ISSN 1392-8619 print/ISSN 1822-3613 online
ÛKIO TECHNOLOGINIS IR EKONOMINIS VYSTYMAS
TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
http://www.tede.vgtu.lt
2006, Vol XII, No 1, 30–36

OPTIMIZATION OF NONLINEAR CONSTRAINED PARTICLE SWARM

António Ismael de Freitas Vaz*,
Edite Manuela da Graça Pinto Fernandes

Department of Production and Systems, Engineering School, Minho University,
Campus de Gualtar, 4710-057 Braga, Portugal.
E-mail: {aivaz,emgpf}@dps.uminho.pt
**Corresponding author*

Received 3 October 2005; accepted 4 January 2006

Abstract. We propose an algorithm based on the particle swarm paradigm (PSP) to address nonlinear constrained optimization problems. While some algorithms based on PSP have already been proposed in this context, the equality constraints have been posing some difficulties. The proposed algorithm is based on the relaxation of the dominance concept introduced in the multiobjective optimization. This concept is used to select the best particle position and the best ever particle swarm position. We propose also a stopping criterion for the algorithm and present numerical results with some problems collected from the literature. The new algorithm is implemented in a solver connected with AMPL, allowing easy coding and solving of problems.

Keywords: Nonlinear constrained optimization, particle swarm optimization, dominance concept.

1. Introduction

A typical nonlinear constrained optimization problem can be defined as

$$\begin{aligned} & \min_{x\in R^n} f(x), \\ s.t.\quad & c^E(x)=0, \\ & c^I(x)\le 0, \\ & l\le x\le u, \end{aligned} \tag{1}$$

where $f:R^n\to R$ is the objective function, $c^E:R^n\to R^{m_E}$ are m_E equality constraint functions, $c^I:R^n\to R^{m_I}$ are m_I inequality constraint functions, and l and u are simple bounds on variables x. We allow $l_i=-\infty$, and/or $u_i=+\infty$, meaning that variable $x_i, (i=1,\ldots,n)$ may not have a lower and/or an upper bound.

Parsopoulos and Vrahatis [1] proposed a penalty framework for constrained optimization based on a particle swarm paradigm (PSP) for solving the resulting unconstrained subproblems.

Hu and Eberhart [2] and later Hu *et al.* [3] also proposed an algorithm based on PSP for nonlinear constrained optimization. The algorithm finds initial feasible population and ensures feasibility during the entire optimization process. However, for small feasible regions (in particular when equality constraints are present), obtaining of initial feasible population can be a difficult task with too many constraint evaluations.

In this paper we propose an algorithm to find the global optimum of problem (1) that uses a relaxed dominance concept adapted from the multiobjective optimization, to be able to assess progress towards feasibility and optimality.

In the next section the particle swarm paradigm for unconstrained optimization is described. Section 3 is used to present the main ideas behind our proposed algorithm and Section 4 reports on the implementation details. Numerical results with some test problems are presented in Section 5 and Section 6 contains the conclusions.

2. The particle swarm algorithm for unconstrained optimization

The particle swarm optimization algorithm (PSOA) was firstly proposed by Eberhart and Kennedy [4, 5] and has deserved some attention during the last years in the global optimization field. PSOA is based on the population of agents or particles and tries to simulate its social behaviour in optimal exploration of problem space.

During time (iterations in the optimization context) each agent possesses a velocity vector that is a stochastic combination of its previous velocity and the distances of its current position to its own best ever position and to the best ever swarm position. The weights of the last two directions are controlled by two parameters called cognitive and social parameters [6].

PSOA belongs to a class of stochastic algorithms for global optimization and its main advantages are the easily parallelization and simplicity. PSOA seems to outperform the genetic algorithm for some difficult programming classes, namely the unconstrained global optimization problems [6].

In spite of the referred advantages, PSOA possesses some drawbacks, namely its parameters dependency and the slow convergence rate in the vicinity of the global minimum.

In this section we briefly describe PSOA. This description is a summary of *gbest* PSOA tested in [6] and the reader is pointed to [6] for other variants and details (see also [7] for recent study).

PSOA is based on the population (swarm) of particles. Each particle is associated with velocity that indicates where the particle is *traveling*. Let t be a time instant. The new particle position is computed by adding the velocity vector to the current position

$$x^p(t+1) = x^p(t) + v^p(t+1), \qquad (2)$$

being $x^p(t)$ particle p position, $p = 1,\ldots,s$, at time instant t, $v^p(t+1)$ new velocity (at time $t+1$) and s is population size.

The velocity update equation is given by

$$\begin{aligned} v_j^p(t+1) = \iota(t)v_j^p(t) \quad & +\mu\omega_{1j}(t)\left(y_j^p(t) - x_j^p(t)\right) \\ & +\nu\omega_{2j}(t)\left(\hat{y}_j(t) - x_j^p(t)\right), \end{aligned} \qquad (3)$$

for $j = 1,\ldots,n$, where $\iota(t)$ is a weighting factor (inertial), μ is the *cognitive* parameter and ν is the *social* parameter. $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform distribution $U(0,1)$, used for each dimension $j = 1,\ldots,n$. $y^p(t)$ is particle p position with the best objective function value so far and $\hat{y}(t)$ is a particle position with the best function value so far. $\hat{y}(t)$ can be described in a more rigorous way as

$$\hat{y}(t) = \arg\min_{a \in A} f(a)$$
$$A = \left\{y^1(t), \ldots, y^s(t)\right\}.$$

PSOA can be described as follows:

Algorithm 1 *PSOA*

1. Randomly initialize the swarm positions

$$\mathrm{X} = \left\{x^1(0), \ldots, x^s(0)\right\}$$

and velocities $\mathrm{V} = \left\{v^1(0), \ldots, v^s(0)\right\}$.

2. Let $t = 0$ and $y^p(t) = x^p(t)$, $p = 1,\ldots,s$.

3. For all p in $\{1,\ldots,s\}$ do:

If $f(x^p(t)) < f(y^p(t))$ then set $y^p(t+1) = x^p(t)$ else set $y^p(t+1) = y^p(t)$.

4. For all p in $\{1,\ldots,s\}$ do:

Compute $v^p(t+1)$ and $x^p(t+1)$, using equations (2) and (3).

5. If the stopping criterion is true, then stop. Otherwise set $t = t+1$ goes to step 3.

The stopping criterion mostly used in the literature is related to the function value at the global optimum. The algorithm stops, if either the objective function at the best ever particle is approximately equal to the known objective minimum, or a maximum number of iterations is exceeded.

PSOA described by Algorithm 1, where all particles are governed by equations (2) and (3), has no guaranteed convergence onto a local or to a global optimum. In [6] modifications to the traditional algorithm are proposed in order to obtain convergence to a local optimum and asymptotic convergence to a global optimum. A correct choice of the cognitive, social and inertial parameters guarantees the algorithm convergence to a point in space.

3. Constrained optimization

A careful inspection of Algorithm 1 reveals that only the objective function is used to see if the new particle position is more favourable than the previous one. In order to extend PSOA to constrained optimization it suffices to replace the test of step 3 in Algorithm 1. The new test must account for two simultaneous objectives: one is to minimize the objective function and the other is to obtain feasibility, the latter being more important.

To measure the infeasibility of a particle we propose the following function:

$$\mathrm{H}(x) = e^{\left(\sum_{i=1}^{m_E} \log(1+|c_i^{\mathrm{E}}(x)|) + \sum_{i=1}^{m_I} \log(1+[c_i^{\mathrm{I}}(x)]_+)\right)},$$

where $[c]_+ = \max\{0, c\}$. The infeasibility function $\mathrm{H}: R^n \to [1,+\infty[$ does not account for the simple bound

constraints as they are addressed by the projection of the particle position. We postpone this matter to the end of this section.

To illustrate the effect of the infeasibility measure consider *hs014* problem from Hock and Schittkowski [5]

$$\begin{aligned} \min_{x\in R^2} \quad & f(x) \equiv (x_1-2)^2 + (x_2-1)^2 \\ s.t. \quad & c^E(x) \equiv x_1 - 2x_2 + 1 = 0 \\ & c^I(x) \equiv \frac{x_1^2}{4} + x_2^2 - 1 \le 0. \end{aligned}$$

The equality part of function H is plotted in Fig 1, the inequality part in Fig 2, while the combined plot is shown in Fig 3.

It is an easy task to prove the following lemma and therefore we omit the proof.

Lemma 1 *For infeasibility function* $\mathrm{H}(x)$ *we have*

$$\mathrm{H}(x)\begin{cases} =1 & \text{if } x \text{ is feasible} \\ >1 & \text{if } x \text{ is infeasible.} \end{cases}$$

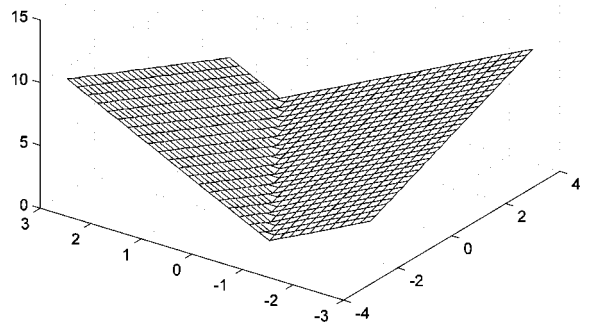

Fig 1. Equality $\mathrm{H}(x)$ plot for *hs014* problem

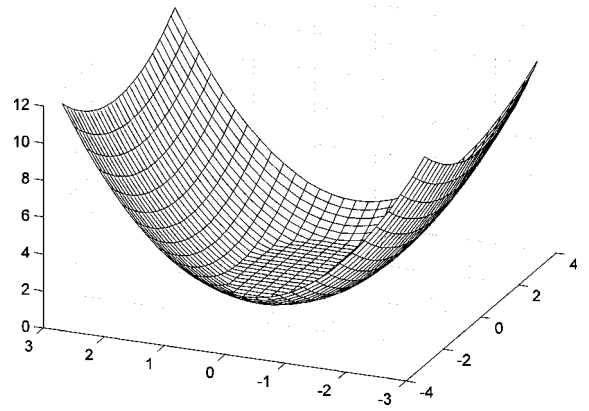

Fig 2. Inequality $\mathrm{H}(x)$ plot for *hs014* problem

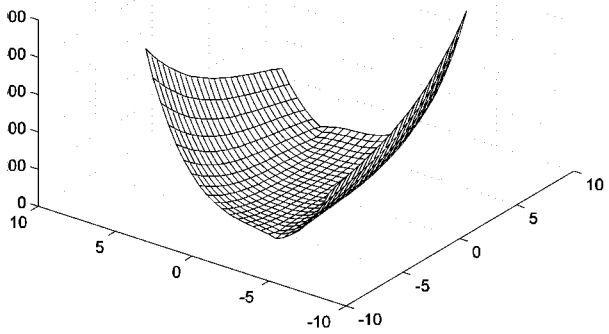

Fig 3. $\mathrm{H}(x)$ plot for *hs014* problem

Thus, problem (1) can then be replaced by the equivalent problem

$$\begin{aligned} & \min_{x\in R^n} f(x) \\ s.t. \quad & \mathrm{H}(x) = 1 \\ & l \le x \le u. \end{aligned} \tag{4}$$

The dominance concept from multiobjective optimization is now adapted to this constrained uniobjective function problem (see [8] for another application of the dominance concept to nonlinear optimization). The description of the dominance concept follows.

Suppose that we have two objective functions $f^1(x)$ and $f^2(x)$ (the extension to more than two objective functions is straightforward) and let us define vector $(f^1(x), f^2(x))$. Pair $(f^1(x^i), f^2(x^i))$ is said to dominate $(f^1(x^j), f^2(x^j))$ if and only if $f^k(x^i) \le f^k(x^j)$, $k = 1,2$, and $f^k(x^i) < f^k(x^j)$ for at least one $k \in \{1,2\}$.

Finding the global solution of problem (4) is somehow equivalent to find point $\bar{x}$ for which $(f(\bar{x}), \mathrm{H}(\bar{x}))$ dominates $(f(x), \mathrm{H}(x))$, for all $x \ne \bar{x}$ and $\mathrm{H}(\bar{x}) = 1$. We therefore must give priority in minimizing $\mathrm{H}(x)$ over minimizing $f(x)$.

For a given particle position x^p we consider that progress was attained if either

$$\mathrm{H}(y^p) > \mathrm{H}(x^p)$$

or

$$\left(\mathrm{H}(y^p) \le \mathrm{H}(x^p) \text{ or } \mathrm{H}(x^p) \le 1+\varepsilon\right) \text{ and } f(y^p) > f(x^p) \tag{5}$$

i.e., we consider that progress was attained, if either the improvement in the feasibility was attained (regardless of obtaining or not the improvement in the objective function) or the improvement in the feasibility was not obtained, but the objective function improvement is forced. The second condition of feasibility in (5) is used to allow a feasible

point with lower function value to be chosen. Getting, by chance, point y^p where $\mathrm{H}(y^p)$ is exactly one would prevent any feasible point with lower objective function value to become a leader, if $\varepsilon > 0$ tolerance was not used.

The simple bound constraints are imposed when computing the new particle position. The new particle is projected onto the region defined by the simple bounds using the following procedure:

$$x_i^p \leftarrow \begin{cases} x_i^p & \text{if} \quad l_i \le x_i^p \le u_i \\ l_i & \text{if} \quad x_i^p < l_i, \\ u_i & \text{if} \quad x_i^p > u_i \end{cases} \qquad i = 1,\ldots,n. \tag{6}$$

We present now the implemented algorithm for constrained nonlinear optimization.

Algorithm 2 *NLCPSOA*

1. Randomly initialize the swarm positions

$$\mathrm{X} = \left\{x^1(0),\ldots,x^s(0)\right\} \quad \text{and velocities}$$

$$\mathrm{V} = \left\{v^1(0),\ldots,v^s(0)\right\}.$$

2. Let $t = 0$ and $y^p(t) = x^p(t)$, $p = 1,\ldots,s$.

3. For all p in $\{1,\ldots,s\}$ do:

If $\mathrm{H}(y^p(t)) > H(x^p(t))$ or $((\mathrm{H}(y^p(t)) \le \mathrm{H}(x^p(t))$ or $\mathrm{H}(x^p(t)) \le 1+\varepsilon)$ and $f(y^p(t)) > f(x^p(t)))$ then set $y^p(t+1) = x^p(t)$ else set $y^p(t+1) = y^p(t)$.

4. For all p in $\{1,\ldots,s\}$ do:

Compute $v^p(t+1)$ and $x^p(t+1)$, using equations (2) and (3) and the projection (6).

5. If the stopping criterion is true, then stop. Otherwise set $t = t+1$ go to step 3.

4. Implementation details

4.1. The initial population

The initial swarm positions are randomly generated. If both simple bounds are finite ($l_i \neq -\infty$ and $u_i \neq \infty$) i coordinate of particle p position is randomly generated by uniform distribution $U(l_i, u_i)$.

If one of the simple bounds is not finite, the algorithm requires initial guess, $\hat{x}$. Coordinate *i* of particle *p* position is then randomly generated by using

$$x_i^p \sim \begin{cases} U\left(-\dfrac{\|\hat{x}\|\hat{x}_i}{2}, \dfrac{\|\hat{x}\|\hat{x}_i}{2}\right) & \text{if} \quad l_i = -\infty \text{ and } u_i = \infty \\ U\left(2\hat{x}_i - u_i, u_i\right) & \text{if} \quad l_i = -\infty \text{ and } u_i \neq \infty \\ U\left(l_i, 2\hat{x}_i - l_i\right) & \text{if} \quad l_i \neq -\infty \text{ and } u_i = \infty. \end{cases} \tag{7}$$

The initial guess, when provided by the user, is included in the initial swarm.

In spite of this strategy for randomly generating swarm we recommend the user to provide, whenever it is possible, bounds on the variables, since that use (7) can reduce or increase the search space without any advantage of solving the problem.

4.2. The stopping criterion

The most used stopping criterion in the particle swarm context has been to stop the algorithm, if a maximum number of iterations has been reached, or if the (known) global optimum has been attained to given tolerance.

In general, this stopping criterion is not appropriate, since the global optimum is not known in advance and stopping the algorithm when a maximum number of iterations is reached causes premature or too late termination.

A typical stopping criterion is based on the magnitude of the search direction. When the magnitude is approximately zero, progress seems no longer possible. This idea can also be used in the particle swarm context. Since in this case we are dealing with the population of points, a possible extension of this criterion is to stop when all the search directions are approximately zero, *i.e.*, the algorithm stops, if the maximum velocity for all particles is approximately zero. More formally we stop if

$$\max_{p\in\{1,\ldots,s\}} \left\| v^p(t+1) \right\| \le \varepsilon \tag{8}$$

where $\varepsilon > 0$ is a small tolerance.

Since the proposed algorithm allows a particle to *travel* outside the feasible region (although in a controlled way) a feasible point may not ever be found. When the condition (8) is satisfied, but the best ever particle position is infeasible ($\mathrm{H}(\hat{y}) > 1$), the swarm is reinitialized. The new swarm includes the best ever found particle and the other particles are randomly initialized following the previously described procedure.

4.3. Connection to AMPL

AMPL [9] is a mathematical programming language that allows the codification of optimization problems in a powerful and easy to learn language. AMPL also provides an interface that allows a wide variety of solvers to communicate with it and automatic differentiation, when requested.

The implemented algorithm is available in NLCPSOA (*NonLinear Constrained Particle Swarm Optimization Algorithm*) solver. The solver contains an interface to connect to AMPL allowing the user to write and solve a problem coded in AMPL in an extremely easy way. The user is referred to AMPL web page *http://www.ampl.com* for more details on AMPL and related solvers. The user can set sev-

eral options for NLCPSOA solver by using the standard argument passing provided by AMPL.

NLCPSOA returns the best ever found solution to AMPL.

5. Numerical results

In this section we provide numerical results with the collection of problems from literature. We are mainly interested in nonlinear constrained global optimization problems both with equality and inequality constraints.

The selected problems were coded in AMPL and are available in the internet web page *http://www.norg.uminho.pt/aivaz/*.

This avoids the need to fully describe the problems and the possible introduction of errors in their transcription.

The parameters used in the following numerical results are: $\mu = \nu = 2$, $\varepsilon = 10^{-4}$ and $\iota(t)$ is linear interpolation between 0.9 and 0.4.

The numerical results for the engineering problems described in [3] are presented in Table 1, Table 2, Table 3 and Table 4. Numerical comparison between a particle swarm penalty approach and genetic algorithms is made in [3] and we chose to report only the numerical results related to therein proposed particle swarm algorithm.

In Vessel design problem we have four variables and four inequality (≤ 0) constraints. f is the objective function value at the found solution. While Hu *et al.* [3] used ranges $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10 \leq x_3 \leq 200$ and $10 \leq x_4 \leq 200$, therein reported solution is out of range. We relaxed the lower bounds on x_1 and x_2 in order to obtain, approximately, the same solution.

In Beam design problem we have four variables and seven inequality (≤ 0) constraints.

In Spring design problem we have three variables and four inequality (≤ 0) constraints.

In Himmelblau's design problem we have five variables and three range constraints, $0 \leq c_1^I(x) \leq 92$, $90 \leq c_2^I(x) \leq 110$ and $20 \leq c_3^I(x) \leq 25$.

Table 1. Vessel design problem

	Vessel design problem in [3]	
	In [3]	NLCPSOA
x_1	8.125000E-01	7.781690E-01
x_2	4.375000E-01	3.846490E-01
x_3	4.209845E+01	4.031960E+01
x_4	1.766366E+02	2.000000E+02
c_1^I	0.000000E+00	-9.945530E-09
c_2^I	-3.588000E-02	-3.807780E-09
c_3^I	-5.820800E-11	-5.848560E-04
c_4^I	-6.336340E+01	-4.000000E+01
f	6.059131E+03	5.885330E+03

Table 2. Beam design problem

	Beam design problem in [3]	
	In [3]	NLCPSOA
x_1	2.057300E-01	2.013810E-01
x_2	3.470490E+00	3.231920E+00
x_3	9.036620E+00	1.000000E+01
x_4	2.057300E-01	2.013810E-01
c_1^I	0.000000E+00	-2.927840E-02
c_2^I	0.000000E+00	-4.972770E+03
c_3^I	-5.551115E-17	-3.586410E-07
c_4^I	-3.432984E+00	-3.326250E+00
c_5^I	-8.072960E-02	-7.638030E-02
c_6^I	-2.355403E-01	-2.390990E-01
c_7^I	-9.094947E-13	-3.917640E-03
f	1.724851E+00	1.814290E+00

Table 3. Spring design problem

	Spring design problem in [3]	
	In [3]	NLCPSOA
x_1 (d)	5.146637E-02	5.000000E-02
x_2 (D)	3.513839E-01	3.104140E-01
x_3 (N)	1.160866E+01	1.500000E+01
c_1^I	-3.336613E-03	-3.309970E-06
c_2^I	-1.097013E-04	-1.737420E-02
c_3^I	-4.026318E+00	-1.862670E+02
c_4^I	-7.312393E-01	-7.597240E-01
f	1.266614E-02	1.319260E-02

Table 4. Himmelblau's design problem

	Himmelblau's design problem in [3]	
	In [3]	NLCPSOA
x_1	7.800000E+01	7.800000E+01
x_2	3.300000E+01	3.300000E+01
x_3	2.707100E+01	2.711060E+01
x_4	4.500000E+01	4.500000E+01
x_5	4.496924E+01	4.500000E+01
c_1^I	9.200000E+01	9.200000E+01
c_2^I	1.004048E+02	9.887260E+01
c_3^I	2.000000E+01	2.001960E+01
f	-3.102556E+04	-3.101210E+04

Table 5 presents the comparison between the results reported in [1] and with our solver for other six problems. The values in the table refer to the objective function value

at the found solution. Some of these problems are from Hock and Schittkowski [10] test suit.

Problem *pso1* has a quadratic objective function with a linear equality constraint and a quadratic inequality constraint. Problem *pso2* has a cubic objective function with two quadratic inequality constraints. Problem *pso3* has a polynomial objective function and four polynomial inequality constraints. Problems *pso4* and *pso5* have quadratic objective functions with three range constraints. Problem *pso6* has a quadratic objective function with two linear inequality constraints.

Table 5. Obtained solutions *vs* best solutions in [1]

Problem	NLCPSOA	Best solution in [1]
pso1	1.393470E+00	1.393430E+00
pso2	-6.961810E+03	-6.961837E+03
pso3	6.806300E+02	6.806390E+02
pso4	-3.066550E+04	-3.154446E+04
pso5	-3.102640E+04	-3.154505E+04
pso6	-2.130000E+02	-2.130000E+02

One of the major drawbacks of stochastic algorithms is a large number of objective and constraint functions evaluations required. No reference on the number of functions or constraint evaluations is made in the previously reported numerical results and to obtain a feasible initial swarm a small number of particles in the swarm is recommended (around 20 in [3]).

In our proposed algorithm, the size of the swarm has a great impact on the ability of the algorithm in finding a feasible optimal solution. Thus, we propose a larger swarm size (around 100-300). In Table 6 we present the number of objective function evaluations required to converge to the solution (*nfun*). The number of constraints evaluations is equal to the number of function evaluations.

The choice of the parameters for the implemented algorithm has guaranteed convergence to a point in space confirmed by numerical experiences. Theoretical convergence to an optimum is not guaranteed.

Table 6. Number of function evaluations

Problem	*nfun*	Problem	*nfun*
vessel	879000	*pso2*	1461900
beam	960300	*pso3*	1689600
spring	757800	*pso4*	975300
himmelblau	784200	*pso5*	792900
pso1	1417200	*pso6*	879900

6. Conclusions

We proposed an algorithm for nonlinear constrained global optimization based on the dominance concept. This concept is used to select the best particle position and the swarm best ever particle position.

The proposed algorithm proved to be reliable and comparable to other already proposed algorithms for constrained optimization.

The algorithm finds the solution that minimizes the constraints violation. When the problem has an empty feasible region, the algorithm is able to present the solution that minimizes the constraints violation.

The preliminary numerical testing looks promising and in the future we will focus on designing a definitive algorithm with guaranteed convergence properties. Borrowing the ideas from **Error! Reference source not found.**], *sufficient* reduction in one of the measures H or f might be required in order to consider that progress was attained instead of simple reduction as described in (5).

Acknowledgments

The work is partially supported by FCT grant POCI/MAT/58957/2004 and Algoritmi research center.

References

1. Parsopoulos, K. E. and Vrahatis, M. N. Particle swarm optimization method for constrained optimization problems. In: Proceedings of the Euro-International Symposium on Computational Intelligence, 2002.
2. Hu, X. and Eberhart, R. Solving constrained nonlinear optimization problems with particle swarm optimization. In: Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando, USA, 2002.
3. Hu, X.; Eberhart, R. and Shi, Y. Engineering optimization with particle swarm. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2003), Indianapolis, Indiana, USA, 2003, p. 53–57.
4. Eberhart, R. and Kennedy, J. New optimizers using particle swarm theory. In: Proceedings of the 1995 6th International Sysmposium on Micro Machine and Human Science, p. 39–43.
5. Kennedy, J. and Eberhart, R. Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center, Piscataway, NJ. http://engr.iupui.edu/~shi/Coference/psopap4.html, p. 1942–1948.
6. Bergh, van den F. An Analysis of Particle Swarm Optimizers. PhD thesis, Faculty of Natural and Agricultural Science, University of Pretoria, November, 2001.
7. Schutte, J. F. and Groenwold, A. A. A study of global optimization using particle swarms. *Journal of Global Optimization*, Vol 31, No 1, 2003, p. 93–108.
8. Fletcher, R. and Leyffer, S. Nonlinear programming without a penalty function. *Mathematical Programming*, Vol 91, No 2, 2002, p. 239–269.
9. Fourer, R., Gay, D. M., and Kernighan, B. W. A modeling language for mathematical programming. *Management Science*, Vol 36, No 5, 1990, p. 519–554.
10. Hock, W. and Schittkowski, K. Test Examples for Nonlinear Programming Codes, *Lecture Notes in Economics and Mathematicl Systems*, Vol 187. New York: Springer-Verlag, 1981.

NELINIJINE TRAJEKTORIJA JUDANČIO DALELIŲ SRAUTO OPTIMIZAVIMAS

A. I. F. Vaz, E. M. G. P. Fernandes

Santrauka

Pagal dalelių srauto judėjimo teoriją sukurtas algoritmas, skirtas nelinijinės optimizacijos problemoms spręsti. Panašūs algoritmai būvo siūlomi ir anksčiau, tačiau kildavo keblumų su apribojimais. Pasiūlytame algoritme pritaikyta dominavimo koncepcija. Ši koncepcija naudojama geriausiai atskiros dalelės ir dalelių srauto padėčiai nustatyti. Taip pat algoritmui pasiūlytas *stop* kriterijus, išspręsti parinkti konkretūs uždavinių pavyzdžiai. Kad būtų paprasčiau suformuluoti ir išspręsti uždavinį, naujasis algoritmas užprogramuotas matematinio programavimo kalba.

Pagrindiniai žodžiai: nelinijinis optimizavimas, dalelių srauto optimizavimas, dominavimo koncepcija.

António Ismael de FREITAS VAZ. Professor. Department of Production and Systems, Engineering School, Minho University. Author and co-author of 20 scientific papers, several technical reports and conference talks.

Research interests: nonlinear, derivative free and global optimization, semi-infinite programming.

Edite Manuela da GRAÇA PINTO FERNANDES. Professor. Department of Production and Systems, Engineering School, Minho University. Author and co-author of 55 scientific papers and several technical reports. Research interests: numerical optimization, global optimization, semi-infinite programming, applied statistics.