Evolutionary Algorithms for Program Module Assignment by Multicriteria Approach

JERZY BALICKI, ZYGMUNT KITOWSKI
The Polish Naval Academy,
Œmidowicza St., 81-919 Gdynia,
POLAND

Abstract - Design problems of distributed computer system are formulated as multiobjective combinatorial optimization tasks, which can be solved by modern heuristic techniques. Software task allocations in multiple computer systems can decrease the total time of program execution by taking advantage of the specific efficiencies of some computers or advantage of load computer states. In a network of workstations or personal computers, two or more program modules may execute concurrently for various periods. A program module is a collection of procedures or subroutines, or could be data files. In the paper, an evolutionary algorithm is applied as a novel approach to a multicriteria assigning of program modules in distributed computer systems. For quality evaluation of module assignment two criteria are used: a total cost of program processing and a performance of a distributed computer system. Then an evolutionary algorithm for finding Pareto-optimal solutions is proposed. Some results of numerical simulations are presented.

Key-Words: - Multiobjective optimization, distributed computer systems, evolutionary algorithms
CSCC'99 Multiconference Proceedings: - Pages: 7571-7576

1. Introduction

In a distributed computer system, another way for the minimization of the total time of program execution is to change the computers for module processing. So, the computer with a power floating point unit can be dedicated for performing tasks with several numerical procedures, and the graphic workstation is suitable for program modules with animation and chart processing. Operating and configurations of several distributed computer systems are very complex. For an assessment of their quality several parameters can be used. If there are considered the set of feasible alternatives of the system structure, we have to solve the instance of multicriteria optimization problem [1].

Genetic algorithms, evolutionary strategies and evolutionary algorithms are the alternative approaches to the other heuristic optimization methods such, as simulated annealing, tabu search, Hopfield models of neural networks, or Lagrangean relaxation. It compels a discussion about the quality of solutions obtained by one of above optimization techniques. A crucial result of studies is using evolutionary algorithms for solving multiobjective optimization problems. There are differences between genetic algorithms, evolutionary algorithms and evolutionary strategies.

Evolutionary algorithms develop genetic algorithms for solving optimization problems by another chromosome representation, more complex operators, and a specific knowledge related with the optimization problem [7].

2. A model of parallel processing

The standard problem of program module allocations is a question how to find the allocation of program modules to minimize the program execution cost. An objective of optimization is the time of program performing, too. Another measure is the amount of computer resource using.

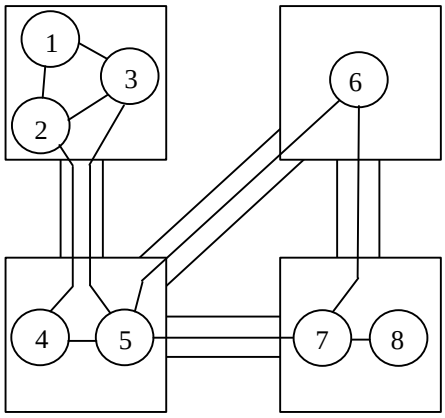

Fig. 1. Allocation of program modules in the distributed computer system with 4 computers

On the fig. 1 an example of distributed program in a computer network with 4 computers is presented. The program was divided on 8 modules

M_1, M_2,..., M_8. Module M_4 is assigned to a computer with the module M_5. It is reasonable solution, if between above pairs of modules great number of interactions is required.

A program module can be activated several times during the program lifetime. The process (operation) is the processing of one activated program module. With the module can be associated some processes. In results, a set of program modules $\{M_1,...,M_m,...,M_M\}$ is mapped to a set of processes $\{m_1,...,m_v,...,m_V\}$ A process can be split on the threats, this parallel processing technique for speed up is not considered.

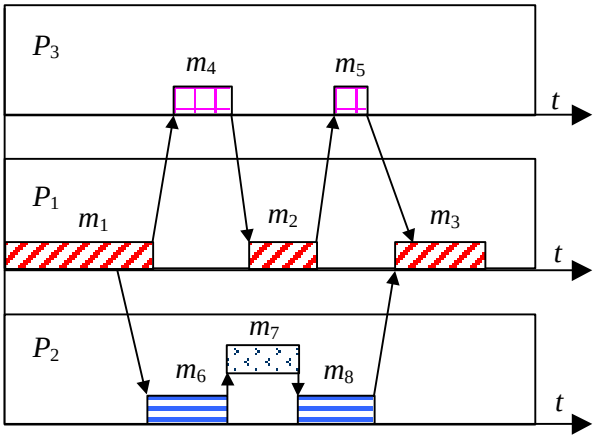

Fig. 2. The performing of a parallel program consisted of 4 modules by 3 computers

On the fig. 2, the performing of a parallel program consisted of 4 modules is presented. The modules are allocated on 3 computers P_1, P_2 and P_3. The module M_1 starts (the process m_1) and finishes (the process m_3) the whole parallel program on the computer P_1. Moreover, the module M_1 (the operation m_2) calls the module M_2 (the operation m_6) on the computer P_2. Then, the module M_1 (the operation m_2) transfers its activity to the module M_4 (the operation m_4) on the computer P_3. Afterwards, the module M_4 (the operation m_4) returns an activity to the module M_1 (the process m_2). In above scheme the processing of parallel program is carried out.

3. Computer allocations

Let us assume that the process m_v can be executed on several sorts of computers taken from the set $\Pi = \{p_1,...,p_j,...,p_J\}$. Computers are able to operate in the fixed nodes included to the set $W = \{w_1,...,w_i,...,w_I\}$, only. On the fig. 2, there are 3 nodes w_1, w_2, w_3. In the node w_1, the computer P_1 is situated. P_1 is taken from the set $\Pi = \{p_1,...,p_j,...,p_J\}$. Computers, which are located in different nodes, can represent the same sort. Above assumption gives the possibilities of consideration, which computer sort is more convenient in any node.

Each pair of computers situated in nodes can communicate to support performing of program module interactions. Furthermore, in each node one and only one computer should be allocated. It implies the computer allocation constraints, as follows:

$$\sum_{j=1}^{J} x_{ij}^{p} = 1,\ i = \overline{1,I}. \qquad (1)$$

where

$$x_{ij}^{p} = \begin{cases} 1 \text{ if } p_j \text{ is assigned to the } w_i \\ 0 \text{ in the other case,} \end{cases}$$

A vector, as below, can describe an allocation of computers:

$$x^{p} = [x_{11}^{p},...,x_{1j}^{p},...,x_{1J}^{p},...,x_{ij}^{p},...,x_{I1}^{p},...,x_{Ij}^{p},...,x_{IJ}^{p}]^{T}, \qquad (2)$$

4. Process allocation constraints

An optimal process allocation should be found for parallel program execution time minimization. A vector can determine process allocation, as follows:

$$x^{m} = [x_{11}^{m},...,x_{1i}^{m},...,x_{1I}^{m},...,x_{vi}^{m},...,x_{VI}^{m}]^{T}, \qquad (3)$$

where

$$x_{vi}^{m} = \begin{cases} 1 \text{ if module } m_v \text{ is assigned to } w_i, \\ 0 \text{ in the other case,} \end{cases}$$

Because each process is allocated to one node, then the process allocation constraints are formulated, as below:

$$\sum_{i=1}^{I} x_{vi}^{m} = 1,\ v = \overline{1,V}. \qquad (4)$$

Now, the following vector can write the allocation of operations to computers:

$$x = [x_{11}^{m},...,x_{1i}^{m},...,x_{1I}^{m},...,x_{vi}^{m},...,x_{VI}^{m}, x_{11}^{p},...,x_{1j}^{p},...,x_{1J}^{p},...,x_{ij}^{p},...,x_{I1}^{p},...,x_{Ij}^{p},...,x_{IJ}^{p}]^{T} \qquad (5)$$

Constraints (1), (4) reduce the number of allocations x from $a=2^{I(V+J)}$ to $b=I^{V}J^{I}$. Let us consider the simplest distributed computer system with 2 computers (I=2). The number of allocations of operations to 2 computers with respect operation number V and computer sorts J=2 is shown in the

table I. Grey cells denote numbers of allocations, which can be compared to find an optimal allocation in 1 hour. For evaluation, a modeling computer comparing 1 pair of allocations during 1 ns [10^{-9} s] was taken. If the number of processes is not smaller than 100, then it is impossible to find optimal allocation by comparing all combinations during reasonable time.

Tab. I. *A number of allocations of operations to 2 computers with respect operation number V*

V	Number of allocations	
	a	b
2	64	16
10	$1.678*10^{7}$	$4{,}096*10^{3}$
100	$2.571*10^{61}$	$5{,}07*10^{30}$
1000	$1.971*10^{603}$	$1.286*10^{301}$

5. Resource constraints

Each computer has resources needed for a program performing. A program module reserves an operational memory during its processing. Also the reserved memory size by a module can be changed during a module execution, then a maximal amount can be estimated. Another resource is a size of hard discs. If modules share the other sort of memories (a tape memory, the ZIP memory etc.), then the capacities of memories can not be exceeded. In the distributed computer system are available the following memories $z_1,...,z_r,...,z_R$. Computers can be equipped in different amounts of memories. So, d_{jr} denotes the capacity of memory z_r in the computer p_j . The value d_{jr} is nonnegative and limited. We assume that the operation m_v reserves c_{vr} units of memory z_r and holds it during a program execution. The value c_{vr} is nonnegative and limited, too.

The memory limit in any computer assigned to the ith node can not be exceeded, what is written, as bellows:

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \le \sum_{j=1}^{J} d_{jr} x_{ij}^{p}, \quad i=\overline{1,2}, \ r=\overline{1,R}. \quad (6)$$

A program module can require the subroutine library, a specific software environment, a heavy duty printer, a high resolution monitor, or the other components. Let $k_1,...,k_s,...,k_S$ denote required components of computers. We assume that the following component coefficients are given:

$$c'_{vs} = \begin{cases} 1 \text{ if } m_V \text{ requires computer component } k_S, \\ 0 \text{ in the other case,} \end{cases}$$

$$d'_{js} = \begin{cases} V \text{ if } p_j \text{ has the component } k_S, \\ 0 \text{ in the other case,} \end{cases}$$

Operational requirements related with the access to computer components can be formulated, as below constraints:

$$\sum_{v=1}^{V} c'_{vs} x_{vi}^{m} \le \sum d'_{js} x_{ij}^{p}, \quad i=\overline{1,2}, \ s=\overline{1,S}. \quad (7)$$

6. Problem formulation

The first criterion used for an allocation evaluation is the cost of parallel program performing, which can be calculated, as below:

$$F_1(x) = \sum_{j=1}^{J}\sum_{v=1}^{V}\sum_{i=1}^{I} t_{vj} x_{vi}^{m} x_{ij}^{p} + \sum_{v=1}^{V}\sum_{u=1}^{V}\sum_{i=1}^{I} t_{vu} x_{vi}^{m} x_{u,3-i}^{m}, \ x \in \mathbb{B}^{M}, \quad (8)$$

where

t_{vj} - the cost of performing the module m_v by the computer p_j,

t_{vj} - the cost of communications between the module m_v and the module m_u,

- the set {0, 1}.

A cost of computers can be calculated according to the following formula:

$$F_2(x^p) = \sum_{i=1}^{I}\sum_{j=1}^{J} k_j x_{ij}^{p}. \quad (9)$$

where k_j represents the cost of computer p_j.

There are several classes of multiobjective optimal solutions related with the preferences for criteria. If criteria are ordered from the most important criterion to the least important criterion, then a hierarchical solution can be found.

If all criteria have the same priority, then Pareto-optimal solutions can be considered. Because of the great number of Pareto-optimal solutions some reducing techniques can be used. For instance, the compromise solutions with the "democracy" parameter p equal 1, 2 or ∞ may be extracted from Pareto set. Moreover, an additional criterion can be used.

Let consider the case of multiobjective optimization problem (χ, F, P) for finding the Pareto-optimal solutions, which are some allocations of program modules and computer types in a distributed

computer system. This allocation of program modules to processors can be called *the operational structure* of processing in distributed computer system. It seems to be very representative for the other combinatorial problems as it was shown in [1]. In this problem some denotations are used, as follows:

1) X - a feasible solutions set

$$X = \{ x \in B^{2I(V+J)} |$$

$$\sum_{i=1}^{I} x_{vi}^{m} = 1, v = \overline{1,V}; \quad \sum_{j=1}^{J} x_{ij}^{p} = 1, i = \overline{1,I};$$

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \le \sum_{j=1}^{J} d_{jr} x_{ij}^{p}, \ i = \overline{1,I}, \ r = \overline{1,R};$$

$$\sum_{v=1}^{V} c_{vs}^{'} x_{vi}^{m} \le \sum d_{js}^{'} x_{ij}^{p}, \ i = \overline{1,I}, \ s = \overline{1,S} \}.$$

2) F - a vector quality criterion

$$F : X \to R^{2}, \qquad (10)$$

where $F(x) = [F_1(x), F_2(x)]^T$ for $x \in X$

$F_1(x)$ is calculated by (8),

$F_2(x)$ is calculated by (9).

3) P - the Pareto relationship [1]

The relationship P for finding Pareto-optimal trajectories is a subset of $Y \acute{} Y$, where $Y=F(X)$. If $a \hat{I} Y$, $b \hat{I} Y$, and $a_n \le b_n$, $n = \overline{1,N}$, then the pair of evaluations $(a,b) \hat{I} P$. Above definition of the Pareto relationship respects the minimization of all criteria. For Pareto-optimal trajectory $x^* \hat{I} X$ there is no trajectory $a \hat{I} X$ such, that $(F(a),F(x^*)) \hat{I} P$.

For solving above optimization problem, three techniques were used. A poor genetic algorithm, an evolutionary algorithm and an evolutionary strategy used similar operators, but there are several differences between them, what cause obtained solutions distinguish in final result. Now, we try to evaluate the quality of above algorithms

7. Genetic algorithm

For solving multiobjective optimization problem (12) with the considered three criteria an evolutionary algorithm can be used. Genetic algorithms are applied for solving several optimization problems. Holland [5] developed this approach and its theoretical foundation. Rosenberg noticed abilities of GA for development many criteria [9].

Schaffer [10] considered GA for solving multiobjective optimization problems by a vector evaluated genetic algorithm VEGA. VEGA is an extension of system GENESIS prepared by Grefenstete [4]. VEGA uses dividing of the population on N subpopulations, where N are the number of criteria. For each nth subpopulation the criterion F_n is a fitness function. But, selection, crossoverring, and mutation are carried out for whole population. This method for fitness evaluation has the disadvantage related with the discrimination of Pareto solutions situated in the interior of the Pareto frontier. Indeed, mainly lexicographic solutions are preferred.

Fourmann considered selection with using hierarchical tournaments, where two randomly chosen solutions are compared, and a hierarchical solution is a winner in this competition, and it is included to a mating pool of potential parents [2]. A selection probability is calculated fot the most important goal. A random choice is carried out twice according to the roulette rule. But similarly to VEGA approach, hierarchical tournaments for target preferences set a'priori supports the solution migration towards lexicographical solutions.

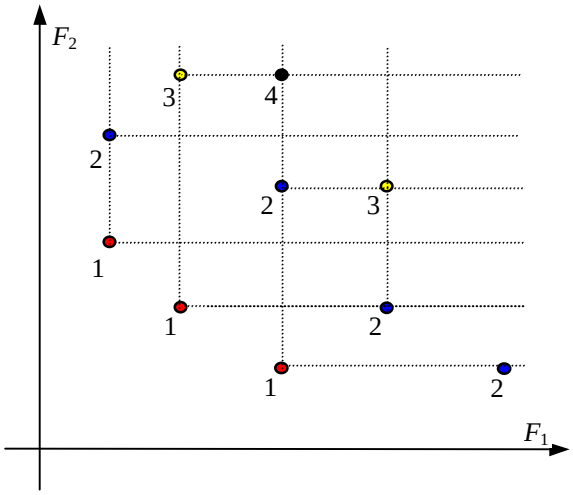

Fig. 3. *Ranking of solution evaluation in a population*

Another Fourmann's selection is based on random choice of a goal for comparison of a taken solutions [2]. Selection probabilities are the same or it can be related with the chosen goal for the other version of selection.

To avoid the discrimination of the interior Pareto solutions Goldberg introduced the ranking system for non-dominated individuals, which is similar to the ranking system for one function [3]. If there are some feasible solutions in a population, then the Pareto-

optimal individuals are sought, and they get the rank 1 (fig. 3). Then they are temporary eliminated from the population. From reduced population the new Pareto-optimal trajectories are found and get the rank 2. This procedure with increasing of the rank is repeated until the set of feasible solutions will be exhausted. That is why, all nondominated solutions have the same rank and the same fitness to reproduction.

If x is non-feasible, it gets the fitness function value $f(x)=1$. If x is feasible, then the fitness function value is calculated, as below:

$$f(x) = -r(x) + L + 1. \qquad (11)$$

where $r(x)$ denotes the rank of a feasible solution.

A genetic algorithm for multicriteria optimization problem can be presented on the fig. 4.

BEGIN
  t:=0, set the size of population L
  randomly generate initial population $\boldsymbol{P(t)}$
  calculate ranks $r(x)$ and fitnesses $f(x)$, $x \in \boldsymbol{P}(t)$
  finish:=FALSE
  WHILE NOT *finish* DO
  BEGIN /* new population */
    t:= t+1, $\boldsymbol{P}(t) := \varnothing$
    calculate selection probabilities $p_S(x), x \in \boldsymbol{P}(t-1)$
    FOR $L/2$ DO
    BEGIN /* reproduction cycle */
      - proportional selection of a potential parent pair (**a,b**) from the population $\boldsymbol{P}(t-1)$
      - simple crossoverring of a parent pair (**a,b**) with assumed crossover probability p_c
      - bit mutation of an offspring pair (**a',b'**) with assumed mutation probability p_m
      - $\boldsymbol{P}(t) := \boldsymbol{P}(t) \cup$ (**a',b'**)

    END
    calculate ranks $r(x)$ and fitnesses $f(x), x \in \boldsymbol{P}(t)$
  IF ($\boldsymbol{P}(t)$ converges OR $t \geq T_{max}$) THEN *finish*:=TRUE
  END

Fig. 4. *Genetic algorithm for multicriteria problems*

Genetic algorithm (GA) in a standard form from fig. 4 is an general approach for solving a wide class of multiobjective optimization problems. If there is software implementation of GA for one criterion optimization problems, then the ranking procedure has to be added before fitness calculation. Binary vectors represent solutions. So, for several combinatorial tasks GA is suitable for applying. Especially, GA can solve the problem (10). Simulation results confirmed, that GA is capable for finding suboptimal in Pareto sense solutions, but the quality of them was non-satisfying. Several experiments with changing the parameters of GA such, as the crossover probability, the mutation probability or selection rules were done.

On the fig. 5, a level of convergence to Pareto frontier is shown for different values of the crossover probability p_c. The best performance was done for the reasonable value p_c=0.2. The size of population L=10 and the mutation probability p_m=0.05. The number of binary decision variables (the length of chromosome) is 24. A search space has 2^{24} elements for this instance.

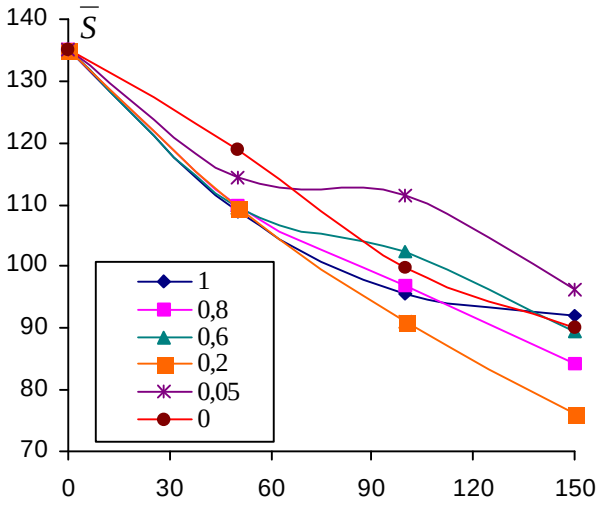

Fig. 5. Finding crossover probability p_c

The level of convergence to Pareto frontier is a closeness measure the nondominated points obtained by GA to the Pareto points $\{P_1, P_2, ..., P_U\}$. For this sort of optimization problem it can be taken, as follows:

$$\overline{S} = \sum_{u=1}^{U} | P_{u1} - A_{u1} |. \qquad (12)$$

where A_{1u} is a best cost of program performing found by GA for the cost of distributed computer system $A_{2u} = P_{2u}$.

8. Evolutionary algorithm

An overview of evolutionary algorithms for multiobjective optimization problems is presented by Fonseca and Flaming [2]. In an evolutionary algorithm (EA) some specific knowledge about considered optimization problem is respected. So, GA can be used for solving the wide class of problems, and EA is rather focused on the special case of task, only. But usually, results are much

better.

A logical scheme is similar to GA, but initial population is constructed to individuals satisfy the constraints $\sum_{i=1}^{I} x_{vi}^{m} = 1, v = \overline{1,V}; \ \sum_{j=1}^{J} x_{ij}^{p} = 1, i = \overline{1,I}$ by introducing integer representation, as follows:

$$X = (X_1^m, ..., X_v^m, ..., X_V^m, X_1^p, ..., X_i^p, ..., X_J^p), \quad (13)$$

where $X_v^m = i$ for $x_{vi}^m = 1$ and $X_i^p = j$ for $x_{ij}^p = 1$

Moreover, $0 < X_v^m \le I$ and $0 < X_i^p \le J$. A simple crossover operator is used, but a bit mutation is substituted by the random exchange of integer value by another from a feasible discrete set. If solution is non-feasible, then the penalty is calculated. The fitness for a non-feasible solution is equal to the difference between maximal penalty in population The fitness for a feasible solution is created by adding maximal penalty in population to a term $-r(x)+L+1$. This evolutionary algorithm gives much better results than GA (fig. 6).

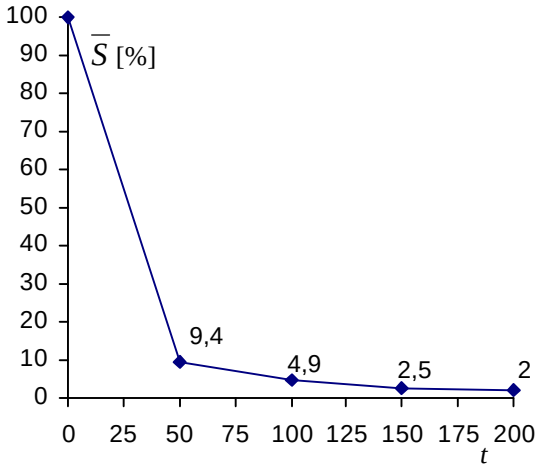

Fig. 6. *Minimization of the the average level of convergence to Pareto frontier by EA*

9. Evolutionary strategy

For solving optimization problems an evolutionary strategy was proposed and developed by Rechenberg [8] and Schweffel [11]. An extension of evolutionary strategies on multi-objective optimization was introduced by Kursawe [6].

Chromosome in evolutionary strategies consists of two main parts, as follows:

$$\overline{X} = (x, s), \quad (14)$$

where

x - decision variable vector,

s - deviation standard vector for x.

The diagram of evolutionary strategy EA is shown in a version $(m+l)$ [7]. A strategic mutation of chromosome changes a value of decision variable x_m by randomly chosen number Dx_m representing value of random variable with a normal distribution $N(0, s_m)$.

10. Concluding remarks

Techniques for solving related multiobjective optimization problems can use the proposed the evolutionary algorithm or evolutionary strategy "with plus". The presented approach seems to be very elastic for adaptation in the other cases.

In the evolutionary algorithm some new mutation operators can be introduced to satisfy some constraints. Moreover, evolutionary strategies can be compared to evolutionary algorithms. A canonical genetic algorithm gave worse results, because it is more general and it do not use the knowledge related with the specific optimization problem.

References:

[1] J. Balicki, Z. Kitowski, A. Stateczny: *Extended Hopfield Model of Neural Networks for Combinatorial Multiobjective Optimization Problems*, Proc. of 1998 IEEE World Congress on Computational Intelligence - IJCNN, Anchorage, May 1998, vol. 2, pp. 1646-1651.

[2] C.M. Fonseca, P.J. Fleming, *An overview of evolutionary algorithms in multiobjective optimization*, Evolutionary Computation, vol. 3, No. 1, 1995, pp.1-16.

[3] D.E. Goldberg, R. Lingle, *Alleles, loci, and the traveling salesman problem*, in Proc. of the Int. Conf. on Genetic Algorithms and Their Applications, Carnegie Mellon University, Pitsburg, 1985, pp. 154-159.

[4] J.J. Grefenstette, GENESIS: *A system for using genetic search procedures*. Proc. of Int. Conf. on Genetic Algorithms and Their Applications, 1984, 161-165.

[5] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[6] E. Kursawe, *A variant of evolution strategies for vector optimization.* In H.-P. Schwefel, R. Manner (Eds.): *Parallel problem solving from nature*, 1st Workshop, Lecture Notes in Computer Science, Vol. 496, 1991, pp. 193-197.

[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer Verlag, 1996

[8] I. Rechenberg, *Evolutionstrategie. Optimie-rung technischer systems nach printzipien der biologischen evolution.* Frommann-Holtzboog Verlag, Stuttgart 1973.

[9] R.S. Rosenberg, *Simulation of genetic populations with biochemical properties*. *I. The Model*, Mathematical Biosciences, vol. 7, pp. 223-257.

[10] J.D. Schaffer, *Multiple objective optimization with vector evaluated genetic algorithm*, Proc. Int. Conf. on Genetic Algorithms and Their Applications, 1985, pp. 93-100.

[11] H.P. Schwefel: *Evolution and optimum seeking.* John Wiley&Sons, Chichester 1995.