Particle Swarm Optimization for Solving Polynomial Equations

***Anuj Kumar, Puneet Mehta, Raghavendra V. Kulkarni**

Faculty of Engineering & Technology, M. S. Ramaiah University of Applied Sciences, Bangalore 560 054

*Contact Author e-mail: anuj.qtx@gmail.com

Abstract

The task of determining the roots of a polynomial equation has been prosed as an optimization problem. Particle swarm optimization algorithm has been applied to approximate the roots of polynomial equations of second and higher degrees. Two versions of PSO, Global-best and Local-best, have been implemented and a comparison of their performances is presented in terms of number of iterations to convergence, computation time and mean square error between expected and obtained roots.

Key Words: Particle Swarm Optimization, Quadratic Equation, Fitness Function, Swarm Intelligence

1. INTRODUCTION

There exists an analytic method to solve quadratic equations of the form $ax^2 + bx + c = 0$. However, as the degree of the polynomial equation increases, the equations for the roots get more and more complicated. Search-based optimization is a good approach to solve polynomial equations of high orders. There exist many deterministic, heuristic and metaheuristic approaches to solve an optimization problem. Swarm intelligence is a source of many algorithms that solve multidimensional optimization problems in a resource-efficient manner. Particle swarm optimization (PSO) algorithm is a shining example of such algorithms [1].

PSO is an iterative algorithm that contains a population of candidate solutions called particles. Particles move in an n-dimensional search space where n represents the number of optimization parameters (which equals the degree of polynomials in this study). Each individual particle learns its behavior influenced from its past behavior as well as the collective behavior of the swarm.

PSO has been applied for optimization in domains. It has been applied in many areas, such as telecommunications, control, design, data mining, power systems, combinatorial optimization and signal processing. Hundreds of publications describing PSO applications and algorithms are documented [2]. However, a majority of PSO applications are applied to solve unconstrained, single-objective optimization problems. PSO has been developed to solve constrained problems and problems with dynamically changing landscapes or multi-objective optimization problems as well. It has also been used to find multiple solutions to a single problem [1]. In this paper, the PSO algorithm has been applied to determine the roots of quadratic equations. Performance evaluation has been conducted on two variants of PSO, namely, global best PSO (GBEST) and local best PSO (LBEST). Further, the algorithm is extended to solving polynomials of degrees 3, 4, and 5. The effect of the increase in problem dimensionality on the algorithm performance has been investigated.

2. RELATED WORK

Finding the roots of a polynomial is an age-old problem. Traditional techniques like linear programming are promising as far as the quality of solution is concerned. However, they are not easily scalable and start to become computationally expensive as the complexity of the problem increases (degree of the polynomial, in this study) [3]. This led researchers to turn to bio-inspired metaheuristic algorithms for solving such problems. One such solution is developed [4] in which genetic algorithm and genetic programming algorithms have been used to solve a quadratic equations.

A comparison of various heuristic algorithms shows that PSO converges fast and is less resource intensive [5]. PSO is also known to have better immunity towards the tendency of getting trapped in local minima [3]. This motivates the authors apply the PSO algorithm to solve polynomial equations.

3. IMPLEMENTATION

A problem should be posed as an optimization problem in order to use the PSO algorithm. In this paper, the PSO is used to minimize the objective function f defined in (1), where a, b and c are the coefficients of the quadratic equation and x_1 and x_2 are the desired roots.

$$f = (ax_1^2 + bx_1 + c)^2 + (ax_2^2 + bx_2 + c)^2 \quad (1)$$

Naturally, the value of the objective function is zero if the PSO algorithm accurately finds the desired roots. The roots are initialized with random values at the start of the algorithm.

The PSO algorithm is implemented in the GNU Octave simulation tool. The search-space is in the range {-10 10}. A swarm of 30 particles is chosen and the impact on the solution is studied by gradually increasing the number of iterations. Acceleration coefficients for cognitive and social components c_1 and c_2 are both set to 2. Weights are initialized randomly and then are updated linearly per iteration. To avoid repeated roots and getting stuck at local minima, a penalty is imposed on the fitness value. The fitness function and penalty function often attain very high values because of their inherent nature. Accordingly, the initial value of the fitness is chosen to be very high (of the order of 10^7). Also, to ensure that the particles remain within the boundary of search space of {-10 10}, velocity clamping is imposed.

After each iteration, best fitness value is computed which is the minimum fitness among all the swarm particles participating in the computation and the index of such particle is taken to update the position of the fittest particle. After successful execution of all the iterations, the position of the particle whose fitness is minimum provides the roots of the solution.

3.1 Global Best Variant of PSO

For the GBest variant, the roots obtained for a second-degree polynomial for iterations 30, 300, 500, 1000, 3000 are shown in Table 1. The corresponding mean square error (MSE) and computation time are shown in Table 2.

Table 1. Expected and obtained roots (GBEST)

Expected Root 1	Expected Root 2	Obtained Root 1	Obtained Root 2
8	-9	8.0015	-8.9983
-6	-8	-6.0080	-7.9970
-9	2	-9.0001	2.0080
4	10	3.9999	10.0000
-2	2	-1.9983	1.9996

Another experiment has been conducted in which the number of iterations is fixed at 3000, and the number of particles is varied as 5, 10, 20, 30 and 50 to record the error and computation time. The results thus obtained are shown in Table 3. The corresponding MSE and computation time are shown in Table 4.

Table 2. Computation time (GBEST)

No. Of Iterations	MSE	Computation Time (S)
30	0.0023	0.2813
300	0.0085	0.5780
500	0.0080	1.1562
1000	0.0001	0.2969
3000	0.0017	0.8125

Table 3. Swarm size versus accuracy (GBEST)

Swarm Size	Expected Root 1	Expected Root 2	Obtained Root 1	Obtained Root 2
5	-2	-3	-1.99697	-2.9953
10	3	8	2.99842	7.99916
20	-2	-9	-2.00015	-9.001
30	9	-9	9.0002	-8.9995
50	9	2	9.00067	1.99963

Table 4. MSE and computation time (GBEST)

Swarm Size	MSE	Computation Time (S)
5	0.005592039	0.1875
10	0.001789413	0.625
20	0.001011187	0.76562
30	0.000538516	1.0625
50	0.000765376	0.96875

3.2 Local Best variant of PSO

Most of the parameters for LBest are kept the same as in GBest variant. An additional parameter called neighborhood size is used for partitioning the swarm into smaller groups. Neighborhood size has been set to 10 in this study. Iterations are varied as 30, 50, 70, 100, 300 and 500. The roots obtained are shown in Table 5. The corresponding MSE and computation time is shown in Table 6.

Like for the GBest variant, the effect of varying the swarm size is studied for LBest also (keeping the number of iterations fixed at 100). Table 7 shows the roots obtained as the swarm size is varied with values 5, 10, 20, 30 and 50. The corresponding MSE and computation time are shown in Table 8.

When swarm size is 5, the neighborhood size is selected as 5 and for rest of the case it is selected as 10.

Results in Table 8 also show that for a swarm size of 20, the error is almost zero while not increasing the computation time drastically.

Table 5. Expected and obtained roots (LBEST)

Expected Root 1	Expected Root 2	Obtained Root 1	Obtained Root 2
-7	-2	-6.99658	-2.0048
7	6	6.99728	5.9913
-2	1	-1.99996	0.99978
-8	3	-8	3
-3	7	-3	7
4	-1	4	-1

Table 6. MSE and computation time (LBEST)

No. of Iterations	MSE	Computation Time (S)
30	0.005893759	0.34375
50	0.009115284	0.53125
70	0.000223607	0.6875
100	0	1.0312
300	0	3.6875
500	0	5.1719

Table 7. Study of number of particles (LBEST)

Expected Root 1	Expected Root 2	Obtained Root 1	Obtained Root 2
-5	7	-5.0024	6.99845
-7	-8	5.76914	0.48954
9	1	9	1
-8	-9	-8	-9
3	5	3	5

Table 8. MSE and computation time (LBEST)

Swarm Size	MSE	Computation Time (S)
5	0.002857009	0.20312
10	15.33372837	0.35938
20	0	0.71875
30	0	1.1875
50	0	1.7188

4. GBEST VERSUS LBEST FOR SECOND DEGREE POLYNOMIAL

For a swarm size of 30, GBest takes up to 1000 iterations to reach an error of 0.0001 (Table 2) while LBest PSO achieves an error of 0.0002 at only 70 iterations as presented in Table 6. However, the time taken by LBest is considerably higher than GBest. While GBest takes 297 milliseconds for 1000 iterations, LBest takes 697 milliseconds for 70 iterations. 1000 iterations seem to be good enough number as when number of iterations increases to 3000, both error and time complexity increases (Table 2). Similarly, particle size 30 seems to be good enough in terms of

error and time complexity for GBest as after 30, error increases slightly (Table 4). For LBest, 70 iterations give quite good result and post 100, error practically becomes zero (Table 6).

Figures 1 and 2 show that the convergence of GBest is unpredictable, while LBest is stable as it maintains its error level after reaching a minimum. It happens because the diversity of GBest variant is less compared to diversity of LBest. Fig. 3 and 4 show that for smaller swarms, GBest does better in terms of predictability, while LBest results are quite volatile. However, as the swarm size increases, LBest shows much more stability, predictability and quality when compared to GBest.

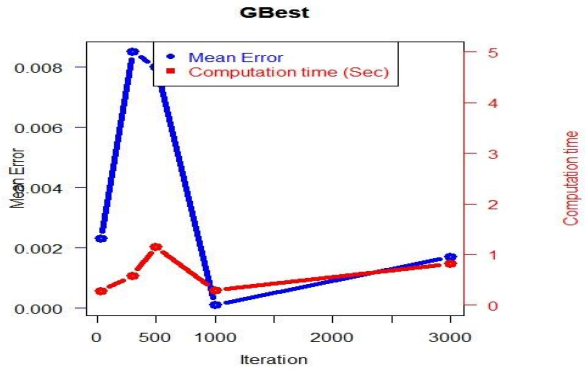

Fig. 1 Error, CompTime vs Iteration for Gbest

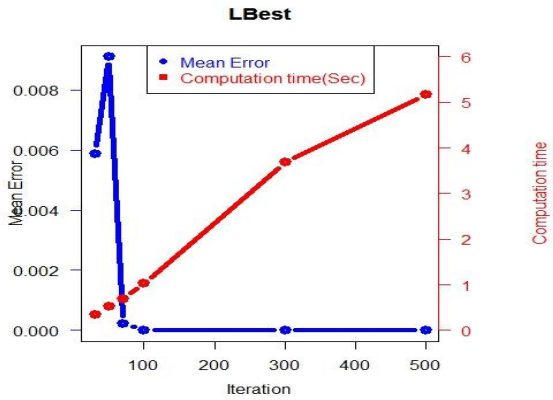

Fig. 2 Error, CompTime vs Iteration for Lbest

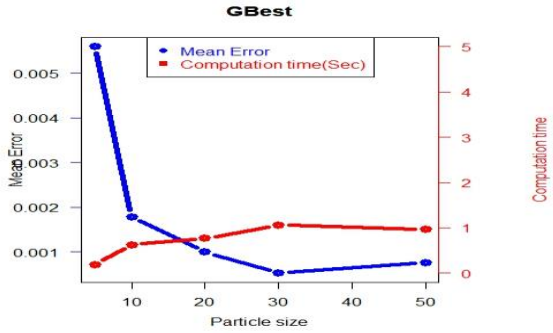

Fig. 3 Error, CompTime vs Particle size for GBest

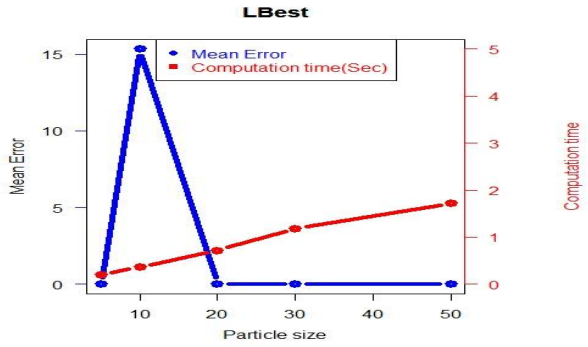

Fig. 4 Error, CompTime vs Particle size for LBest

Figures 2 and 4 show that the computation time increases remarkably in case of LBest due to manifold increase in internal computation and neighborhood size, whereas GBest remains constant (shown in Fig. 1 and Fig. 3). In future, further studies can be carried out for decreasing computation time. One way to attain that could be by reducing number of loops in source code.

5. INVESTIGATION ON SOLUTIONS OF HIGHER ORDER POLYNOMIALS

In this section, the algorithm is extended to problems of higher degree. Performance is compared for $n = 3$ and $n = 5$ for both GBest and LBest separately (n being the number of dimensions).

Results obtained with higher degree polynomial are not very encouraging. An increase in degree is found to make the solutions unstable, along with and increased MSE and computation time.

A comparison of GBest and LBest for $n = 3$ and Table 9, for $n = 5$ has been presented in Table 10.

Table 9. Performance comparison with $n = 3$

Type	Swarm Size	Iterations	MSE	Time (S)
Gbest	30	3000	0.00042	8.8437
Lbest	20	100	17.09430	1.4

Table 10. Performance comparison with $n = 5$

Type	Swarm Size	Iterations	MSE	Time (S)
Gbest	30	3000	12.649	29.14
Lbest	20	100	20.121	2.359

From Tables 9 and 10, it is evident that current PSO implementation is not suitable for higher order polynomials. Although GBest obtains better result than LBest in terms of MSE, its computation time is quite high. LBest, on the other hand, has lower computation time, but higher MSE.

6. CONCLUSION

Both versions of PSO have been successful in delivering satisfactory solutions to quadratic equations. MSE of 0.0017 is achieved for GBest and almost zero for LBest for n=2. This study can be further extended in multiple directions. The reduction of computation time and the accurate solution of higher degree polynomials are the two major directions.

REFERENCES

[1] Eberhart R., Kennedy J. (1995) A new optimizer using particle swarm theory, In *Micro Machine and Human Science, 1995. MHS'95. Proceedings of the Sixth International Symposium on*. pp. 39-43.

[2] Poli R. (2008) Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, doi:10.1155/2008/685175

[3] Kulkarni R.V., Venayagamoorthy G.K. (2011). Particle swarm optimization in wireless-sensor networks: A brief survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41*(2), pp. 262-267.

[4] Nayak T.D. (2012) Solution to Quadratic Equation Using Genetic Algorithm, *Proceedings of National Conference on AIRES.*

[5] Vale Z.A., Ramos C., Faria P., Soares J.P., Canizes B., Teixeira J., Khodr H.M. (2010) *Comparison between Deterministic and Meta-heuristic Methods Applied to Ancillary Services Dispatch,*. Porto: Knowledge Engineering and Decision-Support Research Center.