I Krajowa Konferencja
Algorytmy Ewolucyjne
Murzasichle, 12-15.06.1996

A HYBRID GENETIC ALGORITHM FOR SOLVING MULTIOBJECTIVE OPTIMIZATION PROBLEMS OF DYNAMIC MODULAR PROGRAM ASSIGNMENTS

Jerzy Balicki, Zygmunt Kitowski

Akademia Marynarki Wojennej
81-919 Gdynia 19, ul. Śmidowicza,
Tel.: (48-58) 262532, 262625, Fax: (48-58) 253881, E-mail: amw.@.beta.nask.gda.pl

ABSTRACT

In this paper, a hybrid genetic algorithm for finding Pareto-optimal solutions in multiobjective optimization problems of dynamic modular program assignments is presented. It operates on the population of artificial neural networks generating suboptimal solutions for only one criterion. Initial states of designed neural networks are changed by the genetic algorithm. Finally, Hopfield's analog neural network population for finding Pareto-suboptimal solutions in a simple multiobjective optimization problem for resource dynamic assignments in distributed computer networks have been proposed.

1. INTRODUCTION

In distributed computing system a modular program must have its modules assigned among the processors so as to avoid excessive interprocessor communication while taking advantage of specific efficiencies of some processors in executing some program modules. It makes sense to change assignments dynamically to take advantage of local behavior of programs and the relatively infrequent changes in program locality. Fortunately, the dynamic problem is no harder than the static one, except for the need to solve several static assignment problems instead of a single one.

Several multiobjective optimization problems of static module assignments are NP-hard [1,13]. A promising approach for solving these problems seems genetic algorithms (GA). Genetic algorithms are the alternative approach compare to standard operational research methods [15], simulated annealing [11], Hopfield's neural networks [9], the Boltzman machine [16], and elastic nets [20]. Holland [10] developed GA and its theoretical foundation. Although there are many variants of the basic genetic algorithms [4,14,17], the fundamental underlying mechanism operates on a population of individuals, is relatively standard. Goldberg and Lingle [8] suggested a crossover operator called the „partially mapped crossover". Bac and Perov [2] proposed another crossover operator created by the non-Abel group theory.

Schaffer [18] considered GA for solving multiobjective optimization problems by an vector evaluated genetic algorithm. Multiobjective learning by GA is discussed in [19]. Additionally, an overview of evolutionary algorithms for multiobjective optimization problems is presented by Fonseca and Flaming [6].

In this paper, genetic approach for solving multiobjective optimization problems are presented. This genetic algorithm combining with Hopfield's analog neural networks (HANN) is considered. Several elements of HANN can be transformed by the genetic algorithm. For given HANN presenting an suboptimal solution in an equilibrium point is possible to improve an accuracy of obtained solutions by reproduction, crossover, and mutation. Results of an evolution of HANN initial states are presented.

2. GENETIC ALGORITHMS

Holland [10] noted that reproduction in conjunction with the pressure of natural selection has developed species remarkably well adapted to their environment. The crossover operator as the discovery mechanism played the base role in Holland's genetic algorithm. The basic relatively standard genetic algorithm operates on a population of individuals and consists of the following operations. There are an evaluation of individual fitness, a formulation of a gene pool, and recombination with mutation.

Goldberg and Lingle [8] suggested a crossover operator, which leads to an efficient solution of TSP, as they believed. It is called,, the partially mapped crossover" (PMX), because one randomly chosen gene vector part of parents is mapped into the gene vector part of offspring with the modification by the exchanging of genes between parents. Formally, Liepins and Hilliard specified the genetic algorithm [14]. The more complicated genetic algorithm has been proposed by Bac and Perov [2]. Details of the random replication, selection, crossover, and mutation in [7] are described. It is easy for software implementation and can be used for solving several difficult optimization problems.

Genetic algorithms and artificial neural networks are separately used for solving many optimization problems. Tank and Hopfield considered the "neural" approach for solving the Traveling Salesman Problem (TSP) [12]. Sun and Fu presented the hybrid neural network model consisted of multiprocessor systems for finding solutions of TSP and successfully discovers solutions to the Hamiltonian Cycle [20]. Finally, Hopfield's analog neural networks for solving two similar multiobjective optimization problems in [3] have been proposed. From the other hand, Schaffer [18] proposed to use for solving multiobjective optimization problems, a genetic approach.

The real temptation is a combination between the genetic algorithm and the neural networks population. The main purpose of it is the improving of an accuracy of obtained solutions. For giving more details of this new approach [10,20] neural networks for finding suboptimal solutions with respect to only one criterion are considered.

3. SEVERAL HANNs FOR MULTIOBJECTIVE OPTIMIZATION PROBLEMS

It is possible to use one of known neural methods for minimization problem with only one criterion to solve the multiobjective optimization problem. The main question is how the problem with many criteria lead to the problem with only one criterion. The most popular nonnegative convex method for finding Pareto-optimal solutions is preferred [3].

$$\begin{aligned} &\min_{y \in Y=F(X)} \{Q(y,\alpha) = \sum_{n=1}^{N} \alpha_n y_n = \alpha^T y\} \\ &\alpha_n \geq 0 \quad n = \overline{1,N} \\ &\sum_{n=1}^{N} \alpha_n = 1 \end{aligned} \tag{1}$$

where

X - a feasible solution set,

$Y=F(X)\subset R^N$ - a set of vector evaluations,

$y=[y_1,\ldots,y_n,\ldots,y_N]^T$ - a point in the set Y,

$\alpha=[\alpha_1,\ldots,\alpha_n,\ldots,\alpha_N]^T$ - a vector of nonnegative convex combination coefficients.

The other method for scalarization of multiobjective problems in [1] are described. It is possible to use artificial neural networks for minimization of nonnegative convex combination of particular functions. For linear and quadratic nonnegative convex combination functions, Hopfield's analog networks can be considered. Then genetic algorithm can operate on the different parameters of given optimizing neural networks. For instance, external inputs or synaptic weights can be changed. Moreover, gain coefficients in activation functions, types of activation function or passive inhibitor coefficients can be modified, too. But, we want to considered neural population behaves more intelligent than only competitive populations in simple genetic algorithms. That is why, members of this populations are equipped in poor intelligent dedicated for finding suboptimal solutions in one criterion sense.

We can consider N network sorts for minimization each criterion with respect to constraints and K network sorts for implementation of several known scalarization methods leading from multiobjective problems to minimization problems. Each network category has N(j) representatives, where j denotes the number of network category. All representatives are processed by the genetic algorithm. We assume, that all networks are Hopfield's analog networks.

For finding one local efficient (Pareto-suboptimal) solution of multiobjective optimization problem, we can use the Hopfield's ANN called PHANN. The outputs of neurons represent decision variables x_m (or related dummy variables) or nonnegative convex parameters α_n (or related dummy variables β_n). PHANN can represent one Pareto-optimal solution in an equilibrium point. PHANN satisfies the Cohen-Grossberg's conditions of a global stability [9]. In the analog model of Hopfield's ANN, the behavior of neurons is described by the differential equations [12]:

$$\frac{du_m}{dt} = -\frac{u_m}{\eta_m} + \sum_{i=1}^{M} w_{im} g_i(u_i) + I_m \tag{2}$$

where

η_m - passive inhibitor coefficient for neuron x_m ;

I_m - external input to the neuron x_m .

The synaptic weights w_{im} in a connection between neurons x_i and x_m are symmetric. The activation function is a sigmoid function $g_m(u_m) = \frac{1}{2}[1 + \tanh(\gamma u_m)]$, where γ denotes the gain coefficient.

If external inputs are constant over time, then Hopfield's ANN with feasible parameters reaches the equilibrium point. The energetic function for PHANN implementing nonnegative convex combination method are constructed as below:

$$E(x,\alpha) = \sum_{n=1}^{N} \alpha_n F_n(x) + \beta \sum_{l=1}^{L} h_l(x) \tag{3}$$

where

β- a penalty parameter for no satisfaction of constraints,

$h_l(x)$ - a nonnegative penalty function for no satisfaction lth constraint for solution x.

We can calculate the particular weights and particular external inputs for the feasible solutions as shown in [3]. Moreover, passive inhibitor coefficients and gain parameters are

found by simulation methods [3]. Then networks can work, if initial states of activation levels are given.

On the fig. 1 the minimization trajectory of energy function and the trajectory of the 1st neuron activation level state for the particular HANN satisfied the constraint $\sum_{m=1}^{M} x_m = k$ is presented. Network HANN can be a subnetwork of network PHANN. These results have been obtained by the simulation of differentiable equations (1) for 50 neurons (M=50) and k=3. Synaptic weights are equal to -2, and external inputs are equal to 2k-1 [3]. The gain parameter γ in activation functions is equal to 10. The length of integrity step is assumed as 0.01. The started input activation level state vector is equal to $[50,49,48,47,...,2,1]^T$.

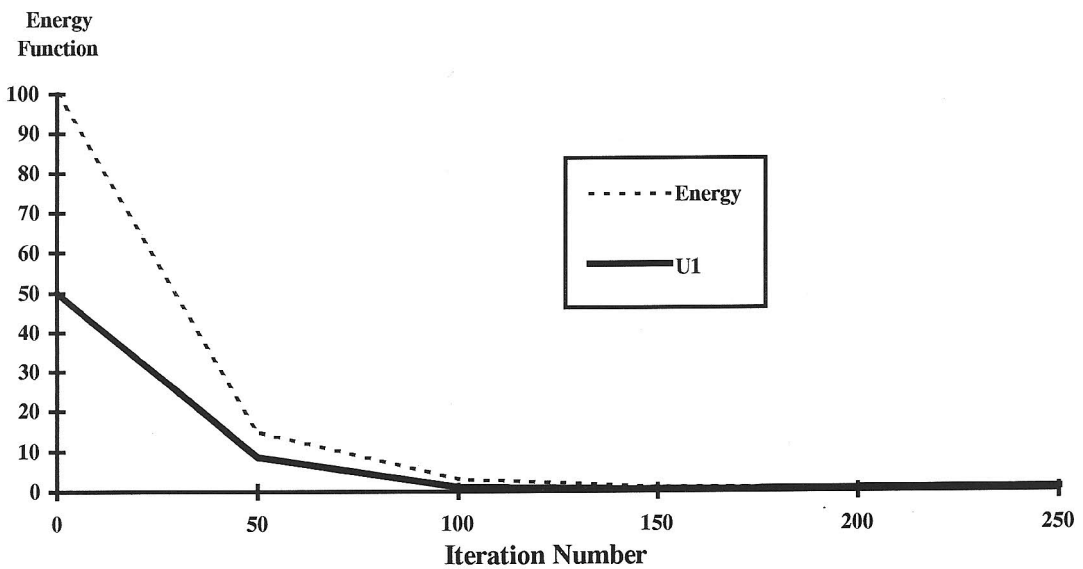

Fig. 1. Simulation of HANN prepared for the constraint $\sum_{m=1}^{M} x_m = k$

Increasing of gain coefficients in activation functions accelerates the convergence of network to the equilibrium point. For the gain parameter γ=100 in activation functions and for the length of integrity step is assumed as 0.2 presented HANN can obtain the equilibrium point only after 3 steps. Even for software implementation of HANN above results are very promising. Networks constructed for performing the constraint $\sum_{m=1}^{M} x_m = k$ can be denoted HANN/k/M.

4. GENETIC ALGORITHM USING PHANN

Now, the new genetic algorithm with using PHANN with given parameters is presented.

---

1. i:=1
2. Set the number of maximum iterations I.
3. Choose a ***PHANN*** population size 2K.

4. Randomly generate the ***initial states*** in the PHANN population.
5. ***Local neural optimization*** in the population of PHANN.
6. Evaluate the objective function Q (nonnegative convex combination function) for each PHANN.
7. Choose the PHANN x* with the minimum of the objective function value Q*.
8. ***Crossover*** K randomly chosen pairs of PHANN ***initial states*** from networks population to receive K pairs of offspring by the Goldberg operator PMX.
9. ***Mutate*** randomly chosen individuals of the PHANN population with the low probability p_m.
10. Local neural optimization in the population of PHANN.
11. Evaluate the objective function Q (nonnegative convex combination function) for each PHANN.
12. Choose the PHANN x** with the minimum of the objective function value Q**.
13. If x** dominates x* in Pareto sense, then i:=i+1, otherwise i:=0.
14. If i>I then STOP. Otherwise go to the step 8.

Initial states of activation levels in artificial neural networks are very important factors for obtaining of optimal solutions. Moreover, they influence on the convergence speed of network to an equilibrium point. On the fig. 2 we can observe examples of several convergence a network HANN/1/50 for different values of initial state vector u.

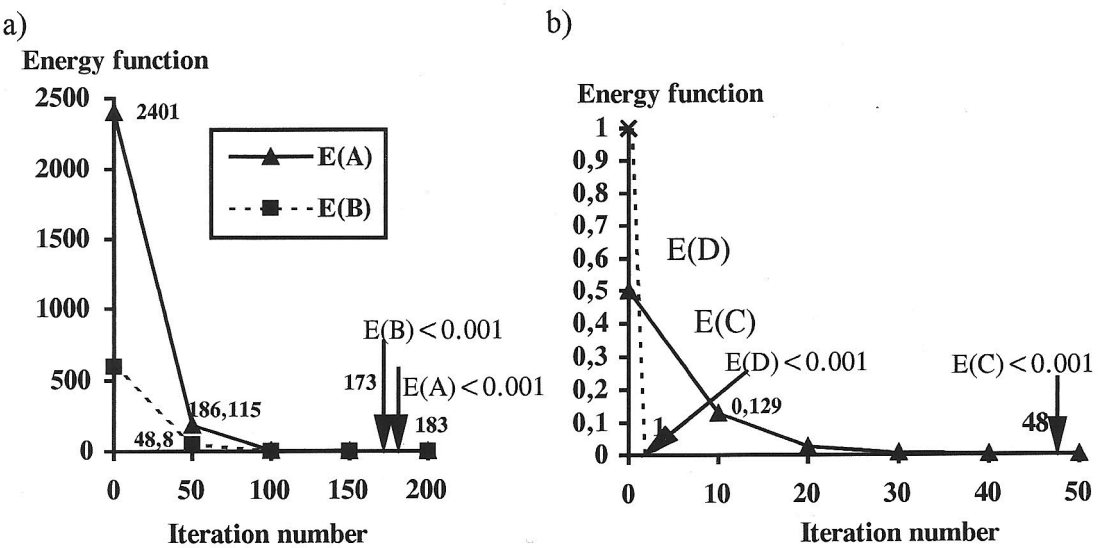

Fig. 2. Convergences of HANN/1/50 for different initial states:
a) $A=[50, 49, \ldots, 2, 1]^T$, $B=[25,24,23,\ldots,-23, -24]^T$,
b) $C=[0, -1,\ldots, -48, -49]^T$, $D=[1.25, -0.76, -0.75,\ldots, -1.23, -1.24]^T$

Similarly to the gradients minimization methods, results of artificial neural networks are related with the value of initial point. This is a main reason for taking these as a subject of GA operations.

5. AN EXAMPLE OF GENETIC-NEURAL METHODS IN RESOURCE DYNAMIC ALLOCATION PROBLEMS

Let us consider an simple example of static resource allocation problem in the multiobjective optimization formulation ($\mathcal{X}$,F,P), where

1) $\mathcal{X}$ - a feasible solutions set

$$\mathcal{X} = \{x \in B^{2(V+J)} \mid \quad x = (x^{m}_{11}, \ldots, x^{m}_{vi}, \ldots, x^{m}_{VI}, x^{\pi}_{11}, \ldots, x^{\pi}_{ij}, \ldots, x^{\pi}_{2J})^{T};$$

$$\sum_{i=1}^{2} x^{m}_{vi} = 1 \qquad v = \overline{1, V}; \qquad \sum_{j=1}^{J} x^{\pi}_{ij} = 1 \qquad i = \overline{1, 2;}\} \tag{4}$$

2) F - a quality criterion

$$F : \mathcal{X} \to R^2, \quad F(x) = [F_1(x), F_2(x)]^T \quad x \in \mathcal{X}$$

$$F_1(x) = \sum_{j=1}^{J}\sum_{v=1}^{V}\sum_{i=1}^{2} t_{vj} x^{m}_{vi} x^{\pi}_{ij} + \sum_{v=1}^{V}\sum_{u=1}^{V}\sum_{i=1}^{2} \tau_{vu} x^{m}_{vi}(1 - x^{m}_{u,3-i}), \quad F_2(x) = \sum_{i=1}^{2}\sum_{j=1}^{J} \kappa_j x^{\pi}_{ij} \tag{5}$$

3) P - a Pareto relationship [1].

Above static problem is solved several times during program performing. Moreover, we assume that execution times t_{vj} of module (operation) m_v on processor π_j are given. In two nodes have to be installed two processors. There is one processor in one node. Processors can perform operations with the different speed. So, we distinguish several sorts of processors π_j with the given cost κ_j. If modules are processed in different nodes, they can require additional given times τ_{vu} to satisfy intermodule references.

Decision variables are x^{m}_{vi} and x^{π}_{ij}. The variable x^{m}_{vi} is equal to 1, if the module m_v is assigned to the node w_i. Similarly, the variable x^{π}_{ij} is equal to 1, if the processor π_j is assigned to the node w_i. In other cases decision variables are equal to zero. The Pareto relationship forces to minimize both the time of execution assigned operations F_1 and the cost of assigned processors F_2. There is a general conflict between minimization the time of module assignment and the cost of processor assignment, because processors with better performance abilities are more expensive. Additionally, some modules can use several properties of special processors to decrease the time of a run operation. Then exceptionally, the run time for a given module on the cheaper processor can be shorter then the run time for the more expensive one.

For solving considered problem we can use several techniques based on the Stone's method, gradient minimization methods, Hopfield's analog networks, simulated annealing or genetic algorithms. On the fig. 3 an example of a criterion space $\mathcal{Y}$=F($\mathcal{X}$) is presented for the following parameters:

V=4, J=3, I=2,

$$T = \begin{bmatrix} 2{,}5 & 1{,}3 & 4{,}0 \\ 1{,}4 & 8{,}9 & 2{,}0 \\ 1{,}2 & 5{,}2 & 3{,}0 \\ 2{,}2 & 1{,}1 & 2{,}0 \end{bmatrix}, \quad \tau = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$\kappa=[1,2,3]^T$.

For above particular assumptions the number of decision variables is equal to M=I(V+J)=14. There are 79 points in criterion space $\mathcal{Y}$. The set $\mathcal{Y}$ has been generated by the modified greedy search method. From the permutations set number 2^{14}=16384 only 256

solutions has been chosen by the directional evaluation. Set $\mathcal{X}$ of feasible solutions consists of 147 points.

Moreover, on the fig. 3 results of a simple genetic algorithm SGA [7] are presented. Non-dominated points for each population are connected by a dashed line with the number of population. SGA used 67 new populations to obtain two suboptimal in Pareto sense solutions presented as points C and D. Each population consisted of 200 solutions. A probability of crossovering for solutions from a gene pool was taken p_c=0,95. A mutation probability for one bit was chosen p_m=0,001. A distribution of solutions in an initial population was uniform. Reached solutions are good, because they are feasible and close to Pareto-optimal solutions. However, Pareto-optimal solutions should be preferred. So, we can use the genetic-neural algorithm GNA to eliminate above disadvantage.

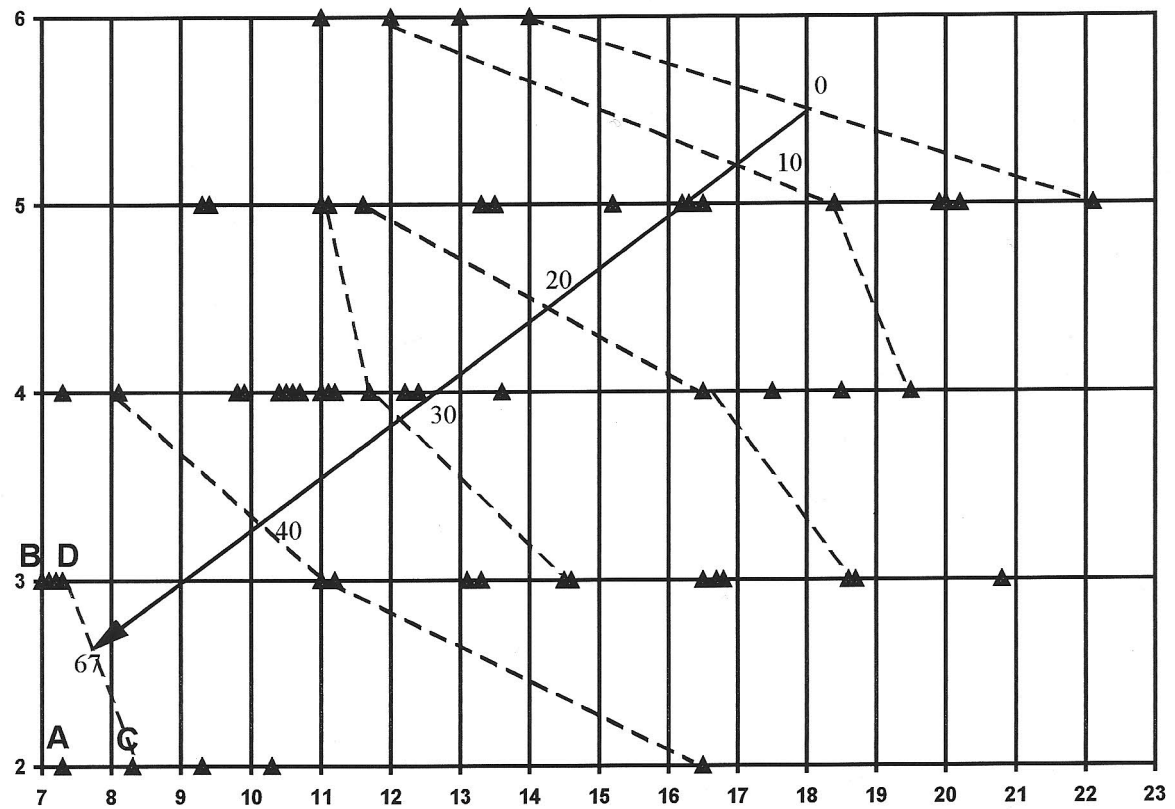

Fig. 3. An example of a criterion space $\mathcal{Y}$

Neural-genetic algorithm GNA is very efficient for solving such optimization problems. In the considered example it found two Pareto-optimal solutions with the evaluations A and B by generating only 12 new populations. Each population consisted of 50 artificial neural networks PHANN dedicated for solving considered optimization problem. GNA changed values of activation levels. So, we can consider GNA as a simply genetic algorithm SGA operated on the initial activation levels of a system consisted of 50 isolated neural networks. After selection, crossover, and mutation states of initial activation levels are changed. Then 50 dedicated neural networks obtain their equilibrium points. Afterwards, fitness function values are calculated and new population of initial activation levels. The main advantage of this approach is improving solutions by neural networks. If neural networks are implemented as VLSI chips, then time of obtaining equilibrium points can be approximately equal to one computer instruction time. Even for simulation of networks by program environment based on computers IBM PC presented GNA method is very powerful.

6. CONCLUDING REMARKS

In this paper, genetic-neural methods for solving dynamic modular program allocation problem have been proposed. PHANN based on nonnegative convex method algorithm for finding Pareto-optimal solution has been presented. Initial experimental results confirm, that genetic-neural algorithms GNA are very useful for solving optimization problems, and especially for solving multiobjective optimization problems.

REFERENCES

[1] Ameljańczyk, A. (1986): „Multicriteria Optimization". WAT, Warszawa, (in polish).
[2] Bac, F. Q. and Perov V.L. (1993): „New Evolutionary Genetic Algorithms for NP-Complete Combinatorial Optimization Problems". Biological Cybernetics, Vol. 69, pp. 229-234.
[3] Balicki, J. (1995): „Artificial Neural Networks for Multicriteria Optimization Problems of Program Modules Allocations". In Proc. of the 8th International Symposium on System-Modelling-Control, Vol.3, (Zakopane, Poland, May), pp.1-6.
[4] Domański, P.D. and Arabas, J. (1995): „A Genetic Approach to the Linguistic Modelling". In Proc. of The 8th International Symposium on System-Modelling-Control, (Zakopane, Poland, May), pp. 224-229.
[5] Fogel D.B., „An Evolutionary Approach to The Traveling Salesman Problem". Biol Cybern 60, (1988), pp. 139-144.
[6] Fonseca, C.M. and Flaming, P.J.: „An Overview of Evolutionary Algorithms in Multiobjective Optimization". Evolutionary Computation". Vol. 3, n° 1, pp. 1-16.
[7] Goldberg, D.E. (1989): „Genetic Algorithms in Search, Optimization, and Machine Learning". Addison_Weslej Publishing Company, Inc. Reading, Massachusetts, USA.
[8] Goldberg, D.E. and Lingle, R. (1985): „Alleles, Loci, and The Traveling Salesman Problem". In Proc. of the International Conference on Genetic Algorithms and Their Applications, Carnegie Mellon University, (Pitsburg), pp.154-159.
[9] Hertz, J. *et. al.* (1993): „An Introduction to Neural Calculations Theory". WNT, Warszawa. (in polish).
[10] Holland, J.H. (1975): „Adaptation in Natural and Artificial Systems". The University of Michigan Press, Ann Arbor.
[11] Kirkpatrick, S. *et. al.* (1985): „Optimization by Simulated Annealing". Science, Vol. 220, pp. 671-680.
[12] Korbicz, J. *et. al.* (1994): „Artificial Neural Networks". AOW PLJ, Warszawa, (in polish).
[13] Lawler, E.L. *et. al.* (1989): „Sequencing and Scheduling. Algorithms and Complexity." Report BSR8909, Center for Mathematics and Computer Science, Amsterdam.
[14] Liepins, G,E. and Hilliard M.R. (1989): „Genetic Algorithms. Foundations and Applications". Ann. Oper. Res., Vol. 21, pp. 31-58.
[15] Lin, S. and Kernigham, B.W. (1973): „An Effective Heuristic Algorithm for The Traveling Salesman". Oper. Res., Vol. 21, pp. 498-516.
[16] Metropolis, N. *et. al.* (1989): „Equations of State Calculations by Fast Computing Machines". J. Chem. Phys., Vol. 21, pp. 1087-1091
[17] Michalewicz, Z. (1992): „Genetic Algorithms + Data Structures = Evolutionary Programs". Springer Verlag.
[18] Schaffer, J.D. (1985): Multiple Objective Optimization with Vector Evaluated Genetic Algorithms". In Proc. of an International Conference on Genetic Algorithms and Their Applications, pp. 93-100.
[19] Schaffer, J.D. and Grefenstette, J.J. (1985): „Multi-objective Learning via Genetic Algorithm". In Proc. of the (th International Joint Conference on Artificial Intelligence, Vol. 1, pp. 593-595.
[20] Tadeusiewicz, R. (1993): „Neural Networks". AOW, Warszawa, (in polish).