On manipulation of initial population search space in heuristic algorithm through the use of parallel processing approach

Marcin Woźniak and Dawid Połap
Institute of Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland
Email: Marcin.Wozniak@polsl.pl, Dawid.Polap@gmail.com

***Abstract*—Increasing use of heuristic algorithms in various fields of science causes numerous modifications of the original algorithms in need for better performance and efficiency. The main problem of heuristics is time required to find optimal solution. For this purpose, we propose to use parallel processing in the initial phase of heuristic methods to decrease computing time. Implemented technique models migration of individuals by adjusting initial population to required conditions. Proposed approach is simulating parallel processes that take place in human brain while solving tasks. In this case, human intelligence is working parallel on various aspects of the problem to compare them in the end before final decision. Proposed approach is simulating this parallelization of thinking threads in the process of optimization. Presented experimental tests have been carried out and discussed in terms of advantages and disadvantages.**

I. Introduction

Artificial intelligence methods are more often used in current technology trends. Continuous modifications and improvements of existing methods allow not only to improve precision and reduce run-time, but also to gain a lot more flexibility. Flexibility is based on the general and multi-tasking, which contributes to the fact that one algorithm with minimal changes can be applied in several different purposes.

Mathematical optimization is a branch of mathematics that involves finding the minimum or maximum value (called global extremes) for a given function in a specific range. The problem of finding the extremes depends mainly on the type of the analyzed function. G.B. Dantzig presented Simplex Algorithm [1]. This problem dates back to the sixteenth and seventeenth century, when Fermat and Lagrange proposed a formula developed for identifying optimum of functions. Then, Gauss and Newton proposed the first iterative methods for finding solutions for the specified optimization problems. It was not until the end of the first half of the twentieth century, the theory of optimization began to gain popularity among scholars and various applications in technology and science.

Popularity of optimization methods was caused by the discovery of more and more numerous applications - from finding extremes, the knapsack problem and the theory of graphs to many other with much more practical aspects. In various areas, more and more complex functions have been created, which contributed to the departure from the classical optimization methods at the expense of genetic algorithms and heuristics that rely on finding approximate solutions. These types of algorithms are included not only in the optimization theory but also in computational intelligence that includes methods for the analysis and processing of diverse information. To this day they find numerous applications along with other methods of computational intelligence, eg. for AGC of multi-area power system [2]. Another important aspects are 2D and 3D graphics processing. In [3], was presented the use of the firefly algorithm for the purpose of image compression. Again, in [4], [5] the authors presented different approaches to 3D face recognition. In [6], the idea of a rapid classification of images is presented.

Other important elements of computational intelligence are neural networks, where heuristics can be used as learning algorithms [7]. In contrast, in [8], the authors described a novel memristor training scheme for such structures. In [9] the authors analyzed the effects of uncertainty parameters and pulse to the global stability of delayed neural networks. For computational intelligence algorithms, knowledge is an important element. Based on the provided knowledge, systems are trained for classification. There are situations that knowledge is incomplete, then the knowledge base should be supplemented. One of such technique is to supplement by generating new samples on the basis of existing one as discussed by Mleczko et al. [10]. Example of classification by similar network is shown in Brivinskas et al. [11] where the classification is based on EEG signals. Computational intelligence of various types is also used in games. The authors of [12], [13], [14] describe the structure of devoted agents for multi-playing game and single-player games.

A drawback of heuristics is time required to find optimal solution. To increase their efficiency and reduce operating time, it is important to reduce the amount of operations on the way to final solution. However it is not always possible, i.e. for large scale optimization problems we are not able to reduce sufficiently object model. Therefore we need to develop another approach, efficient in calculations. Our idea presented in this article is devoted to parallelization of information processing. Parallel processing is a topic that is recently growing together with the improvements in computational machines. This approach allows for a new modeling applied in different topics, where we are able to distribute tasks between available threads and therefore speed up processing. This type of parallel processing can be profitable for algorithms implemented for game tree search [15] and surface modeling method [16]. Use of parallelization proved to be very beneficial in digital information processing, where in various data base systems the amount of information exceeds computational powers of regular approaches and therefore needs devoted processing

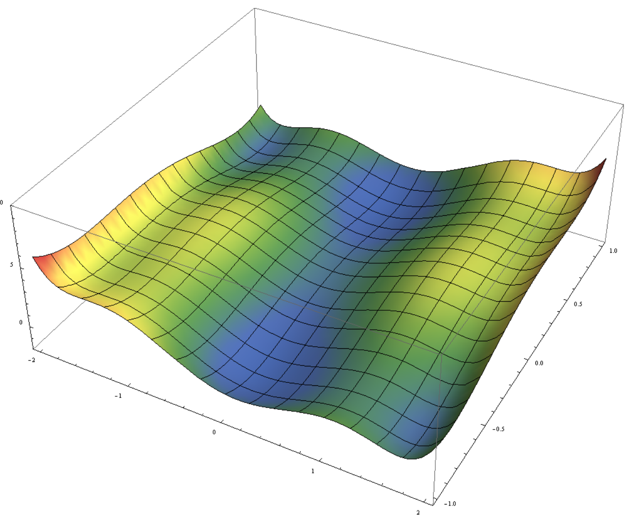


Fig. 1. Graph of the Six-Hump Camel Function.

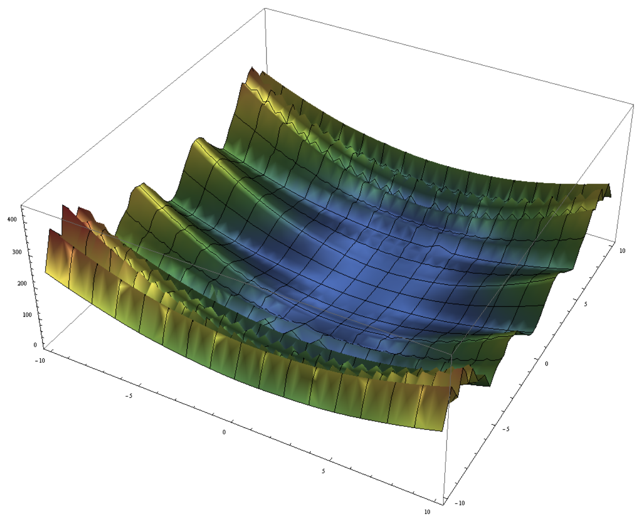


Fig. 2. Graph of the Levy Function N. 13.

[17].

In this article, we present experimental research on a modification of heuristic algorithm by application of parallel processing for the purpose of reducing operation time. Proposed approach introduces separation of information between threads, which are used for local search, and at the end for final decision results of calculations are compared to choose the best solution. This approach varies in efficiency due to the amount of CPU cores involved in actions. This article is to discuss possibility of parallelization of heuristic methods and results in efficiency depending on number of involved CPU cores.

The approach we discuss in this article is connected with human perception, i.e. we often say that somebody has an ability to think of various things parallel. Our approach is based on divisions of the input object into small parts to be presented for the algorithm executed on various cores. This action is somehow similar to various senses of humans, where we can estimate objects looking at them, hearing them, smelling them, etc. Similarly we can also compare the approach to humans working in a group, where each member has a devoted part of the task to do. In the proposed approach applied heuristics are executed on various parts of the input functions to be compared in the end to find the global optimum among them.

II. Heuristic approach to the optimization problem

Building more and more sophisticated methods and algorithms associated with creating complex functions or equations is a visible trend in evolutionary computation in the recent years. We can see the development of new algorithms simulating various animals during hunt or adaption to environmental conditions. In this article we discuss one of the new heuristic algorithms developed to simulate whales while hunting krill. Our approach is examined on three test functions.

A. Test Functions

Optimization algorithms are tested using mathematical functions for which the minimum or maximum value is very difficult to find for the classic methods. Mainly the reason for these difficulties is the shape of test function. One of the classics used in examinations are valley-shaped functions due to the numerous hills, where on ups and downs computational methods can easily get lost. In the test of parallel heuristic approach we have used:

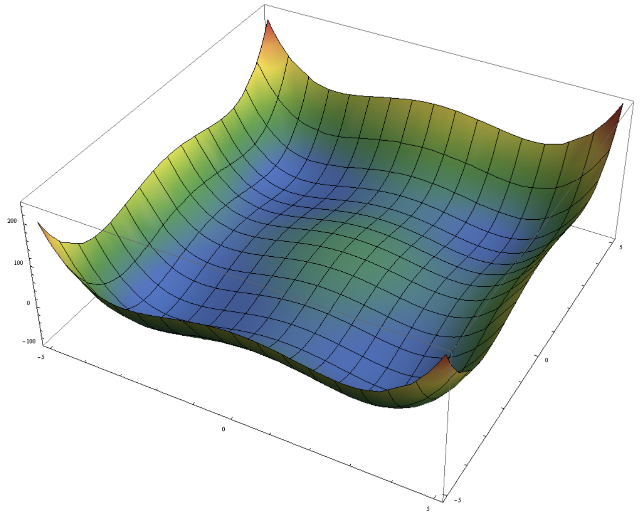


Fig. 3. Graph of the Styblinski-Tang Function.

Six-Hump Camel Function, which is described as

$$f_C(x_1,x_2) = x_1^2\left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) + x_1x_2 + x_2^2(4x_2^2 - 4). \tag{1}$$

Fig. 1 presents this function on a rectangle $x_1 \in [-3,3]$ and $x_2 \in [-2,2]$. This has six local minimums of which only two are global at $(0.0898, -0.7126)$ and $(-0.0898, 0.7126)$ equal -1.0316.

Levy Function N. 13, which is another test function with many local minimums. Similarly to previous one it has the surface usually interpreted as a landscape of mountains with numerous peaks. Example presented in Fig. 2 was depicted in the search space $[-10,10]\times[-10,10]$ according to equation

$$f_L(x_1,x_2) = \sin^2(3\pi x_1) + (x_1-1)^2[1+ \sin^2(3\pi x_2)] + (x_2-1)^2[1+\sin^2(2\pi x_2)], \tag{2}$$

where we have only one global minimum $f_L(x_1,x_2)=0$ at point $(x_1,x_2)=(1,1)$.

Styblinski-Tang Function is a third in the group of functions representing landscape and depressions in the ground. Fig. 3 presents this function in the area of a rectangle $[-5,5]\times[-5,5]$ according to equation

$$f_{S-T}(x_1,x_2) = \frac{1}{2}\sum_{i=1}^{2}(x_i^4 - 16x_i^2 + 5x_i), \tag{3}$$

where we one global minimum $f_{S-T}(x_1,x_2) = -78,33198$ at point $(x_1,x_2)=(-2.9035,-2.9035)$.

B. The Whale Optimization Algorithm - classic approach

One of the last heuristic algorithms is the Whale Optimization Algorithm (WOA) described by S. Mirjalili and A.Lewis in [18]. WOA is inspired by behavior of humpback whales and the strategy of bubble-net hunting. Humpback whales prey on smaller fish close to the surface creating bubbles. In recent years, two maneuvers were observed and called upward-spiral and double-loops [19].

The algorithm assumes that the humpback whale is represented by a point $\overline{x}$ in solution space. An individual can locate their prey and surround it. The prey is considered to be the best solution in the current iteration t of the algorithm, to which every other individual will be heading. Movement of the humpback whale in search of prey is described by the following equation

$$\overline{x^{t+1}} = \overline{x^t} - \alpha\left|2r\overline{x^t_{best}} - \overline{x^t}\right|, \tag{4}$$

where $\overline{x^t_{best}}$ is the best adapted individual in the iteration, r is a random value between $[0,1]$. Parameter α is calculated as

$$\alpha = a(2r-1), \tag{5}$$

where a is coefficient linearly reduced at each iteration from 2 to 0.

During the movement, humpback whales explore the area at random, therefore parameter α can be used to search for prey. The authors assumed that if $|\alpha|<1$ humpback whales move toward prey. Otherwise, they move toward a randomly selected neighbor. This behavior is modeled by

$$\overline{x^{t+1}} = \overline{x^t_{rand}} - \alpha|2r\overline{x^t_{rand}} - \overline{x^t}|, \tag{6}$$

where $\overline{x^t_{rand}}$ means random humpback whale in t iteration.

In a situation where humpback whale is near the prey, a strategy is applied. It consists two steps – shrinking encircling mechanism and spiral updating position. The first step is modeled as a reduction in the value of a parameter a in equation (5). The second step is to calculate distance d between individual $\overline{x^t}$ and prey $\overline{x^t_{best}}$ and simulation helix-shaped movement of humpback whales using the following equation

$$\overline{x^{t+1}} = d(\overline{x^t},\overline{x^t_{best}})\exp(bl)\cos(2\pi l) + \overline{x^t_{best}}, \tag{7}$$

Algorithm 1 Whale Optimization Algorithm

```
 1: Start,
 2: Define number of individuals in population, number of
    iterations $t_{max}$, fitness function $f$ and value of parameter
    $b$,
 3: Create an initial population randomly,
 4: Find best individual in population $\overline{x_{best}}$,
    # Search for the best solution among all located points in
    the current iteration by comparing values of the function
 5: $t := 0$,
 6: while $t < t_{max}$ do
 7:    for each individual $\overline{x}$ in population do
 8:       Reduce value of parameter $a$,
 9:       if $p < 0.5$ then
10:          if $|a| < 1$ then
11:             Move individual using (4),
12:          else
13:             Select a random neighbor $\overline{x_{rand}}$,
14:             Move individual using (6),
15:          end if
16:       else
17:          Move individual using (7),
18:       end if
19:    end for
20:    for each individual $\overline{x}$ in population do
21:       if $f(\overline{x}) < f(\overline{x_{best}})$ then
22:          Replace $\overline{x_{best}}$ with $\overline{x}$,
23:       end if
24:    end for
25:    $t++$,
26: end while
27: Return $\overline{x_{best}}$,
28: Stop.
```

where b is a given parameter defining shape of a logarithmic spiral and l is a random parameter in $[0,1]$. The model assumes that there is a 50% chance to choose one of strategies for each humpback whale in each iteration. The full algorithm is shown in Algorithm 1.

III. Parallelization by manipulating initial population - proposed innovative approach

For practical applications, the cost of optimization using heuristic algorithm will depend upon desired size of solution space, and a particular test function. Very often it happens that time required to find the optimal solution is very long. For this purpose, parallel processing may be appropriate.

In Algorithm 1 - classic version of WOA, individuals in initial population are found randomly over entire solution space. In case of functions where there is a large number of deep local minimums - like for presented test functions, the risk of getting stuck is high. As a remedy to limited ability to move population from local to global minimum a common approach is to use a larger number of individuals and larger number of iterations. Unfortunately these do not guarantee finding the optimum for given parameters. Moreover, the larger solution space, the chance to find the global minimum is smaller.

Let us think how animals plan strategies for survival in natural conditions. Most of animals stick together in groups, which increases their chances of survival (safety – a large herd allows them to avoid the attacks of other animals). Similarly, humpback whales do not live around the ocean but in specific areas. Of course, there is a chance that some individuals may be separated from the herd, but they are just selective cases. Humpback whales migrate, depending on the time of the year. In the summer period, humpback whales reside in colder waters, and migrate to warmer in winter, therefore we can see them i.e., at Atlantic Ocean at the coast of South Africa from June to October and in other months they move up the Globe. This situation can be modeled for the need to increase the chance of obtaining a solution in a much shorter time.

Imagine that the initial population chooses one particular area in the solution space, which is the most attractive at the moment (assume that the area of potential solution are cooler waters during the summer period). Finding such an area is based on the division of the entire solution space on pc parts, where pc i.e. can be the number of processor cores. Suppose that solution space for the function f is $[a_1, b_1] \times [a_2, b_2]$. Such space can be divided into n different rectangles, for example by dividing the space of one variable. For this purpose, the interval can be written as the sum of pc smaller intervals in the following manner

$$[a_1, b_1] = \bigcup_{i=0}^{pc-1} \left[a_1, a_1 + (i+1)\frac{b_1 - a_1}{pc} \right]. \qquad (8)$$

In this way, the solution space can be represented as

$$[a_1, b_1] = \bigcup_{i=0}^{pc-1} \left(\left[a_1, a_1 + (i+1)\frac{b_1 - a_1}{pc} \right] \times [a_2, b_2] \right), \quad (9)$$

and this is the sum of pc intervals of the solution space.

Humpback whales are looking for the coolest area, so instead of starting to generate the entire population consisting of n individuals in one step, we can lessen the space of solutions to generate the initial population. For this purpose, for each of space division, a random population consisting of $\frac{n}{pc}$ humpback whales is created. Each individual is assessed relative to the fitness function f. The best adapted humpback whale in each interval is selected. Each of them is compared, in order to select the best - and thus a smaller division of the solution space, where potential solution may be found. In this way, we obtain an interval of solutions – only in this range developed parallel version of heuristic method will be executed to find a solution.

In the proposed method, parallel processing has an important part. The solution space is divided depending on the number of processor cores. On each of the cores, one thread is created, which is responsible for a particular area. This solution allows to reduce the searched area and the number of operations performed by heuristics due to the smaller number of individuals. The newly developed parallel method is presented in Algorithm 2.

IV. Experimental Research and Performance Results

For experimental research we have implemented original version of WOA and the proposed parallel modification

Algorithm 2 Parallel analysis of the solution space for the initial population

```
1: Start,
2: Define solution space, number $n$ of individuals, number $t$ of iterations and fitness function $f$,
3: Detect number $pc$ of processor cores,
4: Divide solution space using (9),
5: Create an array of $pc$ threads,
6: for each part of solution space do
7:     Create a thread,
8: end for
9: for each thread do
10:    Create a temporary population of $n/pc$ individuals at random,
11: end for
12: Select a thread in which the best adapted individual is located,
13: Replace original solution space with the space of the thread,
14: Create a random population composed of $n/pc$ individuals in new solution space,
15: Return the newly created population,
16: Stop.
```

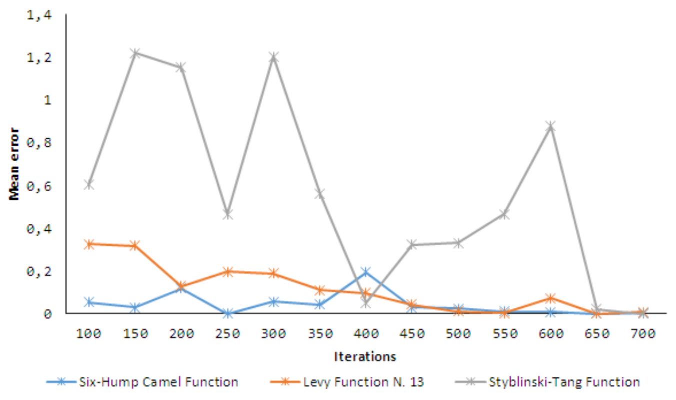
described in Section III using C# language. Both of them were tested using test functions from Section II-A. Tests were carried out on quad-processor 64-bit Intel Core i5-4460 clocked at 3.20 GHz. Aaverage accuracy of solutions and average execution time of algorithm for input parameters have been selected as comparison parameters. The original WOA algorithm and its parallel modification were examined for 2 and 4 CPU cores, for each of test functions (see equations (1), (2) and (3)) with specific parameters, running 30 tests in each experiment. Then, average scores were calculated by applying arithmetic mean. Results for a population of 400 individuals and 200 iterations are shown in Table I. For each

TABLE I. Experimental Optimization Results

	Classic WOA	Parallel WOA	Parallel WOA
Cores	1	2	4
Individuals	400	400→200	400→100
Iterations	200	200	200
	Six-Hump Camel Function		
Average time [ms]	152	98	63
Obtained point	(-0.099;0.6824)	(0.0943;-0.723)	(-0.0987;0.73)
Approximate solution	-1.0238	-1.0307	-1.029
	Levy Function N. 13		
Average time [ms]	275	134	101
Obtained point	(0.9978;0.9987)	(1.0002;0.92)	(1;1)
Approximate solution	0.0004	0.0079	0
	Styblinski-Tang Function		
Average time [ms]	189	121	98
Obtained point	(-2.8972;-2.9034)	(-2.9011;-2.9081)	(-2.9892;-2.9198)
Approximate solution	-78.3316	-78.3319	-78.1972

test function, operation time is reduced when the amount of CPU cores is increased. Obtained solutions are almost accurate for proposed novel parallel modification using 4 CPU cores. In case of original algorithm, and modification for 2 CPU cores, the results are less precise.

Results depend on number of iterations, and mean errors shown in Fig. 4 (population composed of 100 individuals, modification tested for 4 CPU cores). The mean error was



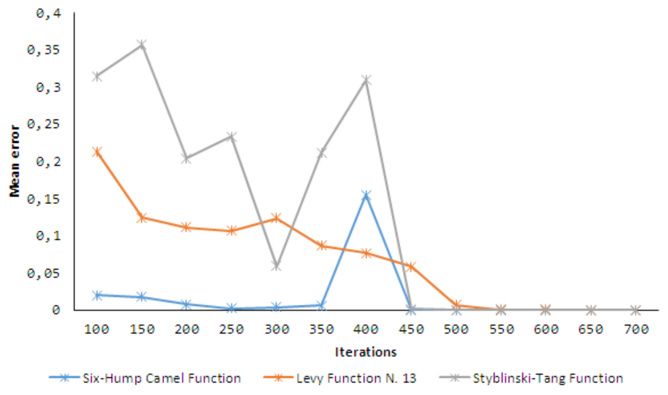


Fig. 4. The mean error measured for the accuracy of experimental results for the classic version of the algorithm (left) and newly proposed parallel version of the algorithm (right).

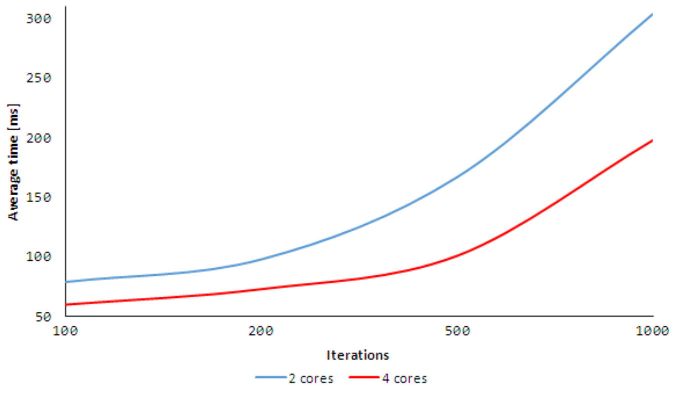


Fig. 5. The average amount of computing time for a number of iterations for 100 individuals.

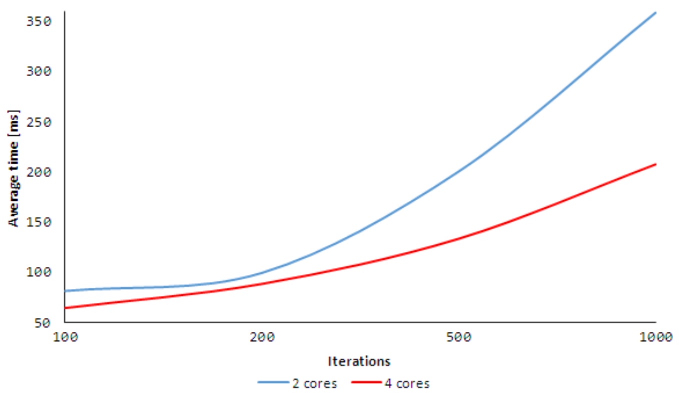


Fig. 6. The average amount of computing time for a number of iterations for 300 individuals.

determined by the following formula

$$|f(\overline{x_{ex}}) - f(\overline{x_{obt}})|, \tag{10}$$

where $\overline{x_{ex}}$ is the exact solution and $\overline{x_{obt}}$ is obtained solution. Based on graph analysis, we can say that obtained solutions are much more precise for proposed technique where the biggest mean error was 0.36 achieved for *Styblinski-Tang Function*. For comparison, the method without modification achieved error equal to 1.23 with the same parameters, which is 3.4 times less precise solution.

As we conclude from the analysis that efficiency of the proposed modification depends on the usage of processor cores as shown in Fig. 5 - Fig. 7. The time of processing increases exponentially for both configurations of involved CPU cores with a population below 500 Individuals. Moreover, tests carried out for a population of 500 individuals showed that average time increases in a linear way.

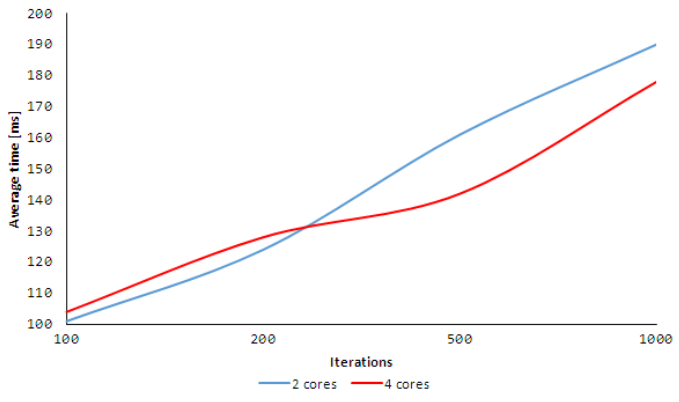


Fig. 7. The average amount of computing time for a number of iterations for 500 individuals.

V. Conclusion

The article presents an application of newly proposed technique of manipulation of initial population in a heuristic algorithm based on parallel processing. Conducted tests have shown that the use of proposed method can significantly reduce execution time of the algorithm in various optimization problems. Moreover, reducing solution space in the initial phase of the algorithm will help the method to quickly converge to the extreme individuals.

The tests were carried out on three functions with different surface landscape. The newly proposed algorithm assumes that individuals are generated in a random way what may cause that solution space will be reduced to such an extent that a global solution will be easier to find using even less individuals. Similarly to this it is possible to find difficulties in parallel method, however in the early stage of the research we can say that proposed modification has a positive impact on calculations. In future work we will focus on more efficient divisions into subspaces for each thread i.e. by flexible estimation of borders that may influence even faster convergence to functions extremes and also consider other computing architectures like these involving on GPUs that shall enable far more efficient parallelization even faster than when run on a CPU.

Acknowledgment

Authors acknowledge contribution to this project of Operational Programme: Knowledge, Education, Development financed by the European Social Fund under grant application POWR.03.03.00-00-P001/15, contract no. MNiSW/2016/DIR/208/NN.

References

[1] G. B. Dantzig, *Linear programming and extensions*. Princeton university press, 1998.

[2] R. K. Sahu, S. Panda, and S. Padhan, “A hybrid firefly algorithm and pattern search technique for automatic generation control of multi area power systems,” *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 9–23, 2015.

[3] M.-H. Horng, “Vector quantization using the firefly algorithm for image compression,” *Expert Systems with Applications*, vol. 39, no. 1, pp. 1078–1091, 2012.

[4] S. Pabiasz, J. T. Starczewski, and A. Marvuglia, “A new three-dimensional facial landmarks in recognition,” *Lecture Notes in Artificial Intelligence - ICAISC’2014*, vol. 8468, pp. 179–186, 2014, DOI: 10.1007/978-3-319-07176-3_16.

[5] S. Pabiasz, J. T. Starczewski, and A. Marvuglia, “SOM vs FCM vs PCA in 3d face recognition,” *Lecture Notes in Artificial Intelligence - ICAISC’2015*, vol. 9120, pp. 120–129, 2015, DOI: 10.1007/978-3-319-19369-4_12.

[6] M. Korytkowski, L. Rutkowski, and R. Scherer, “Fast image classification by boosting fuzzy classifiers,” *Inf. Sci.*, vol. 327, pp. 175–182, 2016, DOI: 10.1016/j.ins.2015.08.030.

[7] E. Valian, S. Mohanna, and S. Tavakoli, “Improved cuckoo search algorithm for feedforward neural network training,” *International Journal of Artificial Intelligence & Applications*, vol. 2, no. 3, pp. 36–43, 2011.

[8] J. Starzyk and Basawaraj, “Memristor crossbar architecture for synchronous neural networks,” *IEEE Trans. on Circuits and Systems*, vol. 61-I, no. 8, pp. 2390–2401, 2014, DOI: 10.1109/TCSI.2014.2304653.

[9] T. Huang, C. Li, S. Duan, and J. Starzyk, “Robust exponential stability of uncertain delayed neural networks with stochastic perturbation and impulse effects,” *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 6, pp. 866–875, 2012, DOI: 10.1109/TNNLS.2012.2192135.

[10] W. K. Mleczko, R. K. Nowicki, and R. Angryk, “Rough restricted boltzmann machine–new architecture for incomplete input data,” *Lecture Notes in Artificial Intelligence - ICAISC’2016*, vol. 9692, pp. 114–125, 2016, DOI: 10.1007/978-3-319-39378-0_11.

[11] D. Birvinskas, V. Jusas, I. Martišius, and R. Damaševičius, “Data compression of eeg signals for artificial neural network classification,” *Information Technology And Control*, vol. 42, no. 3, pp. 238–241, 2013.

[12] K. Waledzik and J. Mandziuk, “An automatically generated evaluation function in general game playing,” *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 6, no. 3, pp. 258–270, 2014, DOI: 10.1109/TCIAIG.2013.2286825.

[13] M. Swiechowski and J. Mandziuk, “Self-adaptation of playing strategies in general game playing,” *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 6, no. 4, pp. 367–381, 2014, DOI: 10.1109/TCIAIG.2013.2275163.

[14] M. Okulewicz and J. Mandziuk, “Two-phase multi-swarm PSO and the dynamic vehicle routing problem,” in *2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence, CIHLI 2014, Orlando, FL, USA, December 9-12, 2014*, 2014, pp. 86–93, DOI: 10.1109/CIHLI.2014.7013391.

[15] L. Li, H. Liu, H. Wang, T. Liu, and W. Li, “A parallel algorithm for game tree search using gpgpu,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, pp. 2114–2127, 2015.

[16] M. Zhao, T. Yue, N. Zhao, X. Yang, Y. Wang, and X. Zhang, “Parallel algorithm of a modified surface modeling method and its application in digital elevation model construction,” *Environmental Earth Sciences*, vol. 74, no. 8, pp. 6551–6561, 2015.

[17] M. Gabryel, “The bag-of-features algorithm for practical applications using the mysql database,” *Lecture Notes in Artificial Intelligence - ICAISC’2016*, vol. 9693, pp. 635–646, 2016, DOI: 10.1007/978-3-319-39384-1_56.

[18] S. Mirjalili and A. Lewis, “The whale optimization algorithm,” *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[19] J. A. Goldbogen, A. S. Friedlaender, J. Calambokidis, M. F. McKenna, M. Simon, and D. P. Nowacek, “Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology,” *BioScience*, vol. 63, no. 2, pp. 90–100, 2013.