Journal of Computational and Applied Mathematics 235 (2011) 1446–1453

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems

W.F. Abd-El-Wahed [a], A.A. Mousa [b,*], M.A. El-Shorbagy [b]

[a] Faculty of Computers and Information, Shebin El-Kom Minufiya University, Egypt

[b] Department of Basic Engineering Science, Faculty of Engineering, Shebin El-Kom Minufiya University, Egypt

ARTICLE INFO

Article history:
Received 8 May 2009
Received in revised form 7 August 2009

Keywords:
Particle swarm optimization
Genetic algorithm
Nonlinear optimization problems
Constriction factor

ABSTRACT

Heuristic optimization provides a robust and efficient approach for solving complex real-world problems. The aim of this paper is to introduce a hybrid approach combining two heuristic optimization techniques, particle swarm optimization (PSO) and genetic algorithms (GA). Our approach integrates the merits of both GA and PSO and it has two characteristic features. Firstly, the algorithm is initialized by a set of random particles which travel through the search space. During this travel an evolution of these particles is performed by integrating PSO and GA. Secondly, to restrict velocity of the particles and control it, we introduce a modified constriction factor. Finally, the results of various experimental studies using a suite of multimodal test functions taken from the literature have demonstrated the superiority of the proposed approach to finding the global optimal solution.

1. Introduction

Constrained optimization problems, especially nonlinear optimization problems, where objective functions are minimized under given constraints, are very important and frequently appear in the real world.

There are some efficient methods to solve nonlinear optimization problems, such as a recursive quadratic programming, a projection method, and a generalized reduced gradient method [1]. These methods assume the differentiability of the objective function. However, it is difficult to apply these methods to problems, of which the objective function is not differentiable or the feasible set is not convex.

Generally, the constrained problems are solved by the combination of a transformation method and a direct search method. The transformation method [1] converts a constrained problem into an unconstrained one, and the direct search method optimizes the objective function by using only the value of it. There are some transformation methods such as a penalty method and a multiplier method. The penalty method is often used to solve optimization problems, because the solutions are often near the boundary of the feasible set and the method is used easily for its simplicity. However, it is difficult to know what value of the penalty coefficient leads to a feasible solution and how much a search point satisfies the constraints [1].

The use and development of heuristic-based optimization techniques have significantly grown. Since they use a population of solutions in their search, it is more likely to find the global solution of a given problem. In addition, it uses only a simple scalar performance measure that does not require or use derivative information. Particle swarm optimization method is one of heuristic-based optimization techniques and was successfully applied in many optimization tasks.

* Corresponding author. Tel.: +20 101296232.
E-mail address: A_mousa15@yahoo.com (A.A. Mousa).

0377-0427/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2010.08.030

Generate an initial population;
Evaluate fitness of individuals in the population;
Do:
    Select parents from the population;
    Recombine (mate (crossover and mutation operators)) parents to produce children;
    Evaluate fitness of the children;
    Replace some or all of the population by the children;
while a satisfactory solution has been found.

Fig. 1. The pseudo code of the general GA algorithm.

GA and PSO are two of the heuristic-based optimization techniques. they are much similar in their inherent parallel characteristics, whereas experiments [2] show that they have their specific advantages when solving different optimization problems. Compared with GA, PSO has some attractive characteristics [3,4]. It has memory, so knowledge of good solutions is retained by all the particles; whereas in GA, previous knowledge of the problem is discarded once the population changes. It has constructive cooperation between particles; that is, particles in the swarm share information among themselves. To date, PSO has been successfully applied to optimizing various continuous nonlinear functions in practice [5].

In recent years there have been a lot of reported works focused on the hybridization of PSO with other heuristic-based optimization techniques. In [6], PSACO (particle swarm ant colony optimization) algorithm was proposed for highly non convex optimization problems. Also in [7], GA has been incorporated into PSO as a hybrid method combining two heuristic optimization techniques for the global optimization of multimodal functions.

The major objective of this paper is to propose a new hybrid algorithm to benefit by the advantages of the two heuristic optimization techniques, PSO and GA.

This paper is organized as follows. In Section 2, nonlinear programming problem is described. Section 3, provides an overview of the genetic algorithms and particle swarm optimization. In Section 4, the proposed algorithm is presented. The results of various experimental studies using a suite of multimodal test functions are described in Section 5. Finally, Section 6 gives a brief conclusion about this study.

2. Nonlinear programming problem (NLPP)

The nonlinear programming problem can be defined as follows [8]:

$$\begin{aligned} NLPP: \quad & \text{Min} \quad f(\bar{x}) \\ & \text{s.t.} \quad F = \{\bar{x} \in R^n | g_i(\bar{x}) \le 0,\ i = 1, 2, \ldots, m\}, \\ & \qquad S = \{\bar{x} \in R^n | l(x_i) \le x_i \le u(x_i),\ i = 1, 2, \ldots, n\} \end{aligned} \tag{1}$$

where $\bar{x} \in \mathbb{F} \subseteq \mathbb{S}$. The set $\mathbb{S} \subseteq \mathbb{R}^n$ defines the search space and the set $\mathbb{F} \subseteq \mathbb{S}$ defines a feasible part of the search space. Usually, the search space $\mathbb{S}$ is defined as n-dimensional rectangle in $\mathbb{R}^n$ (domains of variables defined as lower and upper bounds): $left\ (i) \le x_i \le right\ (i),\ 1 \le i \le n$ whereas the feasible set $\mathbb{F}$ is defined by the search space $\mathbb{S}$ and an additional set of constraints $g_j(\bar{x}) \le 0$.

3. Overview of the PSO and GA optimization technique

PSO shares many similarities with evolutionary computation techniques such as GA. In this section, we give a brief description of PSO and GA, the reader is referred to [9,10].

3.1. Genetic algorithm (GA)

The discovery of genetic algorithms (GA) was dated to the 1960s by Holland and further described by Goldberg [9]. The GAs have been applied successfully to problems in many fields such as optimization design, fuzzy logic control, neural networks, expert systems, scheduling, and many others [11]. For a specific problem, the GA codes a solution as an individual chromosome. It then defines an initial population of those individuals that represent a part of the solution space of the problem. The search space therefore, is defined as the solution space in which each feasible solution is represented by a distinct chromosome. Before the search starts, a set of chromosomes is randomly chosen from the search space to form the initial population. Next, through computations the individuals are selected in a competitive manner, based on their fitness as measured by a specific objective function.

The genetic search operators such as selection, mutation and crossover are then applied one after another to obtain a new generation of chromosomes in which the expected quality over all the chromosomes is better than that of the previous generation. This process is repeated until the termination criterion is met, and the best chromosome of the last generation is reported as the final solution. Fig. 1 shows the pseudo code of the general GA algorithm.

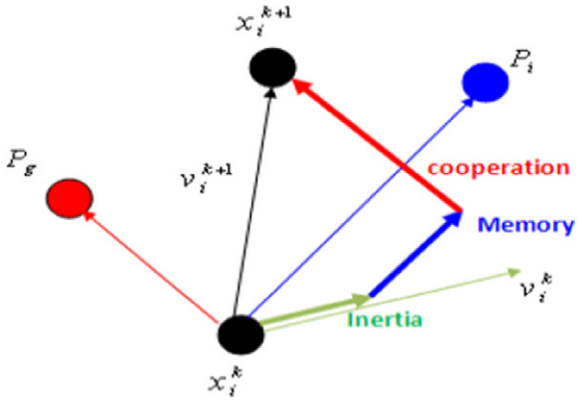

Fig. 2. Description of velocity and position updates in particle swarm optimization for a two-dimensional parameter space.

Randomly initialize positions and velocities of all particles.
Do:
Set P_{best} and G_{best}.
Calculate particle velocity according to equation (2).
Update particle position according to equation (3).
Evaluate the objective function value (fitness value).
while a satisfactory solution has been found

Fig. 3. The pseudo code of the general PSO algorithm.

3.2. The particle swarm optimization method

PSO is an evolutionary computation technique motivated by the simulation of social behavior [10]. Namely, each individual (agent) utilizes two important kinds of information in a decision process. The first one is their own experience; that is, they have tried the choices and know which state has been better so far, and they know how good it was. The second one is other agent's experiences; that is, they have knowledge of how the other agents around them have performed. Namely, they know which choices their neighbors have found are most positive so far and how positive the best pattern of choices was. In the PSO system, each agent makes his decision according to his own experiences and other agent's experiences. The system initially has a population of random solutions. Each potential solution, called a particle (agent), is given a random velocity and is flown through the problem space. The agents have memory and each agent keeps track of its previous best position (called the P_{best}) and its corresponding fitness. There exist a number of P_{best} for the respective agents in the swarm and the agent with greatest fitness is called the global best (G_{best}) of the swarm. Each particle is treated as a point in a n-dimensional space. The ith particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The best previous position of the ith particle (P_{besti}) that gives the best fitness value is represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$. The best particle among all the particles in the population is represented by $P_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$. The velocity, i.e., the rate of the position change for particle i is represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$.

The particles are manipulated according to the following equations (the superscripts denote the iteration):

$$v_i^{k+1} = w \times v_i^k + c_1 \times r_1 \times (p_i - x_i^k) + c_2 \times r_2 \times (p_g - x_i^k), \tag{2}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{3}$$

where $i = 1, 2, \ldots, N$, and N is the size of the population; w is the inertia weight; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; r_1 and r_2 are random numbers uniformly distributed with in the range [0, 1]. Eq. (2) is used to determine the ith particle's new velocity v_i^{k+1}, at each iteration, while Eq. (3) provides the new position of the ith particle x_i^{k+1}, adding its new velocity v_i^{k+1}, to its current position x_i^k. Fig. 2 shows the Description of velocity and position updates of a particle for a two-dimensional parameter space. Fig. 3 shows the pseudo code of the general PSO algorithm.

4. The proposed PSO based on GA algorithm

PSO and GA are much similar in their inherent parallel characteristics, whereas experiments show that they have their specific advantages when solving different problems. What we would like to do is to obtain both their excellent features by integrating the two algorithms. In the proposed approach, the algorithm initialized with a population of random solutions

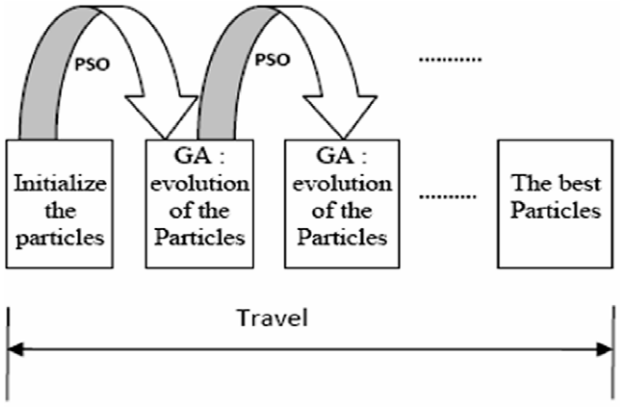

Fig. 4. Diagram model for particle movement.

and searches for optima by traveling into the search space. During this travel an evolution of this solution is performed by integrating PSO and GA. The description diagram of the proposed algorithm is shown in Fig. 4 and it is described as follows:

Step 1. Initialization:

Initialize a population of particles with random positions and velocities on n-dimensions in the problem space.

Step 2. Evaluation:

Evaluate the desired optimization fitness function in n variables for each particle.

Step 3. Setting P_{best} and G_{best}:

Set P_{best} of each particle and its objective value equal to its current position and objective value, and set G_{best} and its objective value equal to the position and objective value of the best initial particle.

Step 4. Updating the velocity and position:

Update the velocity and position of each particle according to Eqs. (2) and (3).

Step 5. Evolution of particles:

To restrict velocity and control it, some authors [12] use a constriction factor χ, which has a constant value to improve the performance of PSO.

We present a modified constriction factor (i.e., dynamic constriction factor) to keep the feasibility of the particles. e.g., Fig. 5 shows the movement of the particle i through the search space with and without a modified factor. Where the particle i start at the position x_i^k with velocity v_i^k in the feasible space, the new position x_i^{k+1} in Fig. 5 depends on velocity v_i^{k+1}.

$$x_i^{k+1} = x_i^k + v_i^{k+1}. \tag{4}$$

Then, v_i^{k+1} makes the particle to lose its feasibility, so we introduce a new modified factor χ such that:

$$\chi = \frac{2}{|-2 - \tau - \sqrt{\tau^2 + \tau}|} \tag{5}$$

where, τ is the age of the infeasible particle (How long it's still unfeasible?). And the new modified position of the particle is computed as:

$$x_i^{k+1} = x_i^k + \chi v_i^{k+1}. \tag{6}$$

For each particle we check its feasibility, if it is infeasible, we implement χ parameter to control its position and velocity, using Eq. (6) where τ is increased with the number of failed trial to keep the feasibility of the particle.

Step 6. Evaluation:

Evaluate the desired optimization fitness function in n variables for each particle.

Step 7. Updating P_{best} and G_{best}:

For each particle, compare its current objective value with the objective value of its P_{best}. If the current value is better, then update P_{best} and its objective value with the current position and objective value. Determine the best particle of the current swarm with the best objective value. If the objective value is better than the objective value of G_{best}, then update G_{best} and its objective value with the position and objective value of the current best particle.

Step 8. Ranking:

Ranks individuals (particles) according to their objective value, and returns a column vector containing the corresponding individual fitness value.

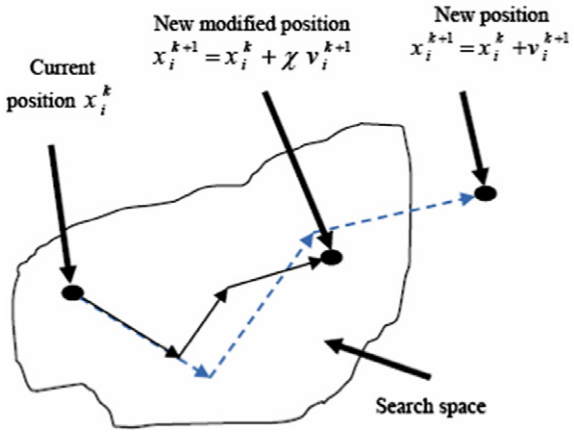

Fig. 5. The movement of the particle *i* through search space.

Step 9. Selection:

Selection is an operator to select two parent strings for generating new strings (*i.e.*, offspring). In the selection, a string with a low fitness value has more chance to be selected as one of parents than a string with a high fitness value. In GAs, parent strings are selected by random choice. The parent strings, however, are not selected by a sheer random choice. The fitness value of each string is utilized for selecting parent strings.

Step 10. Crossover:

Crossover is an operator to generate new strings (*i.e.*, offspring) from parent strings. Various crossover operators have been proposed for GAs [9,13].

Step 11. Mutation :

Mutation is an operator to change elements in a string which is generated by a crossover operator. Such a mutation operator can be viewed as a transition from a current solution to its neighborhood solution in local search algorithms [14].

Step 12. Elitist strategy (Replacing) :

Randomly remove a string from the current population and add the best string in the previous population to the current one.

Step 13. Repairing:

Repair the infeasible individuals of the population to be feasible. The idea of this technique is to separate any feasible individuals in a population from those that are infeasible by repairing infeasible individuals. This approach co-evolves the population of infeasible individuals until they become feasible, the reader is referred to [15]. The pseudo code of the proposed algorithm is shown in Fig. 6.

5. Computational experiment

The performance of the proposed algorithm for a global optimization continuous function is tested on several unconstrained well-known benchmark multimodal problems [7] and two constrained benchmark problems taken from [16]. The algorithm is coded in MATLAB 6.0 and the simulations are run on a Pentium 3 CPU 900 MHz with 128 MB memory capacity.

5.1. Parameter selection

The population size of PSO is often between 10 and 40. The reason for a lower population size is that it significantly lowers the computing time. This is because during initialization, all the particles must be in the feasible space. Randomly initialized particles are not always in the feasible space. So initialization may take a longer time if the population is too large. However, for complex cases, a larger population size is preferred.

In PSO, there are not many parameters that need to be tuned. Only the following parameters need to be taken care of: maximum velocity $V_{\max}$, inertia weight w, acceleration coefficient C_1 and C_2.

In GA, there are many parameters and operators that are to be adjusted. The following parameters need to be taken care of: Generation gap GGAP, Crossover rate *Pc*, Mutation rate *Pm*, Selection operator and Crossover operator. The parameters adopted in the implementation of the proposed algorithm are listed in Table 1.

5.2. Results

The comparison between the results calculated by the proposed approach and the global optimal solutions are reported in Table 2. The results have demonstrated the superiority of the proposed approach to finding the global optimal solution.

Initialize parameters for PSO and GA.
While travel not completed.

PSO algorithm.
While sub-travel not completed.
Evaluation
Set P_{best} and G_{best} .
Update particle velocity v_i^{k+1} and position x_i^k according to Equation (2) and equation (3) of all particles.
Evolve the unfeasible particle according to equation (6).
End while

GA algorithm
While evolution not completed
Evaluation
Keep the best.
Selection.
Recombination.
Mutation.
Repair.
End while

End while

Fig. 6. The pseudo code of the proposed algorithm.

Table 1
The algorithm parameters.

Cognitive parameter	2.8
Social parameter	1.3
Maximum velocity of particles	$V_{\max} = X_{\max} - X_{\min}$
Inertia weight	0.6
Generation gap	0.9
Crossover rate	0.9
Mutation rate	0.7
Selection operator	Stochastic universal sampling
Crossover operator	Single point
Mutation operator	Real-value
PSO iteration	5
GA generation	5

Table 2
Comparison between the global solution and calculated solution.

Test function	No. of iteration	No. of particles	F_{optimal}	$F_{\text{calculated}}$
RC	50	300	0.397887	0.397887
B2	120	300	0	0
ES	100	300	−1	−1
GP	100	300	3	3
SH	300	100	−186.7309	−186.7309
DJ	500	300	0	2.6022E−064
$H_{3,4}$	100	100	−3.86278	−3.86343347
$H_{6,4}$	100	100	−3.32237	−3.322368
$S_{4,5}$	500	600	−10.1532	−10.1532
$S_{4,7}$	300	600	−10.40294	−10.402916
$S_{4,10}$	300	600	−10.53641	−10.5363855
R_2	150	300	0	1.38584E−21
R_5	1000	300	0	1.7476E−11
R_{10}	1000	700	0	1.1367e−9
Z_2	300	100	0	1.8461E−18
Z_5	300	100	0	3.8176E−9
Z_{10}	500	500	0	2.0996E−9
P1	1000	100	−30665.5	−30665.5
P2	150	100	−6961.81	−6961.81

Hybrid genetic algorithm and particle swarm optimization (GA-PSO) [7] and our approach are applied to the suit of 17 unconstrained test problems, and the outcome is tabulated in Table 3. It is found that our approach has a 100% rate of successful minimization for all 17 classical test functions. As a result from Table 3, the average error of proposed approach is smaller than of GA-PSO [15].

Table 3
Results provided by GA-PSO and our approach.

Test function	Average error	
	The proposed approach	GA-PSO
RC	4.59E−7	0.00009
B2	1E−25	0.00001
ES	1E−30	0.00003
GP	−6.3060E−14	0.00012
SH	8.83064E−6	0.00007
DJ	8.443663E−15	0.00004
$H_{3,4}$	0.00003	0.00020
$H_{6,4}$	2E−6	0.00024
$S_{4,5}$	Zero	0.00014
$S_{4,7}$	0.00002	0.00015
$S_{4,10}$	0.00002	0.00012
R_2	1E−30	0.00064
R_5	1E−20	0.00013
R_{10}	1E−18	0.00005
Z_2	1E−15	0.00005
Z_5	1E−17	0.00000
Z_{10}	1E−25	0.00000

Table 4
Results provided by constrained PSO and our approach.

Test function	Error = optimal − calculated	
	The proposed approach	Constrained PSO
P_1	1E−25	Zero
P_2	1E−8	1.759999

On the other hand, the constrained PSO of mechanical systems [16] and our approach are applied to the two constrained problems. It is found that our approach has a 100% rate of successful minimization for the problems. Although, the average error of our approach is the same as the constrained PSO of mechanical systems.

In this subsection, a comparative study has been carried out to assess the proposed approach concerning quality of the solution. On the first hand, evolutionary techniques suffer from the quality of solution. Therefore the proposed approach has been used to increase the solution quality by combining the two merits of two heuristic algorithms (see Table 4).

On the other hand, unlike classical techniques our approach search from a population of points, not single point. Therefore our approach can provide a globally optimal solution. In addition, our approach uses only the objective function information, not derivatives or other auxiliary knowledge. Therefore it can deal with the non-smooth, non-continuous and non-differentiable functions which are actually existed in practical optimization problems.

Another advantage is that the simulation results prove superiority of the proposed approach to those reported in the literature, where it is completely better than the other approaches. Finally, the reality of using the proposed approach to handle complex problems of realistic dimensions has been approved due to procedure simplicity.

6. Conclusions

This paper presents a hybrid method combining two heuristic optimization techniques, PSO and GA. The proposed algorithm integrates the concept of evolving individuals originally modeled by GA with the concept of self-improvement of PSO, where, the algorithm initialized by a set of a individuals which travel in the search space using the PSO. During this travel we implement GA to evolve these individuals. Also, in order to keep the feasibility of the particles, an additional parameter χ is introduced, where the algorithm co-evolves the population of infeasible individuals until they become feasible. Simulated experiments for the optimization of nonlinear multimodal test functions show that the proposed approach is superior in the ability to finding the global optimal solution.

Acknowledgement

The authors are grateful to the anonymous reviewers for their valuable comments and helpful suggestions which greatly improved the paper's quality.

References

[1] S.S. Rao, Engineering Optimization: Theory and Practice, 3rd ed., Wiley, NY, USA, 2003.
[2] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, L.M. Wang, An improved GA and a novel PSO-GA-based hybrid algorithm, Information Processing Letters 93 (2005) 255–261.

[3] M.A. Panduro, C.A. Brizuela, L.I. Balderas, D.A. Acosta, A comparison of genetic algorithms, particle swarm optimization and the differential evolution method for the design of scannable circular antenna arrays, Progress in Electromagnetics Research B 13 (2009) 171–186.
[4] O. Uysal, S. Bulkan, Comparison of genetic algorithm and particle swarm optimization for bicriteria permutation flowshop scheduling problem, International Journal of Computational Intelligence Research 4 (2) (2008) 159–175. ISSN 0973-1873.
[5] X. Yang, J. Yuan, J. Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation 189 (2007) 1205–1213.
[6] P.S. Shelokar, P. Siarry, V.K. Iayaraman, B.D. Kulkarni, Particle Swarm and ant colony algorithms hybridized for improved continous optimization, Applied Mathematics and Computation 188 (2007) 129–142.
[7] Y. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing (2008) 849–857.
[8] S. Koziel, Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation 7 (1) (1999) 19–44.
[9] D.E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, Reading, MA, 1989.
[10] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann, San Francisco, 2001.
[11] K.F. Man, K.S. Tang, S. Kwong, Genetic Algorithms: Concepts and Designs, Springer, London, 1999.
[12] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, Congress on Evolutionary Computing 1 (2000) 84–88.
[13] I. Oliver, D. Smith, J. Holland, A study of permutation crossover operators on the traveling salesman problem, in: Proc. 2nd ICGA, Massachusetts Institute of Technology, USA, July 28–31, 1987, pp. 224–230.
[14] M.F. Bramlette, Initialization mutation and selection methods in genetic algorithms for functions optimization, Proceedings of the ICGA 4 (1991) 100–107.
[15] A.A. Mousa, I.M. El-Desoky, GENLS: co-evolutionary algorithm for nonlinear system of equations, Applied Mathematics and Computation 197 (2008) 633–642.
[16] K. Sedlaczek, P. Eberhard, Constrained Particle Swarm Optimization of Mechanical Systems, in: 6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, 30 May–03 June, Brazil, 2005.