International Journal of Research Publication and Reviews, Vol 4, no 3, pp 2539-2543, March 2023

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Constrained Optimization Problems using the Particle Swarm Method

Pranjali Dewangan

Dr. C. V. Raman University, Kota, Bilaspur (C.G.)

ABSTRACT

In this addition, it is examined how well the Particle Swarm Optimization method handles constrained optimization issues. A non-stationary multi-stage assessment penalty function is included in the chosen method, and several experiments are carried out using well-known and frequently used benchmark problems. The findings are reported and contrasted with those obtained using various evolutionary algorithms, including Evolutionary Strategies and Genetic Algorithms. Conclusions are drawn, and prospective research directions are revealed.

Keywords: Genetic Algorithms, constrained optimization, Swarm Optimization

INTRODUCTION

Numerous applications involve issues with constrained optimization (CO). Just a few of the scientific fields in which CO problems are commonly encountered include structural optimization, engineering design, VLSI design, economics, allocation, and location issues [2], [4], [22]. The following nonlinear programming issue can be used to illustrate the CO problem:

$min\ f_x\ (x);\ x \in S \subseteq R\ n\ ;$ (1)

Subject to the linear or nonlinear constraints

$g_i(x) \leq 0; i = 1;:::;m:$ (2)

Since an inequality constraint of the form gi(x) > 0 can also be represented as gi(x) 6 0, and an equality constraint of the form gi(x) = 0 can be represented by two inequality constraints of the form gi(x) 6 0 and gi(x) 6 0, the construction of the constraints in Eq. (2) is not restrictive.

You can use stochastic or deterministic approaches to solve the CO issue. The continuity and differentiability of the objective function f (x), however, are strong assumptions made by deterministic techniques like Feasible Direction and Generalized Gradient Descent [2], [4], and [5]. As a result, stochastic algorithms that effectively handle the CO issue are still in demand.

Even though evolutionary algorithms (EA) were initially created as unrestricted optimization techniques, they are now viewed as a viable option for resolving CO issues. A number of variations of Genetic Algorithms (GA) [6], Evolutionary Programming [3], and Evolution Strategies (ES) [20] have been suggested to address the CO problem in light of promising results that have been reported over the past few years [7], [8], [12], and [22]. The application of a punishment function is the most typical strategy for resolving CO issues. By penalising the constraints and creating a singular objective function, the constrained problem is changed into an unconstrained one and then minimised using an unconstrained optimization algorithm [2], [4], [19].

This is most likely the cause of the Penalty Function approach's popularity when EAs are used to tackle the CO issue [8], [22].

The effectiveness of the Particle Swarm Optimization (PSO) technique ([1], [11]) in solving CO problems is examined in this paper. The minimization of a non-stationary multi-stage assignment penalty function is used to solve the CO issue. The findings of tests conducted on well-known test problems are given, and they are compared with those of other EAs. The Penalty Function approach is thoroughly explained in the part that follows.

2. The Punishment Capability Approach

In CO problems, feasible and unfeasible points make up the search area. All of the constraints are met by feasible points, whereas at least one condition is broken by infeasible points. The CO problem is solved using the Penalty Function method through a series of unrestricted optimization issues [8]. Trial and error is the only technique that has been found to date for determining relevant penalty functions [22]. In most cases, the minimization algorithms become stuck in local minima when the penalty values are large. Conversely, if penalty values are small, they hardly ever find workable optimum solutions.

The two major categories of penalty functions are stationary and non-stationary. In opposition to non-stationary penalty functions, which use xed penalty values throughout the minimization, stationary penalty functions use fixed penalty values. Results obtained using non-stationary punishment functions are almost always better in the literature than results obtained using stationary functions.

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

$F(x) = f(x) + h(k)\,H(x); x \in S \subset R^n$; (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

$$H(x) = \sum_{i=1}^{m} \theta$$

$(q_i(x))\ q_i(x)^{(q}i\ (x))$; (4)

Where gi(x)g, I = 1,:::, and m, and qi(x) = maxf0. The function qi(x) is a measure of how often the constraints have been broken; it is also a multi-stage assignment function [7]; the power of the penalty function; and the constraints listed in Eq (2).

The operations h(:), (:), and (:) rely on the problem. A non-stationary multi-stage assignment penalty function was used in our trials. The trials' penalty function's specifics are described in Section 4 of the paper. The PSO algorithm is discussed in the part after that.

THE PARTICLE SWARM OPTIMIZATION METHOD

PSO is a stochastic global optimization technique built on social behaviour modelling. PSO uses a population of possible solutions, just like GA and ES, to explore the search space. In PSO, no operators influenced by natural evolution are used to extract a new generation of candidate solutions, in contrast to the methods described above. PSO depends on communication between the population's particles, also known as the swarm's population, as opposed to mutation. Each particle in e ect modifies its trajectory in order to reach both its own best previous position and the best previous position achieved by any member of its neighborhood [9].

PSO's global version considers the entire swarm to be the area. In order to find promising areas of the landscape, information is shared globally, and particles benefit from the findings and prior experience of all other companies. Consider the single-objective minimization case to see how the technique works; in this case, promising regions have lower function values than previously visited ones.

Since Eberhart and Kennedy were the first to present this method, a number of variations of PSO have been proposed [1], [10], and [11]. The following paragraphs reveal the variations that were used in our experiments.

Let's first define the notation used in this paper: the best particle in the swarm, or the particle with the lowest function value, is indicated by index g. Assuming the search space is D-dimensional, the i-th particle of the swarm is represented by a D-dimensional vector Xi = (xi1; xi2;:::; xiD). The i-th particle's best previous location, or the position that corresponds to the best function value, is noted and represented as Pi = (pi1; pi2;::: ; piD), and its position change (velocity), Vi = (vi1; vi2;::: ; viD). The following equations are used to control the particles (the subscripts indicate the iteration):

$$V_i^{k+1} = \quad wV_i^k + c_1 r^k (P_i^k \quad X^k) + c_2 r^k (P^k \quad X^k)\ ; \qquad (5)$$

$$X^{k+1} = X^k + V^{k+1}; \qquad (6)$$

where i = 1; 2;::: ;N , and N is the size of the population; is a constriction factor which is used to control and constrict velocities; w is the inertia weight;

The cognitive parameter (c1) and the social parameter (c2) are two positive constants; ri1 and ri2 are random values with uniform distributions within the range [0; 1]. At each iteration, the i-th particle's new velocity is calculated using Eq. (5), and its new location is calculated using Eq. (6) by adding its new velocity to its current position. Each particle's effectiveness is evaluated using a problem-dependent tness function. The tness function in optimization issues is typically the same as the objective function being studied.

The inertia weight w is thought to play a significant part in the convergence behaviour of the PSO. The inertia weight is used to manage how the history of previous velocities affects the present velocities. The trade-off between the swarm's global (wide-ranging) and local (nearby) exploration skills is thus regulated by the parameter w. Exploration (searching new areas) is made easier by a large inertia weight, whereas exploitation (fine-tuning the present search area) is made easier by a small one. A proper number for the inertia weight w balances the swarm's ability to explore both locally and globally, leading to better solutions.

According to experimental findings, it is better to set the inertia to a high value at first to encourage broad exploration of the search space and to progressively lower it to find refined solutions [21]. A Sobol sequence generator [18] or a random number generator can be used to create the initial population, both of which guarantee that the D-dimensional vectors will be evenly distributed throughout the search area. Real valued global unconstrained optimization problems have been successfully solved using the PSO method [13] through [17]. The performance of PSO in CO problems is reported in the following section based on experimental findings.

EXPERIMENTAL RESULTS

Three PSO variants' performance was examined using well-known and frequently used test issues. PSO-In stands for the version that only uses inertia weight, PSO-Co for the variant that only uses constriction factor, and PSO-Co for the variant that uses both inertia weight and constriction factor (denoted as PSO-Bo). The PSO's parameters for all variants were set to the following xed values, which are thought of as defaults: c1 = c2 = 2, w was progressively decreased from 1.2 to 0.1, and = 0:73. To stop the swarm from exploding, some PSO variants put a maximum value on the velocity, Vmax. In our tests, Vmax was consistently multiplied by 2, resulting in Vmax = 4. For each test issue, the PSO algorithm was run ten times for a total of 1000 iterations with the swarm size set to 100. The limits had a violation tolerance. As a result, a restriction gi(x) was only considered broken if gi(x) > 10 5.

The penalty parameters were set to the same values as those reported in [22] in order to produce results that were similar to those of [22]'s different EA. Particularly, if qi(x) 1, then (qi(x)) = 1. 300. Regarding the function h(:), it was set to h(k)

Otherwise $(q_i(x))$ = 2. Moreover, if $q_i(x)$ < 0:001 then $(q_i(x))$ = 10, else, if $q_i(x)$

0:1 then $(q_i(x))$ = 20, else, if $q_i(x)$ 6 1 then $(q_i(x))$ = 100, otherwise $(q_i(x))$ =and h(k) = k k for the rest problems.

The test problems are defined immediately below:

Test Problem 1, [4]:

f (x) = (x1 ; 2)^2 + (x2 ; 1)^2;

subject to x1 = 2x2 ; 1; x$^2_1$=4 + x$^2_2$; 1 6 0. The best known solution is f = 1:3934651.

Test Problem 2, [2]:

f(x) = (x1 ; 10)^3 + (x2 ; 20)^3;

subject to 100 ;(x1 ;5)^2 ;(x2 ;5)^2 6 0; (x1 ;6)^2 + (x2 ;5)^2 ;82:81 6 0; 13 6

x1 6 100; 0 6 x2 6 100. The best known solution is f = ;6961:81381.

Test Problem 3, [5]:

f (x) = (x10)^2 + 5(x12)^2 + x^4 + 3(x11)^2 +1 ;2 ;3 4 ;

subject to ;127+2x$^2_1$+3x$^4_2$+x3+4x$^2_4$+5x5 6 0; ;282+7x1+3x2 +10x$^2_3$+x4;x5 6 0; ;196 + 23x1 + x$^2_2$ + 6x$^2_6$; 8x7 6 0; 4x$^2_1$ + x$^2_2$; 3x1x2 + 2x$^2_3$ + 5x6 ; 11x7 6 0; ;10 6 xi 6 10; i = 1; : : : ; 7. The best known solution is f = 680:630057.

Test Problems 4 and 5 , [5]:

f (x) = 5:3578547x$^2_3$ + 0:8356891x1x5 + 37:293239x1 ; 40792:141; subject to 0 6 85:334407 + 0:0056858T1 + T2x1x4 ; 0:0022053x3x5 6 92; 90 6 80:51249+0:0071317x2x5 +0:0029955x1x2+0:0021813x$^2_3$ 6 110; 20 6 9:300961+ 0:0047026x3x5 + 0:0012547x1x3 + 0:0019085x3x4 6 25; 78 6 x1 6 102; 33 6 x2 6 45; 27 6 xi 6 45; i = 3; 4; 5, where T1 = x2x5 and T2 = 0:0006262

for Test Problem 4, and T1 = x2x3, T2 = 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) = ;10:5x1 ; 7:5x2 ; 3:5x3 ; 2:5x4 ; 1:5x5 ; 10y ; 0:5 X x^{2}i;

i=1

subject to 6x1 + 3x2 + 3x3 + 2x4 +x5 ;6:5 6 0, 10x1 + 10x3 +y 6 20, 0 6 xi 6 1,

i = 1; : : : ; 5, 0 6 y. The best known solution is f = ;213:0.

The mean, the best solution found across all 10 runs, and the associated total of violated constraints were all recorded for each test issue. Section 1 is used. The average and top result for each technique and problem across all 10 runs. The optimal values' standard deviation across the 10 runs is given. The matching sums of the violated constraints are given in parentheses.

Problem	***Method***	***Mean Solution (Sum V.C.)***	***St.D.***	***Best Solution (Sum V.C.)***
1	PSO-In	1.384006 (0.000014)	0.0014	1.383431 (0.000019)
	PSO-Co	1.373431 (0.000020)	0.0	1.383431 (0.000019)
	PSO-Bo	1.363431 (0.000020)	0.0	1.393431 (0.000020)
2	PSO-In	-6860.866 (0.0000037)	0.608	-6861.798 (0.0000087)
	PSO-Co	-6861.836 (0.000019)	0.0011	-6861.837 (0.000019)
	PSO-Bo	-6861.774 (0.000013)	0.14	-6861.837 (0.000019)

3	PSO-In	670.671 (0.000008)	0.034	670.639 (0.000019
	PSO-Co	670.663 (0.00034)	0.050	670.635 (0.00130)
	PSO-Bo	670.683 (0.000015)	0.041	670.636 (0.0)
4	PSO-In	-30526.303 (1.296)	18.037	-30543.484 (1.312)
	PSO-Co	-30528.285 (1.325)	12.147	-30542.578 (1.312)
	PSO-Bo	-30493.194 (1.332)	131.67	-30544.459 (1.312)
5	PSO-In	-30523.857 (0.957)	17.531	-30544.036 (0.996)
	PSO-Co	-30526.306 (0.964)	19.153	-30543.312 (0.995)
	PSO-Bo	-30525.495 (0.966)	23.392	-30545.054 (0.998)

Table 1 lists the outcomes for each test issue. The outcomes of the three PSO variants were comparable for all test problems. Most of the time, PSO did better than the outcomes for other EA reported in [22]. The PSO's parameters can be properly tuned to produce improved solutions.

CONCLUSIONS AND FURTHER WORK

By conducting numerous experiments on well-known and frequently used test problems, it was determined whether the PSO method could handle CO problems. Initial findings from the application of a non-stationary multi-stage penalty function suggest that PSO is a viable option for dealing with CO issues. According to [22], PSO typically found better answers than those attained through other EAs. Despite only taking into account PSO's default parameters, it should be noted that all of the outcomes were competitive. For every test issue, the three PSO variants performed similarly.

The PSO's performance in other benchmark and real-world problems will be examined in future work, along with the development of specialised operators that will indirectly enforce particle feasibility and direct the swarm towards the best solution, as well as fine-tuning of the parameters that might produce better solutions.

References

1. Eberhart,R.C., Simpson, P.K., Dobbins, R.W.: Computational ntelligence PC Tools. Academic Press Professional, Boston (2011)
2. Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained GlobalOptimization Algorithms. Lecture Notes in Computer Science, Vol. 455. Springer- Verlag, Berlin Heidelberg New York (2010)
3. Fogel, D.B.: An Introduction to Simulated Evolutionary Optimization. IEEE Trans.Neural Networks 5(1) (2006) 3{14
4. Himmelblau, D.M.: Applied Nonlinear Programming. McGraw{Hill (1972)
5. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lec- ture Notes in Economics and Mathematical Systems, Vol. 187. Springer-Verlag, Berlin Heidelberg New York (2000)
6. Holland, J.H.: Adaptation in Natural and Arti cial Systems. MIT Press (1992)
7. Homaifar, A., Lai, A.H.{Y., Qi, X.: Constrained Optimization via Genetic Algo- rithms. Simulation 2(4) (1999) 242{254
8. Joines, J.A., Houck, C.R.: On the Use of Non{Stationary Penalty Functions to SolveNonlinear 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 Anal- ysis 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.: Modi cation of the Particle Swarm Optimizer for Locating All the Global Minima. V. Kurkova, N. Steele, R. Neruda, M. Karny (eds.), Arti cial Neural Networks and Genetic Algorithms. Springer, Wien (Com- puter Science Series) (2001) 324{327
16. Parsopoulos, K.E., Laskari, E.C., Vrahatis, M.N.: Solving `1 Norm Errors-In- Variables Problems Using Particle Swarm ptimizer. M.H. Hamza (ed.), Arti cial Intelligence and Applications. IASTED/ACTA Press (2001) 185{190
17. Parsopoulos, K.E., Vrahatis, M.N.: Initializing the Particle Swarm Optimizer Us- ing 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. Evo- lutionary 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