Evaluating the Performance of Particle Swarm Optimisation Algorithm using Some Single Objective Unconstrained Functions

[1]Shreya Sood, [2]Dr. Arvind Kalia, [3]Naveen Kumar
[1]M.Tech Student, [2]Professor, [3]Research Scholar
Department of Computer Science, Himachal Pradesh Shimla
Shimla, INDIA

Abstract Particle Swarm Optimisation is used for solving non-linear functions. It is simpler than any other evolutionary techniques as it is easy to implement by means of basic mathematical and logic operations. It is a population-based stochastic optimisation technique modelled on the social behaviour observed in flock of bird, school of fish, etc. Test functions hold some diverse properties which helps to find the efficiency, reliability and performance of optimisation algorithms. A good set of test functions used on any new optimisation, helps to validate its performance and compare it with other existing algorithms. These functions are well suited to evaluate the algorithm by comparing its efficiency with other algorithms and test its validity using different parameters. Only few of the test functions have been applied to the particle swarm optimisation algorithm earlier. More test functions applied on the particle swarm optimisation algorithm will help to see the algorithm in vibrant prospect and improvement in its performance. In this paper twenty two test functions have been applied on the particle swarm optimisation algorithm.

Keywords: Optimisation, Optimisation Algorithms, Optimum, Particle Swarm Optimisation, Search Space, Swarm, Test Functions Variants

I. Introduction

Optimisation is the process of finding a set of variables such that those variables can either maximize or minimize a scalar cost function *fx(n)*. The cost function is a value variable and can take multiple values since it depends on more than one decision variable. The n-dimensional decision vector *(x)* is nothing but the *n* decision variables over which decision maker has control. Trial and error technique was used to obtain quality decision variables but the disadvantage was that the method was long and time consuming. Now, optimisation techniques are used to find the decision variable. Optimisation problems have wide applications in the fields of engineering. Sometimes such problems can be very complex depending on the actual and practical nature of the objective function [1].

Optimisation problem consists of maximizing or minimizing a real function by systematically choosing input values from an allowed set and then computing the value of the function. It involves searching for best available value of some objective functions that are given a defined domain. Depending on the nature of variables, optimisation problems can further be divided into two categories i.e. continuous or discrete. Continuous optimisation function are those in which variables take continuous values. Few of the continuous optimization problems are PSO (Particle Swarm Optimisation), ACO (Ant Colony Optimisation), ABC (Artificial Bee Colony Optimisation) etc. Discrete Optimisation Problems are those in which variables take on values from the discrete set, often a subset of integers. Particle Swarm Optimisation (PSO) Algorithm, and Genetic Algorithm are some of the algorithms used to solve such problems. Unconstrained optimisation problems are the problems in which, there are no constraints on the variables [10].

II. Particle Swarm Optimisation Technique

2.1 Particle Swarm Optimisation

PSO is a population-based stochastic optimisation technique modelled on the social behaviour observed, like bird flocking, fish schooling, etc. In PSO, individual particles of a swarm represent potential solutions which move through the problem search space, seeking an optimal solution. The particles broadcast their current positions to the neighbouring particles. The position of each particle is adjusted according to its velocity and the difference between its current positions, the best position found by its neighbours and the best position it has found so far. As the model is iterated, the particles find values of position which are close to the solution. PSO optimizes the problem by iteratively improving the candidate solution with regard to a given measure of quality [19].

It is a meta-heuristics in which particle is considered as a candidate solution and the improvements are made in the candidate solution by moving the particles in the search space. The idea is that position and velocity are not only influenced by particle's best known position but it is also updated by better positions found by other particles in each iteration. Each particle has a position and a velocity. Its position is precisely the candidate solution it currently represents. Its velocity is a displacement vector in a search space which contributes towards a fruitful change in its position in the next iteration [2].

a. PSO Algorithm

The pseudo code of the procedure is as follows:

```
For each particle
   Initialize particle
End
Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in history
         Set current value as the new pBest
   End
   Choose the particle with the best fitness value of all the particles as the gBest
   For each particle
      Calculate particle velocity
      Update particle position
```

i. **Initialisation**: Initialize the population of particles with random position and velocities in *d* dimensional problem space. Give the upper and lower limits of each decision variable to confine the search space. The population of points are initialized with the velocity and position set to fall into the pre-specified or allowed range, satisfying the equality and inequality constraints.

- Evaluate the fitness of each particle in terms of pareto dominance.
- Record the non-dominated solutions found so far and save them in archive.
- Initialize the memory of each individual where the personal best position is stored.
- Choose the global best position from the archive.
- Increase the generation number.

ii. **Velocity Updating**: At each iteration, the velocities are updated.

iii. **Position Updating**: Between successive iterations the positions of all particles are updated accordingly.

- Check all the imposed constraints to ensure the feasibility of all the potential solutions. If any element of individual violates the inequality constraints then the position of the individual is fixed to maximum/minimum operating point.
- Update the archive that stores non dominated solution.

iv. **Memory Updating**: Update particle best position and global best position. Compare particles fitness evaluation with other particles. If current value is better than previous value, then set new value equal to the current value and current location as the location in d dimensional space. Compare fitness evaluation with the population's overall previous best. If current value is better than global best, then reset global position to the current particles array index and value [3].

b. Principles of PSO Algorithm

In case of PSO Algorithm, particles change their state using the principles: (i) to keep its inertia, (ii) to change condition according to its personal best solution, (iii) to change condition according to swarm's global best solution.

The position of each particle in the swarm is affected by the most optimal solution during its own movement (i.e. the personal best solution) and the position of the most optimal particle in its surrounding (i.e. the global best solution of the whole swarm) [4].

c. Analysis of Particle Swarm Optimisation

The basis of PSO Algorithm is based on the research done on the behaviour of the movement of birds. When considering the movement of birds, all birds either get scattered or they go together while searching for food [20]. As soon as a bird finds the food and its location, the same information will be transferred to all the other members of the flock so that all the members can come to the place where the food is. The same information can be applied to the PSO Algorithm as:

i. The bird swarm is considered as the solution swarm.
ii. As the birds move in search of food, the change in their location is considered as the development of the solution swarm.
iii. When the food is found, that location is taken as the most optimisation solution in case of PSO.
iv. Just as birds share the information among themselves about the location of the food, similarly individual particles share information among other particles in order to find the most optimist solution [4].

d. PSO Parameter Control

When PSO is applied to the optimisation problem, there are two steps need to be followed, (i) representation of the solution, (ii) the fitness function. For any function a standard procedure is used to find the optimum. The searching is done iteratively and stop criteria is reached when the maximum iteration number is reached or the minimum error condition is satisfied.
The parameters used to tune PSO are:

i. The number of particles: the typical range is 20-40. But for some special problems it can go to 100-200 particles.
ii. Dimension of the particles: it is determined by the problem to be optimised.
iii. Range of Particles: this parameter also depends upon the problem to be optimised but different ranges could be satisfied for different dimension of particles.
iv. Vmax: it shows the maximum change one particle can take during one iteration. The range is set as the Vmax for example the particle (x1, x2, x3), x1 belongs to [-10, 10] then Vmax=20.
v. Learning Factors: the learning factors c1 and c2 are equal to each other and set in the range of [0,4]
vi. The stop condition: depends on the maximum number of iterations the particle swarm optimisation executes and the minimum error requirement.
vii. Global version versus local version: the global version is faster but may converge to local optimum for some problems. On the other hand the local version is a little bit slower but not easy to be trapped into local optimum.
viii. Inertia weight: it can also be used with appropriate values [19].

III. Literature Review

Eberhart et al. described the optimisation of non-linear functions using particle swarm methodology. Three versions have been tested, the *'GBEST'* version in which every agent has information about the group's best evaluation and the other two variations of the 'LBEST' version. It has been found that original *GBEST* version performed best in terms of median number of iterations to convergence, while the *LBEST* version with a neighbourhood of two was most resistant to local minima [8].

Kennedy et al. previous particle swarm optimisation operated on continuous space where trajectories have been defined as changes in position on some number of dimensions. Kennedy discussed the trajectories in the probability that a co-ordinate would take on a zero or one value. In the binary space, a particle has been seen to move nearer and farther corners of the hypercube by flipping various number of bits and velocity of particles was described by number of bits changed per iteration. An algorithm was able to solve various problems very rapidly whereas new version of the algorithm was able to optimise any function discrete or continuous [9].

Shi et al. introduced a new parameter i.e. inertia weight into particle swarm optimisation. It was required because in previous versions the final solution was heavily dependent on initial population. Hence, it was more likely to exhibit local search. Inertia weight *'w'* was brought into the previous version to solve the problem. The inertia weight played the role of balancing the global and the local search. It has been found that the particle swarm optimisation with the inertia weight in the range [0.9, 1.2] on average would have a better performance i.e. it had a bigger chance to find the global optimum within a reasonable number of iterations [10].

Kaur et al. used test functions to improve the performance of PSO in order to improve its performance and have better results. The details about the characteristics of the test functions and features like search space, global optimum, optimal point, number of optimums have been presented. Five of the test functions have been used to present the results which show performance enhancement of PSO [11].

Barrera et al. as PSO is adopted in more types of application domains, it becomes more significant to have well established methodologies to assess its performance. For that purpose, several test problems have been proposed. Several state-of-the art test function generators that have been used for assessing the performance of PSO variants have been reviewed [13].

Jamil et al. reviewed and compiled a rich set of 175 benchmark functions for unconstrained optimisation problems with diverse properties in terms of modality, separability, and valley landscape. It has been expected that all these functions can be used for testing new optimisation algorithms so as to provide a more complete view about the performance of any algorithms of interest [12].

Ebbesen et al. aimed at reducing fuel consumption but increasing production costs in vehicles by doing electric hybridisation. The results showed that particle swarm optimisation performed significantly better than competing methods [16].

Li et al. proposed a particle swarm algorithm that used a set of interactive swarms to track multiple pedestrians in a crowd. A dynamic social interaction model has been proposed that enhanced the iteration among swarms and also improved the standard particle swarm optimisation [17].

Singh et al. addressed a robust schedule for a flexible job shop scheduling problem with random machine breakdown. Quantum particle swarm optimisation (QPSO) was proposed to generate the predictive schedules that could simultaneously optimise the make span and the robust measures. It came out to be effective in reducing make span in the event that uncertainly came up [18].

Since 1995, PSO has been used to solve various optimisation problems. A lot of variants of PSO have been developed till now. Different parameters like inertia weight and constriction coefficient have been introduced into basic PSO to improve the performance. PSO is used in lot of applications like scheduling problems, travelling salesman problem, etc. Test functions have been used on PSO to have the performance enhancement. Only few of the test functions have been applied till now.

IV. Test Functions

Test functions are vital to validate and compare optimisation algorithms. The test of reliability, efficiency and validation of optimisation algorithms is carried out by using test functions. These functions are well suited to evaluate the algorithm by comparing its efficiency with other algorithms and testing its validity using different parameters [5]. Multimodal test functions are unique as they show some regularity such as it may be symmetrical with respect to one axis, it may have uniform spacing among optima and also increase in the number of global optima with respect to increase in number of decision variables. When these regularities are studied for optimisation algorithms, the degree of difficulty decreases [6].

V. Evaluating Test functions for Assessing the Performance of Particle Swarm Optimisation

Problems in which the search space has several local optima and possibly more than one global optima is called Multimodal problems. Multimodal test functions have few features and shows some regularities. When these regularities are exploited then there is decrease in degree of difficulty [8]. Particle Swarm Optimisation Algorithm is used in a lot of domains so it becomes necessary to have well-established methodologies to assess its performance. Thus, several test problems have been proposed to assess the performance of PSO. Sometimes test problems have regularities, which are easily exploited by PSO that results in excellent performance [7].

VI. Implementation and Results

MATLAB (R2016a) version has been used to implement the PSO algorithm. There are twenty two multimodal test functions, which have been applied one by one. Execution time taken by the algorithm has been captured. *50* particles and *5* unknown variables have been used for the analysis. The maximum change in velocity has been taken to be twenty. *1* has been allotted as the inertia weight and *2* has been used as the values of learning factors.

It has been found that PSO is able to find best cost for all the test functions. As global minimum value are known for each test function, it has been seen that how much value each best cost deviated from global minimum value. For some functions like Easom Function, Goldstein Function, Helical Function etc. the best cost is found much before the 500 iterations are over. For some functions like Carromtable Function, Wood Function, Chichinadze Function, the number of iterations taken are not enough as it is not able to find the minimum value of the function in the given number of iterations. It has been also found that some of the test functions like Eggholder Function got caught in local best solution.

Table 1. Single Objective Unconstrained Test Functions Applied On PSO

Serial No.	**Function Name**	**Function Minimum Value**	**Total Time (sec)**	**Best Cost**
1	Carromtable Function	24.1568	18.293	23.899
2	Cross Function	0	19.163	4.8482e-05
3	Cross-in-tray Function	-2.062612	15.148	-2.0626
4	Cross Leg Table Function	-1	16.282	-0.8743
5	Crowned Cross Function	0.0001	15.822	0.0012
6	Easom Function	-1	13.960	-1
7	Egg Holder Function	-959.64	23.097	-2785.3922
8	Goldstein Price Function	3	16.665	3
9	Helical Valley Function	0	17.529	0
10	Holdertable Function	-19.2085	15.896	-19.2085
11	Levi Function	0	15.433	1.3498e-31
12	Matyas Function	0	16.494	1.0912e-125
13	Mccormick Function	0	15.723	-1.6594
14	Penholder Function	-0.96353	21.155	-0.96353
15	Powell Function	0	19.375	1.2273e-06
16	Schweffel Function	-837.9658	16.545	-837.9658
17	Six Hump Camel	-1.031628	15.793	-1.0316

	Function			
18	Test Tube Holder Function	-10.8723	16.302	-10.8723
19	Three Hump Camel Function	0	16.302	1.3884e-254
20	Trigonometric Function	0	17.694	0
21	Wood Function	0	16.240	0.11268
22	Chichinadze Function	-43.3159	17.287	-42.944

VII. Conclusion and Future Scope

Test functions play a vital role in testing and evaluating any algorithm. These test functions help to evaluate any new algorithm by comparing its efficiency with other algorithms and tests its validity using different parameters. The details about the features of these test functions like search space, global optimum, best cost, total time, etc. are presented here. These properties are used in differentiating the test functions from each other. Here, few of the test functions have been used to present the results which shows the performance enhancement of PSO. Some of the functions have been able to find global minima in five hundred iterations. Few functions needed more than five hundred iterations to find their best minimum value. Still the results seem promising. But for functions like Egg holder, the solution got caught in local optimum value. Increase in the number of iterations will not help finding the global optimum value. For finding the minimum value for such functions, some other variant of PSO can be used.

These test functions can present the algorithm in better light. Many other available test functions can be used for testing the performance of the algorithm in order to have much more results for comparison, results in improved and better particle swarm optimisation algorithm for solving the optimisation problems. These test functions can be applied to check and validate the performance of other optimisation problems as well.

REFERENCES

[1] Lee. W. N. and Park. J. B. 2005. Educational Simulator for Particle Swarm Optimisation and Economic Dispatch Applications. IEEE Transactions on Power Systems.

[2] Eberhart R. C. and Kennedy. J. 1995. A new optimiser using particle swarm theory. Sixth Intl. Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, NJ.

[3] Chung, C. J. and Reynolds. R. G. 1998. CAEP: An Evolution-Based Tool for Real-Valued Function Optimization Using Cultural Algorithms. International Journal on Artificial Intelligence Tool, vol. 7, no. 3, pp. 239-291.

[4] Engelbrecht, A. P and Wiley. J.2006. Fundamentals of Computational Swarm Intelligence. ISBN: 0470091916.

[5] Ronkkonen, J. Li. X. Kyrki. V. and Lampinen. J.2008. A generator for multimodal test functions with multiple global optima. Proceedings of the 7th International Conference on Simulated Evolution and Learning (SEAL '08), Berlin, Heidelberg, Springer-Verlag, pp. 239–248.

[6] Chen, W. N. Zhang. J. Chen. N Zhan. Z. Chung. H. S. Li. Y. and Shi. Y. H. Particle Swarm Optimization with an Aging Leader and Challengers. IBSN: 13231995.

[7] Shi, Y. and Eberhart. R. C. 1999. Empirical study of particle swarm optimization. Proc. IEEE Congress. Evol. Compute. pp. 1945–1950.

[8] Jamil, M. and Yang X. S. 2013. A literature survey of benchmark functions for global optimization problems, Int. Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, pp. 150–194 (2013). DOI: 10.1504/IJMMNO.2013.055204

[9] Kaur A, Kaur M. 2015. A Comprehensive Survey of Test Functions for Evaluating the Performance of Particle Swarm Optimisation Algorithm. International Journal of Hybrid Information Technology Vol.8, No. 5 (2015), pp. 97-104 http://dx.doi.org/10.14257/ijhit.2015.8.5.10

[10] Barrera J., Coello Coello C.A. (2011) Test Function Generators for Assessing the Performance of PSO Algorithms in Multimodal Optimization. In: Panigrahi B.K., Shi Y., Lim MH. (eds) Handbook of Swarm Intelligence. Adaptation, Learning, and Optimization, vol 8. Springer, Berlin, Heidelberg

[11] Eberhart, R. C. and Kennedy. J. 1995. A new optimiser using particle swarm theory", Sixth Intl. Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, NJ.

[12] Shi, Y. and Eberhart R. C. 1998. Parameter Selection in particle swarm optimisation, International Conference on Evolutionary Programming, Evolutionary Programming VII, pp. 591-600.

[13] Eberhart, R. C. and Shi, Y. 2000. Comparing inertia weights and Constriction Factors in Particle Swarm Optimisation, Proceedings of IEEE congress evolutionary computation, San Diego, CA, pp. 84-88.

[14] Ronkkonen J. Li, X. Kyrki, V. and Lampinen J. 2008 A generator for multimodal test functions with multiple global optima", Proceedings of the 7th International Conference on Simulated Evolution and Learning (SEAL '08), Berlin, Heidelberg, Springer-Verlag, , pp. 239–248.

[15] Abedi, M. Chiong, R. Noman N and Zhang, R. 2017. A hybrid particle swarm optimisation approach for energy-efficient single machine scheduling with cumulative deterioration and multiple maintenances. IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, 2017, pp. 1-8, doi: 10.1109/SSCI.2017.8285316

[16] Ye, Y. Li, J. Li. K. & Fu. H. 2018. Cross-docking truck scheduling with product unloading/loading constraints based on an improved particle swarm optimisation algorithm, International Journal of Production Research, 2018, 56:16, 5365-5385, DOI: 10.1080/00207543.2018.1464678

[17] Brownlee, J. Clever Algorithms: Nature-inspired Programming Recipes [Accessed on: December 02, 2018]. Available: http://www.cleveralgorithms.com/nature-inspired/swarm/pso.html.

[18] Singh, M. R. and Mahapatra, S. S. 2015. Robust Scheduling for flexible job breakdowns using a quantum behaved particle swarm optimisation, Int. J. Services and Operations Management, 2015, Vol. 20, No. 1, India.

[19] Types of Optimisation Problems. [Accessed on March 10, 2019]. Available: https://neos-guide.org/optimization-tree

[20] Hui. X. Particle Swarm Optimisation. [Accessed on: December 04, 2018]. Available: http://www.swarmintelligence.org/index.php.

[21] Particle Swarm Optimisation. [Accessed on: December 05, 2018]. Available: http://shodhganga.inflibnet.ac.in/bitstream/10603/28526/9/09_chapter4.pdf