Differential Evolution Strategies with Random Forest Regression in the Bat Algorithm*

Iztok Fister Jr.
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
iztok.fister@guest.arnes.si

Dušan Fister
University of Maribor
Faculty of Mechanical Engineering
Smetanova 17, 2000 Maribor
dusan.fister@uni-mb.si

Iztok Fister
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
iztok.fister@uni-mb.si

ABSTRACT

In this paper, we present a novel solution for the hybridization of the bat algorithm with differential evolution strategies and a random forests machine learning method. Extensive experiments and tests on standard benchmark functions have shown that these hybridized algorithms improved the original bat algorithm significantly.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Student workshop

Keywords

bat algorithm, differential evolution, random forests, machine learning

1. INTRODUCTION

Nature has been always a source of inspiration for mathematicians, computer scientists, and engineers constructing the automatic problem solver. Let us mention only two inspirations from the Nature that have led constructors to reach this goal: the *human brain* and the *Darwinian evolution* [4]. The former tries to solve problems by mimicking the functions of the human brain, and the latter by Darwinian survival of the fittest. As a result, the human brain has influenced the area of artificial intelligence (AI), while Darwinian evolution has influenced evolutionary algorithms (EA). Differential evolution [3, 5] (DE) is one of the most famous classes of evolutionary algorithms.

Additionally, *Swarm Intelligence* (SI) was inspired by collective behavior in self-organized and decentralized natural systems, e.g., ant colonies, bee hives, flocks of birds or schools of fish [1]. Swarm intelligence is an artificial intelligence (AI) discipline concerned with the design of intelligent multi-agent systems. In order to improve a performance of these general problem solvers, these algorithms have been hybridized with problem-specific knowledge by solving a definite problem. For example, an artificial bee colony optimization (ABC) algorithm has been hybridized with two different local search heuristics in order to balance an exploration and exploitation by an large scale function optimization in [7]. The local search heuristic also improved results of the firefly algorithm (FA) by solving the graph 3-coloring in [8]. Interestingly, the hybrid bat algorithm (HBA) [6] improved results of original algorithm, optimizing the suite of function, when hybridized with 'rand/1/bin' DE-strategy.

This paper proposes the hybrid bat with random forest [2] (HBARF). In place of original random walk direct exploitation (RWDE) [12], the HBA uses the 'rand/1/bin' DE strategy for local search step, while the HBARF employs an ensemble of ten DE-strategies and uses the random forest machine learning method to calculate the regression vector. Both hybrid bat algorithms were tested on a suite of five well-known functions and compared with the original BA. The results showed that both hybridizations significantly improved the results of the original BA.

2. BACKGROUND

2.1 Original Bat algorithm

The phenomenon of bats' echolocation can be briefly described as follows. Bats use the time delay between emission and detection of the echo, and the time and loudness difference between their two ears, in order to detect the distance, orientation and moving speed of their target prey. In echolocation, three parameters are important, i.e., the range of frequencies, the rate of pulse emission, and the loudness. The frequency of pulse emission depends on the prey size, i.e. the smaller the prey the higher the frequency. The rate of pulse emission is speed up when bats move near their prey. The loudness is higher when bats hunt their prey and lower when they are homeward bound.

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

The echolocation behavior of bats can be formulated as an new optimization algorithm, where this behavior is captured by an objective function of a specific optimization problem. However, the following approximation of bat behavior are idealized during development:

- All bats use echolocation to sense the distance to a target object.
- Bats fly randomly with the velocity v_i at position x_i, the frequency $Q_i \in [Q_{min}, Q_{max}]$ (also the wavelength λ_i), the rate of pulse emission $r_i \in [0, 1]$, and the loudness $A_i \in [A_0, A_{min}]$. The frequency (and wavelength) can be adjusted depending on the proximity of their target.
- The loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

The original bat algorithm is illustrated in Algorithm 1, where the population of virtual bats consists of NP real-valued D-dimensional vectors representing solutions. In this algorithm, bat behavior is captured in the fitness function of the problem to be solved. It consists of the following components: initialization (lines 2-4), generation of new solutions by moving the virtual bats (lines 6-7), local search (lines 8-11), and acceptance of the new solution with some proximity (lines 12-15).

Algorithm 1 Original Bat Algorithm

1: Objective function $f(x_i)$, $x_i = (x_{i1}, ..., x_{iD})^T$
2: Initialize the bat population x_i and velocities v_i for $i = 1 \dots NP$
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. 1 to 3]
8: if$(rand(0, 1) > r_i)$
9: Select the best solution in the current population
10: Generate a local solution around the best solution
11: end if
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

The movement of virtual bats obeys the following equations:

$$Q_i^{(t)} = Q_{min} + (Q_{max} - Q_{min})N(0, 1), \quad (1)$$

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^t + (\mathbf{x}_i^t - \mathbf{x}^*)Q_i^{(t)}, \quad (2)$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t)}. \quad (3)$$

The local search part implements the kind of random walk with direct exploitation according to the equation:

$$\mathbf{x^{(t)}} = \mathbf{x}^* + \epsilon A_i^{(t)} \mathbf{v}_i^{(t)}, \quad (4)$$

where $N(0, 1)$ denotes the random generated number drawn from interval $[-1, 1]$, ϵ is the scaling factor, $\mathbf{x}^*$ the current best solution and $A_i^{(t)}$ the loudness.

The loudness $A_i^{(t)}$ and the rate of pulse emission $r_i^{(t)}$ can be changed during the search process. Naturally, the loudness decreases and the rate of pulse emission increases when a bat finds its prey. This characteristic can be formulated in the bat algorithm with the following equations:

$$A_i^{(t+1)} = \alpha A_i^{(t)}, \quad r_i^{(t)} = r_i^{(0)}[1 - \exp(-\gamma\epsilon)], \quad (5)$$

where α and γ are constants.

2.2 Differential evolution

Differential evolution (DE) [3] is an evolutionary algorithm appropriate for optimization that has been introduced by Storn and Price in 1995 [13]. The 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:

$$u_i^{(t)} = x_{r0}^{(t)} + F \cdot (x_{r1}^{(t)} - x_{r2}^{(t)}), \quad \text{for } i = 1 \dots NP, \quad (6)$$

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 values 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:

$$z_{i,j} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0, 1) \leq CR \vee j = j_{rand}, \\ w_{i,j}^{(t)} & \text{otherwise}, \end{cases} \quad (7)$$

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, a differential selection can be expressed as follows:

$$w_i^{(t+1)} = \begin{cases} z_i^{(t)} & \text{if } f(Z^{(t)}) \leq f(Y_i^{(t)}), \\ w_i^{(t)} & \text{otherwise}. \end{cases} \quad (8)$$

In a technical sense, crossover and mutation can be performed in many ways in differential evolution. Therefore, a specific notation was used to describe the variety of these methods (also strategies) in general. For example, '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 the mutant vector follows a binomial distribution.

2.3 Random forest

The random forest is a machine learning method that is regarded by ensemble classifiers [2]. The ensemble consists of many decision trees. Each tree gives a classification and the forest chooses those classification that is most often classified. This algorithm was developed by Leo Breiman [2]. His ideas are also applicable to regression.

3. THE PROPOSED BAT ALGORITHM

The original bat algorithm implements two features: a local search, where the current best solution is used to try to make an improvement using the RWDE heuristic [12], and

Table 1: Ensemble of DE-strategies

Strategy	Expression
Best/1/Exp	$x_{i,j}^{(t+1)} = best_j^{(t)} + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$
Rand/1/Exp	$x_{i,j}^{(t+1)} = x_{r1,j}^{(t)} + F \cdot (x_{r2,j}^{(t)} - x_{r3,j}^{(t)})$
RandToBest/1/Exp	$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + F \cdot (best_i^{(t)} - x_{i,j}^{(t)}) + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$
Best/2/Exp	$x_{i,j}^{(t+1)} = best_i^{(t)} + F \cdot (x_{r1,i}^{(t)} + x_{r2,i}^{(t)} - x_{r3,i}^{(t)} - x_{r4,i}^{(t)})$
Rand/2/Exp	$x_{i,j}^{(t+1)} = x_{r1,i}^{(t)} + F \cdot (x_{r2,i}^{(t)} + x_{r3,i}^{(t)} - x_{r4,i}^{(t)} - x_{r5,i}^{(t)})$
Best/1/Bin	$x_{j,i}^{(t+1)} = best_i^{(t)} + F \cdot (x_{r1,i}^{(t)} - x_{r2,i}^{(t)})$
Rand/1/Bin	$x_{j,i}^{(t+1)} = x_{r1,j}^{(t)} + F \cdot (x_{r2,j}^{(t)} - x_{r3,j}^{(t)})$
RandToBest/1/Bin	$x_{j,i}^{(t+1)} = x_{i,j}^{(t)} + F \cdot (best_i^{(t)} - x_{i,j}^{(t)}) + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$
Best/2/Bin	$x_{j,i}^{(t+1)} = best_i^{(t)} + F \cdot (x_{r1,i}^{(t)} + x_{r2,i}^{(t)} - x_{r3,i}^{(t)} - x_{r4,i}^{(t)})$
Rand/2/Bin	$x_{j,i}^{(t+1)} = x_{r1,i}^{(t)} + F \cdot (x_{r2,i}^{(t)} + x_{r3,i}^{(t)} - x_{r4,i}^{(t)} - x_{r5,i}^{(t)})$

Algorithm 2 Modification in Hybrid Bat Algorithm with Random Forest

1: if($rand(0,1) > r_i$)
2: Select the best solution in the current population
3: Generate a test set using the 10 DE-strategies from the best solution
4: Generate a valid set using the candidate solution
5: Apply random forest regression using 10 estimators
6: end if

the replacement of the original solution with the better candidate solution according to the predefined proximity (similar to simulated annealing [9]). In this study, the local search part of the BA algorithm (i.e., lines 8-11 in Algorithm 1) was taken into account. In fact, lines 8-11 in the original algorithm are replaced with lines 1-6 from Algorithm 2.

As can be seen from Algorithm 2, HBARF defines an ensemble of ten DE-strategies (Table 1) that creates ten different solution from the candidate solution, each obtained with another strategy [10]. For the random forest algorithm, these ten solutions represent the training set, while the candidate solution acts as a validation set. Based on these two sets, the random forest regression is launched, which predicts the new candidate solution.

Note that the original BA and HBA algorithms were implemented in the C++ programming language, while the HBARF algorithm combines the implementation of the BA in C++ together with the RF implementation in Python using the scikit-learn python library [11].

4. EXPERIMENTS AND RESULTS

The goal of our experimental work was to show how the hybridization of the original BA with an ensemble of DE-strategies and random forest regression influences its performance. Therefore, the original BA was compared with HBA and HBARF. The algorithms were tested by optimizing the suite of five well-known functions of dimension 10 (Table 2) taken from existing literature [14]. The same suite was also used by HBA in [6]. The task of this optimization is to find the global optimum of each function. Note that all of the functions have the global optimum at the value zero.

The BA parameters in the experiments were set as follows: loudness $A_0 = 0.5$, pulse rate $r = 0.5$, frequencies $Q_i^{(t)} \in [0.0, 2.0]$, while the DE with the strategy 'rand/1/bin' operates with the following parameters: $F = 0.5$, $CR = 0.9$. The random forest regression has been run with 10 estimators. All algorithms have been terminated after $T_{max} = 1,000$ generations. Population size has been set to 10 (i.e., number of fitness evaluations was 10,000 for all algorithms in the experiments). All functions were optimized 25 times.

The results of the experiments are illustrated in Table 3, where the best results according to the best, worst, mean, median, and standard deviation values obtained by the algorithms BA, HBA, and HBARF, when optimizing the functions f_1-f_5 are denoted in bold.

From Table 3 it can be seen that both hybridized algorithms improved the results of the BA. As can be seen from Figure 1, where the Friedman's non-parametric test was performed, this improvement was significant. In this figure, two algorithms are significantly different, if their confidence intervals represented as lines do not overlap. On the other hand, HBARF improved HBA. Comparing the results of both hybridized algorithms with each other, it can be concluded that the HBARF achieved the best results according to the mean and standard deviation values, while the results of HBA are more dissipated around a mean value.

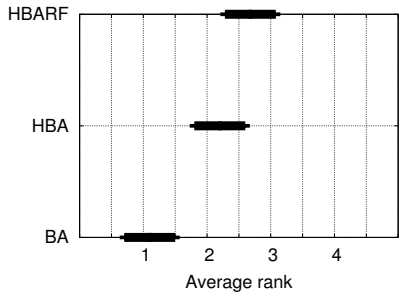

Figure 1: Friedman's non-parametric test

5. CONCLUSION

In this paper two novel ideas for hybridization and improving the bat algorithm were presented. Experiments showed that the random forest method might also be very suitable for evolutionary computation and the swarm intelligence world.

In the future we would like to apply random forest to pure differential evolution and try to create an random forest on ensemble of DE strategies. Moreover, new machine learning methods for using ensemble of DE strategies would be explored in the future work (e.g., Extremely Randomized Trees).

Table 2: Test suite

f	Function	Definition	Range
f_1	Griewangk's function	$F(\vec{x}) = -\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{n} \frac{x_i^2}{4000} + 1$	$-600 \leq x_i \leq 600$
f_2	Rosenbrock's function	$F(\vec{x_i}) = \sum_{i=1}^{n-1} 100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$-15.00 \leq x_i \leq 15.00$
f_3	Sphere function	$F(\vec{x_i}) = \sum_{i=1}^{n} x_i^2$	$-100.00 \leq x_i \leq 100.00$
f_4	Rastrigin's function	$F(\vec{x_i}) = n * 10 + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$-15.00 \leq x_i \leq 15.00$
f_5	Ackley's function	$F(\vec{x}) = \sum_{i=1}^{n-1} \left(20 + e - 20e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))}\right)$	$-32.00 \leq x_i \leq 32.00$

Table 3: The results of the 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
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
HBARF	10	Best	**1.44E-11**	**5.00E-05**	2.36E-06	**3.09E-05**	7.21E-04
		Worst	6.35E-04	**1.99E+00**	5.90E-02	**1.02E+01**	**3.53E-01**
		Mean	3.92E-05	**2.64E-01**	5.92E-03	**5.92E-01**	**3.14E-02**
		Median	**1.05E-06**	**1.58E-01**	**4.50E-04**	**1.07E-01**	**1.27E-02**
		StDev	**1.25E-04**	**5.44E-01**	1.22E-02	**2.00E+00**	**6.76E-02**

6. REFERENCES

[1] C. Blum and X. Li. Swarm intelligence in optimization. In C. Blum and D. Merkle, editors, *Swarm Intelligence: Introduction and Applications*, pages 43–86. Springer Verlag, Berlin, 2008.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

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

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

[5] I. Fister and J. Brest. Using differential evolution for the graph coloring. pages 150–156, 2011.

[6] I. Fister, D. Fister, and X.-S. Yang. A hybrid bat algorithm. *Electrotechnical review*, 2013, In press.

[7] I. Fister, I. Fister, J. Brest, and V. Žumer. Memetic artificial bee colony algorithm for large-scale global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[8] I. Fister, X.-S. Yang, I. Fister, and J. Brest. Memetic firefly algorithm for combinatorial optimization. In B. Filipič and J. Šilc, editors, *Bioinspired optimization methods and their applications : proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications - BIOMA 2012*, pages 75–86. Jožef Stefan Institute, 2012.

[9] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[10] R. Mallipeddi, P. Suganthan, Q. Pan, and M. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696, 2011.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] S. Rao. *Engineering optimization: theory and practice*. John Willey & Sons, New Jersey, 2009.

[13] 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.

[14] X.-S. Yang. Appendix A: Test problems in optimization. In X.-S. Yang, editor, *Engineering Optimization: An Introduction with Metaheuristic Applications*, pages 261–266. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2010.