ELEKTROTEHNIŠKI VESTNIK 80(1-2): 1–7, 2013
ORIGINAL SCIENTIFIC PAPER

A Hybrid Bat Algorithm

Iztok Fister Jr.[1], **Dušan Fister**[1], **Xin-She Yang**[2]

[1]*University of Maribor*
Faculty of electrical engineering and computer science, Smetanova 17, 2000 Maribor, Slovenia
E-mail: iztok.fister@guest.arnes.si
[2]*University of Middlesex*
School of Science and Technology, Middlesex University, London NW4 4BT, United Kingdom
E-mail: xy227@cam.ac.uk

Abstract. Swarm intelligence is a very powerful technique appropriate to optimization. In this paper, we present a new swarm intelligence algorithm, which is based on the bat algorithm. Bat algorithm has been hybridized with differential evolution strategies. This hybridization showed very promising results on standard benchmark functions and also significantly improved the original bat algorithm.

Keywords: swarm intelligence, bat algorithm, differential evolution, optimization

Hibridni algoritem na osnovi obnašanja netopirjev

Inteligenca rojev, (angl. Swarm Intelligence) postaja zelo pomembna optimizacijska tehnika. V članku predstavljamo nov algoritem inteligence rojev, ki temelji na osnovi obnašanja netopirjev, (angl. Bat Algorithm) in je hibridiziran s strategijami diferencialne evolucije. Poleg zelo obetavnih rezultatov na testnih primerih, (angl. Benchmark Functions), hibridizacija prav tako občutno izboljša originalni netopirski algoritem.

1 INTRODUCTION

Nature has always been an inspiration for researchers. In the past, many new nature-inspired algorithms have been developed to solve hard problems in optimization. In general, there are two main concepts developed in bio-inspired computation:

1) evolutionary algorithms,
2) swarm intelligence algorithms.

Evolutionary algorithms are optimization techniques [7] that base on Darwin's principle of survivor of the fittest [5]. It states that in nature, the fittest individuals have the greater chances to survive. Evolutionary algorithms consist of the following disciplines: genetic algorithms, evolution strategies, genetic programming, evolutionary programming, differential evolution.

Although all these algorithms or methods were developed independently, they share similar characteristics (like variation operators, selection operators), when solving problems. In fact, the evolutionary algorithms are distinguished by their representation of solutions. For example, genetic algorithms [15], [16] support the binary representation of solution, evolution strategies [2], [11] and differential evolution [29], [3], [6] work on real-valued solutions, genetic programming [21] acts on programs in Lisp, while the evolutionary programming [13] behaves with finish state automata. Evolutionary algorithms have been applied to a wide range of areas of optimization, modeling, and simulation. Essentially, differential evolution has sucessfully been employed in the following areas of optimization: function optimization [28], large-scale global optimization [4], graph coloring [8], chemical process optimization [1].

Swarm intelligence is the collective behaviour of decentralized, self-organized systems, either natural or artificial. Swarm intelligence was introduced by Beny in 1989. A lot of algorithms were proposed since then. Swarm intelligence algorithms were applied on continuous as well as combinatorial optimization problems [25]. The most well-known classes of swarm intelligence algorithms are as follows: particle swarm optimization, ant colony optimization, artificial bee colony, firefly algorithm, cuckoo search and bat algorithm.

Particle swarm optimization has been successfully applied in problems of antenna design [17] and electromagnetics [27]. Ant colony algorithms were also used in many areas of optimization [20] [26] [22].Artificial bee colony showed good performance in numerical optimization [18] [19], in large-scale global optimization [10], and also in combinatorial optimization [24] [9] [30].

Cuckoo search algorithm is a very strong method for function optimization and also for engineering optimization problems[34] [33]. Firefly algorithm showed promising results in function optimization and it showed good results also in combinatorial optimization [12].

Echolocation is an important feature of bat behaviour. That means, bats emit a sound pulse and listen to

Received 12 April 2013
Accepted 26 April 2013

the echo bouncing back from obstacles whilst flying. This phenomenon has been inspired Yang [36] to develop the Bat Algorithm (BA). The algorithm obtained good results where dealing with lower-dimensional optimization problems, but may become problematic for higher-dimensional problems because it tends to converge very fast initially. On the other hand, differential evolution [23] is a typical evolutionary algorithm with differential mutation, crossover and selection that was successfully applied to continuous function optimization.

In order to improve bat algorithm behaviour for higher-dimensional problems, the original bat algorithm were hybridized with differential-evolution strategies, in this paper. This Hybrid Bat Algorithm (HBA) has been tested on a standard set of benchmark functions taken from literature. Our results of numerical experimental show that the proposed HBA can significantly improve the performance of the original bat algorithm, which can be very useful for the future.

The structure of the paper is as follows. In Section 2, the original bat algorithm together with differential evolution algorithm are introduced. In line with this, some biological foundations of bat behaviour are explained. Section 3 describes our proposed novel approach of hybridizing the bat algorithm with differential evolution strategies. Section 4 illustrates experiments and discusses the results. At the end of the paper, we conclude with future directions and developments with HBA.

2 BAT ALGORITHM

Bat algorithm has been developed by Xin-She Yang in 2010 [35]. The algorithm exploits the so called echolocation of bats. Bats use sonar echoes to detect and avoid obstacles. It is generally known, that sound pulses are transformed to frequency which reflects from obstacle. Bats can use time delay from emission to reflection and use it for navigation. They typically emit short loud, sound impulses. The pulse rate is usually defined as 10 to 20 times per second. After hitting and reflecting, bats transform their own pulse to useful information to gauge how far away the prey is. Bats are using wavelengths, that vary from range [0.7,17] mm or inbound frequencies [20,500] kHz. By implementation, pulse frequency and rate has to be defined. Pulse rate can be simply determined from range 0 to 1, where 0 means there is no emission and by 1, bats are emitting maximum [14], [31], [37].

This behaviour can be used to formulate the new bat algorithm. Yang [35] used three generalized rules for bat algorithms:

1) All bats use echolocation to sense distance, and they also guess the difference between food/prey and background barriers in a some magical way.
2) Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target.
3) Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

Algorithm 1 Original Bat Algorithm

```
1: Objective function f(x), x = (x_1, ..., x_d)^T
2: Initialize the bat population x_i and v_i for i = 1 ... n
3: Define pulse frequency Q_i ∈ [Q_min, Q_max]
4: Initialize pulse rates r_i and the loudness A_i
5: while (t < T_max) // number of iterations
6:   Generate new solutions by adjusting frequency, and
7:   updating velocities and locations/solutions [Eq.(2) to (4)]
8:   if(rand(0,1) > r_i )
9:     Select a solution among the best solutions
10:    Generate a local solution around the best solution
11:  end if
12:  Generate a new solution by flying randomly
13:  if(rand(0,1) < A_i and f(x_i) < f(x))
14:    Accept the new solutions
15:    Increase r_i and reduce A_i
16:  end if
17:  Rank the bats and find the current best
18: end while
19: Postprocess results and visualization
```

The original bat algorithm is illustrated in Algorithm 1. In this algorithm bat behaviour is captured into fitness function of problem to be solved. It consists of the following components:

- initialization (lines 2-4),
- generation of new solutions (lines 6-7),
- local search (lines 8-11),
- generation of a new solution by flying randomly (lines 12-16),
- find the current best solution.

Initialization of the bat population is performed randomly. Generating the new solutions is performed by moving virtual bats according the following equations:

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

where $U(0, 1)$ is a uniform distribution. A random walk with direct exploitation is used for local search that modifies the current best solution according to equation:

$$\mathbf{x^{(t)}} = \mathbf{best} + \epsilon A_i^{(t)}(2U(0,1) - 1), \tag{2}$$

where ϵ is the scaling factor, and $A_i^{(t)}$ the loudness. The local search is launched with the proximity depending

on the pulse rate r_i. The term in line 13 is similar to the simulated annealing behavior, where the new solution is accepted with some proximity depending on parameter A_i. In line with this, the rate of pulse emission r_i increases and the loudness A_i decreases. 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 with following equations:

$$A_i^{(t+1)} = \alpha A_i^{(t)}, \quad r_i^{(t)} = r_i^{(0)}[1 - exp(-\gamma\epsilon)], \quad (3)$$

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

3 Differential evolution

Differential evolution(DE)[29][6] is a technique for optimization which was introduced by Storn and Price in 1995. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand.

DE supports a differential mutation, a differential crossover and a differential selection. In particular, the differential mutation randomly selects two solutions and adds a scaled difference between these to the third solution. This mutation can be expressed as follows:

$$u_i^{(t)} = w_{r0}^{(t)} + F \cdot (w_{r1}^{(t)} - w_{r2}^{(t)}), \quad \text{for } i = 1 \dots NP, \ (4)$$

where $F \in [0.1, 1.0]$ denotes the scaling factor as a positive real number that scales the rate of modification while $r0,\ r1,\ r2$ are randomly selected vectors in the interval $1 \dots NP$.

Uniform crossover is employed as a differential crossover by the DE. The trial vector is built out of parameter values that have been copied from two different solutions. Mathematically, this crossover can be expressed as follows:

$$z_{i,j} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0,1) \leq CR \vee j = j_{rand}, \\ w_{i,j}^{(t)} & \text{otherwise}, \end{cases} \quad (5)$$

where $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution. Note, the relation $j = j_{rand}$ assures that the trial vector is different from the original solution $Y^{(t)}$.

Mathematically, differential selection can be expressed as follows:

$$w_i^{(t+1)} = \begin{cases} z_i^{(t)} & \text{if } f(Z^{(t)}) \leq f(Y_i^{(t)}), \\ w_i^{(t)} & \text{otherwise}. \end{cases} \quad (6)$$

In technical sense, crossover and mutation can be performed on many ways in differential evolution. Therefore, a specific notation was used to describe a variety of these methods (also strategies) in general. For example, "DE/rand/1/bin" denotes that the base vector is randomly selected, 1 vector difference is added to it, and the number of modified parameters in mutation vector follows binomial distribution.

4 Hybrid Bat Algorithm

As we mentioned before, a new bat algorithm, called Hybrid Bat Algorithm (HBA) is proposed in this paper. That is, the original bat algorithm was hybridized using the differential evolution strategies. The pseudo-code of the HBA is illustrated in Algorithm 2.

Algorithm 2 Hybrid Bat Algorithm

1: Objective function $f(x)$, $x = (x_1, ..., x_d)^T$
2: Initialize the bat population x_i and v_i for $i = 1 \dots n$
3: Define pulse frequency $Q_i \in [Q_{min}, Q_{max}]$
4: Initialize pulse rates r_i and the loudness A_i
5: while $(t < T_{max})$ // number of iterations
6: Generate new solutions by adjusting frequency, and
7: updating velocities and locations/solutions [Eq.(2) to (4)]
8: if$(rand(0,1) > r_i)$
9: **Modify the solution using "DE/rand/1/bin"**
10: end if
11: Generate a new solution by flying randomly
12: if$(rand(0,1) < A_i \ and \ f(x_i) < f(x))$
13: Accept the new solutions
14: Increase r_i and reduce A_i
15: end if
16: Rank the bats and find the current best
17: end while
18: Postprocess results and visualization

As a result, HBA differs from the original BA in lines 9, where solution is modified using "DE/rand/1/bin" strategy.

5 Experiments and results

A goal of experiments was to show that HBA significantly improves the results of the original BA. In line with this, two bat algorithms were implemented according to specifications in Algorithms 1 and 2 so that a well-selected set of test functions in the literature are used for optimization benchmarks.

Parameters of both bat algorithms were the same. Dimension of the problem has a crucial impact on the results of optimization. In order to test how the dimension influences on the results, three different sets of dimensions were taken into account, i.e., $D = 10$, $D = 20$, and $D = 30$. The functions with dimension $D = 10$ were limited to 1,000 maximal number of generations, the functions with dimension $D = 20$ twice as much, while the functions with dimension $D = 30$ to

3,000. The initial loudness was set to $A_0 = 0.5$ same as the initial pulse rate ($r_0 = 0.5$). The frequency was taken from interval $Q_i \in [0.0, 2.0]$. Algorithms optimized each function 25 times and results were measured according to the best, worst, mean, and medium values in these runs. In addition, the standard deviation of mean values were calculated as well.

5.1 Test suite

Test suite consists of five standard function taken from the literature [32]. Functions in this test suite are represented in the rest of paper.

5.1.1 Griewangk's function: The aim of the function is overcoming failures, that are optimizing each variable independently. This function is multimodal, since the number of local optima increases with the dimensionality. After sufficiently high dimensionalities ($n > 30$), multimodality seems to disappear and the problem seems unimodal.

$$f_1(\vec{x}) = -\prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{D} \frac{x_i^2}{4000} + 1, \qquad (7)$$

where $-600 \leq x_i \leq 600$. The function has the global minimum at 0.

5.1.2 Rosenbrock's function: The Rosenbrock function, similarly to Rastrigin's has its value 0 at global minimum. The global optimum is located inside a parabolic, narrow shaped flat valley. Variables are strongly dependent from each other, since it is difficult to converge the global optimum.

$$f_2(\vec{x_i}) = \sum_{i=1}^{D-1} 100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \qquad (8)$$

where $-15.00 \leq x_i \leq 15.00$.

5.1.3 Sphere function:

$$f_3(\vec{x_i}) = \sum_{i=1}^{D} x_i^2, \qquad (9)$$

where $-15.00 \leq x_i \leq 15.00$.

5.1.4 Rastrigin's function: Based of the Sphere function, Rastrigin function adds cosine modulation to create many local minima. Because of this feature, the function is multimodal. Its global minimum is at the value 0.

$$f_4(\vec{x_i}) = n * 10 + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)), \qquad (10)$$

where $-15.00 \leq x_i \leq 15.00$.

5.1.5 Ackley's function: The complexity of this function is moderated, since there is exponential term that covers its surface with numerous local minima. It is based on the gradient slope. Only the algorithm that uses the gradient steepest descent will be trapped in a local optima. Search strategy, analyzing wider area, will be able to cross the valley among the optima and achieve better results.

$$f_5(\mathbf{x}) = \sum_{i=1}^{n-1} [20 + e - 20e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))}], \qquad (11)$$

where $-32.00 \leq x_i \leq 32.00$. The global minimum of this function is at 0.

5.2 PC configuration

Configuration of PC, on which experiments have been executed was as follows:

- HP Pavilion g4,
- processor Intel(R) Core(TM) i5 @ 2.40 GHz,
- memory 8 GB,
- implemented in C++.

5.3 The results

The results of our extensive numerical experiments can be summarized in Table 1. The table represents results of BA and HBA algorithms (column 1) solving the test suite of five functions (denoted as f_1, f_2, f_3, f_4, f_5) with dimensions $D = 10$, 20 and 30, respectively, according to best, worst, mean, median, and standard deviation values.

The results of HBA show that this algorithm significantly improved the results of original BA according to almost all measures except the standard deviation in some cases (e.g., by Ackley function). The statistical analysis of the results has not been performed because these are evident better by HBA than by BA.

In order to observe how the results of both algorithms (i.e., BA and HBA) modified with the dimensions of the functions, a mean value of functions f_1 and f_3 with dimensions $D = 10$, $D = 20$, and $D = 30$ are presented in Figs. 1-3. The logarithmic scale is used to display the mean value on y-axis. Higher the mean value, more difficult the function is to solve.

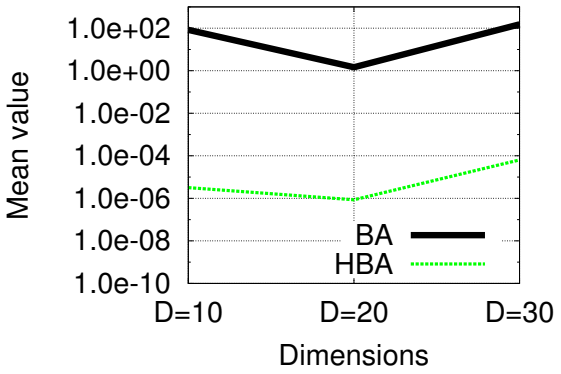

Figure 1. Mean value of function f_1 with various dimensions.

From Fig. 1 it can be seen that the best results are obtained by optimizing the function f_1 with the

Table 1. The results of experiments

Alg.	D	Value	f_1	f_2	f_3	f_4	f_5
BA	10	Best	3.29E+01	1.07E+04	5.33E+01	6.07E+01	1.37E+01
		Worst	1.73E+02	1.58E+06	3.11E+02	5.57E+02	2.00E+01
		Mean	8.30E+01	5.53E+05	1.44E+02	2.27E+02	1.75E+01
		Median	3.91E+01	4.69E+05	6.44E+01	1.06E+02	1.68E+00
		StDev	6.94E+01	4.71E+05	1.48E+02	2.17E+02	1.73E+01
	20	Best	8.77E+01	3.41E+02	2.24E+02	7.28E+01	2.15E+02
		Worst	1.43E+02	1.02E+03	5.72E+02	2.02E+02	5.87E+02
		Mean	1.46E+00	6.87E+02	3.56E+02	1.82E+02	3.38E+02
		Median	1.90E+05	3.16E+06	1.08E+06	9.20E+05	7.50E+05
		StDev	1.64E+01	2.00E+01	1.85E+01	1.21E+00	1.80E+01
	30	Best	1.58E+02	4.95E+02	3.29E+02	8.74E+01	3.39E+02
		Worst	4.18E+02	1.67E+03	7.80E+02	2.64E+02	7.82E+02
		Mean	1.51E+02	1.01E+03	5.17E+02	2.12E+02	4.67E+02
		Median	4.66E+05	6.23E+06	2.10E+06	1.26E+06	2.06E+06
		StDev	1.52E+01	2.00E+01	1.79E+01	1.25E+00	1.76E+01
HBA	10	Best	2.25E-09	6.34E-02	4.83E-09	5.12E+00	6.31E-04
		Worst	3.97E-05	5.10E+02	2.89E-03	2.38E+01	2.00E+01
		Mean	3.18E-06	6.22E+01	1.26E-04	1.55E+01	1.16E+01
		Median	8.66E-06	1.15E+02	5.66E-04	4.46E+00	9.26E+00
		StDev	1.14E-07	7.73E+00	1.66E-07	1.69E+01	1.78E+01
	20	Best	1.01E-07	9.73E-03	4.83E-04	1.89E-03	3.70E-05
		Worst	2.96E+01	9.24E+01	5.47E+01	1.77E+01	5.48E+01
		Mean	8.56E-07	1.10E-01	5.87E-03	2.18E-02	3.82E-05
		Median	3.60E+01	1.44E+03	2.53E+02	3.10E+02	1.41E+02
		StDev	2.17E+00	2.00E+01	1.60E+01	6.18E+00	1.95E+01
	30	Best	6.38E-06	8.28E+00	3.37E-01	1.62E+00	5.43E-04
		Worst	3.57E+01	2.17E+02	9.97E+01	3.98E+01	9.85E+01
		Mean	6.42E-05	6.59E+01	3.09E+00	1.29E+01	2.53E-03
		Median	5.99E+01	4.00E+03	7.67E+02	1.26E+03	2.15E+02
		StDev	3.12E+00	2.00E+01	1.72E+01	5.03E+00	1.94E+01

dimension $D = 20$, while the worst by optimizing the same function with the highest dimension $D = 30$.

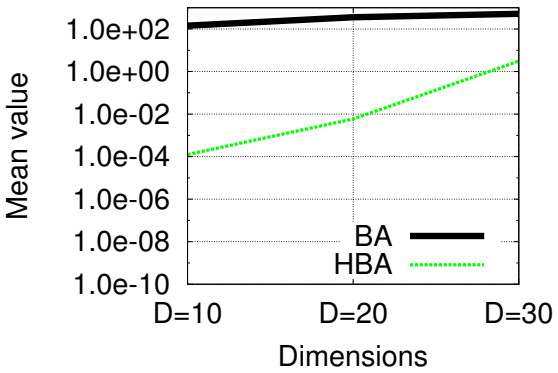

Figure 2. Mean value of function f_3 with various dimensions.

Interestingly, the mean value of a function f_5 with dimension $D = 10$ is the most difficult for the HBA algorithm, while the same function with dimension $D = 20$ is the easiest to solve. In contrast, the results of the original BA algorithm showed that increasing the dimensions also the results become worse.

As can be seen from Fig. 2, difficulty to solve the function f_3 is increased with increasing the dimensionality of the problem. As a result, the most difficult function to solve is the function f_3 with dimension $D = 30$.

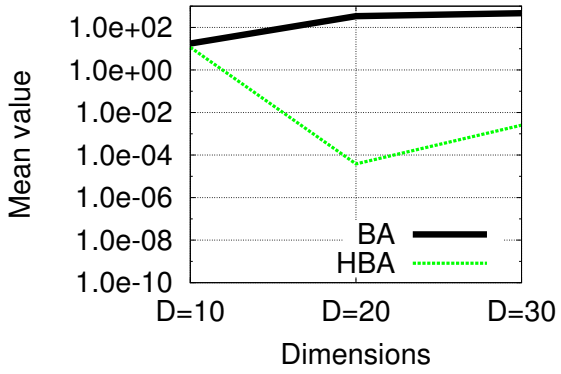

Figure 3. Mean value of function f_5 with various dimensions.

When comparing the results of the BA with the HBA, it can be observed that results of HBA significantly outperformed the results of the original BA algorithm by optimizing the functions f_1, f_3, and f_5. Functions f_2 and f_4 by HBA are also better, but the difference is not outstanding.

6 Conclusion

In this paper, we have improved the bat algorithm by developing a new variant, called hybrid bat algorithm. This new HBA is a hybrid of BA with DE strategies.

Experiments has shown that this algorithm improves significantly the original version of the bat algorithm. In the future, hybrid bat algorithm would be tested on large-scale global optimization. We will also do more extensive testing using more diverse test function sets, together with a detailed parametric study.

REFERENCES

[1] BV Babu and R. Angira. Modified differential evolution (mde) for optimization of non-linear chemical processes. *Computers & chemical engineering*, 30(6):989–1002, 2006.

[2] Thomas Bäck. *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 1996.

[3] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, 2006.

[4] J. Brest, A. Zamuda, I. Fister, and M.S. MaucĚŠec. Large scale global optimization using self-adaptive differential evolution algorithm. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[5] Charles Darwin. *The origin of species*. John Murray, London, UK, 1859.

[6] S. Das and P.N. Suganthan. Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*, 15(1):4–31, 2011.

[7] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, 2003.

[8] I. Fister and J. Brest. Using differential evolution for the graph coloring. In *Differential Evolution (SDE), 2011 IEEE Symposium on*, pages 1–7. IEEE, 2011.

[9] I. Fister, I. Fister, and J. Brest. A hybrid artificial bee colony algorithm for graph 3-coloring. *Swarm and Evolutionary Computation*, pages 66–74, 2012.

[10] I. Fister, I. Fister Jr, and J.B.V. Zumer. Memetic artificial bee colony algorithm for large-scale global optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[11] I. Fister, M. Mernik, and B. Filipič. Graph 3-coloring with a hybrid self-adaptive evolutionary algorithm. *Computational optimization and applications*, pages 1–32, 2012.

[12] I. Fister Jr, X.S. Yang, I. Fister, and J. Brest. Memetic firefly algorithm for combinatorial optimization. *arXiv preprint arXiv:1204.5165*, 2012.

[13] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA, 1966.

[14] A.H. Gandomi, X.S. Yang, A.H. Alavi, and S. Talatahari. Bat algorithm for constrained optimization tasks. *Neural Computing & Applications*, pages 1–17, 2012.

[15] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, MA, 1996.

[16] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[17] N. Jin and Y. Rahmat-Samii. Advances in particle swarm optimization for antenna designs: Real-number, binary, single-objective and multiobjective implementations. *Antennas and Propagation, IEEE Transactions on*, 55(3):556–567, 2007.

[18] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[19] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.

[20] P. Korošec, J. Šilc, and B. Filipič. The differential ant-stigmergy algorithm. *Information Sciences*, 2010.

[21] John R. Koza. *Genetic programming 2 - automatic discovery of reusable programs*. Complex adaptive systems. MIT Press, 1994.

[22] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, 6(4):333–346, 2002.

[23] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33(1–2):61–106, 2010.

[24] Q.K. Pan, M. Fatih Tasgetiren, P.N. Suganthan, and T.J. Chua. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information sciences*, 181(12):2455–2468, 2011.

[25] RS Parpinelli and HS Lopes. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16, 2011.

[26] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, 6(4):321–332, 2002.

[27] J. Robinson and Y. Rahmat-Samii. Particle swarm optimization in electromagnetics. *Antennas and Propagation, IEEE Transactions on*, 52(2):397–407, 2004.

[28] Y. Shi, H. Teng, and Z. Li. Cooperative co-evolutionary differential evolution for function optimization. *Advances in natural computation*, pages 428–428, 2005.

[29] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[30] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, and A.H.L. Chen. A discrete artificial bee colony algorithm for the permutation flow shop scheduling problem with total flowtime criterion. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[31] P.W. Tsai, J.S. Pan, B.Y. Liao, M.J. Tsai, and V. Istanda. Bat algorithm inspired algorithm for solving numerical optimization problems. *Applied Mechanics and Materials*, 148:134–137, 2012.

[32] X.-S. Yang. Appendix a: Test problems in optimization. In X.-S. Yang, editor, *Engineering Optimization*, pages 261–266. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2010.

[33] Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.

[34] Xin-She Yang and Suash Deb. Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4):330–343, 2010.

[35] X.S. Yang. A new metaheuristic bat-inspired algorithm. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74, 2010.

[36] X.S. Yang. Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3(5):267–274, 2011.

[37] X.S. Yang. Review of meta-heuristics and generalised evolutionary walk algorithm. *International Journal of Bio-Inspired Computation*, 3(2):77–84, 2011.

Iztok Fister Jr. was born in 1989, received his B.Sc. from Computer Science in 2011. Currently, he is working towards his M.Sc. degree. His research activities encompasses swarm intelligence, pervasive computing and programming languages.

Dušan Fister was born in 1993 and is a student of first Year of Mechatronics at the University of Maribor. His research activities encompasses GPS solutions, swarm intelligence and operating systems.

Xin-She Yang received his DPhil in Applied Mathematics from University of Oxford. Now 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).