NEW EVOLUTIONARY OPTIMIZATION TECHNIQUES AND TEST FUNCTIONS FOR THEIR EVALUATION

Gábor Zoltán Marcsák[1], Csaba Barcsák[2], Károly Jármai[3],
[1] MSc. student, [2] software engineer, Evosoft Ltd., [3] Professor, *University of Miskolc, Egyetemváros, Hungary*

1. INTRODUCTION

The process of optimization is finding the best solution to a given problem, when the amount of available resources is often restricted. Despite the rapid development of computer science, most optimization problems can't be solved by evaluating all feasible solutions. For example, the class of NP-hard problems, such as the Traveling salesman problem (also known as TSP), might have an enormously large search space, which requires exponential computation time to be fully explored. To solve these kinds of problems, many heuristic algorithms have been developed. Heuristic algorithms can find approximate solutions, even when the search space is excessively huge. In this paper, we benchmarked twelve optimization techniques, to compare their efficiency in finding the global minima of different continuous mathematical test functions. Mathematical function optimization is very important, because most real world optimization problems can be modelled in this general framework. Numerous mathematical test functions can be found in the literature, additionally we created a software solution, which utilizes a novel method to construct customized test functions.

2. BENCHMARK PROBLEMS

As mentioned before, a lot of mathematical test functions can be found in the literature [1]. The complexity of test functions is determined by the number of variables and the number and distribution of local extremes. We studied continuous test functions with two variables, since those problems can be plotted as 3d surfaces. Table 1 summarizes the ten test functions we used in alphabetical order. The Ackley's function has a nearly flat outer region, and a large valley at its centre. This widely used multimodal test function can easily trap heuristic algorithms at one of its local optima. De Jong's function is a very simple convex, unimodal benchmark problem. Drop-Wave function is very complex, with expanding ripples, like when an object is dropped into liquid surface. Easom's function is unimodal like De Jong's, however more complicated, because the global optima is relatively small compared to the search space. Griewangk's function looks similar to De Jong's either, but it has a rugged surface with many regularly distributed local optima. Matyas's function is a plate shaped problem, it doesn't have any local extremes, only the global one, which is relatively easy to find. However, convergence to the global optima is difficult, so that this is a great benchmark problem to measure the accuracy and convergence rate of search algorithms. Rastrigin's function, also known as egg holder is a widely used, highly multimodal problem with regularly distributed local extremes. Rosenbrock's valley is unimodal,

the global minimum can be found in a narrow, parabolic valley. Schaffer's second function is an extremely noisy optimization problem, with a lot of local optima very close to each other. Last but not least, Three-hump camelback looks very much like Rosenbrock's valley, however it has two local extremes.

Table 1.
Numerical data of benchmark problems

Name	Definition	Search range and global optimum
Ackley's function (F1)	$f(x) = -20 \times \exp(-0{,}2 \times \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^2)}$ $- \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + \exp(1)$	$-32{,}768 \le x_i \le 32{,}768$ $x_i = 0, i = 1, \dots, n$ f(x)=0
De Jong's function (F2)	$f(x) = \sum_{i=1}^{n} x_i^2$	$-5{,}12 \le x_i \le 5{,}12$ $x_i = 0, i = 1, \dots, n$ f(x)=0
Drop-Wave function (F3)	$f(x) = -\frac{1 + \cos(12 \times \sqrt{x_1^2 + x_2^2}}{\frac{1}{2}(x_1^2 + x_2^2) + 2} + 1$	$-5{,}12 \le x_i \le 5{,}12$ $x_i = 0, i = 1, 2$ f(x)=0
Easom's function (F4)	$f(x) = -\cos(x_1) \times \cos(x_2)$ $\times \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2) + 1$	$-10 \le x_i \le 10$ $x_i = \pi, i = 1, 2$ f(x)=0
Griewangk's function (F5)	$f(x) = \frac{1}{4000} \times \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 \le x_i \le 600$ $x_i = 0, i = 1, \dots, n$ f(x)=0
Matyas's function (F6)	$f(x) = 0{,}26 \times (x_1^2 + x_2^2) - 0{,}48 \times x_1 \times x_2$	$-10 \le x_i \le 10$ $x_i = 0, i = 1, 2$ f(x)=0
Rastrigin's function (F7)	$f(x) = 10 \times n + \sum_{i=1}^{n}[x_i^2 - 10 \times \cos(2\pi x_i)]$	$-5{,}12 \le x_i \le 5{,}12$ $x_i = 0, i = 1, \dots, n$ f(x)=0
Rosenbrock's valley (F8)	$f(x) = \sum_{i=1}^{n-1}[100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$-2{,}048 \le x_i \le 2{,}048$ $x_i = 1, i = 1, \dots, n$ f(x)=0
Schaffer's N. 2. function (F9)	$f(x) = 0{,}5 + \frac{sin^2(x_1^2 - x_2^2) - 0{,}5}{[1 + 0{,}001 \times (x_1^2 + x_2^2)]^2}$	$-10 \le x_i \le 10$ $x_i = 0, i = 1, 2$ f(x)=0
Three-hump camelback (F10)	$f(x) = 2x_1^2 - 1{,}05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	$-2{,}048 \le x_i \le 2{,}048$ $x_i = 0, i = 1, 2$ f(x)=0

3. COMPOSITION OF TEST FUNCTIONS

In [2] the authors proposed a novel theoretical method to create more complex test functions from given basis functions. A practical framework has been described in [3], which extends the above theory. We have developed a software solution, which utilizes the practical framework to easily create customized arbitrarily difficult test problems.

3.1 Theoretical Method

In order to generate arbitrarily difficult complex test functions, the algorithm requires several input parameters:

- $[X_{\min}, X_{\max}]^D$: search range of the complex function
- D: number of dimensions
- $f_i(\theta)$: list of basic functions
- $[x_{\min}, x_{\max}]^D$: search range of the basic functions
- o_i: the position of the global optima for the i-th basic function
- bias (ψ): a vector to define the global optima. It allows the user to shift the optimum values of the basic functions.

To define the position of global and local optimums, we have to shift the value of the basic function's global optimum points. In order to achieve this, the basic functions have to be evaluated outside the defined search range. Therefore the given global optimum positions of the basic functions have to be independent of the search range. The complex test functions can be determined using the following formulas:

$$F(\theta) = \sum \left[w_i \left(\frac{\mathrm{f_i}(\vartheta_i)}{\mathrm{f_i}(o_i)} + bias_i \right) \right] \tag{1}$$

$$\vartheta_i = \frac{\theta - \left(X_{min} - x_{min_i} \kappa_i + \psi_i \right)}{\kappa_i} \tag{2}$$

Where w_i is the weighting function, which ensures to keep the predefined optimum positions and values. The closer we get to a basic functions global optimum position (o_i), the bigger weighting coefficient it gets. At the same time, the other basic functions get smaller weighting coefficients. We used three different types of weighting functions.

3.1.1 Euclidean distance-based weighting

The first weighting function is based on the Euclidean distance between the complex function's given point (ϑ) and the optimum points (o_i) of the basic functions.

$$w_i = \text{Euclidean\_distance}(\vartheta_\text{i}, o_\text{i}) \tag{3}$$

We have to normalize the distances:

$$w_i = \frac{w_i}{\sum w} \tag{4}$$

$$w_i = 1 - w_i \tag{5}$$

If $w_i \neq maximumDistance(w)$, and n is the number of basic functions:

$$w_i = \frac{1 - maximumDistance(w)}{(n-1)} \tag{6}$$

3.1.2 Gaussian weighting

Smoother edges can be achieved by using the Gaussian functions to determine weighting coefficients.

$$w_i = \text{e}^{\frac{(\vartheta_\text{i} - o_\text{i})^2}{6}} \tag{7}$$

3.1.3 Gabor-like weighting

We can create more difficult optimization problems if the weighting function generates noise. However we have to make sure not to shift the original optimum values. The weighting function should return values between 0 and 1, its global maxima should be at the complex function's global optimum.

$$w_i = \left| \prod_\text{k} \cos(\vartheta_\text{i}(\text{k})\tau_1)\, \text{e}^{-\frac{\sum_\text{k} \vartheta_\text{i}(\text{k})^2}{\tau_2}} \right| \tag{8}$$

Where $\vartheta_\text{i}(\text{k})$ is the k-th element of the ϑ_i vector, τ_1 is the noisiness parameter, and τ_2 is the convergence range. Increasing the τ_1 parameter results in more noise. If $w_i \neq maximumWeight(w)$, and n is the number of basic functions:

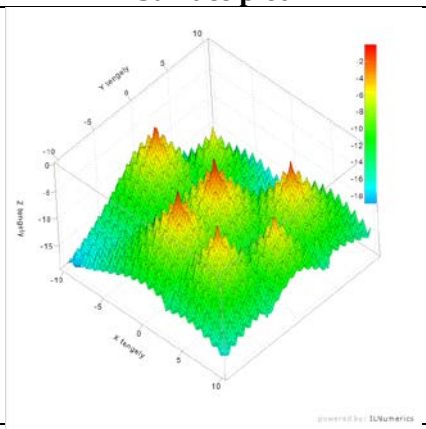
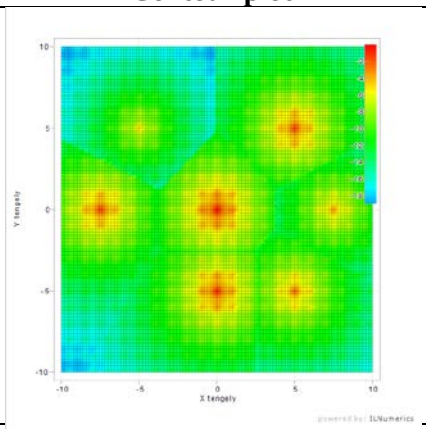
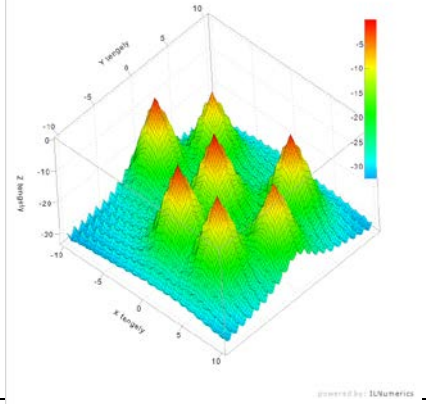
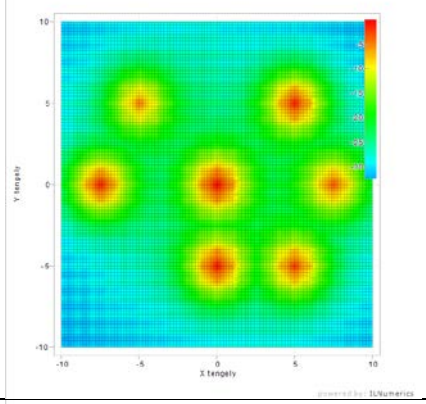
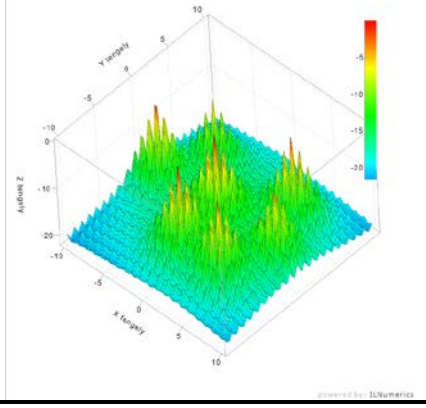
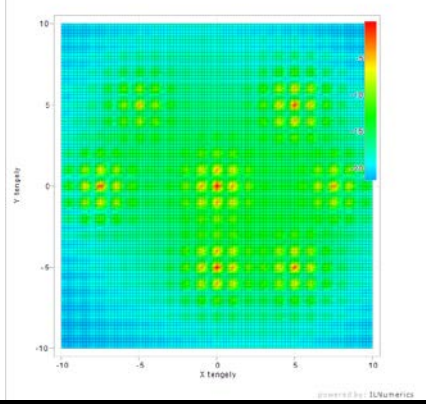
$$w_i = \frac{1 - maximumWeight(w)}{(n-1)} \tag{9}$$

3.2 Practical Example

We have generated a complex test function to demonstrate the method and expand the number of benchmark problems used in this paper. The input parameters are the following:

- $[X_{min}, X_{max}]^D$: the search range of the complex function is $[-10, 10]^D$
- D: the number of dimensions is 2
- $f_i(\theta)$: we used 7 basic functions, all of them are Ackey's functions (F1).
- $[x_{min}, x_{max}]^D$: since we shift the optimum points of the basic functions, we don't have to define the basic functions' search range.
- $o(x)$: the x coordinates for the basic functions: [7,5; 5; 5; 0; -7,5; -5; 0]
- $o(y)$: the y coordinates for the basic functions: [0; 5; -5; -5; 0; 5; 0]
- bias (ψ): a vector to shift the basic functions: [3; 1; 2; 0,5; 1; 4,5; 0]

Table 2.
Surface and contour plot of complex benchmark problems

Name	Surface plot	Contour plot
Complex function with Euclidean distance-based weighting (F11)		
Complex function with Gaussian weighting (F12)		
Complex function with Gabor-like weighting ($\tau_1 = 3, \tau_2 = 8$) (F13)		

In Table 3 you can see the generated Complex functions, which all consists of 7 Ackley's basic functions. The different characteristics of the three weighting functions are also observable. The basic functions global optimum values have been shifted, therefore the complex function's global optimum value is $f(x) = 0$, can be found at $x = 0$, $y = 0$ coordinates.

3.4 Novel software solution to easily generate complex test functions

We have developed a software solution which utilizes the above framework to easily create customized arbitrarily difficult test problems. All the coding was done in C#, the program uses the .NET Framework. The best feature of the software is the automatic source code generation. As the user builds the complex function in a visual editor, at the same time the program generates the function's C# source code. Therefore the newly created optimization problem can be used straight away.

4. MAIN CHARACTERISTICS OF HEURISTIC ALGORITHMS

As mentioned before, the biggest advantage of heuristic algorithms is that they can find approximate solutions, even when the search space is excessively huge. However, finding the global optimum cannot be guaranteed, since they don't evaluate all feasible solutions. A good heuristic algorithm has to maintain balance between local search and global search. On one hand, it has to explore the entire search space properly, on the other hand search around the current best positions efficiently. In other words, quickly find regions with quality solutions, and don't waste too much time in low quality areas. Most of the time, heuristic algorithms have stochastic behaviour. Ideally, the final solutions, through slightly different, will converge to the optimal solution of the given problem. However, the way heuristic algorithms get to the solution is always a bit different because of the stochastic factor. Nowadays a lot of nature inspired heuristic algorithms emerge. We benchmarked twelve optimization techniques, like evolutionary (Differential Evolution, Cultural Algorithm, Memetic Algorithm), physical (Simulated Annealing, Harmony Search), biological (Artificial Immune Network) and swarm intelligence (Bacterial Foraging, Bees Algorithm, Krill Herd, Particle Swarm) optimization methods. We made the algorithms' source code available on the Internet. The algorithms search for the global minima, if the Rastrigin's function in the source code [4].

4.1 Benchmarked heuristic algorithms

- *Artificial Immune Network* (AiNet) algorithm was developed by de Castro and Von Zuben to solve a clustering problem in 2000 [5]. According to its principles, the algorithm is related to the field of Artificial Immune Systems.
- *Bacterial Foraging Algorithm* (BFOA) was first described by Liu and Passino in 2002 [6]. It's a relatively new swarm intelligence search algorithm. These techniques use the collective intelligence of numerous homogenous individuals. In principle, an individual entity may not be able to solve a problem on its own.

However, if a large number of individuals form a group, the group's collective intelligence may be enough to solve the task. Bacterial Foraging is based on the foraging behavior of E. Coli bacteria colonies.

- *Bees Algorithm* (BA) was published by Pham in 2005 [7]. Primarily it was developed to search for the global optima of continuous mathematical functions. It belongs to the field of swarm intelligence procedures. The Bees Algorithm, as the name suggests, was inspired by the foraging behavior of bees.
- *Cultural Algorithm* (CA) was described by Reynolds in 1994 [8]. This evolutionary algorithm simulates the cultural evolution of human society.
- *Differential Evolution* (DE) algorithm was developed by Storn and Price in 1995. It belongs to the field of evolutionary algorithms. Differential Evolution is mainly based on Darwin's Theory of Evolution, because its main principle is natural selection [9].
- *Harmony Search* (HS) was published by Geem, Kim and Loganathan in 2001 [10]. It was inspired by Jazz musicians. When they start a musical performance, they adapt their music to the band, creating musical harmony. If a false sound occurs, the band makes modifications to improve their performance.
- *Krill Herd* (KH) algorithm is a novel swarm intelligence method, developed by Gandomi and Alami in 2012 [11]. It is inspired by the foraging behavior of the Antarctic krill (Euphausia superba). These krill search for food in dense swarms, the number of krill can be up to 10-30 thousand per cubic meters. On one hand, the swarm guarantees protection against predators, on the other hand krill can find food easier, because the swarm can scout larger areas.
- *Memetic Algorihm* (MA) was developed by Moscato in 1989 [12]. The algorithm simulates the creation and inheritance of cultural information among individuals. Meme is the basic unit of cultural information (an idea, discovery, etc), which name derives from the biological term gene.
- *Nelder-Mead* (NM) algorithm was named after its creators. Nelder and Mead created this heuristic in 1965 [13]. In the literature its often referred as Amoeba Method. Basically Nelder-Mead algorithm is a simplex search method [14].
- *Particle Swarm Optimization* (PSO) was developed by Eberhart and Kennedy. Nowadays its one of the most promising metaheuristic optimization algorithms. Particle Swarm's operation was inspired by the foraging movement of bird and fish swarms [15].
- *Random Search* (RS) algorithm, as the name suggests, is a simple random search algorithm. It takes any position in the search space with equal probability. The new solutions are always independent from the previous ones.
- *Simulated Annealing* (SA) method was described by Kirkpatrick, Gelatt and Vecchi in 1983. The operation of the algorithm is based on a physical phenomenon. In metallurgy certain materials gain beneficial properties when heated and then cooled under controlled conditions. The materials' crystal structure is transformed during the process, because the particles take more favorable positions. The heuristic algorithm emulates this process to search for better solutions to a given problem [16].

5. NUMERICAL EXPERIMENTS

Benchmarking heuristic optimization algorithms is a quite difficult and complex process. Due to the algorithms' stochastic factors, we had to use statistical methods to present satisfactory results. The heuristic algorithms did 50 Monte Carlo searches per test function. We specified the iteration limit for each search to be 100. The input parameters for the algorithms are determined based on the proposals found in the literature indicated.

5.1 Statistical results

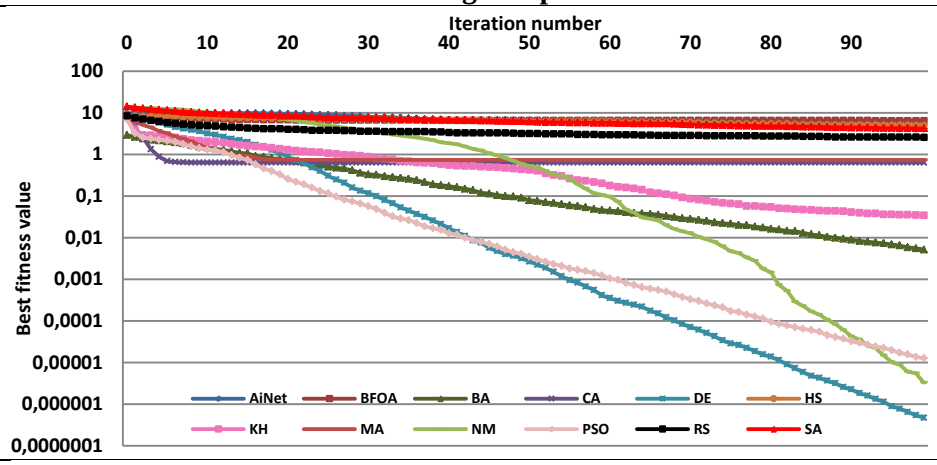
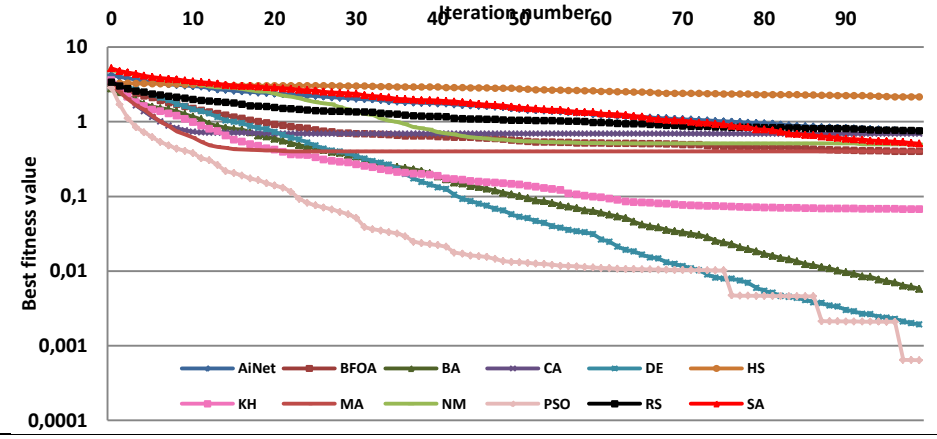
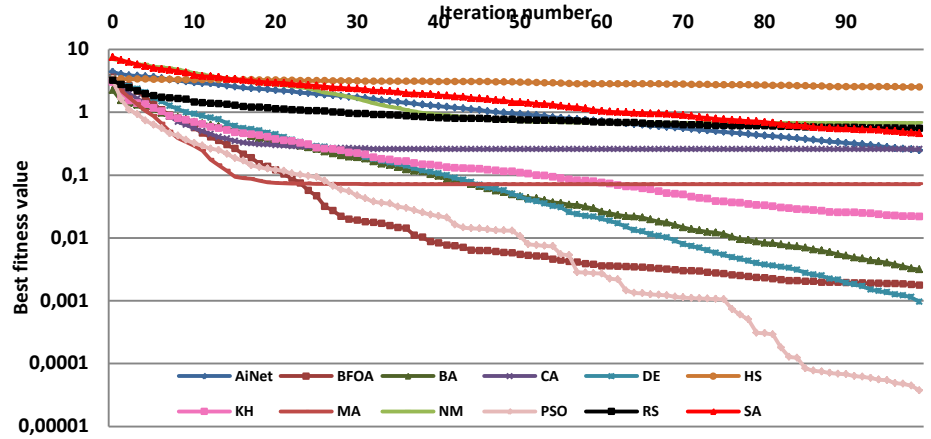
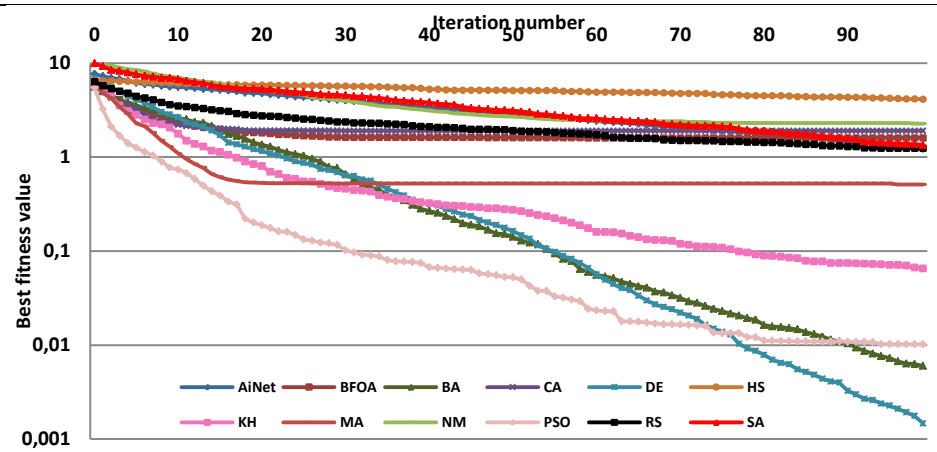
We created a statistical table, which summarizes the performance of algorithms broken down by test functions. Table 4 gives an overview regarding algorithms' efficiency and reliability. The values are normalized, so that the optima in each row is 0, and the maximum is 1. These are not the absolute minima found by each algorithm, but the average minima for 50 Monte Carlo simulations.

Table 3.
Mean normalized optimization results for thirteen benchmark functions.

	AiNet	BFOA	BA	CA	DE	HS	KH	MA	NM	PSO	RS	SA
F1	0,58	1	**0**	0,10	**0**	0,77	0,01	0,11	**0**	**0**	0,40	0,65
F2	0,12	**0**	**0**	0,29	**0**	1	**0**	0,01	**0**	**0**	0,08	0,03
F3	0,28	0,56	**0**	0,31	0,04	1	0,11	0,16	0,35	**0**	0,47	0,71
F4	0,28	0,56	**0**	0,31	0,04	1	0,11	0,16	0,35	**0**	0,47	0,71
F5	0,43	0,05	0,04	0,06	**0**	0,68	0,10	0,30	1	**0**	0,05	0,27
F6	0,08	0,17	**0**	0,01	**0**	0,13	**0**	0,01	0,01	**0**	0,04	1
F7	0,12	**0**	**0**	0,07	0,06	1	0,01	0,13	**0**	**0**	0,04	0,15
F8	0,09	**0**	0,05	0,08	0,01	0,82	0,01	1	**0**	**0**	0,07	0,01
F9	**0**	0,04	**0**	0,11	0,01	1	0,04	0,07	0,41	**0**	0,02	0,37
F10	**0**	**0**	**0**	0,05	0,02	1	**0**	0,03	0,29	**0**	0,01	0,07
F11	0,64	1	**0**	0,10	**0**	0,77	0,01	0,11	**0**	**0**	0,40	0,65
F12	0,10	**0**	**0**	0,10	**0**	1	0,01	0,03	0,27	**0**	0,22	0,18
F13	0,35	0,19	**0**	0,32	**0**	1	0,03	0,18	0,23	0,01	0,35	0,24
Σ	2	5	11	0	7	0	3	0	5	12	0	0

We used convergence plots in Table 5 to measure the convergence rate of search algorithms. Convergence rate shows how quickly the heuristic algorithms can find the optimum. The data points are the best fitness found in each iteration, averaged for 50 Monte Carlo simulations.

Table 4.
Convergence plots for Ackley's (F1) and Complex (F11, F12, F13) functions

Name	Convergence plot
Ackley's function (F1)	
Complex function with Euclidean distance-based weighting (F11)	
Complex function with Gaussian weighting (F12)	
Complex function with Gabor-like weighting ($\tau_1 = 3, \tau_2 = 8$) (F13)	

5.2 Benchmark conclusion

We used twelve search algorithms for the thirteen benchmark problems. The test functions had quite diverse characteristics, like unimodal and multimodal problems with varying number and distribution of local extremes. Three complex benchmark problems were constructed by us with the software we created. After we examined the significant amount of statistical data, we get a picture of the algorithms' overall performance. If we take a look at Table 4, we can see the superiority of swarm intelligence methods. Bees Algorithm (BA) and Particle Swarm Optimization (PSO) almost always found the global optima, even in case of very difficult and noisy functions. The third best algorithm was an evolutionary method, the Differential Evolution, however sometimes it trapped at local optima. To improve the performance of slowly converging methods (HS, SA), we should increase the number of iterations. The rate of convergence could be observed very well on the convergence plots of Table 5. The three best algorithms' rate of convergence is relatively fast, and this characteristic proved to be crucial for success. Furthermore, the convergence plots showed the weighting functions really affect the difficulty of complex functions. The averaged fitness function values revealed the Complex function with Euclidean distance-based weighting (F11) was a thousand times harder to solve then the basis function (F1). However the use of Gaussian weighing function resulted in smoother edges, which decreased the number and distribution of local extremes, making it easier to find the global optima for the search algorithms. Probably the hardest test function was Complex function with Gabor-like weighting, because the weighting function generated significant noise. In case of this test problem, the best algorithm was Differential Evolution, however its solution is ten times worse than the best algorithm of Euclidean distance-based weighting. As final conclusion, we can say that the complex functions with different weighting proved to be great benchmark problems.

6. SUMMARY

The field of numerical optimization is an always evolving science. Several new evolutionary optimization techniques appeared lately. We created a software solution, which provides practically endless possibilities to create arbitrarily difficult complex test functions. We benchmarked twelve optimization algorithms with thirteen test functions. In the future, we would like to create more difficult, self-made test problems, and benchmark further heuristic techniques. Based on the benchmark result, we will try to create novel, more efficient hybrid heuristic algorithms, which could be utilized in real-life structural and logistical optimization problems.

7. ACKNOWLEDGEMENTS

The research was supported by the TÁMOP 4.2.4.A/2-11-1-2012-0001 priority project entitled ‘National Excellence Program - Development and operation of domestic personnel support system for students and researchers, implemented

within the framework of a convergence program, supported by the European Union, co-financed by the European Social Fund. The research was supported also by the Hungarian Scientific Research Fund OTKA T 109860 project and was partially carried out in the framework of the Center of Excellence of Innovative Engineering Design and Technologies at the University of Miskolc.

8. REFERENCES

[1] MOLOGA M., SMUTNICKI C.: **Test functions for optimization needs**, 2005. pp. 1-10, www.bioinformaticslaboratory.nl

[2] LIANG J., SUGANTHAN N., DEB K.: **Novel composition test functions for numerical global optimization**, Swarm Intelligence Symposium, Proceedings, 2005. pp. 68-75.

[3] BARCSÁK CS., JÁRMAI K.: **Benchmark for testing evolutionary algorithms**, 10th World Congress on Structural and Multidisciplinary Optimization, May 19 -24, 2013, Orlando, Florida, USA

[4] **C# source code of heuristic algorithms**: https://drive.google.com/folderview?id=0BxE6yHbGFZBAOHBpN2VIV08yS0k&usp=sharing

[5] DE CASTRO L. N. and VON ZUBEN F. J.: **An evolutionary immune network for data clustering.** In Proceedings Sixth Brazilian Symposium on Neural Networks, IEEE Computer Society, 2000. pp. 84–89.

[6] LIU Y. and PASSINO K. M.: **Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviours**, Journal of Optimization Theory and Applications, 2002. pp. 603–628.

[7] PHAM D. T., GHANBARZADEH A., KOC E., OTRI S., RAHIM S., and ZAIDI M.: **The bees algorithm**. Technical report, Manufacturing Engineering Centre, Cardiff University, 2005.

[8] REYNOLDS R. G.: **An introduction to cultural algorithms.** In Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific Publishing, 1994. pp. 131–139.

[9] STORN R. and PRICE K.:. **Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces**, Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, 1995.

[10] GEEM Z. W., KIM J. H., and LOGANATHAN G. V.: **A new heuristic optimization algorithm: Harmony search. Simulation**, 76:60–68, 2001.

[11] GANDOMI A. H. and ALAVI A. H.: **"Krill herd: a new bio-inspired optimization algorithm"**, Communications in Nonlinear Science and Numerical Simulation, vol. 17, no. 12, 2012. pp. 4831–4845.

[12] MOSCATO P.: **On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.** Technical report, California Institute of Technology, 1989.

[13] NELDER J. A.; MEAD R.: **"A simplex method for function minimization".** Computer Journal 7, 1965. pp. 308–313.

[15] KENNEDY J. and EBERHART R. C.: **Particle swarm optimization**, In Proceedings IEEE int'l conf. on neural networks Vol. IV, 1995. pp 1942–1948.

[16] KIRKPATRICK S.: **Optimization by simulated annealing: Quantitative studies.** Journal of Statistical Physics, 1983. pp 975–986.