Maximal Optimized Solution Using Swarm Intelligence

Hina Agarwal[1] and **Zaved Akhtar**[2]
Department of Computer Science and Engineering, Vishveshwarya Institute of Engineering and Technology, NH-91, Dadri, Gautam Budh Nagar 203 207, Uttar Pradesh India
[1]hina.aditi@gmail.com, [2]javed.gkp@rediffmail.com

Abstract - **Many optimization problems can be found in real life such as agricultural sciences, protein folding, computer vision, economics, pattern recognition etc. These problems need to be optimized either through 'maximize the earned profit, or to minimize the loss incurred. In literature' many optimization algorithms have been proposed to solve these problems to achieve optimal solution within optimal time. To solve optimization problems, many nature inspired optimization algorithms have been proposed like Evolutionary Programming, Swarm Based Optimization Algorithms and Differential Evolution. The performance of all algorithms is problem specific *i.e.* no algorithm is best in solving all optimization problems. So, there is always requirement to establish a new optimization algorithm which can solve some optimization problems within minimum time frame.**

Swarm Intelligence is a technique inspired by biological phenomena such as swarming and fish schooling. Particle Swarm Optimization (PSO) is an optimization technique which incorporates social behavior of swarm observed in bird of flock and school of fish. PSO is a population based optimization technique which is developed to solve various optimization problems.

The objective of any optimization algorithm is to obtain the global optimal solution. This can be achieved by minimizing the search time, improvement in convergence rate, redefinition of parameters according to the given problem statement. In the proposed algorithm named MPSO, we have observed the limitations associated with the basic PSO algorithm. The major problem is to handle stagnation at some point which happened due to premature convergence and may lead to get trapped in local optima. This generally happens in the case of complex multimodal problems. So to improve the performance of the algorithm, it is necessary to handle the convergence speed and also keep the track of the particles position to escape from local optima.

To meet these objectives, a variant of PSO is proposed named as MPSO. Finally auto tuning of random parameters has been done to make its performance independent of the problem.

Keywords: Optimization, Swarm Intelligence, Particle Swarm Optimization, Multimodal Problems

I. INTRODUCTION

OPTIMIZATION is a technique through which we can find the minimum or maximum optimal value of a given problem statement, technically say, as fitness function or objective function. This function helps us to understand that how much effort is required to get the desired result. Objective function needs input values known as variables. All feasible positions or value of these variables form search space. We process these feasible positions with respect to the objective function to get the optimal position (value).

Swarm Intelligence (SI) is an intelligent technique inspired by biological phenomena such as bird flocking, swarming and fish schooling. Particle Swarm Optimization (PSO) is an optimization technique which incorporates social behavior of swarm observed in bird of flock and school of fish. PSO is a population based optimization technique which is developed to solve various optimization problems.

PSO starts with randomly generated population of solutions and searches optimal value by uploading generations. In PSO, three features involved that impact on particles position like Inertia weight, the individual itself (best solution found by an individual so far) and social influences (best solution found by an individual in its neighbor). The main strength of PSO is its fast convergence rate that makes it faster as compared to other optimization algorithms. Many variants of PSO have been proposed till now. Still there is requirement to establish a new variant of PSO due to continued changes in the optimization problems. To apply PSO successfully, it is required to map the problem solution into the particle, which directly affects its feasibility and performance.

II. OBJECTIVE

An attempt has been initiated to propose a new version of PSO algorithm to remove some limitations observed during the literature survey by achieving the following significant things:

- *Improved convergence rate*: - This newly developed algorithm must be as fast as compared to other state-of-the-art algorithms. It should require minimum number of generations to capture global optimal solution.
- *Avoid Local Optima*: - It must be able to escape from local optima. This can be done by proper tuning of random parameters, inertia weight and learning factors.
- *Able to capture optimal solutions with diversity*: - It can be done by proper tuning of random parameters that helps to capture diverse optimal solutions for unimodal and multimodal problems.

- *Able to perform better in higher dimensions*: - It is required to capture global optimal solution in higher dimensions. This can be done by introducing a new algorithm which can capture global optimal solution in higher dimensions.

To meet these objectives, a variant of PSO is proposed named as MPSO. Finally, auto tuning of random parameters has been done to make its performance independent of the problem. We applied this algorithm on COCOMO model to optimize software cost estimation.

The challenge of identifying appropriate parameters for efficient performance of particle swarm optimization algorithms has been studied for many years. In this approach, the parameters of the PSO algorithm (inertia weight, cognitive acceleration coefficient and social acceleration coefficient) are tuned for improving the optimal solution in the search space. Parameter tuning strategy is needed because the basic version of the PSO algorithm was not giving efficient performance on the benchmarks functions. Proper and fine tuning of the parameters may result in faster convergence of the algorithm, and alleviation of the local minima. Initially, the values of the parameters of PSO algorithm were constant. However, experimental results proved that it is better to initially set the parameters to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined optimal solutions. A large parameter's value facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration (fine tuning of the current search area). A suitable value for the parameters usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution.

To develop this new algorithm, we have identified the problems associated with existing operators of PSO. We observe that the operators of PSO need to be redefined because the method to update the position of particle seems very awkward, as same factor is added to each particle without knowing whether the particle is moving in right direction or wrong direction. It affects both the personal best (pbest) position of the particle and global best (gbest) position of the swarm.

In this case, the performance of PSO can be improved if some additional factor is included in velocity vector which stores particle related information such that for how longer pbest positions of particles are not getting updated and gbest position of the swarm is not being improved. If particle is going in right direction, its velocity should be decreased else if it is going in wrong direction it should be discouraged. We have used two, way information sharing technique among particles to explore the whole search space to find multiple optimal regions simultaneously with different weight combinations. This helps PSO to converge towards the best solution efficiently.

We have checked performance of this algorithm on a real life optimization problem. This real life problem is related to software effort estimation and its main objective is to minimize the difference between actual effort and predicted effort.

A common approach to estimate the software cost (effort) with COCOMO model is based on the following equation:

$$\text{Effort} = a \times \text{size}^b$$

where a and b are constants and their values are determined using regression analysis applied to historical data. The values of a and b differ for different types of projects (organic, embedded and semidetached). With the help of MPSO, we have optimized the value of parameters a and b so that difference between predicted effort and actual effort has been minimized.

Parameter Tuning in PSO Algorithm: The challenge of identifying appropriate parameters for efficient performance of particle swarm optimization algorithms has been studied for many years. In this approach, the parameters of the PSO algorithm (inertia weight, cognitive acceleration coefficient and social acceleration coefficient) are tuned for improving the optimal solution in the search space. Parameter tuning strategy is needed because the basic version of the PSO algorithm was not giving efficient performance on the benchmarks functions. Proper and fine tuning of the parameters may result in faster convergence of the algorithm, and alleviation of the local minima. Initially, the values of the parameters of PSO algorithm were constant. However, experimental results proved that it is better to initially set the parameters to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined optimal solutions. A large parameter's value facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration (fine tuning of the current search area). A suitable value for the parameters usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution.

III. MODIFIED PARTICLE SWARM OPTIMIZATION (MPSO)

The objective of any optimization algorithm is to obtain the global optimal solution. This can be achieved by minimizing the search time, improvement in convergence rate, redefinition of parameters according to the given problem statement. In the proposed algorithm named MPSO, we have observed the limitations associated with the basic PSO algorithm. The major problem is to handle stagnation at some point which happened due to premature convergence and may lead to get trapped in local optima. This generally happens in the case of complex multimodal problems. So to improve the performance of the

algorithm, it is necessary to handle the convergence speed and also keep the track of the particles position to escape from local optima.

TABLE 1 -- MPSO TERMINOLOGY.

Term	Description
c1min, c2min,	Minimum value of c1and c2
c1max, c2max	Maximum value of c1and c2
gmax	Maximum number of generation
gcurrent	Current generation
wmin, wmax	Minimum and maximum value of inertia weight

To achieve the desired objective, we have tuned the parameters such as learning factors and inertia weight of PSOM. Learning factors *i.e.* cognitive acceleration coefficient 'c1' and social acceleration coefficient 'c2' are tuned to avoid premature convergence. The formula for c1 and c2 is given in equation (1) and (2). Inertia weight (w) is kept dynamic for making balance between exploration and exploitation techniques of particles. Changes that have been made in the parameters c1 and c2 of basic PSO algorithm are as follows:

$$c1 = c1min + (c2max - c1min) \times (gmax-gcurrent)/gmax \quad (1)$$

$$c2 = c1min + (c1max - c2min) \times (gmax-gcurrent)/gmax \quad (2)$$

The inertia weight "w" can either be a constant or a dynamically changed value. Essentially, this parameter controls the exploration of the search space, so that a high value allows particles to move with large velocities in order to find the global optimum neighborhood in a fast way and a low value can narrow the particles search region. Research has been taken into account in order to find a suitable set of w. The value of w is decreased from a higher initial value (typically 0.9) to a lower value (typically 0.4) linearly as the iteration number increases or the value of w can be chosen randomly from values distributed from 0 to 0.5. Using this strategy, particles can search the design domain more flexibly and widely. For our work, we have kept the inertia weight dynamic. The formula of inertia weight is as given below

$$w = wmin + (wmax-wmin) \times (gmax-gcurrent)/gmax \quad (3)$$

We have added a new parameter "e" in velocity (v) to improve the position of each particle in the iterative generations. Here "e" is an environmental constraint that works on inertia weight to decide the velocity of particles in a particular generation. This parameter manages the value of inertia weight on the basis of the current position of the particles in the iterative generations. If particles confine their search around local minima then parameter "e" helps in managing the value of inertia weight and velocity accordingly to explore the new search regions to avoid local minima.

While on the other side information collected from "e" manage inertia weight that helps to restrict maximum velocity of the particles that seems to go beyond the optimum region, where particles found themselves very near to the optimum and start exploitation around the optimal region. The modified formula for velocity vector, new position of particles and value of "e" is shown in equations (4), (5), (6) and (7) respectively as follows:

$$v_{id}^{i+1} = w * v_{id}^{i} + c_1 r_1 (P_{id}^{i} - x_{id}^{i}) + c_2 r_2 (G_{best}^{i} - x_{id}^{i}) + e \quad (4)$$

$$x_{id}^{i+1} = x_{id}^{i} + v_{id}^{i+1} \quad (5)$$

e = w + p/c1 >= Max of w then e = rand [(w - p/c1), Max of w] for exploration (6)

e = w - p/c2 <= Min of w then e = rand [(w + p/c1), Min of w] for exploitation (7)

Here *p* is a parameter that tracks the current position of particles in the iterative generations. To calculate the value of "p", fitness values of pbesti and new position of all particles are compared.

Initially the population of swarm (all particles) is assigned randomly the velocity and respective positions in n-dimensional search space. To explore the entire search space particles share the information they have to get the optimum position. Every particle has its best position in respect to the optimal position known as pbest and it is assigned to each particle as pbesti.

Among all particles positions, they select best particle of the swarm with best position named gbest. In upcoming generations, particles adjust their position by comparing their respective pbesti with the fitness of new position obtained by them and gbest position of swarm. If the new position is obtained by particles is better as compared to the present pbesti then particles can obtain other better positions near newly obtained position by exploiting the whole region.

This can be possible only by adjusting inertia weight accordingly that decrease velocity to dig out into the good regions to obtain other better positions. Else particles start exploring the whole search space for getting new good regions where possibility of getting the optimal position should be high. This can be possible only through managing the inertia weight which helps to increase the velocity accordingly. Also gbest

value of swarm is changed accordingly. If it is found that after some intermediate generations, particles pbesti and swarm gbest is not improved then parameter "e" helps to explore other unexplored good regions with high possibility of getting good position value. In each generation, particles are forced to select good regions only. The process is applied repeatedly to obtain the optimal solution. The algorithm terminates either the maximum number of generations is reached or optimal position is obtained which comes first.

IV. ALGORITHM FOR MPSO

Step 1: Initialize the iterative number for generation i = 0. Initialize the population size as POPsize. Maximum generation size for termination of the algorithm is Maxgen. Generate initial population of the particles of size POPsize.

Step 2: Calculate the fitness of each particle of the initialized population and for the first generation let Pid be the initialized particles. Select the particle with best fitness value among all particles as Gbest (gbest).

Step 3: Give birth to the particles (obtaining new positions) for the next generation using equation (4) and (5) as Xid. Calculate the fitness of the particles and compare with the best fitness value P_{id}^{i} *i.e.* pbesti in the history. Select the particles with better fitness value and take them as new P_{id}^{i} . Select the particle with best fitness value among all particles as G_{best}^{i} . Check if number of generation "i" is equal to maximum number of generation Maxgen or optimal position is obtained which comes first then go to Step 4 or else i = i + 1 go to Step 2.

Step 4: The global best position of the swarm is the optimal solution *i.e.* Gbest.

V. CONCLUSION

MPSO has been developed which is a variant of basic PSO algorithm. It is developed to remove the shortcomings of PSO. In MPSO, we have fine-tuned the value of parameters such as inertia weight (w), acceleration coefficient (learning factors) *i.e.* c1 and c2. Also we have added a new parameter 'e' in velocity vector to manage the velocity of particles.

MPSO performs well with both unimodal and multimodal problems and provides better results than other algorithms. Unlike PSO and other variants of PSO, MPSO is very fast in unimodal benchmark functions. For multimodal benchmark function, this method is very good and captures global optimal solutions with improved convergence rate.

VI. REFERENCES

[1]. Laskari, Elena C., Konstantinos E. Parsopoulos and Michael N. Vrahatis. "Particle Swarm Optimization For Integer Programming." *Proc. IEEE World Congress Computational Intelligence*, Volume 2, 2002.

[2]. A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence,* John Wiley & Sons, 2006.

[3]. C. Blum and X. Li, *Swarm Intelligence in Optimization*. Springer, 2008.

[4]. S. Chetty and A. O. Adewumi, "Comparison Study of Swarm Intelligence Techniques for The Annual Crop Planning Problem," *IEEE Transactions Evolutionary Computation,* Volume 18, Number 2, 2014, pp. 258–268.

[5]. Narinder Singh and S.B. Singh, "Personal Best Position Particle Swarm Optimization," *Journal of Applied Computer Science and Mathematics*, Volume 12, Number 6, 2012.

[6]. Yao, Jingzheng and Duanfeng Han, "Improved barebones Particle Swarm Optimization with Neighborhood Search and Its Application on Ship Design."*Mathematical Problems in Engineering*, 2013.

[7]. W. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.S.-H. Chung, Y. Li and Y.-H. Shi, " Particle Swarm Optimization with an Aging Leader and Challengers," *IEEE Trans. Comput.,* Volume 17, Number 2, April 2013, pp. 241-258.

Hina Agarwal received B.Tech degree in Computer Science and Engineering from Uttar Pradesh Technical University in 2011. Now She is doing M.Tech from VGI G.B. Nagar, AKTU Lucknow. She is doing her Dissertation in Swarm Intelligence. She has 3 years experience as Lecturer (CSE Department) in Maharja Agarsain Institute of Technology, Ghaziabad. Her area of interest is Software Engineering.

Zaved Akhtar received M.Tech in Computer Science and Engineering. He is pursuing his PhD. from Dr. A.P.J. Abdul Kalam Technical University, Lucknow. He is working as an Associate Professor (CSE Department) in Vishveshwaraya Group of Institutions, Dadri, G.B. Nagar and he has about 13 years experience. His area of interest is Software Engineering, Data mining and Algorithms. Published over 15 research papers.