StarCraft: Brood War - Strategy Powered by the SOMA Swarm Algorithm

Ivan Zelinka, Lubomir Sikora
Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VSB-Technixal University of Ostrava
17. listopadu 2172/15
708 00 Ostrava-Poruba, Czech Republic
Email: ivan.zelinka@vsb.cz
www.ivanzelinka.eu

***Abstract*—This participation is focused on artificial intelligence techniques and their practical use in computer game. The aim is to show how program (based on evolutionary algorithms) can replace a man in the strategy game StarCraft: Brood War. Implementation used in our experiments use classic techniques of artificial intelligence environments, as well as unconventional techniques, such as evolutionary computation. An artificial player, proposed in this paper, is the combination of the decision tree and evolutionary algorithm SOMA. Whole code for experiments was written in the Java programming language. The proposed code provides a simple implementation of the artificial computer player in combination with slightly modified algorithm SOMA. This provides an opportunity for effective, coordinated movement of combat units around the combat landscape. Research reported here has shown potential benefit of evolutionary computation in the field of strategy games.**

I. Introduction

In last years artificial intelligence (AI) plays an important role in various technological applications and branches of science. One of interesting challenge come from game domain that, in fact, represent various mathematical problems with different level of complexity. As an example can serve Chess, Tia Tac Toe amongst the others. Various methods from AI has been used for that purpose. In this paper is discussed use of evolutionary algorithms (EAs), especially Self-Organizing Migrating Algorithm (SOMA) in computer game StarCraft[1]. In this game was SOMA used in realtime regime so that trajectories of an individuals were one-to-one trajectories of game bot warriors. SOMA is evolutionary algorithm, more close to the swarm class algorithms. Evolutionary algorithms are a powerful tool for solving many problems of engineering applications. They are usually used where the solution of a given problem analytically is unsuitable or unrealistic. If implemented in a suitable manner, there is no need for frequent user intervention into the actions of the equipment in which they are used.

The majority of the problems of engineering applications can be defined as optimization problems, for example, finding the optimum trajectory of a robot or the optimum thickness of the wall of a pressure tank or the optimum setting of the regulator's parameters. In other words, the problem solved can be transformed into a mathematical problem defined by a suitable prescription, whose optimization leads to finding the arguments of the objective function, which is its goal.

Countless examples illustrating this problem [1] can be found. The solution of such problems usually requires working with the arguments of optimized functions, where the definition ranges of these arguments may be of a heterogeneous character, such as, for example, the range of integers, real or complex numbers, etc. Moreover, it may happen (depending on the case) that for certain subintervals from the permitted interval of values, the corresponding argument of the optimized function may assume values of various types (integers, real, complex,..). Besides this, various penalizations and restrictions can play a role within optimization, not only for given arguments, but also for the functional value of the optimized function. In many cases, the analytical solution of such an optimization problem is possible, nevertheless, considerably complicated and tedious. Evolutionary algorithms solve problems in such an elegant manner that they became very popular and are used in many engineering fields.

From the point of view of the most general classification, the evolutionary algorithms belong to heuristic algorithms. Heuristic algorithms are either *deterministic* or *stochastic*. The algorithms of the second group differ in that their certain steps use random operations, which means that the results of the solutions obtained with their use may differ in the individual runs of the program. It is therefore meaningful to run the program several times and select the best solution obtained.

Stochastic heuristic methods are sometimes called *meta-heuristics*, because they only provide a general framework and the algorithms of the operation itself must be chosen (for example, by the operation of crossbreeding and mutation in genetic algorithms, operation of neighborhood in simulated annealing, "tabu search", etc.) in dependence on the problem investigated. Because these methods are frequently inspired by natural processes, they are also called evolutionary algorithms. Depending on their strategy, they can be divided in two classes:

1) Methods based on point-based strategy such as, for example, *simulated annealing* ([2], [3], [4]), *hill-climbing algorithm* [5] and *tabu-search* [11]. These algorithms are based on the *neighbourhood* operation of the current solution, in which we are looking for

[1] http://eu.blizzard.com/en-gb/games/hots/landing/

978-1-4799-8622-4/15/$31.00 ©2015 IEEE

a better solution, for example by mutation of the original-previous solution.

2) *Population-based strategy. Genetic algorithms* [7], [8], [9], [10] are based on *population strategy* as well as another algorithms like differential evolution, particle swarm, SOMA, etc.

These methods differ from classic gradient methods by admitting (with a certain probability) a worse solution into the next iteration; in this manner, they try to avoid local minima. For more details, see for example books [11], [8],[9],[12] and [13].

As a representative example of evolutionary algorithms e.g. **Genetic algorithm** (GA) can be used. This algorithm is one of the first successfully applied EAs methods [14], [8]. In GAs the main principles of EAs are applied in their purest form. The individuals are encoded as binary strings and represent possible solutions to the optimization problem under study. Similar, and in fact older, algorithms are **Evolutionary strategies** (ES). This algorithm also belongs to the first successful stochastic algorithms in history. It was proposed at the beginning of the sixties by Rechenberg [26] and Schwefel [15]. It is based on the principles of natural selection similarly as the genetic algorithms. More modern and swarm based algorithm, with very good performance in combinatorial problems is **Ant Colony Optimization** (ACO), [16]. This is an algorithm whose action simulates the behavior of ants in a colony and is one of the first swarm based algorithms as well as **Particle Swarm**(PS) or **SOMA** (Self-Organizing Migrating Algorithm). SOMA is a stochastic optimization algorithm that is modeled on the social behavior of cooperating individuals [17]. It has been proven that the algorithm has the ability to converge towards the global optimum [17]. SOMA works on a population of candidate solutions in loops called *migration loops*. It is vector oriented algorithm based on central individual called Leader that influences behavior of the whole population. SOMA is in fact swarm based algorithm, too, however it can be also catalogized as the **Memetic Algorithms** (MA). This term represents a broad class of metaheuristic algorithms [20], [21], [18], [19]. The key characteristics of these algorithms are the use of various approximation algorithms, local search techniques, special recombination operators, etc. These metaheuristic algorithms can be basically characterized as competitive-cooperative strategies featuring attributes of synergy. Another algorithm is **Scatter Search** (SS), This optimization algorithm differs by its nature from the standard evolutionary diagrams. It is a vector oriented algorithm that generates new vectors (solutions) on the basis of auxiliary heuristic techniques. It starts from the solutions obtained by means of a suitable heuristic technique. New solutions are then generated on the basis of a subset of the best solutions obtained from the start. A set of the best solutions is then selected from these newly found solutions and the entire process is repeated. One of the last mentioned here is **Differential Evolution** (DE). Differential Evolution [22] is a population-based optimization method that works on real-number coded individuals. Differential Evolution is robust, fast, and effective with global optimization ability. It does not require that the objective function is differentiable, and it works with noisy, epistatic and time-dependent objective functions. Along with those well known algorithms also exist another (sometimes similar in principle) algorithms like **Firefly** [24], **CoCoo algorithm** [23], **Bat algorithm** [25] amongst the others.

These algorithms can be divided according to the principles of their action, complexity of the algorithm, etc. Of course, this classification is not the only possibility, nevertheless, because it fits the current state rather well, it can be considered as one of the possible views on the classical and modern optimization methods.

The evolutionary algorithms can be essentially used for the solution of very heterogeneous problems. Of course, for the solution of the optimization problems, there are many more algorithms than were indicated here. Because their description would exceed the framework of this text, we can only refer to the corresponding literature, where the algorithms indicated above are described in more details.

II. Experiment Design

In [29] is SOMA application focused on techniques of artificial intelligence (AI) applications and practical utilization. The goal of the [29] (and its results partially reported here) is to implement computer player replacing human in real time strategy StarCraft: Brood War. The implementation uses ”conventional” techniques from artificial intelligence, as well as unconventional techniques, such as evolutionary computation. The computer player behavior is provided by implementation of decision-making tree together with evolutionary algorithm called SOMA, used to remote movemet of combat units. Code was written in Java programming language in which was created system, that ensures behavior of computer player in an easy way in implementation of artificial intelligence. Particular implementation of SOMA algorithm provides an opportunity for efficient, coordinated movement of combat units over the map. The work [29] and its results reported here has shown great benefit of evolutionary techniques in the field of real time strategy games.

A. Selected Algorithm and its Setting

SOMA (Fig. 1) is a stochastic optimization algorithm that is modeled based on the social behavior of competitive-cooperating individuals [17]. It was chosen because it has been proved that this algorithm has the ability to converge towards the global optimum [17]. SOMA works on a population of candidate solutions in loops, called migration loops. The population is initialized by uniform random distribution over the search space at the beginning of the search. In each loop, the population is evaluated and the solution with the lowest cost value becomes the Leader. Apart from the Leader, in one migration loop, all individuals will traverse the searched space in the direction of the leader. Mutation, the random perturbation of individuals, is an important operation for SOMA. It ensures diversity amongst all the individuals and it also provides a means to restore lost information in a population. Mutation is different in SOMA as compared with other EAs. SOMA uses a parameter called PRT to achieve perturbations. This parameter has the same effect for SOMA as mutation for EAs. The PRT vector defines the final movement of an active individual in the search space. The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position

Parameter	Value
PathLength	1
Step	0.5
PRT	0.8
D	2
PopSize	varying in time
Migration	”infinity”
MinDiv	-1 (i.e. not applied)

TABLE I. SOMA PARAMETER SETTING.

in the corresponding dimension. An individual will travel over a certain distance (called the PathLength) towards the leader in finite steps of the defined length. If the PathLength is chosen to be greater than one, then the individual will overshoot the leader. This path is perturbed randomly. The main principle of SOMA, i.e. traveling of an individual over space of possible solutions is depicted at Fig. 1.

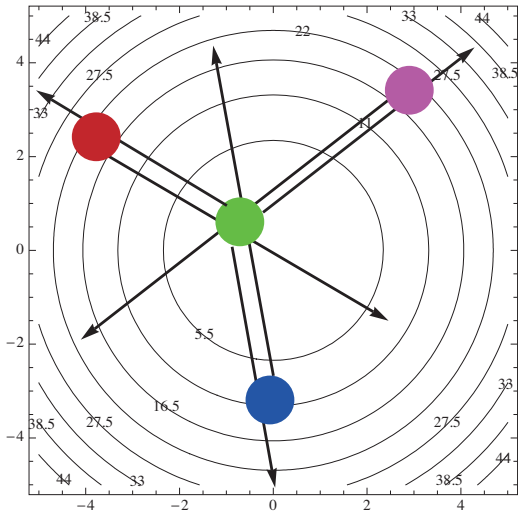

Fig. 1. SOMA principle.

The parameters set of SOMA for our experiments are in Tab. I.

For more details about SOMA, see [17]. SOMA can be also regarded as a member of swarm intelligence class algorithms. In the same class is the algorithm particle swarm, which is also based on population of particles, which are mutually influenced amongst themselves. Some similarities as well as differences are between SOMA and particle swarm, for details see [27], [28].

B. Algorithms Modification for StarCraft

Whole combat dynamics has been considered like searching of global extreme on dynamically changing cost function (battlefield) with static extremes too. For this purpose has to be slightly adapted principles of population size handling. The peculiarities of this adaptation in StarCraft game is a variable size population (parameter PopSize), that is usually constant during the time in the case of classical EAs. Combat units were treated as members of the population at the moment of creation to the population and deleted out of population when they are killed. This is not the only specific modification of the population in this application. The entire population is composed of two, say subspecies. One subspecies is a classic of the population of units, moving around hypersurface - battle field. This population is dynamic. Beside it contains implementations still static subspecies of the population, which is unique in that individuals of this subspecies does not move. These individuals are placed at all points on the battle field where there are natural sources of materials, and also to all points on the battle field that are starting points of the players. This subpopulation avoid its journey and are never destroyed. A leader can be selected from this subpopulation for most of the time. It is due to the fact that under the objective function on all of these positions represent the static local extremes, while dynamic local extremes (enemy combat units moving around the battle field) are found in motion of a dynamic subspecies (combat units). This implementation populations solves all the disadvantages of using the classic version of the algorithm SOMA (and each EA in this case). It consist of individuals that are dynamic warrior/scout subpopulation pseudorandomly set on battle field and static (i.e. subpopulation is set according to the deterministic knowledge and is distributed according to the positions of natural materials and starting positions of players). Cost function is then combination of static and dynamic one. Or better, dynamic with a few static extremes.Thus whole problem of game is then converted into an optimization problem.

The dynamics of SOMA on such space of possible solutions (i.e. battlefield) was as follows. After the start of the algorithm is chosen as the leader of a static subject, corresponding to the second starting position on a map - the enemy’s main base. After the warrior unity was generated (dynamic population algorithm SOMA) and begins its journey directly to the leader (the main enemy base). It is possible to implemented here, that the condition that his journey will start only with a certain dynamic population size to achieve a more concentrated combat force. On the way to the leader (in this moment enemy base) can warriors encounter enemy units, which is a dynamic component of the objective function, and try to destroy them. In this moment can be applies classical combat algorithms as well as algorithms based on AI which will decide whether it is the strength of my troops found sufficient to destroy the enemy army, or put prefer to retreat and preferentially produce more units. Another role of Leader, in the position of home base is the alarm function representing a threat of the enemy, attacking one of our bases. This is a beneficial of that static population, because it contains individuals presenting a position of its own base. Since the weight of enemy units (in defined cost function) are set at a high value, although dynamic population currently attacking the enemy base and the enemy in a greater number attacks our base, a leader is selected from static population, representing the position base , which is currently under heavy attack, thus it is currently attracting dynamic population to the source of hostile attacks.

Race, which was chosen as a test has been Zerg race (see, Fig. 2, 3 and 4). Screenshot in Fig. 5 demonstrate one of many game phases. As reported in [29] and Table II, SOMA driven combats has exhibit high statistical success rather than units driven by computer itself.

Fig. 2. The Zerg combat units (left - Ultralisk, Queen, Defiler; above - Overlord).

Fig. 3. An example of the Zerg base.

C. Cost Function Definition

Game in which an artificial intelligence was created is well-known real time strategy game StarCraft: Brood War. Similarly to other strategic games here it is also crucial to provide sufficient amount of materials for creating army in order to win. Materials are to be found on several places on the map as well as at the starting position. Game provides a choice of selecting from three races. Protoss are strong extraterrestrial beings. They have strong but expensive units. Zergs appear to be primitive and overgrown insects. Although, theirs units are of a cheap price, they are much weaker than units of Protoss. This race finds its victory in numbers. Last race are Terrans; race of humans being a compromise between already mentioned races have balanced prices and efficiency of theirs units. For a creation of the artificial intelligence a Zergs race has been chosen. For the creation of the artificial intelligence were used casual techniques such as decision making tree or unconventional techniques in the form of evolutionary algorithms, specifically SOMA algorithm. The algorithm should have arranged a movement of the combat units on the map.

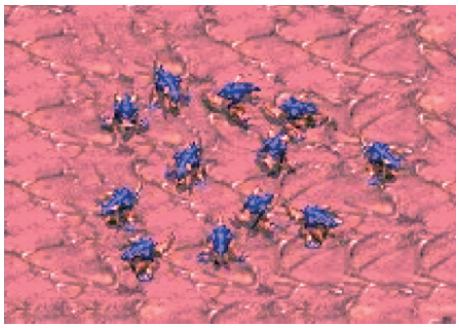

Fig. 4. Zerlings - combat unit for quick, short and near distance attack.

Fig. 5. Combat screenshot (Protos combat units).

This at first appeared as an inappropriate as it would pointlessly search the whole map when an enemy is to be situated only at the places with a raw materials. Each member of the unit would be searching the map individually which is not advantageous as the army would become scattered. Game also further would not allow random placement of a population on the map. Staring positions of the units can be only near the structures meant for production of the units. With an incorrect objective function would be rising problems with the speed of finding the enemy. Only the contact with the enemy would stabilize the searching and the searching units would be attracted to this place. Few of the problems can be solved by the right setting of the parameters of SOMA algorithm and suitable implementation of the objective function. Other problems could not be solved this way. Parameter PathLength was set to a value 1 for the units to stay on the position of the leader. Parameter Step was not a bigger success at this situation as an interface of the game enabled to command the course of a unit. The unit was particularly searching by Step long as its own. Meaning Step = 0.5. Parameter PRT was selected near 1 so the units would not deviate from each other so much and at least some random searching came to pass. PRT was set to value 0.8. Parameter D = 2 because I was looking for the places on the map which were given by coordinates x and y. PopSize was a changeable parameter. Each produced combat unit was added to population and was its part until its death. All the finishing parameters (Migrations, MinDiv) were

unwanted because algorithm SOMA was working through whole game if there were any units present. The positions of the enemy units were included in the fitness function as well as important positions on the map. SOMA algorithm thus optimized dynamic function in terms of mobility of the enemy units. This is how the function looked like:

$$Fitness = \sum_{i=0}^{m} hp_i w_i - \sum_{j=1}^{n} d_{bj} w_j - \sum_{k=1}^{p} d_{Mk} w_k - \sum_{l=0}^{q} d_{IPl} w_l \quad (1)$$

m is a number of units in radius 100 from the individual, hp_i is a number of hit points of the enemy unit in the radius 100 from the individual, n is a number of starting positions, d_{Bj} is a distance of starting positions from the individual, p is a number of positions with the materials, d_{Mk} is a distance of the individual from the position with the materials, q is a number of important positions on the map (e.g. last clash with an enemy, strategically important positions), d_{IPl} is a distance from the important point, w_x is an assigned weight of each item, can be changeable with values $w_i = 10$,$w_j = 5$,$w_k = 0.1$, $w_l = 1$.

The largest value was set to the enemy units. The more there were the enemies in the presence of the individual the bigger was a value of fitness. The closer was the individual to enemys starting position the bigger was its fitness. Starting position itself was set to 0. The biggest problem was a non-random placement of the population and inefficiency of the random search of the map. The position of the headquarters of the enemy is known. It is one of the starting positions. These problems were solved by the division of the population into two classes (as already reported): static and dynamic. Dynamic individual is not interesting at all. It is classical individual who yields to recombination and mutation. It is migrating individual in the terminology of SOMA algorithm. Static individuals on the other hand are not migrating. They are placed on the important positions at the map. Meaning positions with materials and starting positions. They can be also placed on the position of each built structure. Their role is important because of the static individual, at the starting position of the enemy, is calculated the biggest fitness. And they attract all of the units albeit from only one position. If they encounter the enemy on their route this kind of individual is chosen as a leader (in the terms of the highest value of the enemys hit points). If is enemy to attack any of our structures, this structure is chosen to be a leader (there is a static individual) from the same reason as stated. And the leader will attract attacking friendly units from unprotected enemy base. So these individuals create also alarm system. The strategy was quick attack (rush) at unprepared enemy. The results of the fights are demonstrated in the Table II.

III. Results

As already mentioned, the race, which was chosen for testing of SOMA has been the Zerg race. The advantage of this race is the low price and high number of their combat units. As a test, following strategy was chosen. It involves rapid straightforward attacks towards the enemy. Immediately after the start is set up new building, which is necessary for the production of the first type units. This unit is called "Zergling". This unit is shown in Fig. 4. This is a ground unit that from his attack on nearby targets. It is very quick, but it has a low amount of life. One egg produce two units. This strategy uses options of the Zerg race to produce these units in an extremely short time and be quickly focused to the unarmed enemy. Tactic is suited for small maps and for two players. If the map was bigger, or more enemy starting positions, this strategy would not be effective because of the time spent to searching for the enemy - enemy would have sufficient time to prepare the defense. Table II shows the success of this strategy on the combat units, that are in play already implemented. Right value is the number of victories the enemy. Selected examples of screenshots from Zerg battles are in Fig. 6 - 8. Despite the fact that Zerg race (and especially Zergling) has some disadvantages (already partly mentioned), obtained results clearly shows that Zerg powered by SOMA dynamics were the most successful combat units.

Enemy	Score
Protoss	6 : 4
Terran	8 : 2
Zerg	9 : 1

TABLE II. Battle score - wins of race : wins of enemy. The total number of the battle simulations was 10 for each race.

Fig. 6. Combat screenshot with Zerg units, screenshot from Video 1.

IV. Conclusion

In this paper we have demonstrated use of SOMA swarm class algorithm on strategic game StarCraft: Brood War. SOMA dynamics has been modifies for game purpose and used to control movement of the combat units in order to win battle. For that purpose has been used Zerg race and Protoss with Terrans were selected like an enemy. Dynamics of SOMA together with combat problem converted into fitness function has converted that into problem of optimization, that is applicable for EAs on general. Our implementation and obtained results has shown (remember this is not mathematical proof

Fig. 7. Combat screenshot with Zerg units, screenshot from Video 2.

Fig. 8. Combat screenshot with Zerg units, screenshot from Video 3.

but numerical demonstration) that EAs and swarm algorithms (in this case SOMA) are applicable for such class of games with acceptable success ratio.

ACKNOWLEDGMENT

The following grants are acknowledged for the financial support provided for this research: Grant Agency of the Czech Republic - GACR P103/15/06700S and SP2015/142.

REFERENCES

[1] Back T, Fogel B, Michalewicz Z. (1997) Handbook of Evolutionary Computation, Institute of Physics, London

[2] Kirkpatrick S, Gelatt Jr. C, Vecchi, M (1983) Optimization by Simulated Annealing, Science, 220(4598):671-680

[3] Cerny V (1985) Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, J. Opt. Theory Appl. 45(1):41-51

[4] Telfar G. (1996) Acceleration Techniques for Simulated Annealing. MSc Thesis. Victoria University of Wellington, New Zealand.

[5] Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, NJ: Prentice Hall, pp. 111-114, ISBN 0-13-790395-2

[6] Rego C, Alidaee B (2005) Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search, Springer-Verlag, ISBN: 978-1402081347

[7] Davis L (1996) Handbook of Genetic Algorithms, Van Nostrand Reinhold, Berlin

[8] Goldberg D (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company Inc., ISBN 0201157675

[9] Michalewicz Z. (1996) Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin

[10] Chu P (1997) A Genetic Algorithm Approach for Combinatorial Optimisation Problems. Ph.D. Thesis. The Management School Imperial College of Science, Technology and Medicine, London 181

[11] Glover F, Laguna M (1997) Tabu Search, Springer, July, ISBN 0-7923-8187-4

[12] Michalewicz Z, Fogel DB (2000) How to Solve It: Modern Heuristics, Springer-Verlag, Berlin

[13] Reeves C (1993) Modern Heuristic Techniques for Combinatorial Problems. Oxford, Blackwell Scientific Publications, 1993

[14] Holland J (1975) Adaptation in natural and artificial systems, Univ. of Michigan Press, Ann Arbor

[15] Schwefel H (1974) Numerische Optimierung von Computer-Modellen (PhD thesis). Reprinted by Birkhuser, 1977

[16] Dorigo M, Sttzle T (2004) Ant Colony Optimization, MIT Press, ISBN: 978-0262042192

[17] Zelinka I (2004) SOMA - Self Organizing Migrating Algorithm, In: Onwubolu , Babu B (eds), New Optimization Techniques in Engineering, (Springer-Verlag, New York), ISBN 3-540-20167X, 167-218

[18] Goh C, Ong Y, Tan K (2009) Multi-Objective Memetic Algorithms, Springer-Verlag, ISBN 978-3-540-88050-9

[19] Schonberger J, (2005) Operational Freight Carrier Planning, Basic Concepts, Optimization Models and Advanced Memetic Algorithms, Springer-Verlag, ISBN 978-3-540-25318-1

[20] Onwubolu G, Babu B (2004) New Optimization Techniques in Engineering, Springer-Verlag, New York, 167-218, ISBN 3-540-20167X

[21] Hart W, Krasnogor N, Smith J (2005) Recent Advances in Memetic Algorithms, Springer-Verlag, ISBN 978-3-540-22904-9

[22] Price K (1999) An introduction to differential evolution. In: Corne D, Dorigo M, Glover F (eds) New Ideas in Optimisation, McGraw Hill, International (UK), 79-108.

[23] X.-S. Yang; S. Deb (December 2009). "Cuckoo search via Lvy flights". World Congress on Nature and Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210214

[24] Yang, X. S. (2009). "Firefly algorithms for multimodal optimization". Stochastic Algorithms: Foundations and Applications, SAGA 2009. Lecture Notes in Computer Sciences 5792. pp. 169178

[25] X. S. Yang, A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).

[26] Rechenberg I (1971) Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis), Printed in Fromman-Holzboog, 1973

[27] Eberhart R. C., Kennedy J. [1995] A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, 39-43

[28] Clerc M., [2006], Particle Swarm Optimization, ISTE Publishing Company, 2006, ISBN 1-905209-04-5

[29] StarCraft: Brood War - Strategy Powered by the SOMA Swarm Algoritmh, Diploma thesis, VSB-TU Ostrava.