TECHNICAL TRANSACTIONS 9/2017
CZASOPISMO TECHNICZNE 9/2017
MECHANICS
DOI: 10.4467/2353737XCT.17.159.7171

Stanislaw Krenich (krenich@mech.pk.edu.pl)
Institute of Production Engineering, Department of Mechanical Engineering,
CracowUniversity of Technology

Multi-thread evolutionary computation for design optimization

Wielowątkowe ewolucyjne obliczenia równoległe w optymalizacji konstrukcji

Abstract

The paper presents multi-thread calculations using parallel evolutionary algorithms (EA) for single and multicriteria design optimization. This approach was implemented to avoid a negative influence of incorrectly chosen initial and EA's control parameters for the accuracy of generated solutions and thereby to improve the effectiveness of the EA's use. Parallel computation for single optimization problems relies just on running n threads with different randomly chosen parameters in order to find the best final solution. For multicriteria optimization problems, each thread generates a set of Pareto optimal solutions and at the end these sets are combined together, giving a real set of Pareto optimal solutions. During the run of the algorithm, random interactions between threads were applied. The experiments were carried out using ten-thread processes for different examples of single and multicriteria design optimization problems, two of which are presented in the paper.

Keywords: parallel computation, evolutionary algorithms, design optimization.

Streszczenie

W artykule przedstawiono wielowątkowe obliczenia równoległe z wykorzystaniem algorytmów ewolucyjnych (AE) dla jedno- i wielokryterialnej optymalizacji konstrukcji. Przedstawioną metodę wykorzystano w celu uniknięcia negatywnego wpływu niewłaściwie dobranych parametrów inicjujących i sterujących w algorytmie ewolucyjnym na dokładność obliczeń, a tym samym w celu poprawy efektywności działania algorytmu. Obliczenia równoległe dla optymalizacji jednokryterialnej polegają na uruchomieniu n wątków z losowo dobranymi parametrami AE z przyjętych zakresów i zbiorów dyskretnych. Dla optymalizacji wielokryterialnej każdy wątek generuje niezależny zbiór rozwiązań Pareto, a następnie na końcu zbiory te są łączone w finalny zbiór rozwiązań Pareto. W trakcie obliczeń wprowadzono losowe interakcję między wątkami. Eksperymenty przeprowadzono z wykorzystaniem 10 wątków równoległych dla wielu przykładów, dwa przedstawiono w artykule.

Słowa kluczowe: obliczenia równoległe, algorytmy ewolucyjne, optymalizacja konstrukcji

1. Introduction

Evolutionary algorithms (EA) are powerful and widely used stochastic optimization techniques which rely on analogies to natural processes. They can often outperform conventional optimization methods when applied to difficult real-world optimization problems. Many different evolutionary algorithm based strategies have been developed recently to find the optimum for nonlinear programming problems including design optimization [1, 4, 7, 8]. There are some problems which belong to the area of computational expensive tasks, where objective functions and constraints require large computing power, for example solved by means of Finite Element based Method (FEM). The only use of simple evolutionary algorithms in order to generate the optimal solution or the set of Pareto solutions is often ineffective due to long calculation time or required calculation accuracy. Moreover, results obtained while running simple evolutionary algorithms in order to solve all optimization tasks depend significantly on setting initial parameters, such as initial population, type and probability of crossover and mutation, type of reproduction mechanisms, population size etc. An incorrect choice of initial parameters for single criterion optimization problems may lead to a local optimum which can be far from the real global optimum. For multicriteria optimization problems the set of optimal solutions is very often either far away from the real set of Pareto optimal solutions or not all Pareto optimal solutions are generated. The latter case refers mainly to discrete and integer programming models. In the literature there are not scientifically proven rules how to set these parameters, however some attempts are made [2–4]. In majority of the cases described so far, these parameters are assigned experimentally during the optimization process. Thus, in the paper the parallel evolutionary algorithms have been implemented to avoid the problems discussed above.

2. Problem formulation

The approach presented in the paper was applied to design optimization but it can be used to solve any other nonlinear optimization problems formulated as follows:

Find the vector of decision variables:

$$\mathbf{x} = [x_1, x_2, ..., x_N] \tag{1}$$

which will satisfy the K inequality constraints:

$$g_k(\mathbf{x}) \geq 0 \text{ for } k = 1, 2, \ldots, K \tag{2}$$

and the M equality constraints:

$$h_m(\mathbf{x}) = 0 \quad \text{for } m = 1, 2, \ldots, M \tag{3}$$

and minimize the vector of the objective functions $\mathbf{f}(\mathbf{x})$, where

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_l(\mathbf{x})] \tag{4}$$

The method can be used to solve the following optimization models: with continuous decision variables, with integer decision variables, with discrete decision variables, with mixed continuous – integer decision variables, with mixed continuous – discrete decision variables.

3. Multi-thread computation algorithm

Generally, there are two main ways of running parallel evolutionary computing called synchronous and non-synchronous algorithms [1, 5, 6, 9, 10]. The first one deals with a common population in the main thread, where the evolutionary process including selection, crossover and mutation operations is implemented. The sub-threads are responsible for the calculation of objective functions, constraints and additional parameters. In this case, high-speed communication between the main and sub-threads is required. The non-synchronous way of parallel computing consists of n independent threads. In each thread, an evolutionary algorithm searching selected sub-domain is implemented. It is allowed to exchange any information between all threads. For example, each thread can be run with different population of chromosomes, parameters, objective functions, constraints, etc.

In the paper, the parallel computation was applied as a non-synchronous multi-thread process, whose flow diagram is presented in Figure 1.

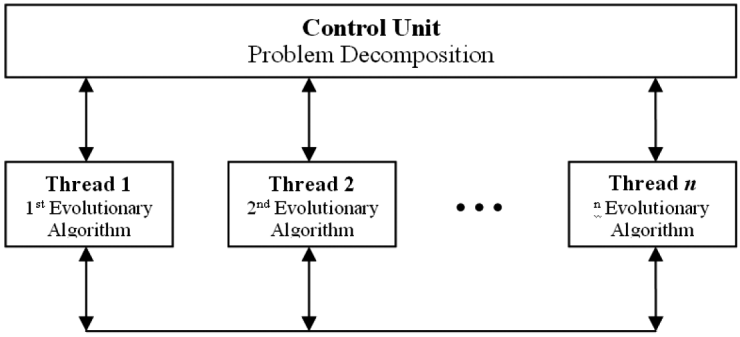

Fig. 1. Non-synchronous parallel evolutionary algorithm

The idea of parallel computations for single optimization problems consist in running *n* threads with different randomly chosen initial and control parameters in order to find the best final solution. For multicriteria optimization problems, each thread generates a set of Pareto optimal solutions and at the end these sets are combined together. During the parallel run of the threads, every *k* generations, an interaction between them is applied. This interaction is based on a random exchange of chromosomes (representing potential solution) selected from current sub-populations in different threads. The run of threads can be realized on a single computer using software or hardware decomposition to many threads or can be performed by

any computer network using the master-slave procedure. For the proper run of the proposed parallel algorithm some of parameters are required to be set. There are several parameters mentioned above which have significant influence on the effectiveness of evolutionary algorithms. The most important are selection pressure and quality of local and global search, which correspond respectively to crossover and mutation types and their settings. During all numerical experiments for these parameters, their ranges or values were assumed arbitrarily from the list of several types of evolutionary operators and for each thread were as follows:

- Range of crossover probability: $0.3 \leq RC \leq 0.8$
- Range of mutation probability: $0.0 \leq RM \leq 0.3$
- Range of initial parameter for a generator of random numbers: $1 \leq seed \leq 100$.
- List of possible selection types (refer to the selection pressure): {proportional, simple tournament, constraint tournament} [4, 7].

The remaining parameters were considered as the constant values: number of generations = 400, population size = 400, one-point crossover, uniform mutation, external penalty function with penalty parameter = 10000, numbers of threads = 10 and rate of thread exchange $k = 10$ (every k generations). The computations were carried out on the multi-core processor using C++ language.

4. Numerical Experiments

4.1. Single criteria design optimization

The problem deals with the minimum volume design of the six-step beam (Fig. 2). The results of running evolutionary algorithms for this problem while running different threads are shown in Table 1. These results show that by using the proposed approach we can avoid the influence of incorrectly chosen initial parameters and, simultaneously, we can improve the effectiveness of the use of evolutionary algorithms.

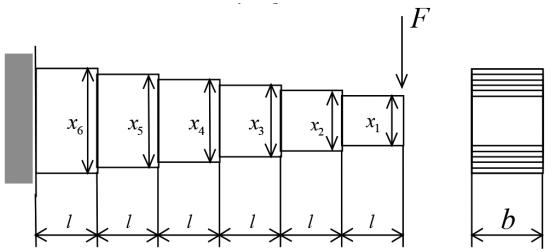

Fig. 2. Scheme of the six-step beam

The vector of decision variables is as follows:

$$X = [x_1, x_2, \ldots, x_N]^T \tag{5}$$

where x_n is the thickness of the n-th part of the beam.

The objective function is the volume of the beam:

$$f(x) = bl\sum_{n=1}^{N} x_n \tag{6}$$

The constraints are:

- Shear stress constraints:

$$\frac{6F \times nl}{bx_n^2} \leq \sigma_g \quad \text{for } n = 1, 2, ..., 6 \tag{7}$$

- Geometrical constraints

$$0 \leq x_1, \; x_n \leq x_{n+1}, \; x_N \leq d \quad \text{for } n = 1, 2, ..., 6 \tag{8}$$

The problem was considered as a continuous programming problem and was run for the following data: $N = 6$, $l = 50$ [mm], $b = 50$ [mm], $F = 10000$ [N], $E = 2.06105$ [N/mm^2], $\sigma_g = 360$ [N/mm^2], $d = 32$ [mm].

All generated solutions are presented in Table 1. The calculations were carried out for randomly chosen EA's control parameters from the sets and ranges given above.

Table 1. Results of the parallel computing for the beam design problem

Thread	Objective function	Decision variables	Parameters			
	$f(x)$	$X = [x_1, x_2, x_3, x_4, x_5, x_6]$	R_C	R_M	seed[1]	Method of selection
1	371.910	[14.175, 21.698, 24.160, 27.489, 29.168, 32.073]	0.73	0.15	14	Proportional
2	349.656	[12.913, 18.262, 22.364, 25.821, 28.873, 31.628]	0.44	0.08	100	Constraint tournament
3	354.351	[13.750, 18.440, 23.120, 25.940, 28.870, 31.620]	0.65	0.05	87	Simple tournament
4	411.203	[15.120, 25.37, 288.810, 31.141, 31.984, 31.990]	0.31	0.29	28	Proportional
5	381.972	[14.540, 24.011, 24.780, 28.015, 29.460, 31.990]	0.56	0.21	3	Constraint tournament
6	406.471	[15.180, 24.810, 28.239, 30.469, 31.931, 31.970]	0.38	0.29	69	Proportional
7	359.967	[13.000, 20.130, 23.441, 26.020, 29.410, 31.980]	0.79	0.01	35	Simple tournament
8	349.595	[12.910, 18.260, 22.360, 25.820, 28.870, 31.620]	0.60	0.04	1	Constraint tournament
9	393.935	[14.300, 24.100, 26.740, 28.920, 31.520, 32.000]	0.36	0.18	43	Simple tournament
10	363.328	[13.000, 18.510, 23.440, 28.220, 30.190, 31.960]	0.53	0.13	52	Proportional
The final result	349.595	[12.910, 18.260, 22.360, 25.820, 28.870, 31.620]	0.60	0.04	1	Constraint tournament

[1] Seed decides on the initial population, R_c – crossover rate, R_M – mutation rate. All constraints satisfied

4.2. Multicriteria design optimization

Let us consider an example of the bicriterion optimization of the robot gripper mechanism presented in Fig. 3. It is assumed that all elements of the mechanism are stiff and friction forces are not considered. In order to build an optimization model, geometrical and force dependencies are calculated.

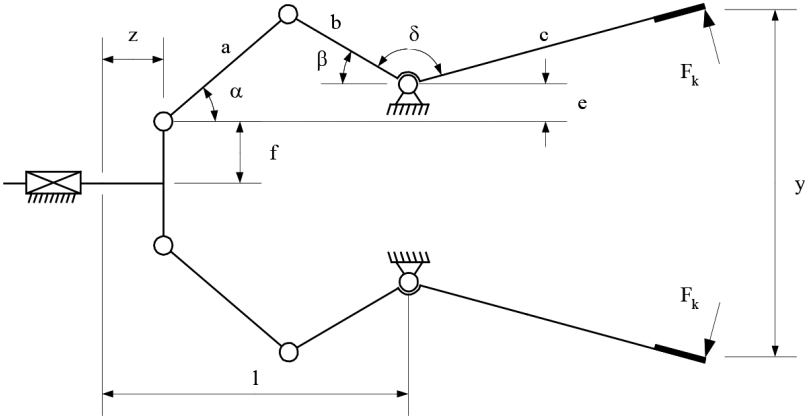

Fig. 3. Scheme of the robot gripper mechanism

The geometrical dependencies of the gripper mechanism are as follows:

$$g=\sqrt{(l-z)^2+e^2},\ \varphi=\arctan\left(\frac{e}{l-z}\right) \tag{9}$$

$$\alpha=\arccos\left(\frac{a^2+g^2-b^2}{2\cdot a\cdot g}\right)+\phi,\ \beta=\arccos\left(\frac{b^2+g^2-a^2}{2\cdot b\cdot g}\right)-\phi \tag{10}$$

$$y(\mathbf{x},z)=2\cdot\left[e+f+c\cdot\sin(\beta+\delta)\right] \tag{11}$$

The force dependencies are as follows:

$$R_{AX}=\frac{P}{2},\ R_{AY}=\frac{P\cdot\sin(\alpha)}{2\cdot\cos(\alpha)} \tag{12}$$

$$R_{BX}=\frac{P}{2},\ R_{BY}=\frac{P\cdot\sin(\alpha)}{2\cdot\cos(\alpha)} \tag{13}$$

$$R_{CX}=\frac{P}{2}\cdot\left(1-\frac{b\cdot\sin(\alpha+\beta)\cdot\sin(\beta+\delta)}{c\cdot\cos(\alpha)}\right) \tag{14}$$

$$R_{CY} = \frac{P}{2} \cdot \left(\frac{\sin(\alpha)}{\cos(\alpha)} + \frac{b \cdot \sin(\alpha+\beta) \cdot \cos(\beta+\delta)}{c \cdot \cos(\alpha)} \right) \tag{15}$$

$$R_A = \sqrt{\left(\frac{P}{2}\right)^2 + R_{AY}^2}, \quad R_B = \sqrt{R_{BX}^2 + R_{BY}^2}, \quad R_C = \sqrt{R_{CX}^2 + R_{CY}^2} \tag{16}$$

$$F_K = \frac{P \cdot b \cdot \sin(\alpha+\beta)}{2 \cdot c \cdot \cos(\alpha)} \tag{17}$$

For the dependencies given above, the optimization model is presented below. The vector of decision variables is as follows:

$$x = [a, b, c, e, f, l, \delta]^T \tag{18}$$

where:

a, b, c, e, f, l – dimensions of the gripper,

δ – the angle between b and c elements of the gripper.

The objective functions can be evaluated as follows:

- Maximization of minimal force transmission ratio:

$$\max\ f_1(\mathbf{x}, z) = \frac{\min\limits_{z} F_k(\mathbf{x}, z)}{P}, \text{ for } Z_{\min} \le z \le Z_{\max} \tag{19}$$

- Minimization of maximal value of force reaction:

$$\min f_2(\mathbf{x}, z) = \max\left\{R_A(\mathbf{x}, z), R_B(\mathbf{x}, z), R_c(\mathbf{x}, z)\right\} \text{ for } Z_{\min} \le z \le Z_{\max} \tag{20}$$

Note that both objective functions depend on the vector of decision variables and on the displacement z. Thus, for the given vector x, values of the functions have to be evaluated for different values of z, which makes the objective functions computationally expensive and the problem becomes more complicated than a general nonlinear programming problem. From the geometry of the gripper, the following constraints can be derived:

- constraints based on the mechanism movement:

$$g_1(\mathbf{x}) = y(\mathbf{x}, z) \ge 0 \text{ for each } z, \text{ were } Z_{\min} \le z \le Z_{\max} \tag{21}$$

$$g_2(\mathbf{x}) = \alpha - \phi \ge 0 \text{ for each } z, \text{ were } Z_{\min} \le z \le Z_{\max} \tag{22}$$

$$g_3(\mathbf{x}) = \frac{\pi}{2} - \alpha \ge 0 \text{ for each } z, \text{ were } Z_{\min} \le z \le Z_{\max} \tag{23}$$

$$g_4(\mathbf{x}) = \frac{\pi}{4} - |\gamma| \ge 0 \text{ for each } z, \text{ were } Z_{\min} \le z \le Z_{\max} \tag{24}$$

$$h_1(\mathbf{x}) = z + a \cdot \cos(\alpha) + b \cdot \cos(\beta) - l = 0 \text{ for each } z, \text{ were } Z_{\min} \le z \le Z_{\max} \tag{25}$$

$$h_2(\mathbf{x}) = a \cdot \sin(\alpha) - b \cdot \sin(\beta) - e = 0 \text{ for each } z, \text{ were } Z_{min} \le z \le Z_{max} \quad (26)$$

- border constraints:

$$g_5(\mathbf{x}) = y(\mathbf{x}, Z_{max}) - Y_{min} \ge 0 \quad (27)$$

$$g_6(\mathbf{x}) = Y_{max} - y(\mathbf{x}, 0) \ge 0 \quad (28)$$

$$g_7(\mathbf{x}) = l - Z_{max} \ge 0 \quad (29)$$

where:

$y(\mathbf{x}, z)$ – displacement of the gripper ends,
Y_{min} – minimal dimension of the griping object,
Y_{max} – maximal dimension of the griping object,
Z_{max} – maximal displacement of the gripper actuator.

Assumed values: Y_{min} = 0 [mm], Y_{max} = 0 [mm], Y_{max} = 2000 [mm], Z_{max} = 50 [mm], P = 1 [N]. Note that input force value P has proportional impact to functional characteristics. In order to show only the force transmission ratio of the mechanism, its value was set as equal to 1 [N] .The calculations were executed as a ten-thread run for randomly chosen EA's control parameters taken from the sets and ranges given in chapter 3. In each of the ten processes, an individual set of Pareto solutions was generated using different EA's control parameters. These sets are presented in Figure 4. At the end, these sets were combined together giving a final set of Pareto optimal solutions. This set of solutions is much more accurate than the individual sets.In this case, the front of Pareto solutions was built on the basis of threads 1, 2, 8.

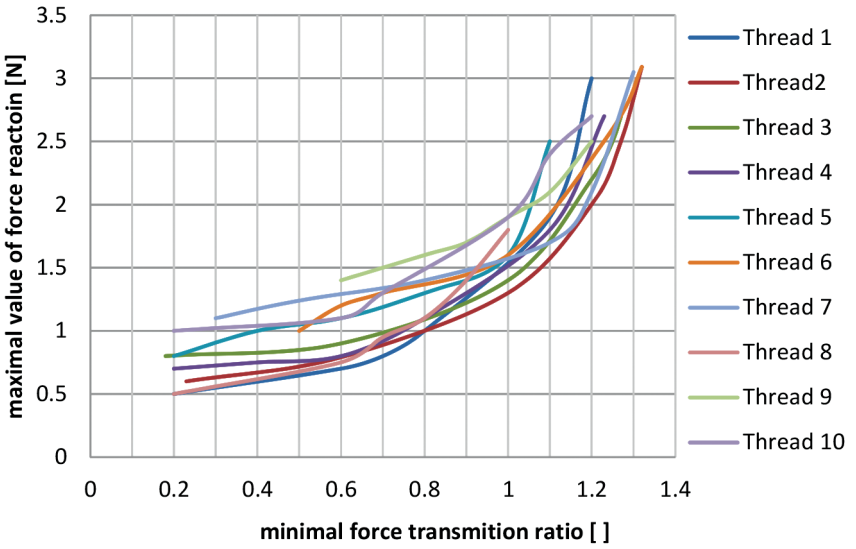

Fig. 4. Sets of Pareto optimal solutions generated while parallel computation for the gripper mechanism

Table 2. Border solutions for the gripper mechanism generated by parallel computing

Solution	Objective function	Decision variables	Control parameters			
	$f(\mathbf{x}, z) = [f_1(\mathbf{x}, z), f_2(\mathbf{x}, z)]$	$\mathbf{X} = [a, b, c, e, f, l, \delta]$	R_C	R_M	*seed*[1]	Method of selection
First	[0.20, 0.5]	[94.47, 70.55, 199.44, 0.37, 108.43, 100.78, 2.14]	0.58	0.12	84	constraint tournament
Last	[1.32, 3,09]	[174.32, 128.14, 150.52, 71.22, 5.17, 115.84, 2.53]	0.65	0.09	29	simple tournament

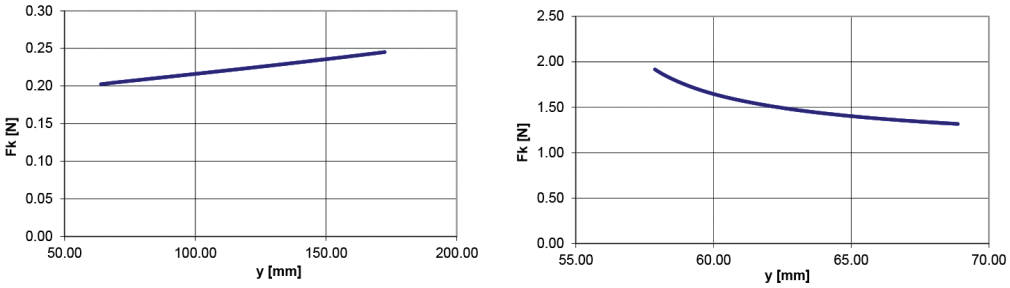

Fig. 5. Force characteristics for the border solutions (respectively the first and the last solution)

5. Conclusions

The evolutionary algorithm based on the parallel computation approach was proposed in the paper. There were many tests of single and multicriteria design optimization carried out to confirm the effectiveness of the algorithm considering its accuracy. The results obtained so far indicate that the presented method can be successfully implemented to avoid an influence of incorrectly chosen initial or control parameters of EA and, simultaneously, to improve the effectiveness of the algorithm. Interactions between different threads during the evolution of each population improved the quality of each sub-population and at the end yielded better value of the objective function or the front of Pareto solutions. But from the other point of view, the presented approach has a disadvantage. The run of the parallel algorithm requires large computation power, thus calculations on a single machine are longer. The calculation time can be improved by the use of a multi-device cluster in the local area network (LAN).Considering all properties of the method, it is clear that it provides the designer with a very effective tool for solving fairly complicated tasks. The method has a universal character and can be applied to solve a wide range of single and multicriteria design optimization problems.

References

[1] Burczynski T., Dlugosz A., Kus W., *Parallel Evolutionary Algorithms in Shape Optimization of Heat Radiators*, Journal of Theoretical and Applied Mechanics 44, 2, Warszawa 2006, 351–366.

[2] Grefenstette J., *Optimization of Control Parameters for Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 16, No. 1, 1986, 122–128.

[3] Kieś P., *Selection of genetic algorithm parameters using off-line method(In Polish)*, Instytut Naukowo-Badawczy ZTUREK, Warszawa 2000.

[4] Krenich S., *Genetic Algorithms in Parametrical Optimization of Robot Gripper Mechanisms*, Ph.D. thesis (in Polish), Wydział Mechaniczny, Politechnika Krakowska, Kraków 2002.

[5] Lis J., Lis M., *Self-adapting Parallel Genetic Algorithm with Dynamic Mutation Probability, Crossover Rate and Population Size*, Proceedings of the First Polish National Conference on Evolutionary Computing, J. Arabas (ed.), Politechnika Warszawska, Warszawa 1996, 324–329.

[6] Miki M., Hiroyasu T., Hatanaka K., *Parallel Genetic Algorithms with Distributed-Environment Multiple Population Scheme*, The 3rd World Congress on Structural and Multidisciplinary Optimization, Buffalo 17–22 May, USA, 1999.

[7] Osyczka A., *Evolutionary Algorithms for Single and Multicriteria Design Optimization*, Springer-Verlag Physica, Berlin Heilderberg, 2002.

[8] Osyczka A., Krenich S., *Evolutionary Algorithms for Global Optimization*, [in:] *Global Optimization – Selected Case Studies*, J. Pinter (ed.), Kluwer Academic Publishers, Dordrecht/Boston/London 2007.

[9] Osmera P., Lacko B., Peter M., *Parallel Evolutionary Algorithms*, Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003.

[10] Sadecki J., *Parallel algorithms for optimization and testing of their effectiveness* (in Polish), Oficyna Wydawnicza Politechniki Opolskiej, Opole 2001.