Competitive Evaluation of Selected Evolutionary Algorithms and SOMA

Pavel Vařacha, Martin Pospíšilík, Ivo Motýl, Michal Bližňák, Dalibor Slovák, Jakub Krampl and Jan Kolek

***Abstract*—** In this paper three evolutionary algorithms are compared, in particular, of the Self-Organizing Migration Algorithm (SOMA) as a most important one is put into the contrast with Differential Evolution (DE) and Particle Swarm Optimization (PSO). In order to compare performances of the above-mentioned algorithms, selected objective functions have been tested. In total 15 different benchmark functions were used and each considered algorithm was employed 100 times on each one producing 4500 set experimental runs. Acquired results were then statistically evaluated and compared.

The paper also describes individual parameters, strategies, and some of the termination criteria of the algorithms

***Keywords*—**Optimization, Evolutionary algorithms, Objective function, SOMA, DE, PSO.

I. Introduction

Evolutionary algorithms are a fitting device for solving and optimizing functions with multiple extremes. As the name suggest, they draw inspiration from the mechanics of evolution, whereby they strive to gradually arrive at the best possible solutions while employing crossbreeding, mutagenic processes, and other methods.

Manuscript received July 30, 2012: Revised version received August 16, 2012. This paper is supported by the Internal Grant Agency at TBU in Zlin, project No. IGA/FAI/2012/041, No. IGA/FAI/2012/056 and project No. IGA/FAI/2012/019 and by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

P. Vařacha is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail:. varacha@fai.utb.cz).

M. Pospíšilík is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail: pospisilik@fai.utb.cz).

I. Motýl is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail: motyl@fai.utb.cz).

M. Bližňák is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail: bliznak@fai.utb.cz).

D. Slovák is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail: slovak@fai.utb.cz).

J. Krampl is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail: j_krampl@fai.utb.cz).

J. Kolek is with the Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T. G. Masaryka 5555, 760 01 Zlin Czech Republic (e-mail: kolek@fai.utb.cz).

The most significant advantage of evolutionary algorithms is their ability to solve even highly complex optimization problems while focusing on finding global, rather than local extremes, as it is with numerical methods. Among their disadvantages is a certain dependence on chance, which prevents us from predicting whether the solution we have found is the best one possible. That is why it is advisable to draw on experience with these algorithms.

II. Evolutionary Algorithms

Optimization algorithms make a very strong and powerful tool in the search for an optimum solution where the analytic route is often very complicated and sometimes border-line unrealistic.

Most engineering problems can be reduced to a mathematical problem described by a corresponding function formula whose parameters we try to optimize in respect to the value of the object function.

Last two decades have seen the development of a group of some very powerful algorithms, known as the evolutionary algorithms. These algorithms employ given rules for using cycles (generations) of populations of individuals in order to arrive at the best possible solution. [1][2]

A. The Population

The population is the cornerstone of all evolutionary algorithms and its members are generated based on a predefined template known as the specimen. We can visualize a population as a matrix consisting of the its members × the number of parameters. Each single individual then represents a solution to the defined problem. To each such solution, an objective function value is assigned, together describing the quality of a given individual. [1]

B. The Specimen

The specimen is a sample individual, based on which the initial population is generated. It has a predefined variable type (real, integer, discrete, etc.) with a predefined range of permissible values for each parameter of an individual. A poorly chosen range may give rise to unsubstantiated or physically unrealistic solutions.

The initial population of individuals is made based on formula (1) where *rnd* is a random number from the interval <0;1>; *i* marks an individual and *j* its parameter; *Hi* and *Lo* denote the top and bottom boundaries of a given parameter. [1]

$$x_{i,j} = rnd \cdot \left(x_{i,j}^{Hi} - x_{i,j}^{Lo}\right) + x_{i,j}^{Lo} \qquad (1)$$

III. Self-Organizing Migration Algorithm

SOMA (**S**elf-**O**rganizing **M**igrating **A**lgorithm) is an algorithm made in 1999 and has many successful applications on various problems of multiobjective-optimization, e.g. [3 - 6]. It is an evolutionary algorithm, although no children are created in course of its evolutionary cycles (migration cycle). A more fitting description of the SOMA algorithm, however, would be as a memetic algorithm. Such algorithms can be described as competition strategies and cooperation strategies.

The foundation of SOMA can interpreted as the behavior of a certain group of intelligent individuals, wherein such individuals collaborate in order to solve a common problem, such as finding a food source. It is therefore based on cooperative search of an enclosed space.

SOMA's self-organizing progression is the result of mutual interference among the population members, as they determine the direction they will take when searching for a better solution in the area. [1]

A. Parameters

PathLength determines the distance from the leading member at which an active individual will stop.
Step determines the step size of an active individual who is on its way to the leading member.

PRT is one of the most important and most sensitive parameters, and it denotes perturbation. The perturbation vector (PRTVector), which describes the direction in which an active individual will be moving, is generated based on this quantity.

The *dimension* of a problem, the number of parameters (arguments) of the object function that the algorithm aims to optimize to the best values possible in light of the object function's value.

PopSize determines the number of individuals present in a population that take part in the migration cycles with the goal of finding the global extreme.

The *Migrations* gives the number of migration cycles in which reorganization—a rebirth of the whole population aimed at finding fitter individuals occurs. [1]

Parametr	Recommended Range
PathLength	<1.1,5>
Step	<.11,PathLength>
PRT	<0,1>
D	problem-specific
PopSize	<10,user-defined>
Migrations	<10,user-defined>

Table 1, Summary of SOMA Parameters [1]

B. Strategies

The *AllToOne* is characteristic for having a specific leading individual (a leader) towards whom the other population members migrate. Once the initial population has been created, each individual is evaluated by the object function, and its value will determine the future leading individual (the leader) of the next migration cycle. Other population members will then start moving in steps towards the leader.

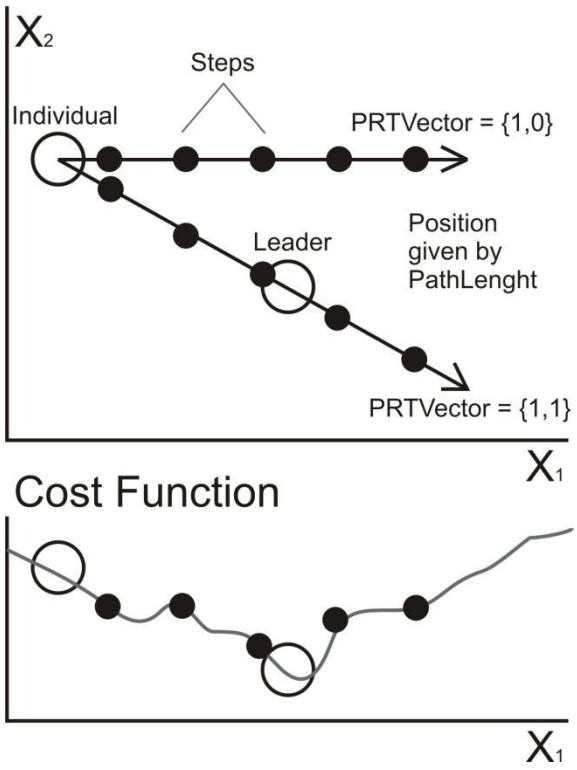

Fig. 1: PRTVector and its action on individual movement

The *AllToAll* lacks a leading individual. Population members thus do not migrate towards the fittest member; rather they move towards all other members. If they find a better extreme on their way, they will complete their journey towards the other members, and only after they have done so will they return to the position at which the best extreme was found. It is at this position that they start at in the next migration cycle.

AllToAllAdaptive is the same strategy as the *AllToAll*. The difference is at the point of return to the better position found. If members find a better extreme in their locomotion, they will complete their migration to the one member they are just migrating to. After that, they will return to the position at which the better extreme was found, and they begin they migration cycle there.

AllToOneRand is based on the principle of individuals migrating towards a leader, where such leader is not the individual with the best value based on the object function, but rather it is some randomly chosen member of the whole population. [1]

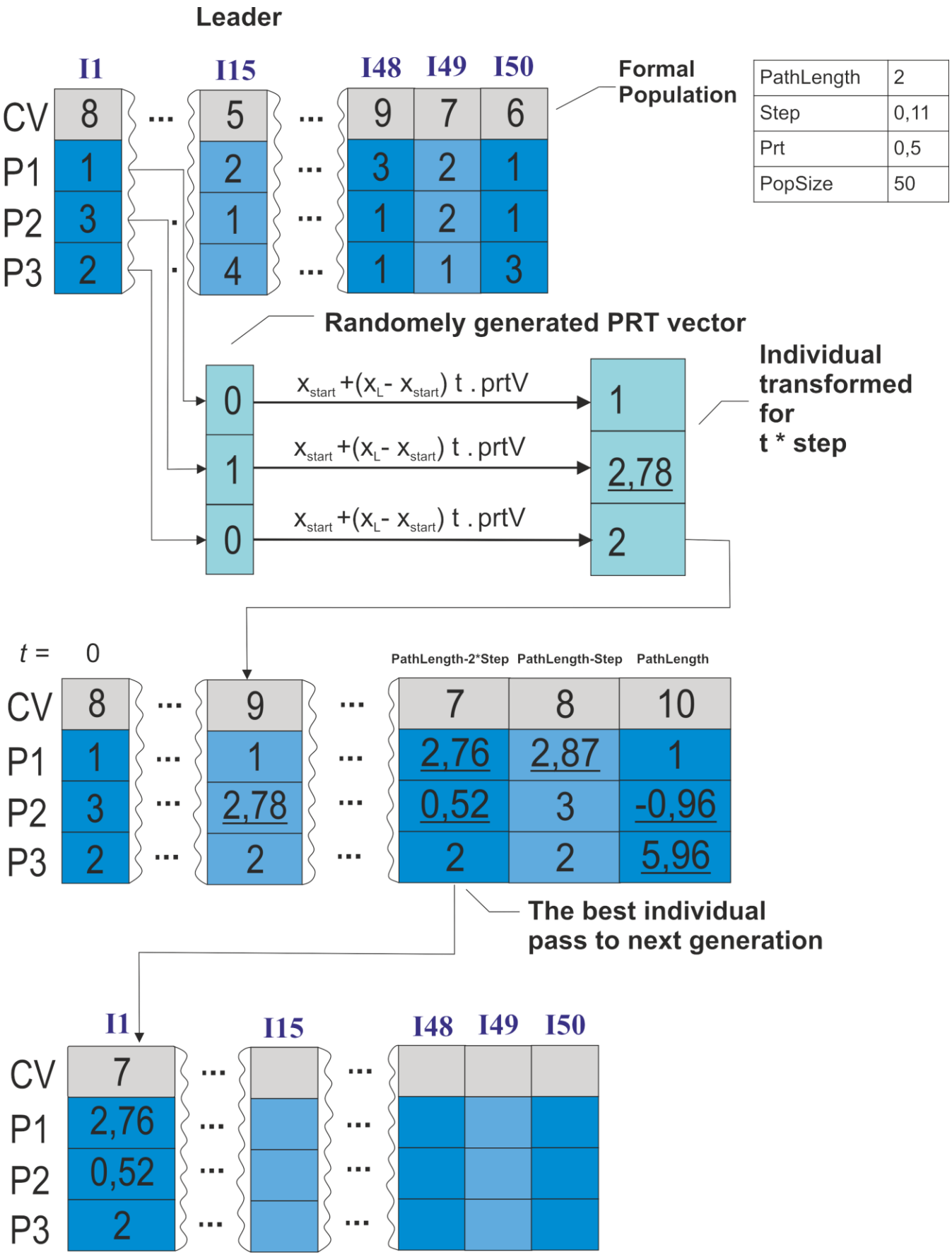

Fig. 2: Main SOMA principles depiction

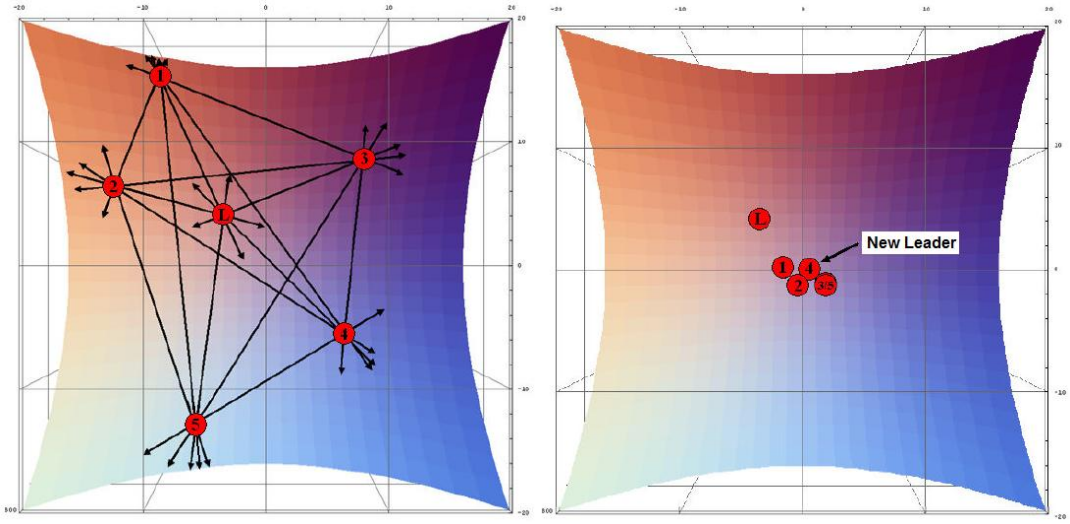

Fig. 3: All-to-One SOMA migration loop

I. Differential Evolution

DE (**D**ifferential **E**volution) is an evolutionary algorithm with its origins dating back to 1995. It has characteristics similar to those of genetic algorithms: in particular, the generation of children or the use of generations. On the other hand, unlike genetic algorithms, it does not limit itself to using only two individuals for the creation of offspring, but it rather employs combinations of four others to create a new individual.

In principle, it is based on the so-called genetic annealing, which was later on changed by switching representation from the binary to the decimal system and by changing logic operations to vector operations. By means of these modifications, an algorithm suitable for numeric optimization was created. The Differential Evolution was later enhanced when the so-called differential mutation was found. [1].

A. Parameters

The *dimension* parameter determines the number of parameters in the object function.

PopSize determines the total number of those individuals that participate in the evolutionary cycle, aiming to find the fittest individual.

The *mutation constant* affects the calculation of the noise vector. Its value determines how much and how many parents will take part in the creation of a new individual.

Based on the value of the crossbreeding limit, crossbreeding of an active individual with the mutated noise vector will give rise to a new testing vector, which will be the result of hybridization, based on the crossbreeding limit value.

The *Generations* parameter determines the number of evolutionary cycles, in which the whole population is generally cultivated. [1]

Parameter	Interval	Optimum
F	<0,2>	<.3,.9>
CR	<0,1>	<.8,.9>
D	problem-specific	-
PopSize	<2D,100D>	10D
Generations	user-defined	-

Table 2, Set of Parameters DE [1]

B. Versions

Differential evolution offers a whole score of versions that mainly differ in how they calculate the noise vector and compose the testing vector after that.

DE/best/1/exp

$$v_j = x^G_{best,j} + F \cdot (x^G_{r2,j} - x^G_{r3,j}) \quad (2)$$

DE/rand/1/exp

$$v_j = x^G_{r1,j} + F \cdot (x^G_{r2,j} - x^G_{r3,j}) \quad (3)$$

II. Particle Swarm Optimization

PSO (**P**article **S**warm **O**ptimization) is a stochastic optimization technique developed in 1995. Like many other algorithms, the technique is inspired by nature. Its foundations in this case are based on the behavior of bird and fish swarms.

In many aspects, it is similar to genetic algorithms where the foundation is comprised of random solutions populations whose members take part in the search for the optimum solution. In PSO, such potential solutions (individuals) are called particles. Unlike genetic algorithms, PSO has no "evolutionary methods" such as crossbreeding or mutation. [7]

A. Parameters

The *Dimension* parameter describes the number of parameters for the object function whose optimum combination we seek.

The *Particles* parameter determines the number of particles that take part in the search for the best solution. This parameter is known as the *PopSize* in SOMA and in the differential evolution algorithm.

The *vMax* parameter defines the maximum velocity of a particle in the course of an iteration cycle.

C1 and *C2* are learning factors that determine the direction of the journey that a particle will take. C1 attracts particulars towards its so-far best solution found. C2, on the other hand, strives to direct all particles towards the best global solution.

It proved to be difficult in more complicated optimization problem to focus the particles around the most promising solution in the last phase of the optimization procedure, as the particles' velocity was too high. The inertia parameter (w_{start}, w_{end}), was therefore introduced to the calculation of a particle's velocity, thereby reducing gradually the velocity of the particle with each iteration.

MaxIteration denotes the number of cycles during which the best solution is being found. We view this as a termination parameter. This parameter is designated *Migrations* or *Generations* in the SOMA and DE algorithms respectively. [7]

Parameter	Recommended Range
D	problem-specific
Particles	<20, 40>
vMax	determined by the search space
C1, C2	<0, 4>
wstart	.9
wend	.4
MaxIterations	user-defined

Table 3, Review of Parameters PSO [7]

III. Termination Criteria

Termination criteria are a tool for terminating a running optimization algorithm. Unlike setting a maximum number of iterations, these criteria are more adaptive and flexible, given the state of optimization at a given time.

A. Minimal Diversion

MinDiv determines the maximum permissible diversion between the objective function values in the best and the worst ranking individual in a population. [8]

B. Maximum Distance

MaxDist (maximum distance) denotes the maximum permissible distance of any individual in the population from the best solution found.

If we are only considering a given fraction of the population, e.g. 90 percent, then we are talking the *MaxDistQuick* version. [4]

$$dist_i = \sqrt{\sum_{j=1}^{D} \left(x_{i,j} - gBest_j\right)^2} \qquad (5)$$

C. Standard Deviation

StdDev (Standard Deviation) or the standard of objective function values in the whole population. Should it go bellow our predefined value, the optimization process will be terminated. [8]

$$s = \sqrt{\frac{1}{PopSize - 1} \cdot \sum_{i=1}^{PopSize} (pBestCV_i - \bar{x})^2} \qquad (6)$$

D. MinDiv + MaxDist

Some of the criteria can even be readily combined. For example, the *MinDiv+MaxDist* combination starts by comparing the difference in objective function values in the fittest and the worst individual, using our predefined permissible value (*MinDiv*). Once that is satisfied, only then the second criterion is applied (*MaxDist*). [8]

IV. Testing

The algorithms were tested using an application that was developed in the .NET environment (the C# programming language) as part of the bachelor thesis [9].

Parameter settings and the SOMA and DE strategies were used from tests carried out under [1]; PSO was set according to range recommendations (Table 3) with the number of migration loops having been set to the same value as with DE. Each experiment was repeated 100 times, and the best instance was always selected (Tab. 5).

Testing Function	Distance from the Global Extreme in %		
	SOMA	DE	PSO
1st De Jong	.01	.053	.024
3rd De Jong	.24	.98	.56
4th De Jong	.00081	1.77	.0011
Ackley function	.012	1.99×10^{-15}	.1
Egg holder[1]	-58927.25	-29637.74	-37304.29
Griewangk function	.45	0	.44
Masters's cosine wave function	5.84	26.68	12.83
Michalewicz function	.3	59.85	4.41
Pathological function[1]	32.37	38.36	33.56
Rana function[1]	-24914.55	-18193.55	-22231.81
Rastrigin function	.051	.043	.2
Rosenbrock's saddle	.026	.026	.025
Schwefel function	.00097	.00097	17.56
Stretched V sine	16.33	17.88	14.2

wave function			
Test Function (Ackley)	1.44	1.72	1.3

Table 4, Test Results [9]

[1] % could not be determined as the global extreme is unknown. Absolute values are given instead.

Used benchmark functions are depicted by following figures:

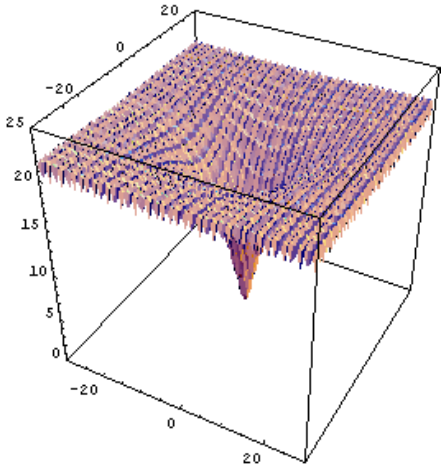

Fig. 4: Ackley 3D visualization

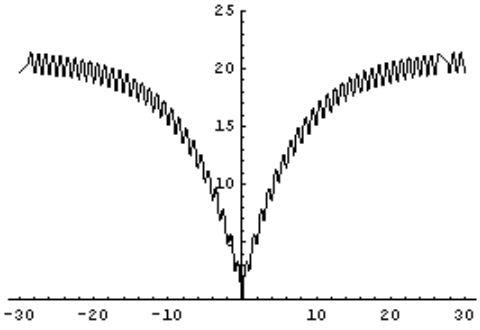

Fig. 5: Ackley 2D visualization

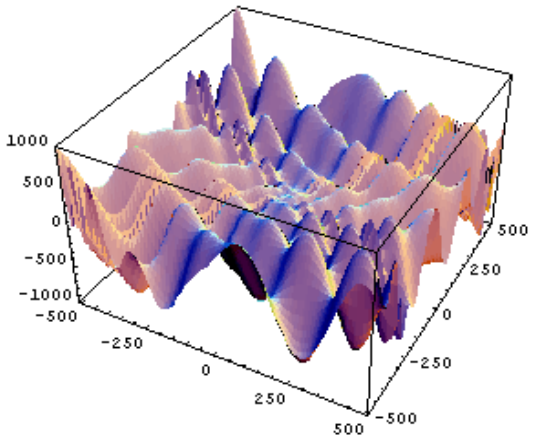

Fig. 6: EggHolder 3D visualization

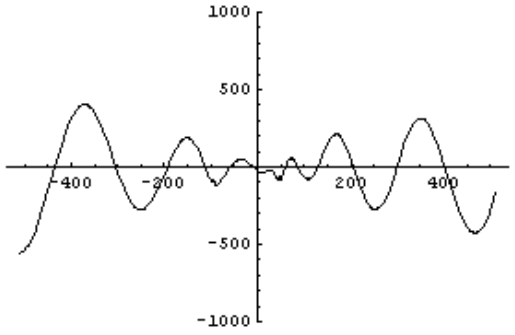

Fig. 7 EggHolder 2D visualization

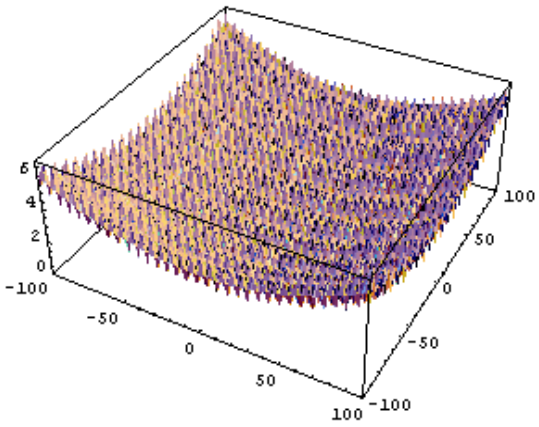

Fig. 8: Michalewicz 3D visualization

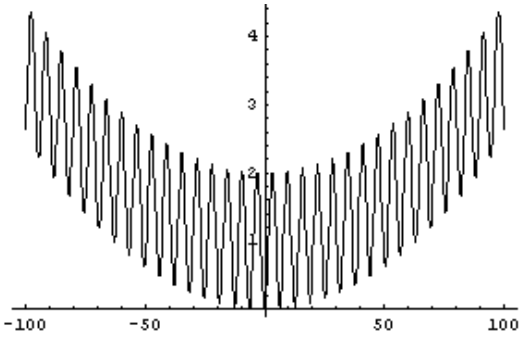

Fig. 9: Michalewicz 2D visualization

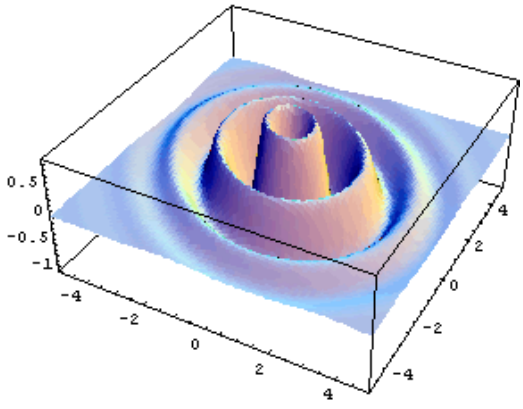

Fig. 10: Masters 3D visualization

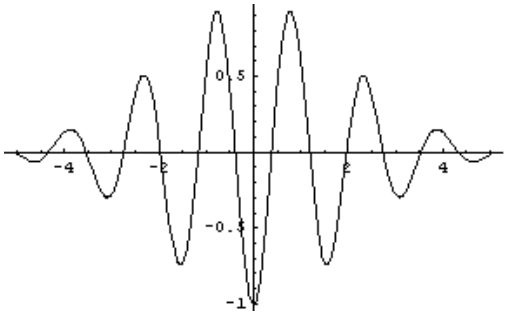

Fig. 11: Masters 2D visualization

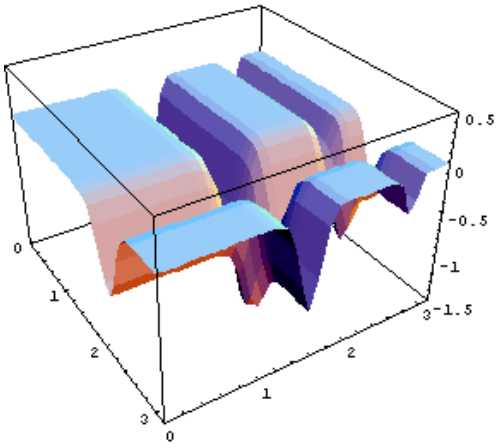

Fig. 12: Michalewicz 3D visualization

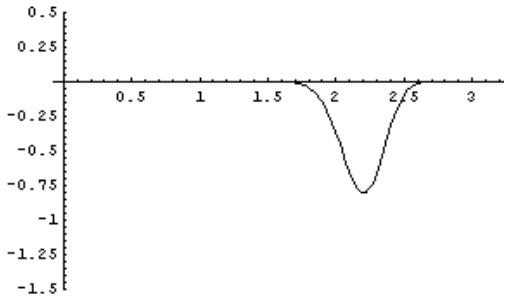

Fig. 13: Michalewicz 2D visualization

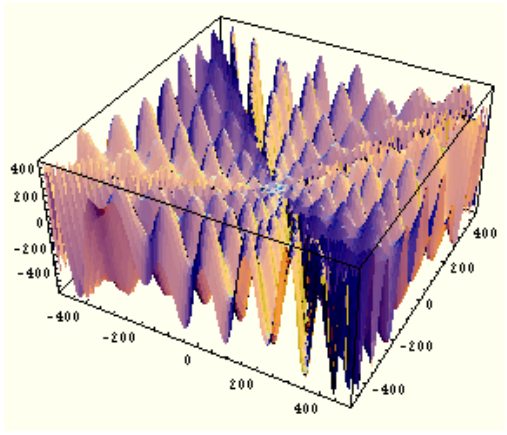

Fig. 14: Rana 3D visualization

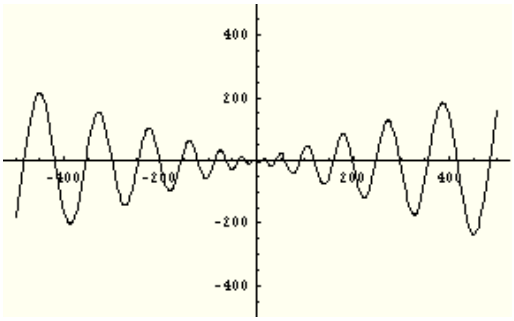

Fig. 15: Rana 2D visualization

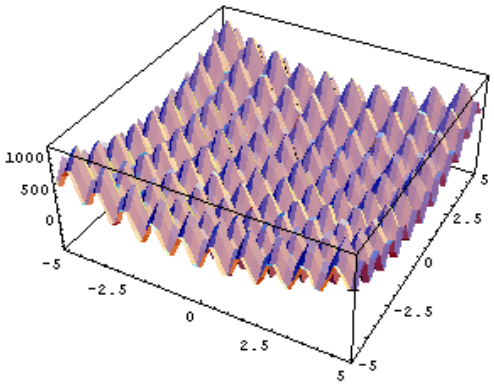

Fig. 16: Rastrigin 3D visualization

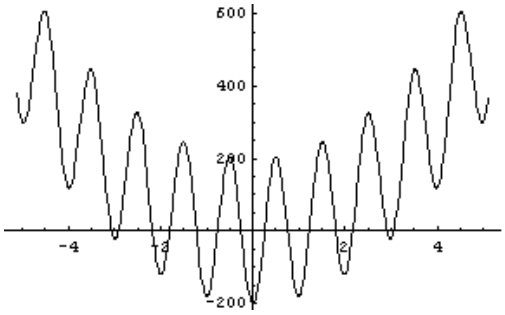

Fig. 17: Rastrigin 2D visualization

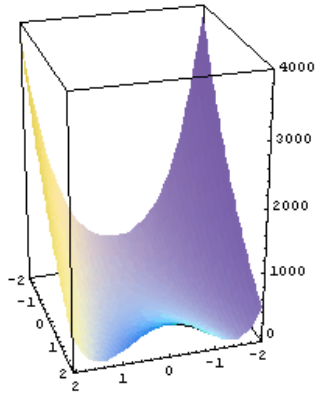

Fig. 18: Rosenbrock 3D visualization

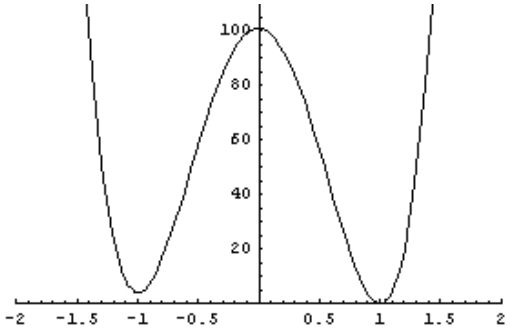

Fig. 19: Rosenbrock 2D visualization

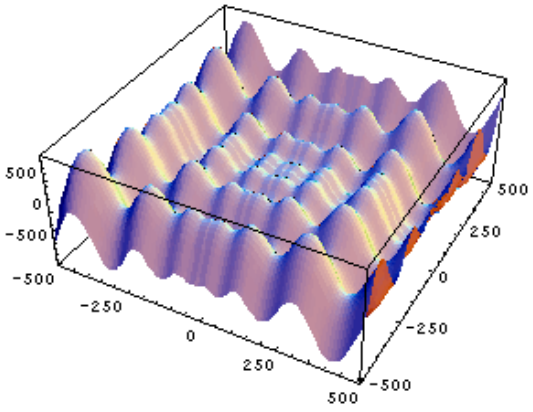

Fig. 20: Schwefel 2D visualization

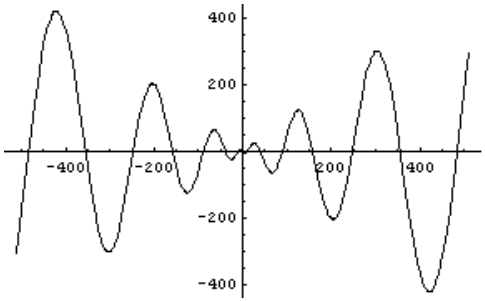

Fig. 21: Schwefel 2D visualization

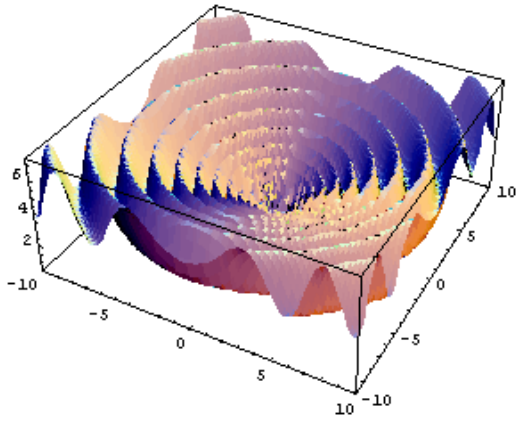

Fig. 22: SineWave 3D visualization

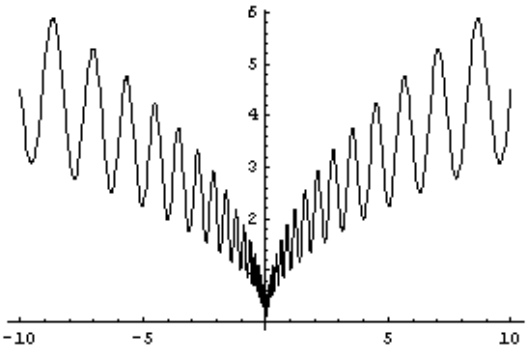

Fig. 23: SineWave 2D visualization

V. Conclusion

One can conclude, based on the tests having been carried out, that test functions have their specific properties and not all algorithms deal with them in the same manner. As an example, we may take the Michalewicz function where some SOMA give a solid result of .3 and DE gives 59.85. As a result, we must carefully consider which of these algorithms to use.

Evolutionary algorithms are particularly sensitive to their parameter settings. Even a minute change can give rise to radically different results. That is why it is very important to pay much attention to their parameter settings as well.

References

[1] ZELINKA I. Umělá inteligence v problémech globální optimalizace. Praha: BEN, 2002. ISBN 80-7300-069-5.

[2] KVASNIČKA V., POSPÍCHAL, J., TIŇO, P. Evoluční algoritmy. Bratislava: STU, 2000. ISBN 80-227-1377-5.

[3] SENKERIK R., OPLATKOVA Z., ZELINKA I., DAVENDRA D. Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming, Mathematical and Computer Modelling, Available online 27 May 2011, ISSN 0895-7177, 10.1016/j.mcm.2011.05.030.

[4] CHRAMCOV B., BERAN P., DANÍČEK L., JAŠEK R.. A simulation approach to achieving more efficient production systems. International Journal of Mathematics and Computers in Simulations, 2011, year 5, issue 4, page 299-309. ISSN 1998-0159.

[5] KRÁL E., VAŠEK, L., DOLINAY V., ČÁPEK P. Usage of Peak Functions in Heat Load Modeling of District Heating System. In Recent Researches in Automatic Control. Montreux : WSEAS Press, 2011, p. 404-406. ISBN 978-1-61804-004-6.

[6] POSPÍŠILÍK M., KOUŘIL L., MOTÝL I., ADÁMEK M. Single and Double Layer Spiral Planar Inductors Optimisation with the Aid of Self-Organising Migrating Algorithm. In Proceedings of the 11th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision. Venice : WSEAS Press (IT), 2011, p. 272 - 277. ISBN 978-1-61804-027-5.

[7] HU, Xiaohui. Particle Swarm Optimization: Tutorial [online]. 2006 [cit. 2011-05-09]. PSO Tutorial.
WWW: <http://www.swarmintelligence.org /tutorials.php>.

[8] ZIELINSKI K., LAUR R. Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimization Algorithm. Informatica : An International Journay of Computing and Informatics [online]. 2007, 31, 1, [cit. 2011-05-11].
WWW: <http://www.informatica.si/PDF/31-1/16_Zielinski-Stopping%20Criteria%20for%20a%20Constrained...pdf>.

[9] KRAMPL J. Implementace vybraných evolučních algoritmů v prostředí .NET. Zlín, 2011. Bachelor thesis (Bc.). Tomas Bata University in Zlín, Faculty of Applied Informatics