An Adaptive Approach for Modifying Inertia Weight using Particle Swarm Optimisation

Rajesh Ojha[1] and Madhabananda Das[2]

[1] Computer Science and Engineering, Biju Patnaik University Of Technology, Gandhi Institute Of Technology And Management
Bhubaneswar, Odisha 752054 , India

[2] Computer Science And Engineering, KIIT University, School Of Computer Engineering
Bhubaneswar, Odisha 751024, India

Abstract

Particle Swarm Optimization is a comparatively recent heuristic technique, introduced by Kenedy and Eberthart in 1995. It is very similar to Genetic Algorithm and it is also a population based method. Many developments have been carried out to the standard Particle Swarm Optimization algorithm. Due to the less computational effort PSOs are very widely being used as an optimization tool. The GA is discrete in nature where as the PSO is inherently continuous. As many variations of the PSO are being popular the motto of this paper is to make analysis of the existing modified versions of standard Particle Swarm Optimization algorithm and to suggest a new variant of PSO. This paper is divided into two parts. The first part is doing analysis of the time variant inertia weight methods suggested by different researchers. In the second part a new method of updating the inertia weight has been proposed. It is also implemented using Mat Lab and proven as worthy than the existing weight updating methods

Keywords: *Particle Swarm Optimization, Swarm Intelligence, Meta Heuristic Algorithm, Inertia Weight.*

1. Introduction

Scientists have always used nature as a source of inspiration. Several scientists have created computer simulations of various interpretations of movement of organisms in a bird flock or in a fish school. Particularly Ratnaweera et al.[1], have presented simulation of bird flocks. Heppner being a zoologist was interested in discovering rules which allow large flocks to move synchronously often suddenly changing direction scattering and regrouping, which is an unpredictable groups dynamics of bird social behavior and based upon manipulation of inter individual distances i.e. synchrony of flock behavior was thought to be a function of bird efforts to keep an optimal distance between themselves and their neighbours. To achieve this they use the method of social sharing of information among members of a same group which has been fundamental to the PSO development i.e. proper use of group intelligence.

The concept of Swarm Intelligence (SI) was first used in the field of cellular robotic systems [9]. In this context, simple agents occupied one- or two-dimensional grid environments and self organized through closest neighbor interactions. In 1999, (Bonabeau et al.) noted that the term "swarm intelligence" extends that definition. Using the expression "swarm intelligence" to describe only this work seems unnecessarily restrictive: "that is why we extend its definition to include any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of insect colonies and other animal societies".

Swarm Intelligence could be defined as any attempt to design algorithms or distributed problem-solving devices whose behavior emerges from the social interaction between local neighbors. The word swarm loosely describes a collection of interactive individuals. The classical example of a swarm is bees swarming around their hive; nevertheless the metaphor can easily be extended to any other system with a similar architecture. As ant colonies can be thought of as a swarm whose individuals are ants, so can a flock of birds. The concept of swarm can be extended to an even more general one: that of any type of collective behavior. Thus, a swarm might occur in high-dimensional cognitive spaces, where collision is no longer a concern and could simply mean agreement. Swarm intelligence is to simulation of social interaction between individuals what evolutionary algorithms are to the simulation of evolution. In Swarm Intelligence, metaphors from successful behaviors of

animal or human societies are applied to problem solving. As their cousins, the goal is not to faithfully mimic the phenomena themselves but to use some of their aspects in practical applications[15][16].

Particle Swarm Optimization (PSO) is paradigm for designing Meta heuristic algorithms for optimization problems. A Meta heuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems [3]. It is an algorithm which, in order to escape from local optima, drive some basic heuristic: either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one. The Meta heuristic part permits the low-level heuristic to obtain solutions better than those it could have achieved alone, even if iterated. Usually, the controlling mechanism is achieved either by constraining or by randomizing the set of local neighbor solutions to consider in local search [6].

In other words, a Meta heuristic is a general-purpose algorithmic framework that can be applied to different optimization problems with relatively few modifications [1]. Examples of Meta heuristics include Particle Swarm Optimization, simulated annealing and ant colony optimization.

Particle Swarm Optimization (PSO) was invented by (Kennedy et al.) in the 1990s while attempting to simulate the graceful motion of swarms of birds as part of a socio cognitive study investigating the notation of "collective intelligence "in biological population [17]. PSO is inspired by the ability of flocks of birds, schools of fish and herds of animals to adapt to their environment, find rich sources of food and avoid predators by implementing an " information sharing " approach, hence, developing an evolutionary advantage[1].

In PSO a set of randomly generated solutions known as swarms propagates on the design space towards the optimal solution over a number of iterations known as moves based on large amount of information about the design space that is gathered and shared by all members of the swarm. Particle swarm optimization received its inspiration from bird flocking, fish schooling and swarming theory, which is based on group intelligence and the capability of storing information in the form of local memory. Besides swarm theory, PSOs have roots in other Artificial Life algorithms such as evolutionary strategies. Particle swarm optimization shares many similarities with evolutionary computation techniques in general and Genetics Algorithms (GAs) in particular[7][8][13]

2. Literature Review and Background Work

There are many works have been carried out in the field of Particle Swarm Optimization and this chapter focuses on the development and variations in the standard Particle Swarm algorithm as well as also compares the PSO with other heuristic techniques.

A strong comparison has been made between the Particle Swarm Optimization method and Genetic Algorithm by Hassan et al. [13] by taking some standard numerical optimization problems. This paper suggests the value of the self confidence factor from 1.5 to 2, value of the swarm confidence factor from 2 to 2.5 and the value of the weight from 0.4 to 1.4 which is also referred by other papers. This paper carried out two tests with eight standard bench mark functions to evaluate the performance of PSO and GA both. The first test is the effectiveness test, which measures the quality of the solutions found by the heuristic algorithm with respect to known solutions for the test problems. This test investigates whether the quality of the solutions obtained is greater than 99%. The second test is the efficiency test, which investigates whether the computational effort of PSO is less than that of the GA for the sample problem set using the same convergence criteria.

A study is made by Karl O. Jones on the performance of both Genetic Algorithms and Particle Swarm Optimization, demonstrating their ability to generate fermentation process feed profile based on a number of objective functions [2][7]. Fermentation process is associated with the formation of yeast, pharmaceuticals and chemicals etc. The problem to be optimized here was to produce maximum biomass in the shortest time using the minimum amount of raw material out of which the organic carbon source is the most expensive component. This paper uses these two recent techniques, applied with the same objective functions and found that the PSO performs was much better than the GA in finding the feed profile. In this experiment he used a swarm size of 200, the weight as 0.95 at the start of PSO iteration and reduced to 0.4 at final iteration.

Hu et al. suggested some modifications to the standard Particle Swarm Optimization algorithm to automatically trap various changes in a dynamic system and named it as Adaptive Particle Swarm Optimization [16]. According to them a situation may be there when the gBest becomes constant for a number of iterations and at that time the parameters should be reset to drive the gBest out of that. In this method the PSO finds the optimum first and records the number of iterations needed to reach the required error level, then dynamic changes are applied and the PSO continues to find out the new optima and records the number of iterations needed to reach the required error level.

The concept of Adaptive Particle Swarm Optimization was implemented for static and dynamic economic load dispatch by Panigrahi et al. by introducing a time variant weight[2]. This paper suggests that in the adaptive PSO, the particle position is adjusted such that the highly fitted particle (best particle) moves slowly when compared to the lowly fitted particle. This can be achieved by selecting different W values for each particle according to their rank, between w_{min} and w_{max} as in the following form:

$$w_i = w_{\min} + \frac{(w_{\max} - w_{\min}) * Rank_i}{Totalpopul\ ation} \qquad (1)$$

The best particle takes the first rank, and the inertia weight for that particle is set to the minimum value while that for the lowest fitted particle takes the maximum inertia weight, which makes that particle move with a high velocity.

Falco et al. [4] implemented Particle Swarm optimization technique to solve classification type problems ,which was being solved using Artificial Neural Networks or K-mean method earlier. When in a multi-dimensional space a class prototype is represented by a centroid, the classification can be seen as the problem of finding the optimal positions of all the class centroids, i.e. determining any centroid is determining its optimal coordinates. PSO is known from literature to be usually very effective in facing such problems.

Where w_{max} is the maximum weight, w_{min} is the minimum weight t and T_{max} are the current iteration and maximum number of iterations. They used the PSO to face a set of 13 databases well known in literature taken from UCI database repository, and its results are compared against those provided by nine classical classification techniques. The experimental set up was carried out with no. of particles n =50, T_{max} = 1000, V_{max} = 0.05, V_{min} = -0.05, C1 = 2.0, C2 =2.0, w_{max} = 0.9 and w_{min} =0.4.

While optimizing the PI controller gains the inertia weight coefficient in velocity updating is employed to manipulate the impact of the previous history of velocities on the current velocity [12].Therefore, ω(*t*) resolves the tradeoff between the global and local exploration ability of the swarm. A large inertia coefficient encourages global exploration while small one promotes local exploration. Experimental results suggest that it is preferable to initialize it to a large value, giving priority to global exploration of search space, and gradually decreasing as to obtain refined solution. This paper suggests the initial value of inertia weight coefficient as 1 and to go on decreasing the inertia weight to a small magnitude nearly zero at first iteration.

Particle Swarm Optimization Time Varying Inertia Weight was proposed by Obaidy et al [11]. Like the references [4] and [5] they also suggest to start with a higher inertia weight and to go on decreasing iteration wise but the new weight is calculated as:

$$w_{iter} = (weight - 0.4) \times \frac{(MAXITER - iter)}{MAXITER} + 0.4 \qquad (2)$$

Where w_{iter} =weight for iteration number

The weight of iter is the constant weight (value suggested is 0.9)

MAXITER= total number of iterations

iter= iteration number.

They have also implemented time variant acceleration coefficients (C_1 & C_2) whose values generally remains constant in standard PSOs and their values generally considered as between 0.5 and 2. So their calculated as :

$$C_1 = (C_1 \min - C_1 \max) \frac{iter}{MAXITER} + C_1 \min \qquad (3)$$

$$C_2 = (C_2 \min - C_2 \max) \frac{iter}{MAXITER} + C_2 \min \qquad (4)$$

Where C_{1min} and C_{2min} are taken as constant (0.05) C_{1max} and C_{2max} are also taken as constant.

A comparison of some PSO variants on a set of common benchmark problems, which is based on a detailed empirical performance analysis from which one can identify algorithmic components that provide a positive effect on some performance aspect[9]. They have designed and evaluate a new composite algorithm, called Franken-stein's PSO, which integrates the algorithmic components that were identified during the first phase. The final evaluation consists in comparing Frankenstein's PSO with the variants from which its components were taken. Dorigo suggests the weight updation as:

$$w^t = \frac{wt_{\max} - t}{wt_{\max}} (w_{\max} - w_{\min}) + w_{\min} \qquad (5)$$

where Wt_{max} marks the time at which $W^t = W_{min}$ and W_{min} are the maximum and minimum weight.

This is a decreasing variant case but the constricted PSO can be considered as a special case of this variant but with a constant inertia weight. If the value of W_{max} and W_{min} will be interchanged it will be an increasing variant PSO. This paper implemented changes to different parameters of the standard PSO , carried out tests using some bench mark functions to ensure their convergence and then derived a composite PSO which is a collaboration of these techniques.

Experimental results suggest that it is better to set the inertia to a large initial value, in order to promote global exploration of the search space, and gradually decrease it to obtain refined solutions[5][7][8].

The weight update is suggested by some researchers[4] as:

$$w_i = w_{\min} + \frac{(w_{\max} - w_{\min}) * Rank_i}{Totalpopul\ ation} \quad (6)$$

Where the best particle is given the 1st rank and the W_{max}, W_{min} are the maximum and minimum possible value for inertia weight.

The weight update is suggested by Stefan and Martin [7] as:

$$w_k = \frac{(w_{\max} - w_{\min}) \cdot k}{h-1} + w_{\min} \quad (7)$$

$$w_k = \frac{(w_{\min} - w_{\max}) \cdot k}{h-1} + w_{\max} \quad (8)$$

The values W_k are determined using either (8) or (9), with level k Є [0,h-1] (the root is on level 0) and the resulting W_k Є [.W_{min} ; W_{max}] .The algorithm using (8) has the values decreasing, from bottom to top of the hierarchy, with the root particle using W_{min}. Whereas the algorithm using (9) inverts the assignment with the root particle using W_{max} .

The weight update is suggested by Falco et al.[4] as :

$$w(t) = w_{\max} - \left((w_{\max} - w_{\min}) \frac{t}{T_{\max}} \right) \quad (9)$$

Where W_{max} is the maximum weight, W_{min} is the minimum weight t and T_{max} are the current iteration and maximum number of iterations.

3. Proposed Work

3.1 Proposed Approach

The role of the inertia weight w is considered crucial for PSO's convergence behavior because the inertia weight is employed to control the impact of the history of velocities on the current velocity. In this way, the parameter w regulates the tradeoff between the global (wide-ranging) and the local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine tuning the current search area. A suitable value for the inertia weight w provides balance between the global and local exploration ability of the swarm, resulting in better convergence rates.

As the value of inertia weight w is playing a very important role in the calculation of the velocity, a time varying inertia weight can help to come out quickly from a region where the velocity becomes stagnant.

The inertia weight coefficient in velocity updating is employed to manipulate the impact of the previous history of velocities on the current velocity. Therefore, a varying inertia weight w resolves the tradeoff between the global and local exploration ability of the swarm faster.

So this paper suggests a new method of updating the inertia weight. In some papers it is preferred to start with a larger inertia weight, which enhances the global exploration rather than choosing a small inertia weight which emphasizes on local exploration.

$$w(t) = w_{\max} - \Delta w \quad (10)$$

In this method the fraction of weight Δw is subtracted from the maximum weight W_{max} instead of adding to the lower bound i.e. the minimum inertia weight W_{min}. So this method is approaching from high weight to low weight iteration wise gradually.

$$\Delta w = \left((w_{\max} - w_{\min}) \frac{T_{\max} - t}{T_{\max}} \right) \quad (11)$$

Where W_{max} is the maximum weight, W_{min} is the minimum weight t and T_{max} are the current iteration and maximum number of iterations.

where $0 \le t < T_{max}$

3.2 Proposed Algorithm

Step-1 Do Parameter settings and initialize a n-dimensional PSO
Step-2 $Y_i \leftarrow X_i$
Step-3 Z ←min (Y_1, Y2……Y_i….Y_s)
Step-4 t ←1
Step-5 While (t<= no. of iterations) do
Step-6 Update inertia weight w
Step-7 For i ←1 to S , consider each particle i
Step-8 if $f(X_i) < f(Y_i)$
Step-9 then $Y_i \leftarrow X_i$
Step-10 if $f(Y_i) < f(Z)$
Step-11 then $Z \leftarrow Y_i$
Step-12 End For
Step-13Update the velocity of particles
Step-14 Update the position of particles
Step-15 t ←t+1
Step-16 End While
Step-17 Exit

3.3 Parameter Setting

SL.NO	Name of the parameter	Value of the parameter
1	Self Confidence(c_1)	1.8
2	Swarm confidence (C_2)	2.3
3	Minimum inertia weight (w_{min})	0.4
4	Maximum inertia weight(w_{max})	1.5
5	Size of the population (S)	100
6	No.of Generations (Maximum no.of iterations)	User specified

Table 1 Representing the values of the parameters used in the PSO

4. Results

The six benchmark functions are tested with the modified time variant PSO. The range of the bench mark functions, dimensionality etc. The simulation result of the optimization of the objective functions are represented in the form of two dimensional graph, where the X-axis represents the No. of iterations carried out and the Y-axis represents the Fitness value with respect to the iteration number. Each of the bench mark function is evaluated four times and the result is plotted for 1000 iterations.

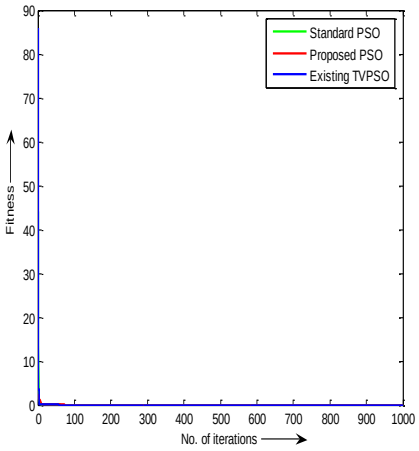

Figure 1: The Sphere Function after 1000 iterations

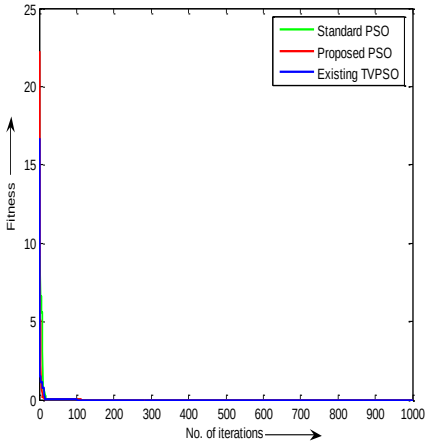

Figure 2: The Rosenbrock Function after 1000 iterations

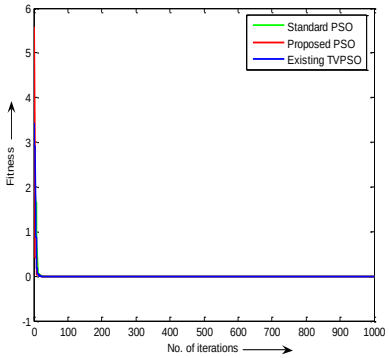

Figure 3: The Ackley Function after 1000 iterations

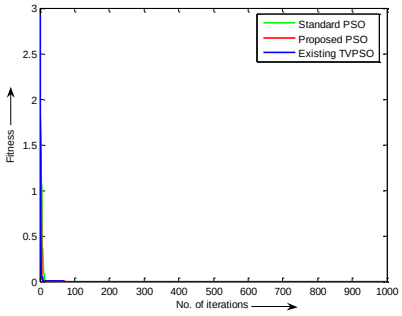

Figure 4: The Rastrigin Function after 1000 iterations

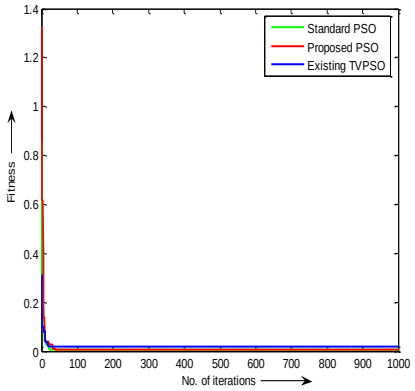

Figure 5: The Griewank Function after 1000 iterations

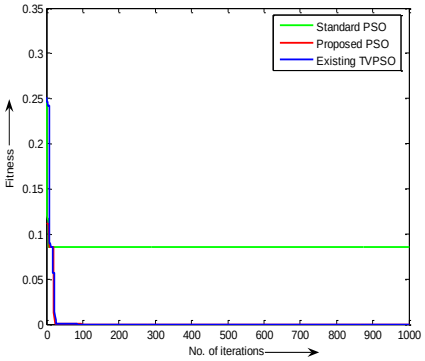

Figure 6: The SchaferF6 Function after 1000 iterations

5. Conclusion And Future Work

There are many modifications have been done to the standard Particle Swarm Optimization algorithm. Here a very simple technique is used to update the inertia weight. The proposed method is an approach to get better convergence, which has been implemented using Matlab and tested with some of the standard problems like Rosenbrock's Banana function, Sphere function ,Shaffer f6 function, Generalized Restrigrin function, Ackley function and Generalized Griewank function etc. these are frequently used to test the efficiency of the new heuristic search algorithms.

The new time variant version of PSO has been tested with some numerical optimization problems, but it has not been applied to any practical or real world problem. So the future work includes the implementation of the modified weight variant PSO to solve a real world problem with lots of complexity and to compare the efficiency of the modified PSO with another recent optimization technique.

References

[1] A. Ratnaweera, S. K. Halgamuge, H. C. Watson (2004) "Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients", IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3,June

[2] B. K. Panigrahi, V. R. Pandi, S. Das (2008) " Adaptive particle swarm optimization approach for static and dynamic load despatch", Conference on Energy Conversion and Management, pp.1407-1415.

[3] E. C. Laskari, K. E. Parsopoulos and M. N. Vrahatis (2007) "Particle Swarm Optimization for Integer Programming", Conference on Advance Computing

[4] D. Falco, A. D. Cioppa, E. Tarantino,(2007)" Facing classification problems with Particle Swarm Optimization ",Applied Soft Computing ,pp. 652–658

[5] J. Sidabras, "A Survey on Particle Swarm Optimization", MSCS 252- Deterministic Models of Operations Research.

[6] J. Wang, (2007) "A Novel Discrete Particle Swarm Optimization Based on Estimation of Dstribution",ICIC 2007, LNAI 4682, pp. 791-802

[7] K. O. Zones (2006) "Comparison of Genetic Algorithm and Particle Swarm Optimization for fermentation feed profile determination", International Conference on Computer System and Technologies, comp sys tech.

[8] M. A. Panduro, C. A. Brizuela., L. I. Balderas and D. A. Acosta , (2009)"A comparison between Genetic Algorithms, Particle Swarm Optimization and the Differential Evolution method for the design of scan able circular antenna arrays", Progress In Electromagnetic Research B, Vol. 13, 171-186,

[9] M. A. Montes, T. Stutzle, M. Birattari, M. Dorigo, (2008) "Frankensein's PSO: A Composite Particle Swarm Optimization Algorithm",IRIDIA- Technical Report Series ISSN 1781-3794, January 2008.

[10] M. Settles and T. Soule1 "Breeding Swarm a GA/PSO Hybrid" Department of Computer Science, University of Idaho, Moscow, Idaho, USA

[11] M. Al. Obaidy and A. Ayesh (2008) "The Implementation of Optimization Algorithm for Energy Efficient Dynamic Ad hoc Wireless Sensor Networks", Proceedings of the Fourth International Workshop on Advance Computation for Engineering Applications, Japan, pp. 17-23.

[12] N. Farokhnia, R. Khoraminia and G.B. Gharehpetian (2010) " Optimization of PI Controller Gains in Nonlinear Controller of STTCOM Using PSO and GA", International conference on Renewable Energies and Power Quality,Granada(Spain),23rd to 25th March.

[13] R. Hassan, B. Kohanim and O. d. Weck (2004) "A comparison of Particle Swarm Optimization and Genetic Algorithm", Massachusetts Institute of Technology, Cambridge.

[14] S. Consoli, J. A. Moreno Perez, K. Darby-Dowman and N. Mladenovic, "Discrete Particle Swarm Optimization for the minimum labeling Steiner tree problem" .

[15] S. Jason and M. Middenford, (2005) "A Hierarchical Particle Swarm Optimizer and Its Adaptive Variant", IEEE transaction on systems,Man and Cybernetics-PartB:,Vol. 35, No, 6, December 2005.

[16] X. Hu and R. C. Eberthart (2002) " Adaptive Particle Swarm Optimization: Detection and Ressponse to Dynamin Systems",Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE Press,pp.1666-1670.

[17]X. Cai, Y. Cui and Y. Tan, (2009) "Predicted Modified PSO with time- varying accelerator coefficients",Int.J.Bio-Inspired Computation, Vol.1.Nos. 1/2.

First Author Rajesh Kumar Ojha had received his B.Tech in Computer Science and Engineering from Biju Patnaik University of Technology in 2006 and M.Tech in Computer Science and Engineering From KIIT University in 2010. He had worked at PKACE bargarh in between 2006 and 2008 and now working as an Assistant Professor and Head of the Department of Computer Science and Engineering in Gandhi Institute of Technology and Management since 2008. He is a member of IEEE, secretary of IEEE EDS Bhubaneswar chapter and a member of IETE. His research interests is in the field of Soft Computing and Service Oriented Architechture.

Second Author Madhabananda Das had received his Phd from KIIT university in 2010. He had worked at College of Engineering and Technology in Computer Science till 2002 and now he is a Senior Professor in KIIT University since 2002. His research interests is in the field of soft computing.