International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 10 (2014), pp. 965-972

http://www. irphouse.com

A Hybrid of Random Search and PSO for Solving Constrained Multi-Objective Optimization Problems

Ashok Pal[1], and S.B.Singh[2]

[1,2]Department of Mathematics
Punjabi University, Patiala (Punjab)-India

Abstract

Suitable solutions for multi-objective optimization problems are investigated as a set of solutions in the literature, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. The proposed algorithm in this paper is a hybrid of the particle swarm optimization algorithm [1] and a random search technique with quadratic approximation formula[2] named Random Search Quadratic approximation Particle Swarm Optimization (RQPSO) algorithm. In this proposed algorithm, a probability having a certain value provided by the user has been fixed. In every iteration, if the uniformly generated random number r(0,1) is less than that value , then the velocity vector is generated by the standard PSO algorithm otherwise it is generated by random search technique with quadratic approximation formula [2]. The proposed algorithm is tested on 13 test problems taken from the literature and are listed in the paper. Results from the proposed algorithm are compared with the known results from the literature and it has been observed that the proposed algorithm improves its performance in number of cases.
Keywords- Particle Swarm Optimization(PSO), Multi-objective optimization problems(MOOP) and Random Search Quadratic approximation Particle Swarm Optimization (RQPSO).

1. Introduction

An art of selecting the best alternative amongst a set of options is called an optimization i.e. the process to maximize the profit and minimize the losses is an optimization. In most of the non linear programming problems (NLPP), a global optimal solution rather than a local optimal solution is desired.These days with the growing awareness of the advantages of optimization and easy availability of

computers a new category of optimization techniques known as 'heuristic based optimization technique' has also introduced in optimization theory. No Free Lunch theorem [3] states that the performance of all optimization algorithms for the set of all possible optimization functions, is equivalent. There does not exist a computational algorithm which can guarantee to find the global optimal solution of each and every optimization problem in a finite number of steps. Therefore in practical real life optimization problems, there is always a need for developing more robust numerically oriented computational techniques which could be used to solve different types of optimization problems arising in different real life situations[4].

When an optimization problem involves more than one objective function, the task of finding one or more optimal solutions is known as multi-objective optimization. Multi-objective optimization problems consist of several objectives that are necessary to be handled simultaneously. Such problems arise in many applications, where two or more, sometimes competing and/or incommensurable, objective functions have to be minimized/ maximized concurrently. A general formulation of a multi-objective optimization problem consists of a number of objectives with inequality and/or equality constraints. Mathematically, the problem can be expressed as follows:

Min/Max $F(x) = [f_1(x), f_2(x), \ldots, f_m(x)]$ subject to the constraints $g_k(x) \leq c_k$, $k = 1, 2, \ldots, K$ and $x_i^{(L)} \leq x_i \leq x_i^{(U)}$ $i = 1, 2, \ldots, n$. where $x = (x_1, x_2, \ldots, x_n)$ is solution vector in X, which is a set of feasible solutions. Objective vector $F(x) = [f_1(x), f_2(x), \ldots, f_m(x)]$ maps solution vector $x = (x_1, x_2, .., x_n)$ in decision space to objective space for $m \geq 2$ i.e. the problem is to find $x = (x_1, x_2, \ldots, x_n)$ which optimize $f_1(x), f_2(x), \ldots, f_m(x)$ such that $g_k(x) \leq c_k$. In general, no solution vector X exists that minimizes all the m objective functions simultaneously. There is no single solution to the problem rather, we get a set of solutions known as Pareto-optimal set/Pareto-front i.e. the optimal solution is not a single point but it is a region in the design space called the pareto-front.

PSO is a robust stochastic optimization technique developed by Dr. James Kennedy (social-psychologist in USA) and Dr. Russell Eberhart (Professor of electrical engineering in USA) [5] inspired by social behavior of bird flocking or fish schooling. Individuals in a population learn from previous experiences and the experiences of those around them. The direction of movement is a function of current position, velocity, location of individuals 'best' success and location of neighbors 'best' successes. Therefore, each individual in a population will gradually move towards the 'better' areas of the problem space and the overall population moves towards 'better' areas of the problem space.The two basic equations in the working of the standard particle swarm optimization (SPSO) algorithm [6] are the velocity and position vectors which are given as

$$v_i(t+1) = \overbrace{wv_i(t)}^{\substack{inertial\\ weight\\ component}} + \overbrace{c_1 r_1[p_i(t) - x_i(t)]}^{\substack{cognitive\\ component}} + \overbrace{c_2 r_2[p_g(t) - x_i(t)]}^{\substack{social\\ component}} \ldots (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Where, $x_i = (x_{i1}, x_{i2}, \ldots\ldots x_{in})$ is the position of the ith particle, $p_i = (p_{i1}, p_{i2}, \ldots\ldots p_{in})$ is the best position of the ith particle achieved based on its own experience so far and

$p_g = (p_{g1}, p_{g2}, \ldots\ldots p_{gn})$ is the position of the best particle based on the overall swarms experience. Shi and Eberhart [6]suggested that a value between 0.8 and 1.2 provided good results and Eberhart and Shi [7] shown it decreases linearly between 0.9 to 0.4.

Clerc and Kennedy [8][1] used a constriction factor C to generate the new velocity and observed that there can be many ways to use the constriction coefficient C. One of the simplest methods of using C is the following :

$$v_i(t+1) = C(v_i(t) + c_1 r_1(p_i(t) - x_i(t)) + c_2 r_2(p_g(t) - x_i(t))) \ldots\ldots\ldots (3)$$

2. The Proposed Algorithm

The proposed algorithm is a hybrid of the particle swarm optimization algorithm [1] and a random search technique with quadratic approximation formula [2] named Random Search Quadratic aproximation Particle Swarm Optimization (RQPSO) algorithm. In this proposed algorithm, a probability having certain value provided by the user has been fixed. In every iteration, if the uniformly generated random number r(0,1) is less than that value, then the velocity vector is generated by the equation(3) of the PSO algorithm otherwise it is generated by equation (4) of random search technique with quadratic approximation formula[2] as given below:

$$p = 0.5 * \frac{[f(b_1)(b_2^2 - b_3^2) + f(b_2)(b_3^2 - b_1^2) + f(b_3)(b_1^2 - b_2^2)]}{[f(b_1)(b_2 - b_3) + f(b_2)(b_3 - b_1) + f(b_3)(b_1 - b_2)]} \ldots\ldots (4),$$

where p gives the extremal point of the quadratic curve passing through the points b_1, b_2 and b_3.

The flow of the proposed algorithm is as under.

BEGIN:

Create and Initialize an n-dimensional swarm S

$\{x_i(t)$: =1to S$\}$ uniformly between 0 and 1.

Set constriction factor C=0.719 and $c_1 = c_2$ =2.0.

For i=1 to S,

For d=1 to n,

Assign some value to P between 0 and 1.

If r (0, 1) < P , then

generate velocity vector using equation (3) of PSO algorithm[1], else generate it using equation (4) random search with quadratic approximation [2].

Calculate particle position as $x_i(t+1) = x_i(t) + v_i(t+1)$

End- for-d;

Compute fitness of updated position; if needed, update historical information for P_i and P_g;

End-for-i;

Terminate if P_g meets problem requirements;

END

3. Performance Evaluating Criteria And Parameter Settings

AFE: Used average number of function evaluations for successful runs.

SR = Success Rate = (No of successful runs (NR) /Total runs)*100 = % age of successful runs to total runs.

AE = Average Error = $\frac{\sum_{n}(f_{\min} - f_{opt})}{NR}$, where NR is the number of runs, SD: Standard deviation of the error.

Parameters Used: max. no. of functions evaluations(MFE) taken=50000, no. of runs(NR)=20,constriction factor(C) =0.719, accelerating coefficients $c_1 = c_2 = 2$, error tolerance(ε)=0.001.

PC Configuration: Processor- Intel Dual Core, RAM-2 GB, Operating System-Window7, Software used- C++/Visual Studio and the random numbers are generated using inbuilt rand () function for the algorithm.

4. Test Functions

S.N.	Test Problem
1. [9]	$F(x) = Min[f_1(x), f_2(x)], f_1(x) = x_1^2 - x_2^2, f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \; s.t. \; x_1 \geq -5, x_2 \leq 10$
2. [9]	$F(x) = Min\ [f_1(x), f_2(x)], f_1(x) = \sqrt{x_1}, f_2(x) = \sqrt{x_2}, s.t. \quad x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \geq 5$
3. [9]	$F(x) = Min[f_1(x), f_2(x), f_3(x)], f_1(x) = 25 - (x_1^3 + x_1^2(1 + x_2 + x_3) + x_2^3 + x_3^3)/10$ $f_2(x) = 35 - (x_1^3 + 2x_2^3 + x_2^2(2 + x_1 + x_3) + x_3^3)/10, f_3(x) = 50 - (x_1^3 + x_2^3 + 3x_3^3 + x_3^2(2 + x_1 + x_2)/10$ $s.t. \quad x \geq 0, \; 12 - x_1^2 - x_2^2, \; -x_3^2 \geq 0$
4. [9]	$F(x) = Min[f_1(x), f_2(x), f_3(x)], f_1(x) = x_1 + 9x_2 + 10x_3 + x_4 + 3x_5,$ $f_2(x) = 9x_1 + 2x_2 + 2x_3 + 7x_4 + 4x_5, \quad f_3(x) = 4x_1 + 6x_2 + 7x_3 + 4x_4 + 8x_5$ s.t $3x_1 + 9x_2 + 9x_3 + 5x_4 + 3x_5 \leq 1039 \;, -4x_1 - x_2 + 3x_3 - 3x_4 - 2x_5 \leq 94$ $3x_1 - 9x_2 - 9x_3 - 4x_4 \leq 61, \; 5x_1 + 9x_2 + 10x_3 + x_4 - 2x_5 \leq 942,$ $3x_1 - 3x_2 + x_4 + 5x_5 \leq 420$, *all* $x_i \geq 0$.
5. [9]	$F(x) = Min\ [f_1(x), f_2(x), f_3(x)], f_1(x) = x_1^2 + (x_2 - 1)^2, f_2(x) = x_1^2 + (x_2 + 1)^2 +$ $f_3(x) = (x_1 - 1)^2 + x_2^2 + 2, s.t. \quad x_1 \geq 2 \; and \quad x_2 \leq 2$
6. [10]	$Min \quad f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2, f_2(x) = 9x_1 - (x_2 - 1)^2$ $x_1^2 + x_2^2 \leq 225 \;, x_1 - 3x_2 + 10 \leq 0$ $-20 \leq x_i \leq 20 \;, \forall_i = 1,2$
7.[10]	$Min(f_1, f_2) = [-(25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2), (x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2)]$ $s.t. \; x_1 + x_2 - 2 \geq 0 \qquad x_1 + x_2 - 6 \leq 0, \qquad x_1 - x_2 + 2 \geq 0, \; x_1 - 3x_2 - 2 \leq 0,$ $(x_3 - 3)^2 + x_4 - 4 \leq 0,$ $(x_5 - 3)^2 + x_6 - 4 \leq 0, \; 0 \leq x_1, x_2, x_6 \leq 10, \; 1 \leq x_3, x_5 \leq 5, \; 0 \leq x_4 \leq 6$

8. [11]	*Max* $[f_1(x), f_2(x)]$ $f_1(x) = -35x_1 - 40x_2 - 38x_3$, $f_2(x) = 20x_1 + 22x_2 + 25x_3$ $s.t.\ 2x_1 + 1.75x_2 + 2.10x_3 \le 2\times10^4$, $0.5x_1 + 0.6x_2 + 0.5x_3 \le 15\times10^3$, $35x_1 + 40x_2 + 38x_3 \le 45\times10^4$ $3000 \le x_1 \le 5000$, $4000 \le x_2 \le 7000$, $2000 \le x_3 \le 4000$.
9. [11]	*Min* $[f_1(x), f_2(x)]$ $f_1(x) = x_3 x_1^{-1} x_2^{-1} + 25\, x_2^{-1} x_1$, $f_2(x) = 0.1 x_1 x_2$. $s.t.$ $5x_1x_3^{-1} + 5x_2x_3^{-1} \le 1,\ x_1, x_2, x_3 \ge 0$
10. [11]	*Min* $[f_1(x), f_2(x)]$ $f_1(x) = 20x_1 - 10.0 x_2^2 x_5^{-1}$, $f_2(x) = -30\, x_1 x_3^{-1}$, $s.t.\ -10x_1^{-1}.x_3.x_4^{-1} + 5x_2 \le 1,\ x_1^{-0.2} x_5^{-1} x_2 x_4^{-0.5} \le 1, x_1, x_2, x_3, x_4, x_5 > 0$
11. [11]	$Min\ [f_1(x,t), f_2(x,t)]$, $f_1(x,t) = 0.775x_1^{-1} + 0.743x_1^{-1}t_1 + 28.3t_2$, $f_2(x,t) = 140.8x_1 + 140.8x_2$ $s.t.\ x_2^{-1}e^{-t_1} \le 1, x_1^{-1}e^{-t_2} \le 1\ \Big], x_1, x_2, t_1, t_2) \ge 0$
12. [11]	$Min\ f_1 = 2x_1^2 x_n + x_1^{-2}.x_2^2.x_2^{-2}t_1^{-1}t_2^{-1}e^{-t_1} - x_2, x_3x_4.e - 2(3t_1 - t_2)$, $f_2 = x_5x_1t_1t_2 + x_5^{-1}x_6e^{-t_1}$ $s.t.\ x_1^{-1}x_2^{-2}x_6^{-1}t_1 - x_2^{-1}x_3^{-1}t_2^{-2}.e^{2t_2} \le -1,\ (x_1, x_2, \ldots\ldots, x_6, t_1, t_2) > 0$
13. [11]	*Max* $[f_1(x), f_2(x), f_3(x), - - - -, f_6(x)]$, $f_1(x) = 4.75 + 2.27(x_1 - 0.3)$ $f_2(x) = 2.0 + 0.524(x_1 - 0.3) + 2.79(x_2 - 0.3) + 0.882(\omega_1 - 0.3) + 2.65(\omega_2 - 0.3)$ $f_3(x) = 5.1 + 0.177(x_1 - 0.3) + 0.978(x_2 - 0.3) + 0.216(\omega_1 - 0.3) + 0.768(\omega_2 - 0.3)$ $f_4(x) = 7.5 - 0.042(59/(1.09 - x_1^2) - 59)$, $_5(x) = -0.0018(532/(1.09 - x_2^2) - 532)$ $f_6(x) = -0.0025(450/(1.09 - x_3^2) - 450)$ $s.t.\ 1.0 + 0.0332(x_1 - 0.3) + 0.0186(x_2 - 0.3) + 3.34(x_3 - 0.3) + 0.0204(\omega_1 - 0.3)$ $+ 0.778(\omega_2 - 0.3) + 2.62(\omega_3 - 0.3) \ge 305$, $0.3 \le x_i \le 1.0, i = 1,2,3$ & $w_i = 0.39/(1.39 - x_i^2)$

5. Discussion on Results

For the solutions of all the constrained multi-objective optimization benchmark functions in this paper a C++ code has been developed and compiled in Microsoft visual C++ compiler and the data recorded as per the table given below.

Pb. N.	**Solution by Literature Optimum (f_1, f_2, .., f_m)**	**Solution using RQPSO Optimum (f_1, f_2, .., f_m)**	**Statistical data recorded in RQPSO**			
			SR	AFE	AE	SD
1.	(0.13,45.13) [9]	(0,0)	100	1113	0.000466	0.000318
2.	(2.13,0.67) [9]	(0,0)	100	42	0.000000	0.000000
3.	(-123.19, 33.71,-335.74) [9]	(18.6,24,35)	100	78	0.000000	0.000000
4.	(10,2,-7) [9]	(0,0,0)	100	726	0.000000	0.000000
5.	(1.96,1.36,3.16) [9]	(4,0,3)	100	326	0.000000	0.000000
6.	(58.6,32.8) [10]	(165,-621)	100	46	0.000000	0.000000
7.	(42.6,29.3) [10]	(-274,4)	100	7679	0.000767	0.000187
8.	(-341000,198000) [11]	(-34100, 233200)	100	12353	0.016781	0.048321
9.	(2.157,2.157) [11]	(2.15,0)	100	94	0.000000	0.000000

10.	(-63.117,-0.985) **[11]**	(-63.12, -1)	100	159	0.000000	0.000000
11.	(45.712,45.712) **[11]**	(45,45)	100	2312	0.000816	0.000118
12.	(0.543,0.543) [11]	(0.543,0.543)	100	227	0.000170	0.000292
13.	(6.,5,6,6.15, -1.623,-1.40, 2.979) **[11]**	(6,6.79,6,7.5,0,-1.28)	100	1050	0.000428	0.000238

6. Conclusions

In the present paper an algorithm which is a hybrid of the standard particle swarm optimization (SPSO) algorithm [1] and a random search technique with quadratic approximation formula [2] named Random Search Quadratic approximation Particle Swarm Optimization (RQPSO) algorithm has been proposed and tested on 13 constrained multi-objective test problems taken from the literature and are listed in section 4 of this paper. Experimental results are compared with the results obtained from the literature and it has been observed that the performance of the proposed algorithm improved in maximum cases based on the optimal function values, average number of function evaluations and rate of success.

References

[1] Clerc, M. and Kennedy, J., 'The particle swarm: explosion, stability and convergence in a multi-dimensional complex space', IEEE Transactions on Evolutionary Computation, 6, 58-73, 2002.

[2] Mohan C. and Shankar K., A control random search technique for global optimization using quadratic approximation, Asia pacific journal of operational research, 11, 93-101,1994.

[3] Wolpert, D. H. and Macready, W. G., 'No free lunch theorems for optimization', IEEE Transactions on Evolutionary Computations, 1, 67-82,1997.

[4] Mohan, C. and Deep, K., 'Optimization Techniques', New Age Publishers, New Delhi, 2009.

[5] J. Kennedy, R. Eberhart, Particle Swarm Optimization, Proceedings IEEE International Conference Neural Networks, vol. 4, pp. 1942-1948, 1995.

[6] Shi, Y. H., Eberhart, R. C.,(1998), A Modified Particle Swarm Optimizer, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998.

[7] Eberhart, R. C. and Shi, Y., 'Comparing inertia weights and constriction factors in particle swarm optimization', Congress on Evolutionary Computing, 1, 84-88, 2000.

[8] Clerc, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings, 1999 ICEC, Washington, DC, pp 1951-1957, 1999.

[9] Andrejs Zujevs, Janis Eiduks, New Decision Maker Model for Multi-obective optimization Interactive Methods,1- 8, 2006.

[10] JI Chunlin, A Revised Particle Swarm optimization Approach for Multi-objective and Multi-constraints optimization, 1-6,2005.
[11] Tyagi S., Optimization techniques and their use in solving a class of engineering design problems, Ph.D. Thesis, University of Roorkee, Roorkee(India),1988.