Particle Swarm Optimization Method for Constrained Optimization Problems

Konstantinos E. Parsopoulos and Michael N. Vrahatis

Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), GR–26110 Patras, Greece
{kostasp, vrahatis}@math.upatras.gr

Abstract. The performance of the Particle Swarm Optimization method in coping with Constrained Optimization problems is investigated in this contribution. In the adopted approach a non–stationary multi–stage assignment penalty function is incorporated, and several experiments are performed on well–known and widely used benchmark problems. The obtained results are reported and compared with those obtained through different evolutionary algorithms, such as Evolution Strategies and Genetic Algorithms. Conclusions are derived and directions of future research are exposed.

1 Introduction

Constrained Optimization (CO) problems are encountered in numerous applications. Structural optimization, engineering design, VLSI design, economics, allocation and location problems are just a few of the scientific fields in which CO problems are frequently met [2], [4], [22]. The CO problem can be represented as the following nonlinear programming problem:

$$\min_x f(x), \quad x \in S \subset \mathbb{R}^n, \tag{1}$$

subject to the linear or nonlinear constraints

$$g_i(x) \leqslant 0, \quad i = 1, \ldots, m\,. \tag{2}$$

The formulation of the constraints in Eq. (2) is not restrictive, since an inequality constraint of the form $g_i(x) \geqslant 0$, can also be represented as $-g_i(x) \leqslant 0$, and an equality constraint, $g_i(x) = 0$, can be represented by two inequality constraints $g_i(x) \leqslant 0$ and $-g_i(x) \leqslant 0$.

The CO problem can be addressed using either deterministic or stochastic methods. However, deterministic methods such as Feasible Direction and Generalized Gradient Descent, make strong assumptions on the continuity and differentiability of the objective function $f(x)$ [2], [4], [5]. Thus, there is an ongoing interest for stochastic algorithms that can tackle the CO problem effectively. Although Evolutionary Algorithms (EA) have been developed primarily as unconstrained optimization methods, they are considered as a good alternative for

solving CO problems. Promising results have been reported during the past few years and several variants of Genetic Algorithms (GA) [6], Evolutionary Programming [3], and Evolution Strategies (ES) [20], have been proposed to cope with the CO problem [7], [8], [12], [22].

The most common approach for solving CO problems is the use of a penalty function. The constrained problem is transformed to an unconstrained one, by penalizing the constraints and building a single objective function, which in turn is minimized using an unconstrained optimization algorithm [2], [4], [19]. This is most probably the reason behind the popularity of the Penalty Function approach when EAs are used to address the CO problem [8], [22].

In this paper, the performance of the Particle Swarm Optimization method (PSO) [1], [11], in solving CO problems is investigated. The CO problem is tackled through the minimization of a non–stationary multi–stage assignment penalty function. The results of experiments performed on well–known test problems are reported and discussed in comparison with results obtained by other EAs. In the next section, the Penalty Function approach is briefly described. In Section 3, the Particle Swarm Optimization method is presented, and, in Section 4, the test problems as well as the experimental results are reported. The paper ends with conclusions and ideas for future research, reported in Section 5.

2 The Penalty Function Approach

The search space in CO problems consists of two kinds of points: feasible and unfeasible. Feasible points satisfy all the constraints, while unfeasible points violate at least one of them. The Penalty Function technique, solves the CO problem through a sequence of unconstrained optimization problems [8]. Up to date, no other method for defining pertinent penalty functions, than trial–and–error, exists [22]. If the penalty values are high, the minimization algorithms usually get trapped in local minima. On the other hand, if penalty values are low, they can hardly detect feasible optimal solutions.

Penalty functions are distinguished into two main categories: stationary and non–stationary. Stationary penalty functions, use fixed penalty values throughout the minimization, while in contrast, in non–stationary penalty functions, the penalty values are dynamically modified. In the literature, results obtained using non–stationary penalty functions are almost always superior to those obtained through stationary functions.

A penalty function is, generally, defined as [22]

$$F(x) = f(x) + h(k)\, H(x), \quad x \in S \subset \mathbb{R}^n, \tag{3}$$

where $f(x)$ is the original objective function of the CO problem in Eq. (1); $h(k)$ is a dynamically modified penalty value, where k is the algorithm's current iteration number; and $H(x)$ is a penalty factor, defined as

$$H(x) = \sum_{i=1}^{m} \theta(q_i(x))\, q_i(x)^{\gamma(q_i(x))}, \tag{4}$$

where $q_i(x) = \max\{0, g_i(x)\}$, $i = 1, \ldots, m$. The function $q_i(x)$ is a relative violated function of the constraints; $\theta(q_i(x))$ is a multi–stage assignment function [7]; $\gamma(q_i(x))$ is the power of the penalty function; and $g_i(x)$ are the constraints described in Eq. (2).

The functions $h(.)$, $\theta(.)$ and $\gamma(.)$, are problem dependent. In our experiments, a non–stationary multi–stage assignment penalty function was used. Details concerning the penalty function used in the experiments, are given in Section 4. In the next section, the PSO algorithm is described.

3 The Particle Swarm Optimization Method

PSO is a stochastic global optimization method which is based on simulation of social behavior. As in GA and ES, PSO exploits a population of potential solutions to probe the search space. In contrast to the aforementioned methods in PSO no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead of mutation PSO relies on the exchange of information between individuals, called *particles*, of the population, called *swarm*. In effect, each particle adjusts its trajectory towards its own previous best position, and towards the best previous position attained by any member of its neighborhood [9]. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape. To visualize the operation of the method consider the case of the single–objective minimization case; promising regions in this case possess lower function values compared to others, visited previously.

Several variants of PSO have been proposed up to date, following Eberhart and Kennedy who were the first to introduce this method [1], [10], [11]. The variants which were applied in our experiments are exposed in the following paragraphs.

Initially, let us define the notation adopted in this paper: assuming that the search space is D–dimensional, the i-th particle of the swarm is represented by a D–dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the best particle of the swarm, i.e. the particle with the lowest function value, is denoted by index g. The best previous position (i.e. the position corresponding to the best function value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the position change (velocity) of the i-th particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

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

$$V_i^{k+1} = \chi \left(w V_i^k + c_1 r_{i1}^k (P_i^k - X_i^k) + c_2 r_{i2}^k (P_g^k - X_i^k) \right), \tag{5}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \tag{6}$$

where $i = 1, 2, \ldots, N$, and N is the size of the population; 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; r_{i1} and r_{i2} are random numbers uniformly distributed within the range $[0, 1]$. Eq. (5) is used to determine the i-th particle's new velocity, at each iteration, while Eq. (6) provides the new position of the i-th particle, adding its new velocity, to its current position. The performance of each particle is measured according to a fitness function, which is problem–dependent. In optimization problems, the fitness function is usually identical with the objective function under consideration.

The role of the inertia weight w is considered important for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. Thus, the parameter w regulates the trade–off between the global (wide–ranging) and the local (nearby) exploration abilities of the swarm. A large inertia weight facilitates exploration (searching new areas), while a small one tends to facilitate exploitation, i.e. fine–tuning the current search area. A proper value for the inertia weight w provides balance between the global and local exploration ability of the swarm, and, thus results in better solutions. Experimental results imply that it is preferable to initially set the inertia to a large value, to promote global exploration of the search space, and gradually decrease it to obtain refined solutions [21]. The initial population can be generated either randomly or by using a Sobol sequence generator [18], which ensures that the D-dimensional vectors will be uniformly distributed within the search space.

The PSO technique has proven to be very efficient for solving real valued global unconstrained optimization problems [13]–[17]. In the next section experimental results of the performance of PSO in CO problems are reported.

4 Experimental Results

The performance of three variants of PSO was investigated on well–known and widely used test problems. One variant utilizes only inertia weight (denoted as PSO-In), a second variant utilizes only constriction factor (denoted as PSO-Co), and the last one utilizes both inertia weight and constriction factor (denoted as PSO-Bo). For all variants, fixed values, considered as default, for the PSO's parameters were used: $c_1 = c_2 = 2$; w was gradually decreased from 1.2 towards 0.1; $\chi = 0.73$. Some variants of PSO, impose a maximum value on the velocity, V_{max}, to prevent the swarm from explosion. In our experiments V_{max} was always fixed, to the value of $V_{max} = 4$. The size of the swarm was set equal to 100, 10 runs were performed for each test problem, and the PSO algorithm ran for 1000 iterations, in each case. A violation tolerance was used for the constraints. Thus, a constraint $g_i(x)$ was assumed to be violated, only if $g_i(x) > 10^{-5}$.

Regarding the penalty parameters, the same values as the values reported in [22] were used, to obtain results comparable to the results obtained using different EA, in [22]. Specifically, if $q_i(x) < 1$, then $\gamma(q_i(x)) = 1$, otherwise $\gamma(q_i(x)) = 2$. Moreover, if $q_i(x) < 0.001$ then $\theta(q_i(x)) = 10$, else, if $q_i(x) \leqslant 0.1$ then $\theta(q_i(x)) = 20$, else, if $q_i(x) \leqslant 1$ then $\theta(q_i(x)) = 100$, otherwise $\theta(q_i(x)) =$

300. Regarding the function $h(.)$, it was set to $h(k) = \sqrt{k}$ for Test Problem 1, and $h(k) = k\sqrt{k}$ for the rest problems.

The test problems are defined immediately below:

TEST PROBLEM 1, [4]:

$$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2,$$

subject to $x_1 = 2x_2 - 1,\ x_1^2/4 + x_2^2 - 1 \leqslant 0$. The best known solution is $f^* = 1.3934651$.

TEST PROBLEM 2, [2]:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3,$$

subject to $100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leqslant 0,\ (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leqslant 0,\ 13 \leqslant x_1 \leqslant 100,\ 0 \leqslant x_2 \leqslant 100$. The best known solution is $f^* = -6961.81381$.

TEST PROBLEM 3, [5]:

$$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + \\ +10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,$$

subject to $-127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0,\ -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0,\ -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0,\ 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0,\ -10 \leqslant x_i \leqslant 10,\ i = 1, \ldots, 7$. The best known solution is $f^* = 680.630057$.

TEST PROBLEMS 4 AND 5 , [5]:

$$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,$$

subject to $0 \leqslant 85.334407 + 0.0056858T_1 + T_2x_1x_4 - 0.0022053x_3x_5 \leqslant 92,\ 90 \leqslant 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110,\ 20 \leqslant 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25,\ 78 \leqslant x_1 \leqslant 102,\ 33 \leqslant x_2 \leqslant 45,\ 27 \leqslant x_i \leqslant 45,\ i = 3, 4, 5$, where $T_1 = x_2x_5$ and $T_2 = 0.0006262$ for Test Problem 4, and $T_1 = x_2x_3$, $T_2 = 0.00026$ for Test Problem 5. The best known solution for Test Problem 4 is $f^* = -30665.538$, while for Test Problem 5 it is unknown.

TEST PROBLEM 6, [12]:

$$f(x, y) = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 - 10y - 0.5\sum_{i=1}^{5} x_i^2,$$

subject to $6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 - 6.5 \leqslant 0,\ 10x_1 + 10x_3 + y \leqslant 20,\ 0 \leqslant x_i \leqslant 1,\ i = 1, \ldots, 5,\ 0 \leqslant y$. The best known solution is $f^* = -213.0$.

For each test problem, the mean and the best solution obtained in all 10 runs, as well as the corresponding sum of violated constraints, were recorded. The

Table 1. The mean and the best solution found in all 10 runs, for each method and problem. The standard deviation of the optimal values for the 10 runs is reported. In the parentheses the corresponding sums of violated constraints are reported

Problem	Method	Mean Solution (Sum V.C.)	St.D.	Best Solution (Sum V.C.)
1	PSO-In	1.394006 (0.000014)	0.0015	1.393431 (0.000020)
	PSO-Co	1.393431 (0.000020)	0.0	1.393431 (0.000020)
	PSO-Bo	1.393431 (0.000020)	0.0	1.393431 (0.000020)
2	PSO-In	-6960.866 (0.0000037)	0.608	-6961.798 (0.0000087)
	PSO-Co	-6961.836 (0.000019)	0.0011	-6961.837 (0.000019)
	PSO-Bo	-6961.774 (0.000013)	0.14	-6961.837 (0.000019)
3	PSO-In	680.671 (0.000008)	0.034	680.639 (0.000019)
	PSO-Co	680.663 (0.00034)	0.050	680.635 (0.00130)
	PSO-Bo	680.683 (0.000015)	0.041	680.636 (0.0)
4	PSO-In	-31526.304 (1.297)	18.037	-31543.484 (1.311)
	PSO-Co	-31528.289 (1.326)	12.147	-31542.578 (1.311)
	PSO-Bo	-31493.190 (1.331)	131.67	-31544.459 (1.311)
5	PSO-In	-31523.859 (0.958)	17.531	-31544.036 (0.997)
	PSO-Co	-31526.308 (0.965)	19.153	-31543.312 (0.996)
	PSO-Bo	-31525.492 (0.968)	23.392	-31545.054 (0.999)
6	PSO-In	-213.0 (0.0)	0.0	-213.0 (0.0)
	PSO-Co	-213.0 (0.0)	0.0	-213.0 (0.0)
	PSO-Bo	-213.0 (0.0)	0.0	-213.0 (0.0)

results for all test problems are reported in Table 1. In all test problems, the three variants of PSO exhibited similar results. In most cases PSO outperformed the results reported in [22] for other EA. Proper fine–tuning of the PSO's parameters may result in better solutions.

5 Conclusions and Further Work

The capability of the PSO method to address CO problems was investigated through the performance of numerous experiments on well–known and widely used test problems. Preliminary results, obtained through the use of a non–stationary multi–stage penalty function, imply that PSO is a good alternative for tackling CO problems. In most cases PSO detected superior solutions than those obtained through other EAs, as reported in [22]. It should be mentioned that the results were competitive in all cases, despite the fact that only the default parameters of PSO were considered. The performance of the three PSO's variants was similar for all test problems.

Future work will include investigation of the PSO's performance in other benchmark and real–life problems, as well as the development of specialized operators that will indirectly enforce feasibility of the particles and guide the swarm towards the optimum solution, as well as fine–tuning of the parameters that may result in better solutions.

References

1. Eberhart, R.C., Simpson, P.K., Dobbins, R.W.: Computational Intelligence PC Tools. Academic Press Professional, Boston (1996)
2. Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained Global Optimization Algorithms. Lecture Notes in Computer Science, Vol. 455. Springer-Verlag, Berlin Heidelberg New York (1987)
3. Fogel, D.B.: An Introduction to Simulated Evolutionary Optimization. IEEE Trans. Neural Networks **5(1)** (1994) 3–14
4. Himmelblau, D.M.: Applied Nonlinear Programming. McGraw–Hill (1972)
5. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, Vol. 187. Springer-Verlag, Berlin Heidelberg New York (1981)
6. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press (1992)
7. Homaifar, A., Lai, A.H.–Y., Qi, X.: Constrained Optimization via Genetic Algorithms. Simulation **2(4)** (1994) 242–254
8. Joines, J.A., Houck, C.R.: On the Use of Non–Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's. Proc. IEEE Int. Conf. Evol. Comp. (1994) 579–585
9. Kennedy, J.: The Behavior of Particles. Evol. Progr. VII (1998) 581–587
10. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proc. IEEE Int. Conf. Neural Networks. Piscataway, NJ (1995) 1942–1948
11. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (2001)
12. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer–Verlag, New York (1992)
13. Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Objective Function "Stretching" to Alleviate Convergence to Local Minima. Nonlinear Analysis TMA **47(5)** (2001) 3419–3424
14. Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Stretching Technique for Obtaining Global Minimizers Through Particle Swarm Optimization. Proc. Particle Swarm Optimization Workshop. Indianapolis (IN), USA (2001) 22–29
15. Parsopoulos, K.E., Vrahatis, M.N.: Modification of the Particle Swarm Optimizer for Locating All the Global Minima. V. Kurkova, N. Steele, R. Neruda, M. Karny (eds.), Artificial Neural Networks and Genetic Algorithms. Springer, Wien (Computer Science Series) (2001) 324–327
16. Parsopoulos, K.E., Laskari, E.C., Vrahatis, M.N.: Solving ℓ_1 Norm Errors-In-Variables Problems Using Particle Swarm Optimizer. M.H. Hamza (ed.), Artificial Intelligence and Applications. IASTED/ACTA Press (2001) 185–190
17. Parsopoulos, K.E., Vrahatis, M.N.: Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method. A. Grmela, N.E. Mastorakis (eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation. WSEAS Press (2002) 216–221
18. Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P.: Numerical Recipes in Fortran 77. Cambridge University Press, Cambridge (1992)
19. Rao, S.S.: Optimization: Theory and Applications. Wiley Eastern Limited (1977)
20. Schwefel, H.–P.: Evolution and Optimum Seeking. Wiley (1995)
21. Shi, Y., Eberhart, R.C.: Parameter Selection in Particle Swarm Optimization. Evolutionary Programming VII (1998) 591–600
22. 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 Heidelberg New York (1997) 201–211