse rate $r = 0.5$, frequencies $Q_i^{(t)} \in [0.0, 2.0]$. The population size $Np = 100$ was used. All algorithm variants terminated after the $MAX\_FE = 1,000 \cdot D$ fitness function evaluations. The number of independent runs was 25. The mentioned parameter setting represented the best values as found after some preliminary parametric study.

The remainder of the section is structured as follows. In the next subsection, a test suite used in our experiments is illustrated. Then, the results of three tests mentioned earlier are presented. In addition, the comparisons of the results are validated using Friedman non-parametric statistical tests.

4.1 Test suite

In our experimental work, the RBA algorithm was applied to function optimisation problems, which belongs to a class of continuous optimisation problems and is defined as follows.

Let us assume, an objective function $f(\mathbf{x})$ is given, where $\mathbf{x} = (x_1, \ldots, x_D)$ is a vector of D design variables in a decision space S. The design variables $x_j \in \{lb_j, ub_j\}$ are limited between their lower $ub_j \in \mathbb{R}$ and upper bounds $ub_j \in \mathbb{R}$. The task of the function optimisation was to find the global optimums of each function in a test suite.

The test suite is a Black-Box (BBOB) (Hansen et al., 2013) standard benchmark suite, consisting of 24 functions. Their global optima are shifted and rotated by the BBOB benchmark functions and, therefore, they are harder to find, in general. A detailed description of the characteristics of BBOB benchmark functions is not the focus of this paper, and such description can be found in Finck et al. (2010).

4.2 Characteristics of the mentioned randomisation methods

In this test, the characteristics of the randomisation methods, like the uniform and Gaussian distributions, Lévy flights, Kent and Logistic chaotic maps, and RSiTFC method (Fister et al., 2014c), are taken into account. In line with this, five million random numbers were generated for each randomisation method, while the results are aggregated in Figure 1.

Interestingly, Gaussian, Lévy and Cauchy probability distributions have intriguing mathematical property; i.e., all of these distributions are stable. This means that a linear combination of two Gaussian (or Lévy or Cauchy) random variables is itself a Gaussian (or Lévy or Cauchy) random variable. Each of the stable distribution is parametrised by four parameters: an index of stability $\alpha \in (0, 2]$, a skewness parameter $\beta \in [-1, 1]$, a scale parameter $\gamma \geq 0$ and a location parameter $\delta \in \mathbb{R}$. α and β are shape parameters that determine the form of a distribution, while γ and δ are equivalent to the role of the mean μ and standard deviation σ in the Gaussian distribution.

For instance, a Gaussian distribution $N(\mu, \sigma)$ is stable with $(\alpha = 2, \beta = 0, \gamma = \mu, \delta = \sigma/\sqrt{2})$, a Cauchy$(\gamma, \delta)$ distribution is stable with $(\alpha = 1,\ \beta = 0,\ \gamma, \delta)$ and a Levy(γ, δ) distribution is stable with $(\alpha = 1/2, \beta = 1, \gamma, \delta)$. When $\alpha < 1$, the tails of the distribution become extremely wide. Therefore, the Lévy symmetric alpha-stable distribution with $(\alpha = 1.2, \beta = 0, \gamma = 0, \delta = 1)$ is used in our tests.

On the other hand, the RSiTFC randomisation method is controlled by three parameters: scaling factor L, number of levels H and the number of of sub-pieces for each piece N. In order to find the proper parameter setting for this method, a lot of experiments were conducted on the already mentioned test suite of benchmark functions BBOB. The results of this experimental work showed that the most promising results can be obtained by $L = 8$, $H = 4$, and

$N = 3$. Obviously, this parameter setup was used for the RSiTFC method during our experimental work.

The figure is divided into six diagrams, where the results of a specific randomisation method are presented in a histogram. A histogram is considered as a natural representation for a distribution of random numbers. These distribution data define a variable statistically. A frequency in a histogram denotes the number of occurrences for some intervals of values. Here, 101 intervals of a width of 0.2 in the interval $[-9.9, 9.9]$ are presented in x-axis. Each histogram consists of intervals coated on the x-axis denoting the value of the variable, and their frequencies coated on the y-axis. Indeed, the intervals of real values $[-9.9, 9.9]$ numbered from –49 to 49 are presented on the x-axis with the maximum frequency of 100,000. The interval zero comprises the range $[-0.1, 0.1]$, while intervals –50 and 50 capture values < -9.9 and > 9.9, respectively.

By looking at Figure 1 closely, we can draw the following conclusions:

1 At a first glance, a Kent chaotic map is similar to a uniform distribution, but the frequencies of the intervals slightly differ between each other from the former. On the other hand, the Logistic chaotic map is an inversion of the previously mentioned, because the border intervals 0 and 5 exhibit higher frequencies than the inner part.

2 The Gaussian distribution is more compact than Lévy flights because the latter enables to generate random numbers outside the intervals –50 and 50.

3 The RSiTFC randomised method plot exhibits more peaks. This behaviour might be useful for solving multi-modal problems.

In summary, three randomisation methods, i.e., uniform, Kent and Logistic chaotic maps, generate random numbers in the interval $[0, 1]$, while the other three can generate random numbers in the interval $-\infty + \infty$. For example, the Lévy flights and RSiTFC can generate values in intervals –50 and 50, while about 99.7% of generated values are usually within the three standard deviations of the mean by the Gaussian distribution. Therefore, this distribution does not allow the big changes of the variable in general.

4.3 Impact of randomisation methods on the results of RBA

In this experiment, the impact of various randomisation methods on the results of the RBA algorithms was verified. Therefore, all six variants of the RBA algorithms were applied to the BBOB benchmark test suite. Thus, the experimental setup was the same as presented at the beginning of this section.

Although the functions with dimensions $D = 10$, $D = 30$, and $D = 50$ were optimised, only those results optimising the functions with dimension $D = 50$ are presented in Table 2, due to the length limitation of the paper. For similar reasons, only the mean and standard deviation are presented in the table, although the results were accompanied, according to the best, worst, mean, standard deviation and median measures. The best results are in italics in this table.

Figure 1 Results of the various randomisation methods, (a) uniform (b) Gaussian (c) Lévy flights ($\alpha = 1.2$) (d) Kent chaotic map (e) logistic chaotic map (f) RSiTFC ($L = 8, H = 4, N = 3$) (see online version for colours)

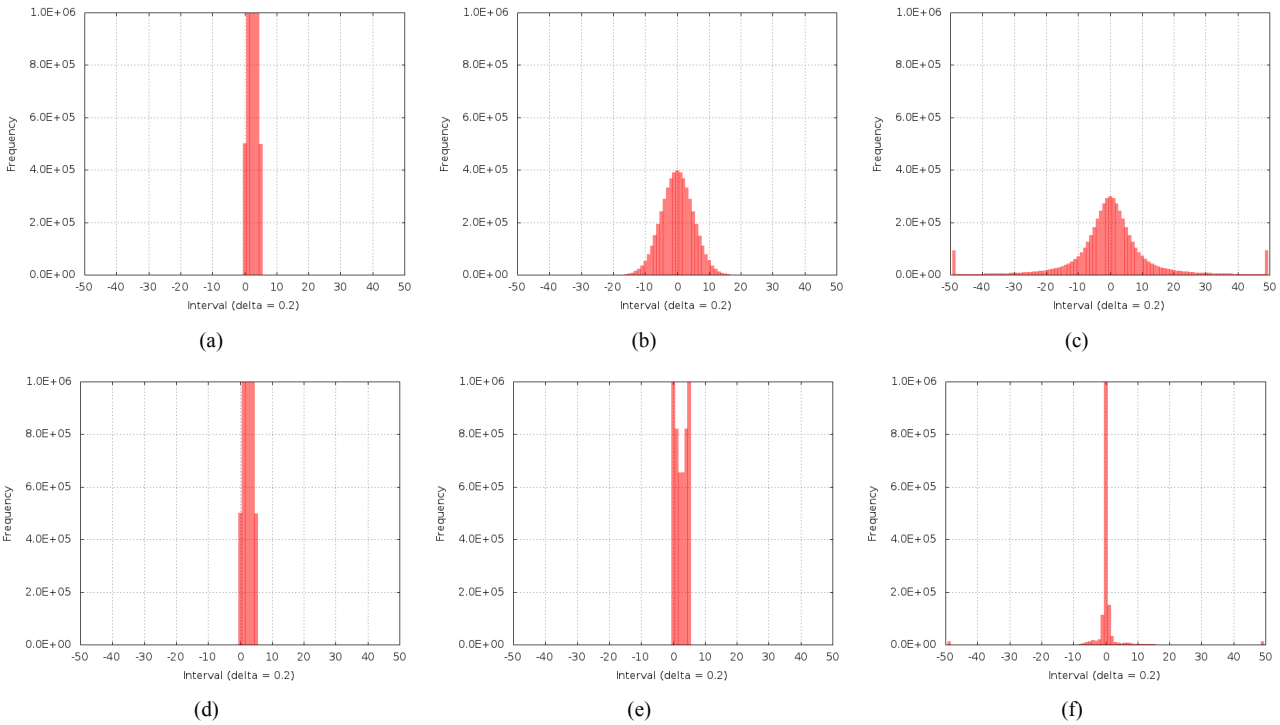
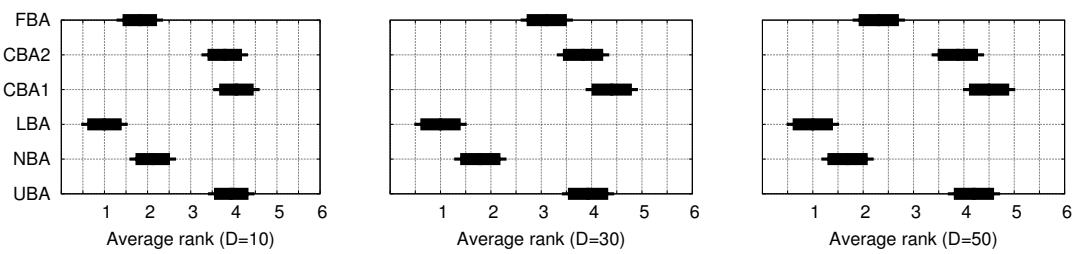


Table 2 Results of RBA according to various randomisation methods (D = 50)

Function	*Meas.*	*UBA*	*NBA*	*LBA*	*CBA1*	*CBA2*	*FBA*
f_1	Mean	1.91E+02	4.66E+00	*2.81E–04*	1.87E+02	1.20E+02	9.25E+01
	Stdev	3.60E+01	6.31E+00	3.20E–05	3.90E+01	2.82E+02	1.78E+01
f_2	Mean	3.74E+06	1.62E+05	*4.71E+02*	3.89E+06	1.66E+06	8.65E+05
	Stdev	1.28E+06	1.47E+05	2.27E+02	1.44E+06	6.18E+06	7.28E+05
f_3	Mean	1.44E+03	1.08E+03	*2.14E+02*	1.50E+03	1.04E+03	1.00E+03
	Stdev	2.04E+02	1.92E+02	5.33E+01	1.84E+02	1.67E+03	1.95E+02
f_4	Mean	1.68E+03	1.25E+03	*2.50E+02*	1.59E+03	1.14E+03	1.29E+03
	Stdev	2.47E+02	2.27E+02	7.58E+01	1.99E+02	1.90E+03	2.14E+02
f_5	Mean	2.64E+02	2.79E+02	*2.31E–03*	2.85E+02	2.12E+02	1.84E+02
	Stdev	7.97E+01	7.63E+01	2.39E–03	4.03E+01	3.62E+02	5.95E+01
f_6	Mean	3.40E+05	3.50E+03	*2.71E+01*	4.94E+05	5.46E+04	6.20E+04
	Stdev	1.70E+05	5.70E+03	2.72E+01	2.38E+05	9.90E+05	7.02E+04
f_7	Mean	1.08E+03	9.52E+02	*3.24E+02*	1.11E+03	6.26E+02	9.25E+02
	Stdev	2.98E+02	2.28E+02	8.09E+01	4.01E+02	2.07E+03	2.66E+02
f_8	Mean	1.05E+05	3.17E+02	*9.00E+01*	1.10E+05	5.20E+04	3.21E+04
	Stdev	3.14E+04	2.60E+02	3.07E+01	3.58E+04	1.97E+05	1.45E+04
f_9	Mean	1.83E+04	5.84E+01	*5.59E+01*	1.92E+04	6.38E+03	3.34E+03
	Stdev	5.82E+03	1.82E+01	1.90E+01	8.92E+03	3.89E+04	1.46E+03
f_{10}	Mean	3.11E+06	3.66E+05	*1.65E+04*	4.33E+06	1.61E+06	6.28E+05
	Stdev	1.06E+06	2.98E+05	4.57E+03	1.73E+06	7.22E+06	4.66E+05
f_{11}	Mean	7.04E+02	3.33E+02	*2.92E+02*	6.46E+02	2.97E+02	6.48E+02
	Stdev	1.78E+02	1.76E+02	1.12E+02	1.29E+02	7.69E+02	1.25E+02
f_{12}	Mean	3.26E+08	1.85E+07	*4.58E+02*	3.42E+08	1.63E+08	2.04E+08
	Stdev	6.90E+07	1.80E+07	3.58E+02	1.07E+08	5.06E+08	3.69E+07
f_{13}	Mean	2.80E+03	5.49E+02	*1.07E+01*	2.71E+03	2.21E+03	1.92E+03
	Stdev	2.41E+02	3.44E+02	6.62E+00	2.44E+02	3.00E+03	4.43E+02
f_{14}	Mean	4.90E+01	8.03E+00	*5.36E–03*	4.73E+01	2.83E+01	2.38E+01
	Stdev	1.71E+01	4.91E+00	6.46E–04	1.09E+01	6.85E+01	7.69E+00
f_{15}	Mean	1.40E+03	*1.05E+03*	1.23E+03	1.36E+03	1.10E+03	1.15E+03
	Stdev	2.11E+02	2.78E+02	2.81E+02	1.87E+02	1.79E+03	2.57E+02
f_{16}	Mean	4.26E+01	3.79E+01	3.22E+01	4.17E+01	*3.17E+01*	3.68E+01
	Stdev	4.55E+00	6.40E+00	5.08E+00	5.77E+00	4.54E+01	5.15E+00
f_{17}	Mean	1.19E+01	*7.28E+00*	9.23E+00	1.19E+01	8.47E+00	8.47E+00
	Stdev	2.52E+00	1.27E+00	1.97E+00	1.97E+00	1.44E+01	1.54E+00
f_{18}	Mean	4.72E+01	3.33E+01	3.74E+01	4.59E+01	*3.04E+01*	3.59E+01
	Stdev	7.73E+00	8.82E+00	8.31E+00	5.94E+00	5.38E+01	7.30E+00
f_{19}	Mean	9.00E+00	6.97E+00	7.53E+00	8.83E+00	*6.92E+00*	8.23E+00
	Stdev	1.13E+00	1.10E+00	9.91E–01	1.02E+00	9.40E+00	1.07E+00
f_{20}	Mean	2.32E+04	2.59E+00	*1.91E+00*	2.60E+04	9.77E+03	6.59E+03
	Stdev	1.11E+04	2.98E–01	2.20E–01	1.12E+04	4.81E+04	6.53E+03
f_{21}	Mean	7.83E+01	9.97E+00	*6.91E+00*	7.80E+01	6.72E+01	6.83E+01
	Stdev	6.06E+00	1.24E+01	1.22E+01	6.25E+00	7.90E+01	9.47E+00
f_{22}	Mean	8.15E+01	1.49E+01	*1.05E+01*	8.25E+01	7.21E+01	7.19E+01
	Stdev	6.10E+00	9.16E+00	7.61E+00	5.98E+00	8.37E+01	9.48E+00
f_{23}	Mean	5.38E+00	2.45E+00	*2.01E+00*	4.80E+00	3.47E+00	4.73E+00
	Stdev	5.95E–01	7.27E–01	1.02E+00	9.11E–01	5.56E+00	8.43E–01
f_{24}	Mean	7.95E+02	*4.65E+02*	5.95E+02	8.06E+02	7.08E+02	4.95E+02
	Stdev	6.62E+01	8.13E+01	1.26E+02	6.57E+01	8.75E+02	6.81E+01

Figure 2 Results of the Friedman non-parametric tests on different variants of RBA algorithms



As can be seen from Table 2, the LBA algorithm randomised with Lévy flights outperformed the results of other variants of the RBA algorithms except for functions f_{15}, f_{17} and f_{24}, where the NBA algorithm was better, and for f_{16}, f_{18} and f_{19}, where the CBA2 was better. A closer look at the characteristics of these functions (Finck et al., 2010) shows that these functions are multi-modal. From this fact, it could be speculated that the NBA and CBA2 variants are more suitable to solving multi-modal functions, while other functions in the test suite are better solved by the LBA variant of RBA. In summary, selecting the suitable randomisation method may depend on the problem to be solved.

The standard Friedman tests were conducted using a significance level 0.05. Here, six classifiers are compared in terms of $24 * 5 = 120$ variables, where the number of functions is 24 and the number of measures is 5. The results of the Friedman non-parametric tests are presented in Figure 2 where the three diagrams show the ranks and confidence intervals (critical differences) for the algorithms under consideration. The closer to the rank one, the better the algorithm is. The diagrams are organised, according to the dimensions of functions.

The first diagram in Figure 2 shows that the LBA significantly outperforms the results of all other variants of RBA, except for NBA and FBA, when optimising functions with dimension $D = 10$. The second diagram represents the ranks when optimising functions with dimension $D = 30$, and it can be seen that two algorithms, i.e., LBA and NBA, significantly outperform the results of the other algorithm variants in tests. Finally, for functions with dimension $D = 50$, LBA, NBA and FBA perform significantly better than other algorithms in tests.

Interestingly, the obtained results are different from the results as reported earlier by Fister et al. (2014a), where the original FA randomised with RSiTFC works well especially the multi-modal functions. This study seems to prefer the randomisation techniques with Lévi flights in place of the RSiTFC, as in Fister et al. (2014a), and leads to speculation that the selection of the suitable randomisation method may also depend on the algorithm itself. That is to say, some algorithms can be better enhanced with one class of the randomisation techniques, while other algorithms may be better with other randomisation techniques.

4.4 Comparative study

In this experiment, the BA algorithm originally used the Gaussian probability distribution (also NBA) was compared with other well-known algorithms such as FA, DE, and ABC, as well as the more promising variants of the RBA algorithm from the last experiments, like LBA and FBA.

While the RBA used the same parameter setup as presented at the beginning of this section, the other algorithms applied the following parameters. The FA parameters were set as follows: the randomised parameter $\alpha = 0.1$, the attractiveness $\beta = 0.2$, and the fixed light absorption $\gamma = 0.9$. The DE parameters were set as follows: the amplification factor of the difference vector $F = 0.5$, and the crossover control parameter $CR = 0.9$. The percentage of onlooker bees for the ABC algorithm was 50% of the colony, the employed bees represented another 50% of the colony, while the generation of scouts were regulated by parameter *limits* $= 100$. That means, when a bee position is not improved in 100 successive generations, this bee becomes scout. In order to make the comparison as fair as possible, each algorithm in test used the same population size $Np = 100$ and terminates after $MAX\_FE = 1,000 \cdot D$ number of fitness function evaluations. That means that the maximum number of fitness function evaluations depends on the dimensionality of the problems. For instance, for $D = 10$, $MAX\_FE = 10,000$ is used. Similarly, $MAX\_FE = 30,000$ for $D = 30$ and $MAX\_FE = 50,000$ for $D = 50$.

Table 3. It is worth pointing out that only this instance of data is illustrated in the table, although the experiments were conducted on all three different dimensions, i.e., $D = 10$, $D = 30$ and $D = 50$. Moreover, the results are presented in terms of the mean and standard deviation, although these were also accompanied by all five measures, as mentioned before. The best results of the algorithms are written in italics.

As it can be seen from Table 3, the LBA algorithm outperformed the results of the other algorithms for eight times (functions), DE for ten times and ABC for six times. 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 Figure 3, which is also divided into three diagrams, according to the dimensions of the functions.

The first diagram presents the results of the Friedman test comparing the results obtained by optimising the functions with dimensions $D = 10$. From this diagram, it can be shown that ABC, DE and LBA outperformed the results of the other algorithms significantly. A similar situation also appears by optimising the functions with dimension $D = 30$. Finally, on the functions with dimension $D = 50$, the DE and LBA achieved significantly better results than all other algorithms except ABC. Note that promising results were also reached by NBA.

5 Conclusions

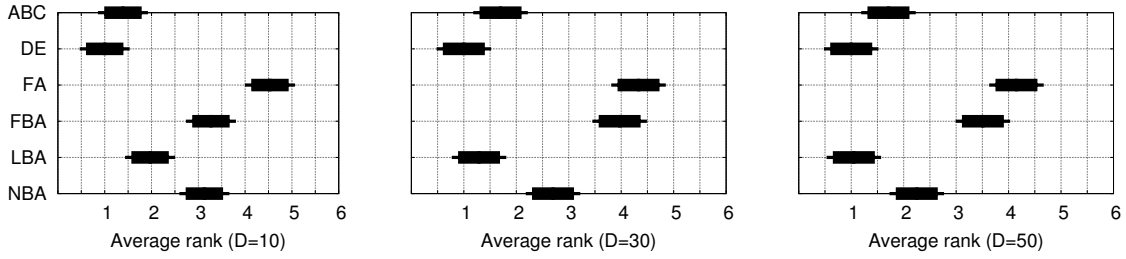
In this paper, we have carried out extensive numerical experiments for new variants of the bat algorithm. The present work implies that

- the selection of randomisation methods has a crucial impact on the results of the original BA algorithm
- the results of the best variants of the RBA algorithm are comparable with the results of the other well-known algorithms.

Table 3 Comparing the results of RBA with the results of other algorithms (D = 30)

Function	*Meas.*	*NBA*	*LBA*	*FBA*	*FA*	*DE*	*ABC*
f_1	Mean	6.94E+00	*7.10E–05*	6.73E+01	1.02E+02	9.91E–03	7.27E–02
	Stdev	5.49E+00	1.10E–05	1.67E+01	2.61E+01	4.07E–03	1.21E–01
f_2	Mean	1.18E+05	3.89E+02	1.24E+06	2.86E+06	*8.03E+00*	2.34E+02
	Stdev	7.11E+04	2.25E+02	7.39E+05	8.84E+05	2.62E+00	2.91E+02
f_3	Mean	4.85E+02	1.80E+02	5.82E+02	7.27E+02	2.16E+02	*4.37E+01*
	Stdev	1.10E+02	4.99E+01	1.23E+02	1.01E+02	8.18E+00	1.17E+01
f_4	Mean	6.41E+02	2.51E+02	7.80E+02	7.65E+02	2.38E+02	*5.09E+01*
	Stdev	1.84E+02	1.01E+02	1.52E+02	1.05E+02	1.06E+01	7.70E+00
f_5	Mean	7.41E+01	3.46E–02	7.85E+01	4.23E+02	*1.67E–02*	1.77E+00
	Stdev	5.53E+01	3.58E–02	4.29E+01	1.48E+01	4.76E–03	1.98E+00
f_6	Mean	3.79E+03	*4.04E+01*	4.13E+04	2.59E+05	4.68E+01	1.89E+02
	Stdev	7.16E+03	3.79E+01	4.86E+04	1.08E+05	1.07E+01	4.95E+01
f_7	Mean	4.06E+02	1.58E+02	3.95E+02	6.71E+02	*1.12E+01*	1.14E+02
	Stdev	1.44E+02	7.09E+01	1.50E+02	2.83E+02	2.25E+00	2.31E+01
f_8	Mean	5.12E+02	5.12E+01	2.42E+04	4.04E+04	*3.58E+01*	1.41E+02
	Stdev	4.67E+02	4.58E+01	1.79E+04	1.26E+04	4.29E+00	4.39E+01
f_9	Mean	6.55E+01	*3.85E+01*	3.97E+03	8.47E+02	4.00E+01	1.35E+02
	Stdev	4.16E+01	2.63E+01	2.17E+03	2.43E+02	6.44E+00	7.89E+01
f_{10}	Mean	2.15E+05	*9.43E+03*	6.02E+05	2.42E+06	7.81E+04	1.30E+05
	Stdev	2.18E+05	3.49E+03	4.23E+05	7.79E+05	1.93E+04	3.04E+04
f_{11}	Mean	2.73E+02	*1.88E+02*	4.12E+02	2.50E+02	2.10E+02	2.17E+02
	Stdev	1.31E+02	8.38E+01	1.29E+02	1.44E+02	3.51E+01	2.22E+01
f_{12}	Mean	1.96E+07	*1.72E+03*	8.96E+07	1.73E+08	2.08E+04	8.70E+04
	Stdev	1.86E+07	6.19E+03	3.89E+07	7.25E+07	9.24E+03	5.09E+04
f_{13}	Mean	5.59E+02	*1.90E+01*	1.55E+03	2.05E+03	4.23E+01	3.92E+02
	Stdev	2.72E+02	1.82E+01	2.58E+02	2.51E+02	8.97E+00	6.86E+01
f_{14}	Mean	6.50E+00	*2.72E–03*	1.85E+01	3.61E+01	5.84E–02	9.93E–01
	Stdev	3.54E+00	2.46E–04	5.96E+00	1.61E+01	1.05E–02	7.79E–01
f_{15}	Mean	4.66E+02	5.71E+02	5.83E+02	7.20E+02	*2.36E+02*	4.52E+02
	Stdev	1.38E+02	1.74E+02	1.22E+02	1.56E+02	1.35E+01	4.65E+01
f_{16}	Mean	3.06E+01	2.57E+01	3.04E+01	3.89E+01	2.74E+01	*1.39E+01*
	Stdev	5.03E+00	4.89E+00	4.35E+00	3.51E+00	3.89E+00	2.24E+00
f_{17}	Mean	6.72E+00	7.95E+00	8.61E+00	1.01E+01	*6.81E–01*	8.42E+00
	Stdev	1.43E+00	1.44E+00	1.41E+00	1.81E+00	1.64E–01	1.07E+00
f_{18}	Mean	2.62E+01	3.05E+01	3.47E+01	3.61E+01	*4.36E+00*	3.25E+01
	Stdev	5.51E+00	8.01E+00	9.12E+00	9.28E+00	1.49E+00	5.35E+00
f_{19}	Mean	6.74E+00	7.10E+00	7.72E+00	6.65E+00	*6.61E+00*	9.68E+00
	Stdev	1.15E+00	1.35E+00	1.30E+00	5.89E–01	5.88E–01	1.00E+00
f_{20}	Mean	2.29E+00	1.84E+00	4.21E+03	5.73E+03	3.23E+00	*1.36E+00*
	Stdev	3.30E–01	1.90E–01	3.60E+03	3.47E+03	1.97E–01	1.73E–01
f_{21}	Mean	1.57E+01	7.71E+00	5.96E+01	7.55E+01	8.04E+00	*2.37E+00*
	Stdev	9.50E+00	8.30E+00	1.15E+01	4.75E+00	8.58E+00	1.33E+00
f_{22}	Mean	1.58E+01	1.21E+01	6.62E+01	8.02E+01	4.15E+00	*3.42E+00*
	Stdev	7.83E+00	1.76E+01	8.87E+00	4.71E+00	4.20E+00	3.07E+00
f_{23}	Mean	1.38E+00	*1.28E+00*	3.45E+00	4.94E+00	3.12E+00	2.39E+00
	Stdev	6.34E–01	4.18E–01	8.85E–01	1.02E+00	4.46E–01	3.17E–01
f_{24}	Mean	2.75E+02	3.90E+02	3.57E+02	3.78E+02	*2.61E+02*	4.96E+02
	Stdev	4.19E+01	7.88E+01	6.00E+01	2.79E+01	1.36E+01	3.58E+01

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



The original BA algorithm (NBA variant of RBA) was randomised by using five various randomisation methods or distributions (uniform, Lévy flights, Kent and Logistic chaotic maps, RSiTFC). Each of these six algorithms were applied to the BBOB benchmark function suite so as to show that the obtained results are significantly different. Experiments on 24 BBOB benchmark functions confirmed that the selection of randomisation methods has a crucial impact on the results of the RBA algorithms.

In addition, the RBA algorithms with the best randomisation methods from the previous experiment were compared with the other well-known algorithms, like FA, DE, and ABC. It turned out that the results of the LBA improved significantly, especially for high-dimensional problems. This mean that the results of LBA by dimension $D = 50$ are almost equally to the results of the DE that obtained the best results by optimising the functions of all dimensions. As a result, LBA performs equally well as the best DE algorithm.

Our numerical experiments also showed that the selection of the appropriate randomisation method may affect the results of the RBA algorithms. That may lead to the assumption that the proper randomisation method is influenced by the problem to be solved. For instance, this study seems to prefer the RBA randomised by Lévy fligths (LBA), while the NBA variant of the RBA is better for solving multi-modal functions. However, this new conclusion is different from the earlier work by Fister et al. (2014a) that reported that the FA randomised with RSiTFC randomisation method works well, especially, for multi-modal functions. This may imply that the suitable randomisation methods may also depend on the algorithm under the application of randomisation techniques.

Obviously, future work can also focus on the application of these randomisation methods would to metaheuristic algorithms, like ABC, PSO, etc. It would be useful to verify if the proper selection of the randomisation methods should be linked with the type of problems and the algorithm itself. On the other hand, parameter control and tuning can also affect the performance of an algorithm, which needs further control and parameter tuning for different randomisation variants. All these can form active research topics in the coming years.

References

Alihodzic, A. and Tuba, M. (2014) 'Improved bat algorithm applied to multilevel image thresholding', *The Scientific World Journal*, Vol. 2014, Article ID 176718, 16 pp., doi: 10.1155/2014/176718.

Beekman, M., Sword, G.A. and Simpson, S.J. (2008) 'Biological foundations of swarm intelligence', in Blum, C. and Merkle, D. (Eds.): *Swarm Intelligence: Introduction and Applications*, pp.3–41, Springer-Verlag, Berlin.

Beni, G. and Wang, J. (1989) 'Swarm intelligence in cellular robotic systems', in *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, pp.26–30, Tuscany, Italy.

Blum, C. and Li, X. (2008) 'Swarm intelligence in optimization', in Blum, C. and Merkle, D. (Eds.): *Swarm Intelligence: Introduction and Applications*, pp.43–86, Springer-Verlag, Heidelberg.

Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V. (2006) 'Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems', *IEEE Trans. Evol. Comput.*, Vol. 10, No. 6, pp.646–657.

Cai, X., Wang, L., Kang, Q. and Wu, Q. (2014) 'Bat algorithm with Gaussian walk', *International Journal of Bio-Inspired Computation*, Vol. 6, No. 3, pp.166–174.

Črepinšek, M., Mernik, M. and Liu, S.H. (2011) 'Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees', *Int. J. Innovative Comput. Appl.*, Vol. 3, No. 1, pp.11–19.

Das, S. and Suganthan, P.N. (2011) 'Differential evolution: a survey of the state-of-the-art', *IEEE Trans. Evol. Comput.*, Vol. 10, No. 6, pp.646–657.

Demšar, J. (2006) 'Statistical comparisons of classifiers over multiple data sets', *J. Mach. Learn. Res.*, Vol. 7, pp.1–30.

Dorigo, M. and Di Caro, G. (1999) 'The ant colony optimization meta-heuristic', in Corne, D., Dorigo, M. and Glover, F. (Eds.): *New Ideas in Optimization*, pp.11–32, McGraw Hill, London, UK.

Eiben, A.E. and Smith, J.E. (2003) *Introduction to Evolutionary Computing*, Springer-Verlag, Berlin.

Elmegreen, B. G. (1997) 'The Initial stellar mass function from random sampling in a turbulent fractal cloud', *Astrophys. J.*, Vol. 486, No. 2, pp.944–954.

Feldman, D.P. (2012) *Chaos and Fractals: An Elementary Introduction*, OUP Oxford, UK.

Finck, S., Hansen, N., Ros, R. and Auger, A. (2010) *Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions*, Technical Report 2009/20, Research Center PPE, 2009, updated February.

Fister, I., Fister Jr., I., Brest, J. and Žumer, V. (2012a) 'Memetic artificial bee colony algorithm for large-scale global optimization', in *IEEE Congress on Evolutionary Computation*, Brisbane, Australia, pp.3038–3045, IEEE Publications.

Fister Jr., I., Yang, X-S., Fister, I. and Brest, J. (2012b) 'Memetic firefly algorithm for combinatorial optimization', in Filipič, B. and Šilc, J. (Eds.): *Bioinspired Optimization Methods and Their Applications: Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications – BIOMA*, pp.75–86, Jožef Stefan Institute.

Fister, I., Yang, X-S., Brest, J. and Fister Jr., I. (2013a) 'Memetic self-adaptive firefly algorithm', in Yang, X-S., Xiao, R.Z.C., Gandomi, A.H. and Karamanoglu, M. (Eds.): *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, pp.73–102, Elsevier, Amsterdam.

Fister, I., Fister Jr., I., Yang, X-S. and Brest, J. (2013b) 'A comprehensive review of firefly algorithms', *Swarm and Evolutionary Computation*, available via ScienceDirect.

Fister Jr., I., Fister, D. and Yang, X-S. (2013c) 'A hybrid bat algorithm', *Electrotechnical Review*, Vol. 80, No. 1, pp.1–7.

Fister Jr., I., Fister, D. and Fister, I. (2013d) 'A comprehensive review of cuckoo search: variants and hybrids', *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 4, No. 4, pp.387–409.

Fister, I., Yang, X.S., Brest, J. and Fister Jr., I. (2013e) 'Modified firefly algorithm using quaternion representation', *Expert Syst. Appl.*, Vol. 40, No. 18, pp.7220–7230.

Fister Jr., I., Fister, D. and Fister. I. (2013f) 'Differential evolution strategies with random forest regression in the bat algorithm', *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, ACM.

Fister, I., Yang, X-S., Brest, J. and Fister Jr., I. (2014a) 'On the randomized firefly algorithm', in *Cuckoo Search and Firefly Algorithm*, Springer-Verlag, Berlin (in press).

Fister Jr., I., Yang, X-S., Ljubič, K., Fister, D., Brest, J. and Fister, I. (2014b) 'Towards the novel reasoning among particles in PSO by the use of RDF and SPARQL', *The Scientific World Journal*, Vol. 2014, Article ID 121782, 10 pp., doi: 10.1155/2014/121782.

Fister, I., Yang, X. S., Brest, J. and Fister Jr., I. (2014c) 'On the randomized firefly algorithm', in *Cuckoo Search and Firefly Algorithm*, pp.27–48, Springer International Publishing.

Fister Jr., I., Yang, X.S., Fister, D. and Fister, I. (2014d) 'Cuckoo search: a brief literature review', in *Cuckoo Search and Firefly Algorithm*, pp.49–62, Springer International Publishing.

Fister, I.J., Fong, S., Brest, J. and Fister, I. (2014e) 'Towards the self-adaptation in the bat algorithm', in *Proceedings of the 13th IASTED International Conference on Artificial Intelligence and Applications*.

Fister, I., Fong, S., Brest, J. and Fister, I. (2014f) 'A novel hybrid self-adaptive bat algorithm', *The Scientific World Journal*, Vol. 2014, Article ID 709738, 12 pp., doi: 10.1155/2014/709738.

Fister, I., Rauter, S., Yang, X.S., Ljubič, K. and Fister Jr., I. (2014g) 'Planning the sports training sessions with the bat algorithm', *Neurocomputing*.

Fister, I. (2013) *A Comprehensive Review of Bat Algorithms and Their Hybridization*, Master's thesis, University of Maribor, Slovenia.

Friedman, M. (1937) 'The use of ranks to avoid the assumption of normality implicit in the analysis of variance', *J. Am. Stat. Assoc.*, Vol. 32, No. 200, pp.675–701.

Friedman, M. (1940) 'A comparison of alternative tests of significance for the problem of m rankings', *An. Math. Stat.*, Vol. 11, No. 1, pp.86–92.

Galassi, D. et al. (2011) *GNU Scientific Library: Reference Manual*, Edition 1.15, Network Theory, Ltd.

Gandomi, A.H., Yang, X-S., Talatahari, S. and Alavi, A.H. (2013) 'Firefly algorithm with chaos', *Commun. Nonlinear Sci. Numer. Simul.*, Vol. 18, No. 1, pp.89–98.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co., New York, USA.

Hansen, N., Auger, A., Ros, R., Finck, S. and Pošík, P. (2013) *Black-Box Optimization Benchmarking (BBOB)*, available via INRIA.

He, X-s., Ding, W-J. and Yang, X-S. (2013) 'Bat algorithm based on simulated annealing and Gaussian perturbations', *Neural Computing and Applications*, Vol. 25, No. 2, pp.459–468.

Hertz, A., Taillard, E. and de Werra, D. (2003) 'Tabu search', in Aarts, E. and Lenstra, J.K. (Eds.): *Local Search in Combinatorial Optimization*, pp.121–136, Princeton University Press, New Jersey.

Hoos, H.H. and Stützle, T. (2004) *Stochastic Local Search: Foundations & Applications*, Morgan Kaufmann, San Francisco.

Jamil, M. and Zepernick, H. (2013) 'Lévy flights and global optimization', in Yang, X-S., Xiao, R.Z.C., Gandomi, A.H. and Karamanoglu, M. (Eds.): *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, pp.49–72, Elsevier, Amsterdam.

Karaboga, D. and Basturk, B. (2007) 'A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm', *J. Global Optim.*, Vol. 39, No. 3, pp.459–471.

Kashi, S., Minuchehr, A., Poursalehi, N. and Zolfaghari, A. (2014) 'Bat algorithm for the fuel arrangement optimization of reactor core', *Annals of Nuclear Energy*, Vol. 64, pp.144–151.

Kennedy, J. and Eberhart, R.C. (1999) 'The particle swarm optimization: social adaptation in information processing', in Corne, D., Dorigo, M. and Glover, F. (Eds.): *New Ideas in Optimization*, pp.379–387, McGraw Hill, London, UK.

Kennedy, J.F. and Eberhart, R.C. (2001) *Swarm Intelligence*, Morgan Kaufman, Burlington, MA.

Laamari, M.A. and Kamel, N. (2014) 'A hybrid bat based feature selection approach for intrusion detection', *Bio-Inspired Computing-Theories and Applications*, pp.230–238, Springer, Berlin Heidelberg.

Taha, A.M., Mustapha, A. and Chen, S-D. (2013) 'Naive Bayes-guided bat algorithm for feature selection', *The Scientific World Journal*, Vol. 2013, Article ID 325973, 9 pp., doi: 10.1155/2013/325973.

Wang, G. and Guo, L. (2013) 'A novel hybrid bat algorithm with harmony search for global numerical optimization', *Journal of Applied Mathematics*, Vol. 2013, Article ID 696491, 21 pp., doi: 10.1155/2013/696491.

Wang, G., Guo, L., Duan, H., Liu, L. and Wang, H. (2012) 'A bat algorithm with mutation for UCAV path planning', *The Scientific World Journal*, Vol. 2012, Article ID 418946, 15 pp., doi: 10.1100/2012/418946.

Wolpert, D.H. and Macready, W.G. (1997) 'No free lunch theorems for optimization', *Evolutionary Computation, IEEE Transactions on*, Vol. 1.1, No. 1, pp.67–82.

Xie, J., Zhou, Y. and Zheng, H. (2013) 'A hybrid metaheuristic for multiple runways aircraft landing problem based on bat algorithm', *Journal of Applied Mathematics*, Vol. 2013, Article ID 742653, 8 pp., doi: 10.1155/2013/742653.

Yang, X-S. and Deb, S. (2009) 'Cuckoo search via Levy flights', in *World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp.210–214, IEEE Publications.

Yang, X-S. and He, A. (2013) "Bat algorithm: literature review and applications', *International Journal of Bio-Inspired Computation*, Vol. 5, No. 3, pp.141–149.

Yang, X-S., Karamanoglu, M. and Fong, S. (2012) 'Bat algorithm for topology optimization in microelectronic applications', *Future Generation Communication Technology (FGCT), International Conference on*, IEEE.

Yang, X-S. (2008) 'Firefly algorithm', in Yang, X-S. (Ed.): *Nature-Inspired Metaheuristic Algorithms*, pp.79–90, Wiley Online Library.

Yang, X-S. (2009) 'Firefly algorithms for multimodal optimization', in *Stochastic Algorithms: Foundations and Applications*, pp.169–178, Springer-Verlag, Berlin.

Yang, X-S. (2010a) 'A new metaheuristic bat-inspired algorithm', in Cruz, C., González, J.R., Krasnogor, N., Pelta, D.A. and Terrazas, G. (Eds.): *Nature Inspired Cooperative Strategies for Optimization (NISCO)*, Vol. 284, pp.65–74, Springer-Verlag, Berlin.

Yang, X-S. (2010b) 'Appendix a: test problems in optimization', in Yang, X-S. (Ed.): *Engineering Optimization*, pp.261–266, John Wiley & Sons, Inc.

Zhou, Q., Li, L., Chen, Z-Q. and Zhao, J-X. (2008) 'Implementation of LT codes based on chaos', *Chin. Phys. B*, Vol. 17, No. 10, pp.3609–3615.