Particle Swarm Optimization Method for Engineering Nonlinear Constrained Optimization

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization
8-10 Averescu Avenue, Bucharest 1, Romania
E-mail: nandrei@ici.ro

March 15, 2017

Abstract. For nonlinear constrained optimization a particle swarm optimization (PSO) method with a new penalty function method is presented. The penalty function includes a composite penalty factor, which introduce a linear progressive penalization subject to the values of the constraint violation into the current point. This new composite penalty function is used into the frame of the particle swarm optimization method. To improve the direct search of PSO a local coordinatewise search is used. The numerical experiments with this new penalization for 10 real nonlinear constrained optimization applications are reported. The obtained results are compared versus the well known direct search methods COBYLA, DFL and the primal-dual interior-point algorithm with a filter line-search method IPOPT. The conclusion is that minimizing a special penalty function using particle swarm optimization method yield a competitive algorithm being more efficient versus COBYLA and DFL.

Keywords: Particle Swarm Optimization; Penalty function; Composite function; COBYLA; DFL; IPOPT.

1. Introduction

Engineering nonlinear optimization problems are present in very numerous applications. These problems can be represented as:

$$\min f(x)$$

subject to: (1)

$$c_i(x) \le 0, \ \ i = 1,\ldots,m,$$
$$x \in \Omega = \{x : l \le x \le u\},$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $c_i \in \mathbb{R}^n \to \mathbb{R}$, $i = 1,\ldots,m,$ are nonlinear functions, called functional constraints, and $l, u \in \mathbb{R}^n$ are lower and upper simple bounds on variables $x \in \mathbb{R}^n$, respectively. The above formulation is not restrictive because the inequality constraints of the form $c_i(x) \ge 0,$ can also be represented as $-c_i(x) \le 0,$ and an equality constraint, $c_i(x) = 0,$ can be represented by two inequality constraints $c_i(x) \le 0$ and $-c_i(x) \le 0.$

For solving these problems plenty of methods are known. These can be classified as *gradient methods* using the information given by the gradients and the Hessians of the functions f and c_i, $i = 1,\ldots,m,$ and *direct methods* using only the values of these functions along a sequence of points from Ω.

The main methods based on derivative information (the first and the second order) can be classified as: penalty and augmented Lagrangian methods [1, 2, 3], sequential quadratic programming [4, 5], interior-point methods [5], filter methods [6], etc. One of the most common methods for nonlinear constrained optimizations is based on penalty function. In the penalty method the constraints are penalized by building an extended objective function, which is

minimized by means of the unconstrained optimization algorithms (see [5, 7, 8]). The penalty function aims to penalize infeasible solutions by increasing their fitness values proportionally to their level of constraints violation. Using the same principle, an extension of the penalty methods is given by the augmented Lagrangian methods [2, 3, 9]. Methods based on augmented Lagrangian often use gradient techniques for minimizing the augmented Lagrangian [10, 11, 12, 13], and rarely they are combined with heuristics that rely on a population of points where the Lagrangean function is evaluated [14, 15, 16].

On the other hand, the methods using only the values of the functions defining the problem are much diversified, the main ones being based on: genetic algorithms, particle swarm optimization, ant colony optimization, evolutionary algorithms, bacterial foraging algorithms, tabu search, electromagnetism-like mechanism, etc. An excellent survey on bio inspired optimization algorithms is given in [17].

In this paper we consider the Particle Swarm Optimization method for solving the nonlinear constrained optimization problems (1) using a special penalty method where a linear assignment of penalization is used. Plenty of particle swarm optimization (PSO) methods for solving nonlinear constraint optimization problems are known [18]. A particle-swarm optimization using modified BFGS updating [19, 20] for constrained nonlinear optimization was presented in [21]. Here, a hybrid algorithm which integrates the modified BFGS into the particle swarm optimization is used to minimize the augmented Lagrangian penalty function. The structure of this hybrid algorithm for the augmented Lagrangian minimization is taken from [7]. Another approach based on augmented Lagrangian which solves a sequence of simple bound constrained sub-problems whose objective function penalizes equality and inequality constraints violation and depends on the Lagrange multipliers and a penalty parameter was given by Rocha and Fernandes [22]. Each sub-problem is solved by a population-based method that uses an electromagnetism-like mechanism introduced in [23]. An application of swarm optimization for nonlinear programming using the evaluation of infeasible particles is described in [24]. Another penalty function approach was described in [25]. Here the constraints are penalized by means of a multi-stage assignment function. Other approaches for solving constrained nonlinear optimization problems have been suggested. For example, a hybrid multi-swarm particle swarm optimization method is proposed in [26], where the current swarm is partitioned into several sub-swarms and the particle swarm optimization is used as the search engine for each sub-swarm. In order to explore more promising regions of the search space differential evolution is incorporated in order to improve the personal best of each particle. A dynamic-objective particle swarm optimization for constrained optimization problems is presented in [27]. This method converts the original constrained optimization problem into a bi-objective optimization problem and then it enables each particle to dynamically adjust its objectives according to its current position in the search space. In [28] a measure of the infeasibility of a particle is computed using an exponential function which contains the logarithms of the modified constraints.

For solving nonlinear constrained optimization problems, in this paper we consider a modification of the penalty function introduced in [25]. The penalty function used here is a composite function in which the constraints are penalized by means of a linear assignment function. In Section 2 we present the penalty function method used in this paper. Section 3 is dedicated to give the main ideas of particle swarm optimization method in conjunction to this new penalty function. In order to improve the best point generated by the PSO algorithm a procedure for the local search around the best point is used. The numerical results of this method subject to a number of 10 nonlinear constrained optimization applications are presented in Section 4. Comparisons of our method versus the direct search methods: COBYLA [29] and DFL [30], as well as versus IPOPT [31, 32] illustrate the performances of our algorithm.

2. The Penalty Function Method with a linear progressive penalization

The penalty function method solves the nonlinear constrained optimization problem (1) by solving a sequence of unconstrained optimization sub-problems. The unconstrained optimization sub-problems are solved using the PSO method. In PSO a non-stationary penalty function with a multi-sage assignment function was introduced by Homaifar, Lai and Qi [33] and used in [25] and [34]. In this paper, for problem (1), we modify this non-stationary penalty function and consider the following one:

$$F(x) = f(x) + h(k)P(x), \tag{2}$$

where $f(x)$ is the original objective function, $h(k)$ is a variable coefficient dynamically modified at every iteration k, and $P(x)$ is a composite penalty factor defined as:

$$P(x) = \sum_{i=1}^{m} \theta(q_i(x)) \left(q_i(x)\right)^{\gamma(q_i(x))}, \tag{3}$$

where $q_i(x) = \max\{0, c_i(x)\}$, $i = 1, \ldots, m$. Observe that the function $q_i(x)$ gives a measure of the violation of constraint c_i into the point x. If the constraint c_i is satisfied, then it has no influence into the penalty factor, i.e. $P(x) = 0$ for every admissible x. $\theta(q_i(x))$ is an assignment function which introduce a progressive penalization subject to the values of the constraint $c_i(x)$ into the current point. In order to emphasize the penalization of the violation of the constraints around 1, the function $\gamma(q_i(x))$ is introduced as:

$$\gamma(q_i(x)) = \begin{cases} 1, & \text{if } q_i(x) < 1, \\ 2, & \text{if } q_i(x) \geq 1. \end{cases} \tag{4}$$

The variable coefficient $h(k)$ is dependent by the number of iteration k. Simple expressions of this coefficient can be $h(k) = k\sqrt{k}$, or $h(k) = \sqrt{k}$. Observe that the functions $\theta(.)$ and $\gamma(.)$ are dependent by the constraint functions defining the problem.

The function $\theta(t)$ which is a penalization of the constraint $c_i(x) \leq 0$ is defined as:

$$\theta(t) = \begin{cases} u, & \text{if } t < 0.001, \\ \dfrac{U-u}{0.999}t + \dfrac{u - 0.001U}{0.999}, & \text{if } 0.001 \leq t < 1, \\ U, & \text{if } t \geq 1, \end{cases} \tag{5}$$

where u and U are the positive penalty coefficients ($u < U$). In our numerical experiments we have considered $u = 10$ and $U = 300$. In other words, we consider a moderate linear evolution of the penalization coefficients associated to the constraints. Clearly, this continuous evolution of function $\theta(t)$ may be modified, by changing the values of parameters u and U, in order to accentuate or diminish the penalization of the constraints in $P(x)$. The simple bound constraints defined by Ω are also introduced in function $P(x)$ as functional constraints. Observe that $P(x)$ is positive for any value of x. With these, the penalty function method for solving (1) consists of minimizing the penalty function (2), viewed as an unconstrained optimization problem, using the particle swarm optimization method, which we present in the next section.

3. The Particle Swarm Optimization Method

Particle Swarm Optimization is a stochastic optimization method based on simulation of the social behavior [18, 35, 36]. This is a technique that can be likened to the behavior of a flock of birds or the sociological behavior of a group of people. PSO is a population based optimization technique, where the population is called swarm. Each particle represents a possible solution to the optimization problem. Along the iterations each particle accelerates in the direction of its own personal best solution computed so far, as well as in the direction of the global best position discovered by any of the particles in the swarm. The idea is that if a particle finds a better new solution, then all other particles in the swarm will move closer to it, thus intensively exploring the corresponding region.

The PSO algorithm has three main steps: generating the particles' positions and velocities, velocity update and finally position update. Suppose that the searching space is $\mathbb{R}^n$ and the population of particles consists of $npop$ particles $X_1, X_2, \ldots, X_{npop}$, where $X_j \in \mathbb{R}^n$, $j = 1, \ldots, npop$. Each particle defines the position of the searching point from $\mathbb{R}^n$, that is $X_j = [X_{j,1}, \ldots, X_{j,n}]$. Therefore, at the k-th iteration, for all the particles the matrix of positions $X^k = [X_1^k, \ldots, X_{npop}^k] \in \mathbb{R}^{n \times npop}$ is defined. The best previous position that is the position of the best value of the minimizing function corresponding to the particle j, at iteration k, is denoted as P_j^k, $j = 1, \ldots, npop$. Clearly, $P_j^k \in \mathbb{R}^n$. At the same time, the best position found by any other particle at iteration k, is denoted by G^k. $G^k \in \mathbb{R}^n$ and represents the best position found by all the population of particles. Every particle, at iteration k, is characterized by a velocity $V_j^k \in \mathbb{R}^n$, $j = 1, \ldots, npop$. Therefore, for all population of $npop$ particles, at iteration k, the matrix of velocities $V^k = [V_1^k, \ldots, V_{npop}^k] \in \mathbb{R}^{n \times npop}$ can be defined.

The PSO method is characterized by the following evolution of velocities:

$$V_j^{k+1} = \chi \left[wV_j^k + c_1 r_j^k \otimes (P_j^k - X_j^k) + c_2 \bar{r}_j^k \otimes (G^k - X_j^k) \right], \tag{6}$$

for $j = 1, \ldots, npop$, where: χ is a *constriction factor* which is used to control and constrict velocities; w is the *inertia weight*; c_1 and c_2 are two positive constants, called the *cognitive* and *social* parameter respectively, also known as acceleration constants. The constriction coefficient is determined as [37]:

$$\chi = \frac{2}{\left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right|}, \quad \phi = c_1 + c_2.$$

$r_j^k \in \mathbb{R}^{n \times npop}$ and $\bar{r}_j^k \in \mathbb{R}^{n \times npop}$ are two matrices with random numbers uniformly distributed in the range [0,1]. $\otimes$ is product on components. The first term in (6), wV_j^k is defining the current motion, the second one $c_1 r_j^k \otimes (P_j^k - X_j^k)$ is defining the particle memory influence and the last one $c_2 \bar{r}_j^k \otimes (G^k - X_j^k)$ is the swarm influence.

The inertia factor w is for controlling the impact of the previous history of velocities on the current velocity. It is used to balance the global (wide-ranging) and local (nearby) search abilities and was introduced to improve the convergence rate of the PSO algorithm. A large inertia weight is more appropriate for global search facilitating the exploration (searching new

areas), while a small one is more appropriate for local search facilitating the exploitation (fine tuning the current search area). A linear decreasing of the inertia weight over the course of search was proposed by Shi and Eberhart [38]. The parameters r_k^j and $\overline{r}_k^j$ are used to maintain the diversity of the population of the swarm and they are uniformly distributed in the range [0,1]. The constant parameters c_1 and c_2 are not critical in PSO, but a fine tuning of them may result in faster convergence and alleviation of local minima. Better results are obtained by choosing a larger cognitive parameter c_1, than a social parameter c_2, but with $c_1 + c_2 \le 4$ [34].

The position update of each particle is computed using the velocity vector as:

$$X_j^{k+1} = X_j^k + V_j^{k+1}, \quad j = 1,\ldots,npop. \tag{7}$$

The velocity update, the position update and the evaluation of the minimizing function in all particles are repeated until a desired convergence criterion is met.
With these, the particle swarm optimization procedure for composite function $F(x)$ can be summarized as follows:

Algorithm CPSO - Constrained Particle Swarm Optimization

Step 1.	Initialize: c_1, c_2, w, χ, $k_{\max}$ and set $k = 1$.
Step 2.	Generate a swarm of *npop* particles with random positions $X_j \in \mathbb{R}^n$, $j = 1,\ldots,npop,$ and with random velocities $V_j \in \mathbb{R}^n$, $j = 1,\ldots,npop.$
Step 3.	Evaluate the penalty function $F(X_j)$, $j = 1,\ldots,npop.$
Step 4.	Find the best particle X_g, where $g = \arg\min\{F(X_j) : j = 1,\ldots,npop\}$, is the index of the best evaluated particle.
Step 5.	Modify the velocities and the positions of the particles by (6) and (7), respectively.
Step 6.	Evaluate the penalty function F for all particles and update the best position for each particle and its index g.
Step 7.	If $k \le k_{\max}$, then set $k = k + 1$ and go to step 3; otherwise stop and output the best particle. ♦

Convergence analysis and stability studies on PSO have been reported by Clerc and Kennedy [37], Trelea [39], Yasuda et al. [40], etc. Research on performance improvements subject to the parameters variations and topological structures have been considered by Eberhart and Shi [41, 42], Li and Engelbrecht [43].

The CPSO algorithm gives only local solutions to problem (1). Therefore, possible this solution can be improved by local search. Since the value of the penalty function (2) is zero in any feasible point, a procedure for improving the solution given by CPSO is not based on the penalty function (2). A simple local search procedure we adopt here is a coordinatewise one applied to the best point. Let x be the best solution generated by CPSO algorithm. For each component i, x is assigned to a temporary point y. Then in y a movement of length $\delta > 0$ is carried out in coordinate i. If the point y satisfies the simple bounds and is feasible, and if the value of the objective function f is reduced, then x is replaced by y. When the point y is infeasible, or the value of the objective function computed in y is not improved, then it is rejected and another coordinate is tried. The local search stops when all coordinates were tried.

The local search algorithm for searching around the solution given by CPSO is very simple and possible in some cases it can improve the performances of CPSO.

4. Numerical results

Test problem suite

In this paper we have considered a number of 10 applications of nonlinear constrained optimization. Table 4.1 presents the name of the application and references where they are described.

Table 4.1. Name of the applications and references.

ALKI	Optimization of an alkylation process, [44, Problem No. 114, page 123]
CAM	Shape optimization of a cam, [45, Application 5.21, page 117]
MSP3	3-stage membrane separation [44, Problem No. 116, page 124]
MSP5	5-stage membrane separation process, [46], [47, Application 6, page 983]
PREC	Optimal Reactor Design. [44, Problem No. 104, page 113], [45, Application 7.1, page 161]
PPSE	Static Power Scheduling [44, Problem No. 107. page 116], [45, Application 6.4, page 144]
TRAFO	Transformer design [44, Problem No. 93. page 108], [45, Application 6.1, page 137]
LATHE	Multi-spindle automatic lathe [45, Application 5.13, page 90]
BRAKE	Optimal design of a disc brake [45, Application 5.8, page 83]
SPRING	Minimizing the weight of a tension/compression spring [45, Application 5.3, page 73]

Numerical results with CPSO-Constrained Particle Swarm Optimization

In our numerical experiments we have considered the following values for the parameters in the particle swarm optimization: $\chi = 0.73$, $w = 0.7$, $c_1 = 2$, $c_2 = 2$. The penalty function (5) is defined with $u = 10$ and $U = 300$. The size of swarm, i.e. the number of particles in the swarm, is different for each application. Empirical studies shown that the number of particles can influence the results of the optimization. For some problems we see an improvement of performances as the size of the swarm is increased, while for others better results are obtained by smaller swarms. The number of iterations is limited to $k_{\max} = 50000$. For each application we have considered 50 independent experiments. An experiment was considered to be successful only if a feasible solution was obtained. The best solution corresponds to the minimum value of the objective function found in these 50 experiments. Table 4.2 shows the performances of CPSO algorithm, where n is the number of variables, m is the number of constraints (including the simple bounds on variables), *size* is the number of the particles in the swarm, *iter* is the number of iteration to get the best solution, *vfo* is the best value of the objective function obtained by the algorithm and *vfoa* is the best value of the objective obtained by local search.

Table 4.2. Performance of the CPSO algorithm.

	n	*m*	*size*	*Iter*	*Vfo*	*vfoa*
ALKI	10	34	150	1172	-1760.975932	-1763.528164
CAM	10	43	100	5923	-43.379410	-43.523641
MSP3	13	41	50	394	97.558862	97.558862
MSP5	16	53	20	367	183.211378	183.024078
PREC	8	22	50	45	4.040709	4.022867
PPSE	9	30	50	>50000	5055.897360	5055.897360
TRAFO	6	14	10	111	135.761724	135.761724
LATHE	10	36	25	498	-4433.63308	-4433.63308
BRAKE	4	14	9	6	0.147958	0.147323
SPRING	3	10	5	70	0.0126904	0.0126904

The column *vfoa* in Table 4.2 represents the value of the objective function f obtained by local search around the best solution given by PSOA using different values for δ. Observe that in some cases the local search is not effective.

Comparisons

In Table 4.3 a comparison of CPSO versus COBILA [29], DFL [30] and IPOPT [31, 32] subject to the value of the objective function of the above constrained nonlinear optimization application is presented. COBYLA is a direct search optimization method that models the objective and the constraint functions by linear interpolation. Each iteration forms such a linear approximation at the vertices of a simplex and a trust region bound restricts each change to the variables. DFL solves the constrained nonlinear optimization problem by a sequence of approximate minimizations of a merit function where penalization of constrained violation is progressively increased. On the other hand, IPOPT is a primal-dual interior-point algorithm with a filter line-search method for nonlinear programming.

Table 4.3. Comparisons: COBILA, DFL, CPSO, IPOPT

	COBILA	DFL	CPSO	IPOPT
ALKI	-1550.38851	-931.120	-1763.528164	-1768.8069
CAM	-43.859947	-45.54602	-43.523641	-43.859947
MSP3	97.587578	50.0	97.558862	97.587509
MSP5	175.619466	209.96775	183.024078	174.786994
PREC	3.951163	4.1431963	4.022867	3.951163
PPSE	5055.0118	8063.5088	5055.897360	5055.01180
TRAFO	135.075962	136.27182	135.761724	135.075962
LATHE	-3434.70855	-4429.152978	-4433.63308	-4430.08793
BRAKE	0.1274	0.131321	0.147323	0.1274
SPRING	0.0126652	0.0126897	0.0126904	0.0126652

Comparing CPSO versus IPOPT, from Table 4.3 we see that the sum of the absolute difference of the objective values obtained by these two algorithms for solving the above 10 applications is 19.0891. Comparing DFL versus IPOPT the sum of the absolute difference is 3932.965. Finally, comparing COBYLA versus IPOPT we get the difference 1214.630. Observe that subject to the value of the objective function the CPSO algorithm is closest to the primal-dual interior-point algorithm with a filter line-search method IPOPT. However, COBYLA was able to get the same value for the objective function as IPOPT in 6 out of 10 applications considered in this numerical study.

5. Conclusions

PSO is one of the methods for solving in an approximate way the optimization problems. The algorithm is able to search an optimal value of a constrained minimizing function by comparing only the values of a penalty function associated to the original problem. The PSO algorithm contains some random factors. Therefore, the algorithm can be regarded as a stochastic one. In this paper for general nonlinear constrained optimization we introduced a new penalty function based on an assignment function which considers a linear progressive penalization subject to the violation of the constraints into the current point. The PSO method is used to minimize this penalty function. To improve the results of the algorithm a local coordinatewise search is considered around the best point obtained by the CPSO algorithm. Numerical experiments proved

that this local coordinatewise search is dependent by the problem. Some remarks are in order. CPSO is very simple to be implemented and it is able to generate good enough approximate solutions. It is dependent by the first random number with which the searching process starts. Therefore better results are obtained by trying to solve the problem using different initial randomly selected population and by comparing the results. In general swarms of small sizes give better results, but this is dependent by the problem. The CPSO method is a competitive alternative to the very sophisticated gradient methods used for solving nonlinear constrained optimization problems.

References

[1] Fiacco, A.V., McCormick, G.P., Nonlinear Programming. Sequential Unconstrained Minimization Techniques. John Wiley & Sons, Inc., New York, 1968.

[2] Hestens, M.R., Multiplier and gradient methods. Journal of Optimization Theory and Applications, vol.4, 1969, pp.303-320.

[3] Powell, M.J.D., Multiplier and gradient methods. Journal of Optimization Theory and Applications, 4 (1969) 303-320.

[4] Bertsekas, D.P., Nonlinear Programming. Athena Scientific, Belmont, MA, second ed., 1999.

[5] Nocedal, J., Wright, S.J., Numerical optimization. Springer Series in Operations Research. Springer Science+Business Media, New York, Second ed., 2006.

[6] Fletcher, R., Leyffer, S., Nonlinear programming without a penalty function. Mathematical Programming, Series A, 91 (2002), pp.239-269.

[7] Bazaraa, M.S., Sherali, H.D., Shetty, C.M., (1993) Nonlinear Programming Theory and Algorithms. John Wiley, New York, 2nd edition, 1993.

[8] Luenberger, D.G., (1984) Introduction to linear and nonlinear programming. Addison-Wesley Publishing Company, Reading, Second ed., 1984.

[9] Rockafellar, R.T., Augmented Lagrange multiplier functions and duality in nonconvex programming. SIAM Journal on Control and Optimization, 12 (1974) 268-285.

[10] Andreani, R., Birgin, E.G., Martínez, J.M., Schuverdt, M.L., On augmented Lagrangian methods with general lower-level constraints. SIAM Journal on Optimization, 18 (2007) pp.1286-1309.

[11] Birgin, E.G., Castillo, R., Martínez, J.M., Numerical comparison of augmented Lagrangian algorithms for nonconvex problems. Computation Opotimization and Applications, 31 (2004), pp.31-56.

[12] Birgin, E.G., Floudas, C.A., Martínez, J.M., Global minimization using an augmented Lagrangian method with variable lower-level constraints. Mathematical Programming, Serie A, 125 (2010) pp.139-162.

[13] Luo, H., Sun, X., Wu, H., Convergence properties of augmented Lagrangian methods for constrained global optimization. Optimization Methods and Software, 23 (2008) pp. 763-778.

[14] Sedlaczek, K., Eberhard, P., Augmented Lagrangian particle swarm optimization in mechanism design. Journal of System Design and Dynamics, 1 (2007) pp.410-421.

[15] Wah, B.W., Wang, T., Efficient and adaptive Lagrange-multipliers. Journal of Global Optimization, 14 (1999) pp.1-25.

[16] Wang, T., Wah, B.W., Handling inequality constraints in continuous nonlinear global optimization. In: Integrated Design and Process Science, 1996, pp. 267-274.

[17] Binitha, S., Sathya, S.S., A survey of bio inspired optimization algorithms. International Journal of Soft Computing and Engineering (IJSCE), 2 (2012) 137-151.

[18] Kennedy, J., Eberhart, R.C., Particle swarm optimization. Proc. IEEE Int. Conf. Neural Networks, 1995, pp. 1942-1948.

[19] Powell, M.J.D., A method for nonlinear constraints in optimization problems. In: Fletcher, R., (Ed.) Optimization. Academic Press, New York, 1969, pp. 283-297.

[20] Powell, M.J.D., A fast algorithm for nonlinearly constrained optimization calculations. In: Watson, G.A. (Ed.) Numerical Analysis, Dundee, Springer-Verlag, 1977, pp. 144-157.
[21] Nezhad, A.M., Shandiz, R.A., Jahromi, A.E., A particle swarm-BFGS algorithm for nonlinear programming problems. Computers & Operations Research, 40 (2013), pp.963-972.
[22] Rocha, A.M.A.C., Fernandes, E.M.G.P., Numerical study of augmented Lagrangian algorithms for constrained global optimization. Optimization,
[23] Birbil, S.I., Fang, S.-C., An electromagnetism-like mechanism for global optimization. Journal of Global Optimization, 25 (2003), pp. 263-282.
[24] Dong, Y., Tang, J., Xu, B., Wang, D., An application of swarm optimization to nonlinear programming. Computers & Mathematics with Applications, 49 (2005), 1655-1668.
[25] Yang, J.-M., Chen, Y.-P., Horng, J.-T., Kao, C.-Y., Applying family competition to evolution strategies for constrained optimization. Lecture Notes in Computer Science, vol.1213, Springer-Verlag, Berlin, (1997) pp. 201-211.
[26] Wang, Y., Cai, Z., A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. Front. Comput. Sci. China, 3 (2009), pp. 38-52.
[27] Lu, H., Chen, W., Dynamic-objective particle swarm optimization for constrained optimization problems. J. Comb. Optim., 12 (2006), pp. 409-419.
[28] Vaz, A.I. de Freitas., Fernandes, E.M.de Graça Pinto, Optimization of nonlinear constrained particle swarm. Technological and Economic Development of Economy, 12 (2006) 30-36.
[29] Powell, M.J.D., A direct search optimization method that models the objective and constraint functions by linear interpolation. Department of Applied Mathematics and Theoretical Physics, University of Cambridge, April 1992, NA5 (revised August 1993).
[30] Liuzzi, G., Lucidi, S., Sciandrone, M., Sequential penalty derivative-free methods for nonlinear constrained optimization. SIAM Journal on Optimization, 20 (2010) 2614-2635.
[31] Wächter, A., Biegler, L.T., Line search filter methods for nonlinear programming: Motivation and global convergence. SIAM Journal on Optimization, 16 (2005) 1-31.
[32] Wächter, A., Biegler, L.T., On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106 (2006) 25-57.
[33] Homaifar, A., Lai, A.H., Qi, X., Constrained optimization via genetic algorithms. Simulation, 2, (1994) pp.242-254.
[34] Parsopoulos, K.E., Vrahatis, M.N., Particle swarm optimization method for constrained optimization. In P. Sincak, J. Vascak, V. Kvasnicka, Pospichal, J. (Eds.) Intelligent technologies-theory and application (New trends in intelligent technologies). Frontiers in artificial intelligence and applications. Vol. 76, pp. 214-220, (2002) Amsterdam, IOS press,
[35] Eberhart, R.C., Kennedy, J., A new optimizer using particle swarm theory. Proc. 6th Int. Symp. Micromachine Human Sci. Nagoia, Japan, 1995, pp. 39-43.
[36] Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N., Memetic particle swarm optimization. Ann. Oper. Res., 156 (2007) pp.99-127.
[37] Clerc, M., Kennedy, J., The particle swarm-explosion, stability and convergence on a multidimensional complex space. IEEE Trans. Evol. Comput. Vol.6, no.1, 2002, pp.58-73.
[38] Shi, Y., Eberhart, R.C., A modified particle swarm optimizer. Proc. IEEE Congr. Evol. Comput., 1998, pp. 69-73.
[39] Trelea, I.C., The particle swarm optimization algorithm: Convergence analysis and parameter selection. Inf. Process Lett., vol.85, no.6, 2003, pp.317-325.
[40] Yasuda, K., Ide, A., Iwasaki, N., Stability analysis of particle swarm optimization. Proc. 5th Metaheuristics Int. Conf., 2003, pp.341-346.
[41] Eberhart, R.C., Shi, Y.H., Particle swarm optimization: Developments, applications and resources. Proc. IEEE Congr. Evol. Comput., Seoul, Korea, 2001, pp.81-86.
[42] Eberhart, R.C., Shi, Y.H., Guest editorial. IEEE Trans. Evol. Comput. – Special Issue Particle Swarm Optimization, vol.8, no.3, 2004, pp. 201-203.

[43] Li, X.D., Engelbrecht, A.P., Particle swarm optimization: An introduction and its recent developments. Proc. Genetic Evol. Comput. Conf. 2007, pp. 3391-3414.
[44] Hock, W., Schittkowski, K., Test examples for nonlinear programming codes. Lecture Notes in Economics and Mathematical Systems, vol.187. Berlin, Springer, 1981.
[45] Andrei, N., Nonlinear Optimization Applications using the GAMS Technology. Springer Optimization and Its Applications 81, Springer Science+Busines Media, New York, 2013.
[46] Dembo, R.S., A set of geometric programming test problems and their solutions. Mathematical Programming, 10 (1976) pp.192-213.
[47] Andrei, N., Criticism of the Constrained Optimization Algorithms Reasoning (in Romanian). Editura Academiei Române, Bucharest, 2015.