Escape Local Minima with Improved Particle Swarm Optimization Algorithm

K. Darshana Abeyrathna[a] and Chawalit Jeenanunta[b]
Department of Information and Communication Technology,
University of Agder, Grimstad, Norway[a].
School of Management Technology, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand[b].
darshana.abeyrathna@uia.no[a], chawalit@siit.tu.ac.th[b]

Abstract

Particle Swarm Optimization (PSO) is a powerful meta-heuristic technique which has been maneuvered to solve numerous complex optimization problems. However, due to its characteristics, there is a possibility to trap all particles in a local minimum in the solution space and then they cannot find the way out from the trap on their own. Therefore, we modify the traditional PSO algorithm by adding an extra step so that it helps PSO to find a better solution than the local minimum that they undesirably found. We perturb all the particles by adjusting parameter values in the traditional algorithm when there is no improvement of the objective value over the training iterations, assuming that particles have stuck in a local minimum. In this research, we mainly focus on adjusting the learning factors. However, the parameter values have to be used in an effective way to perturb the particles. The behavior of the proposed modification and its parameter adjustments are studied using a function which has a large number of local minima - Schwefel's function. Results show that 2 out of 3 PSO attempts trap in local minimum and slight changes on learning factors do not help them to get out from the traps. However, perturbances made with large learning factors can find better solutions than the local minima that they stuck in and help to find the global minimum eventually.

1. Introduction

Particle Swarm Optimization (PSO) is a global optimization technique where it is used to solve complex optimization problem with highly non-linear objective functions. It has been successfully applied by researchers and engineers in number of domains to solve complex optimization problems, including weights optimization of Artificial Neural Networks [1, 2].

In most of the applications, we try to minimize the objective value of the optimization problems. In many of those cases, it can be adjusting parameters to find the minimum distance, minimum cost, minimum time required, minimum error, etc. However, with PSO, a large number of solutions for the objective function are created by particles in the PSO at one generation. Generation by generation, all particles evolve to an optimum which provide the optimum parameter values.

Once the objective function is ready with a set of constraints, fitness of each particle is calculated. This is the objective output for the selected particle. At end of the fitness value calculation of all the particles, personal best (pbest) and the global best (gbest) solutions for the current population is updated. Each particle has a best solution that has been obtained so far. This best solution of each particle is called the pbest. However, gbest is the best particle that provides the best parameter values to the objective function out of all other particles from all generations. These updated pbest and gbest are used to calculate the positions and velocities of each particle at the next generation using the following equations.

$$V^i(t+1) = w \times V^i(t) + c1 \times rand1 \times (p - X^i(t)) + c2 \times rand2 \times (g - X^i(t)) \quad (1)$$

This paper was presented at the NIK 2019 conference. For more information see http://www.nik.no/

$$X^i(t+1) = X^i(t) + V^i(t+1) \quad (2)$$

The velocity and position of i^{th} particle at generation t is given by $V^i(t)$ and $X^i(t)$, respectively while pbest and gbest given with p and g in the above equations. The effect of current velocity for the next generation is decided by the inertia weight w. With a higher value for the inertia weight, current velocity has a higher impact from the previous velocity and vice versa. Random values, *rand1*, *rand2* lie between 0 and 1. Learning factors *c1* and *c2* have to be predetermined and different values of them alter the performance of the PSO algorithm significantly.

However, when the learning procedures of all the particles in PSO decide by the Eq. (1) and (2), parameter values can be adjusted so that, all particles closely follow the global best or they follow their own best to beat the global best solution separately. Both these procedures have their own advantages and disadvantages. The highlighting limitation of the first procedure is once the global best particle traps in a local minimum, all particles will follow that specific particle and trap in the same local minimum. Since they closely follow the global best particle, none of the other particles will find a way to get out from the trap. In the second procedure, each particle competes each other to becomes the best particle of the entire set. In that case, PSO requires large number of training cycles to find the global best solution and will stop at a random location where they find the best at that specific training iteration, otherwise.

Therefore, in this research, we propose a solution for the above limitations of the PSO algorithm combining the discussed properties of the individual procedures. The algorithm starts following the first procedure where all particles closely follow the global best particle. Once they trap in local minima, we switch the parameter values of the Eq. (1) so that they find a completely new solution set for the next generation. In that way, one or some of the particles can find better solutions than the current global best where it is actually a local best of the actual solution space. In this research, we manly focus on adjusting the learning factors *c1* and *c2*. We also conduct an experiment to find the best learning factor combination.

The rest of the paper is organized as follows. The works related to the proposed algorithm and their usage are discussed in the following section (Sec. 2). Steps of the proposed algorithm are presented in detail in the Methodology section (Sec. 3). A detailed description about the experiment setup is given in Sec.4. Results of the conducted experiment are presented and discussed in the Results and Discussion section (Sec. 5). The conclusion for the entire research is made at the end (Sec. 6).

2. Related Works

Finding the best hyper parameter combination for any algorithm is important and difficult. The selected parameters decide the performance and final outcome of the algorithm. This is critical for the PSO algorithm compared to others since it has collectively 5 hyper parameters to be adjusted (Eq. (1)). Adjusting them separately while keeping others in a fixed position will consume a lot of time and might not find the best combination. However, researchers have conducted different experiments to select the best hyper parameters for PSO [3, 4]. The effect of different inertia weights on the performance of PSO has been tested in [3]. The performance has been evaluated by varying inertia weight with an additional constraint call "maximum velocity", V_{max}. According to the empirical results, they identify, when the chosen maximum velocity is small (≤ 2), inertia weight should be fixed approximately at 1. In contrast to the above,

if the maximum velocity is not small (≥ 3), 0.8 for the inertia weight is a better choice. According to the experiment setup in [4], two sets of parameters are selected to test the PSO performance based on the selected parameters. In both sets, learning factors are considered as qual (*c1 = c2*). In the parameter set 1, the inertia weight is 0.6 and learning factors are 1.7 each. In the parameter set 2, the inertia weight is 0.729 and learning factors are 1.494 each. Test results reveal better performance of parameter set 1 while showing the increase of performance parallel to the increase of the number of particles in a generation. However, still it's a comparison of results between selected sets, but it is not the best way to find the best parameter set when the best parameter set is not included in the testing sets.

In addition to findings of best hyper parameters, some researchers have tried to modify the algorithm in order to get better results. A cooperative approach to the PSO algorithm has been presented in [5]. In this proposed approach, different sections of the solution vector are optimized using different individual swarms in PSO. The proposed algorithm is used to solve a benchmark optimization problem and compared against the traditional PSO approach to show the value of the proposed approach. Similar to the above approach, [6] presents a modified version of PSO called Quantum Delta-Potential-Well-based Particle Swarm Optimization (QDPSO) algorithm. In their approach, they identify the quantum behavior of particles. Then they put a higher attention on the particles which are far from the global best. According to their argument, those particles have a higher potential to find better solutions than particles which closely following the global best. Another approach call Adaptive Particle Swarm Optimization (APSO) is presented in [7]. The proposed approach has following steps: fitness of each particle is calculated, four evolutionary states are identified based on the population distribution – 1. exploration, 2. exploitation, 3. convergence, and 4. jumping out. Based on these states, hyper parameters of Eq. (1) are automatically controlled. Then next generation is created based on the adjusted parameter values.

The performance of the algorithm has been improved not only by modifying the algorithm parameters, but also by combing it with some well know algorithms such as Genetic Algorithm (GA) [8]. The procedure in [8] is similar to the procedure in this paper. However, it is complex in nature as it uses GA operations once they stuck in local minima. In [9], a selection mechanism from evolutionary computation has been used to improve the PSO performances. A score for a particle is calculated based on its current position and positions of the other particles. Once the scores of all particles are calculated, they are sorted based on the calculated score. Then the positions and velocities of worst 50% of particles are replaced with the remaining 50% best particles. At each training iteration, this additional procedure is added to the algorithm. In the testing phase, the proposed algorithm shows better performances on three out of four testing functions. A similar approach to enhance the PSO performance is given in [10] by combining PSO and Chaos theory. According to the simulation results, chaotic PSO (CPSO) outperforms standard PSO and some other meta-heuristics algorithms.

The concept of the proposed algorithm in this research is quite similar to the methods in [8, 10]. Compared to [8], the proposed method is easy and computationally inexpensive. Compared to [10], the proposed algorithm starts with traditional PSO and let all particles (moving closer to the global best) to reach an optimum point (global or local) which is faster compared to their research. Then, only when it's required, particles are perturbed to get out from the trap in case if they have stuck in local minima. This proposed algorithm is discussed in detail in the next section.

3. Methodology

The proposed algorithm starts similar way to the traditional PSO approach. Parameters in the objective function collectively form the particles in PSO. Each particle makes a solution to the objective function and all particles collectively moves towards the global optimum during the training phase to find the optimum parameter set.

The algorithm starts creating *n* random solutions to the objective function and *n* is equal to the number of particles in a population. The fitness value of each particle is calculated using the objective function. The particle with the best fitness value is considered as the global best and they also update their personal bests at every training iteration as explained in the introduction section. Likewise, with the aid of Eq. (1) and (2), generation by generation they reach to the global optimum in the solution space. However, it is not possible to say whether particles have reach to the global optimum or traps in a local optimum when it shows minor or no improvements at some adjacent training iterations. However, with the traditional approach, they cannot get out from the local minimum in case if they trap in one. Therefore, in this algorithm, we add an additional step to perturb the particles if there is no or les improvement in the global best for 10 consecutive training iterations (10 stall iterations) by changing learning factors, *c1* and *c2*. However, we also save the current personal best and global best solutions in case if they have already reach to the global solution. This step is applied two times to the traditional approach and assume that it's enough for them to escape local minima and reaches to the global minimum. Then the training procedure stops when it reaches to the maximum training iterations. The states of the proposed algorithm can be well learned by using the following flowchart.

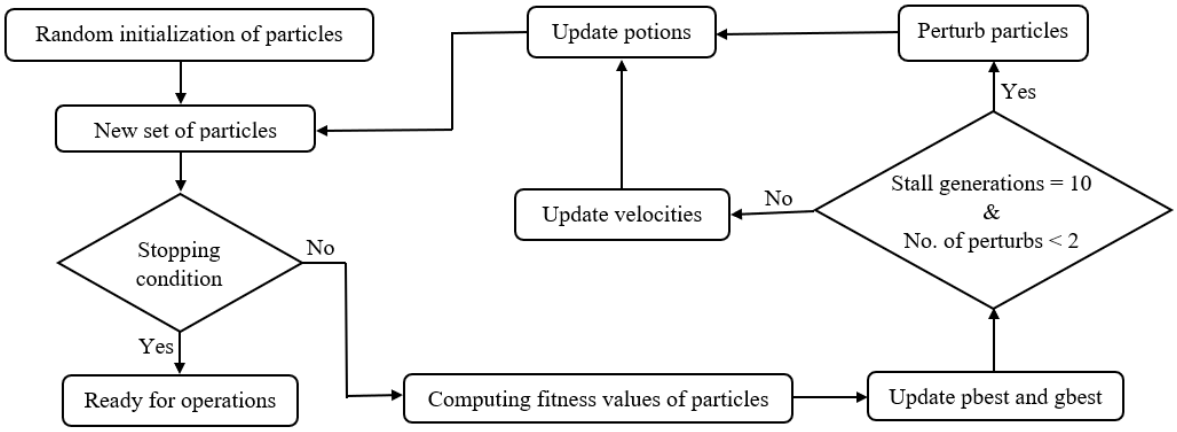


Figure 1 Flowchart of the improved PSO algorithm

However, the best learning-factor combination, *c1* and *c2*, to perturb particles are still unknown. Therefore, to find the best combination of learning factors and to check the performance of the proposed algorithm, an artificial dataset is created. Detail of the experiment setup is given in the next section.

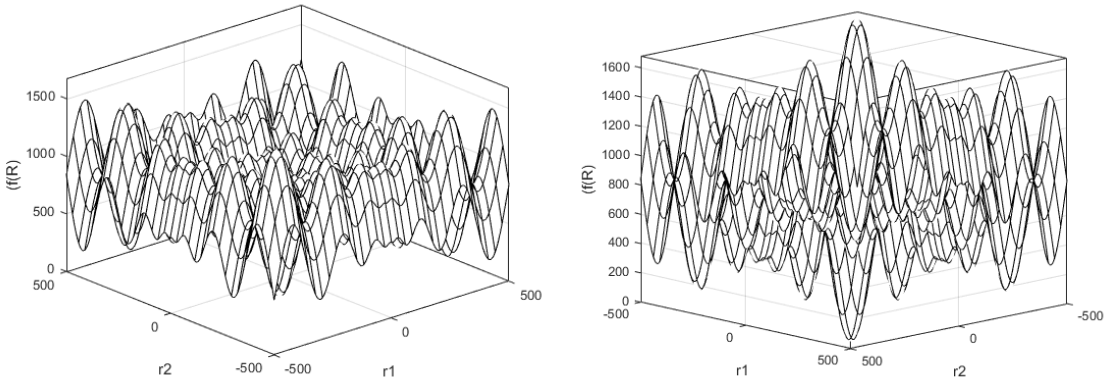
4. Experimental Setup

The objective of the proposed algorithm is to improve the traditional PSO algorithm which, sometimes, shows poor convergence ability with local minima in the solution space. Therefore, to demonstrate the optimization qualities of the proposed algorithm, a

function with multiple local minima, which is famous in terms of its toughness to find the global optimum [11] is selected: Schwefel's function. While Eq. (3) gives the Schwefel's function, Figure 2 shows some angles when it is plotted for two variables.

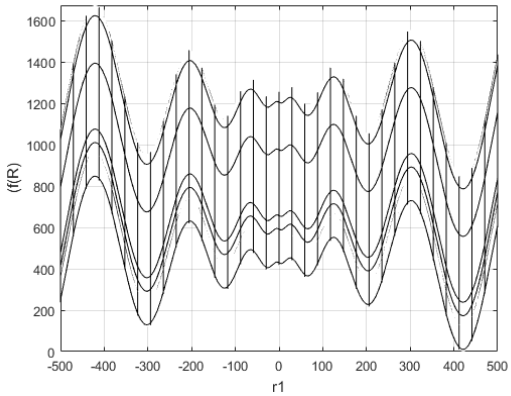
$$f(R) = \quad 418.9829 \times d - \sum_{i=1}^{d} r_i \times \sin(\sqrt{|r_i|}) \tag{3}$$

In this equation, to reach the global optimum, 0, we set a constraint where the maximum and minimum limits of each variable r_i are 500 and -500, respectively. Then, with any number of variables, d the objective value $f(R)$ reaches to 0 at the optimum R^* [$R^* = (r_1^*, r_2^*, \ldots\ldots r_d^*) = (420.9687, 420.9687, \ldots.. 420.9687)$]. However, the following figure shows different angles of the Schwefel's function when it is drawn with two variables (d = 2).



a. An angle from the top

b. An angle from the bottom



c. Side view from r_1 direction

Figure 2 Schwefel's function for two variables

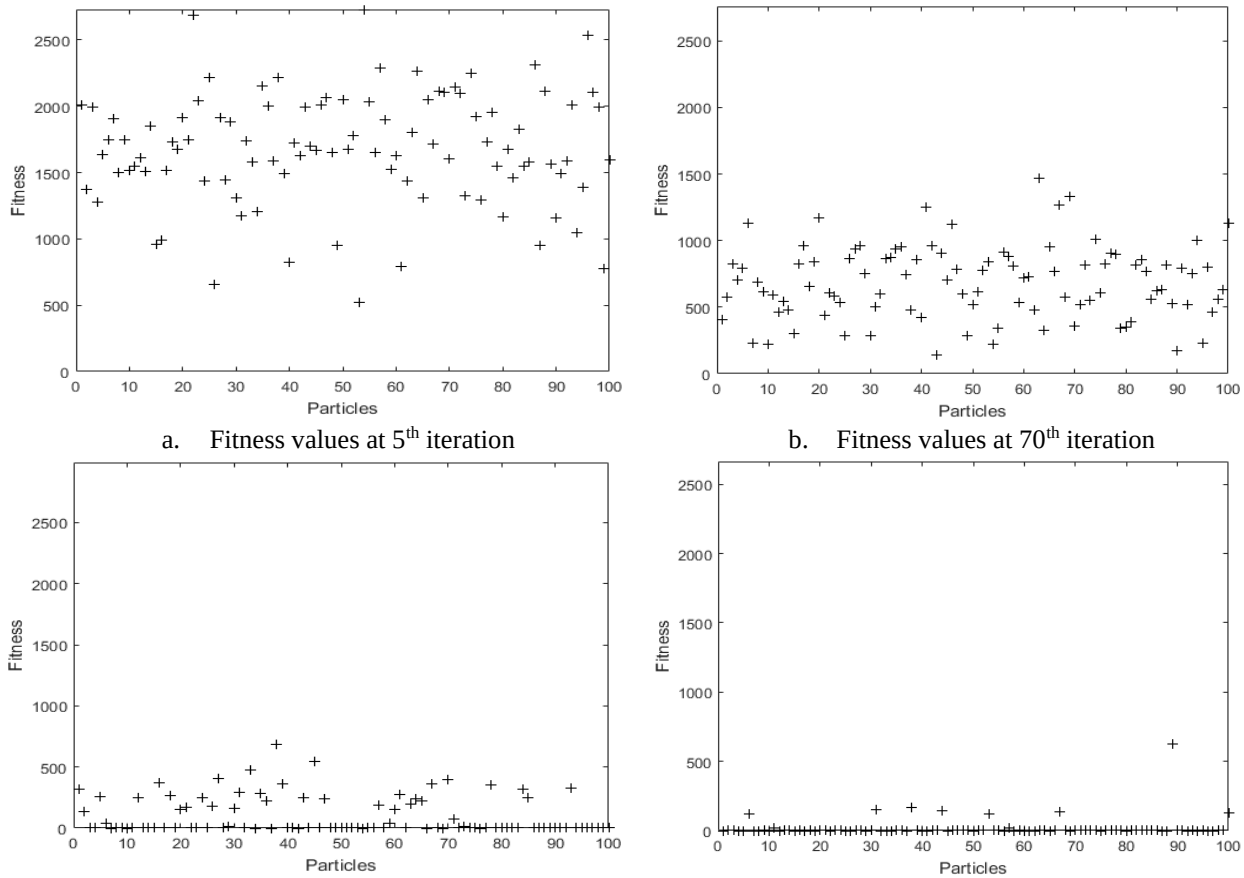
Figure 2 reveals the difficulty of finding the global minimum of the Schwefel's function since it has many local minima. However, in the experiment, we make it more difficult adding two more variables (d = 4) to find the global optimum with both the proposed and traditional PSO algorithms.

The proposed algorithm is similar to the traditional PSO at the beginning. To initiate with, 100 particles are created randomly with the considered constraint for the variables ($r_i \epsilon$ [-500, 500]). Fitness values of those particles are calculated using the Schwefel's

function. Based on the fitness values, pbest and gbest are updated. Velocities and positions for the next generation are calculated using Eq. (1) and (2), respectively. Considering the best parameter values identified at [4], we fix the inertia weight, *w* and learning factors, (*c1 = c2*) at 0.6 and 1.7, respectively. The traditional PSO algorithm runs 500 training iterations (generations), if does not meet 10 continuous stall generations, without converting to the proposed algorithm. Once it meets 10 continuous stall generations, all the particles in the current generation are perturbed to create the next generation. However, in this research, particles are perturbed by only adjusting learning factors. Therefore, these values are changed starting from 2 (*c1 = c2* = 2, 2 is slightly higher than the current value 1.7). The other values selected for learning factors are, 10, 20, 50, 100, 500, and 1000. The results obtained with these changes are presented and discussed in the next section.

5. Results and Discussion

The fitness values of each particle at each training iteration is studied separately. At the beginning, fitness values of all particles are higher and generation by generation they all reach to a minimum point (global or local). The evolution of these particles of a successful PSO attempt are captured at four different training iterations and resented in Figure 3.



a. Fitness values at 5th iteration

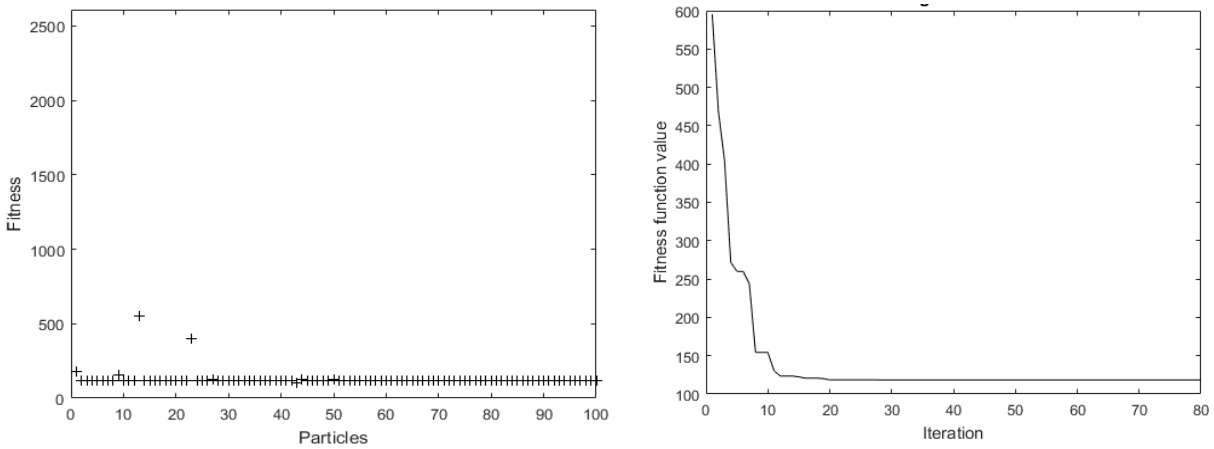
b. Fitness values at 70th iteration

c. Fitness values at 150th iteration

d. Fitness values at 170th iteration

Figure 3 Evolution of particles at different training iterations

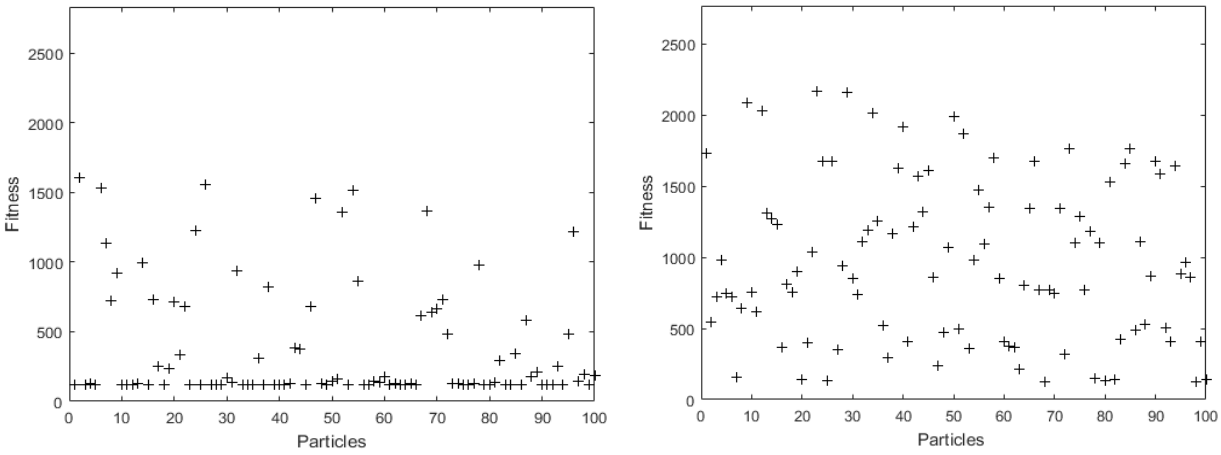
However, when the global best moves towards to a local minimum, all particles follow it and trap in the same local minimum (in case the other particles unable to find a better solution) as shown by Figure 4. Figure 4.a shows that almost all particles have reached to the same local minimum at end of the training iterations and Figure 4.b shows the evolution of the fitness value of the global best over the training iterations.



a. Fitness values of all particles at 80th iteration b. Global best fitness over the iterations

Figure 4 Evolution of particles towards a local minimum

We perturb all particles when they stick in these kinds of traps. However, different values for the learning factors (*c1* and *c2*) perturb particles in the Figure 4.a in different ways. For instance, variation of fitness values of particles, when learning factors during the perturbance process are equal to 10 and 500, are given in Figure 5.a and 5.b, respectively.



a. Perturb with *c1* = *c2* = 10 b. Perturb with *c1* = *c2* = 500

Figure 5. Perturb particles when they trap in local minima

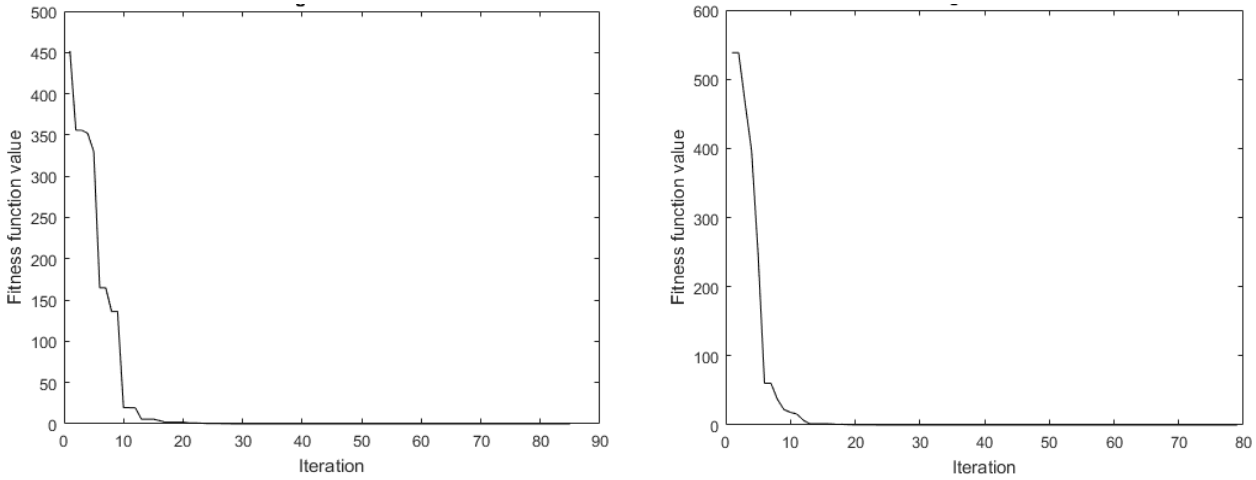
Therefore, the effect of the perturbances depends on the values of the used learning factors. The Table 1 provides the average minimum values of each training algorithm (traditional PSO and proposed algorithm with different learning factor values) obtained after 30 runs.

The table shows that 2 out of 3 PSO attempts trap in local minima and small perturbances do not help them to get out from those traps. With higher values for learning

factors, PSO starts finding new and better solutions than the local minima that they stuck. Figure 6, 7 and 8 provide selected examples for the cases where (Figure 6) PSO found the global optimum itself, (Figure 7) PSO stuck in local minima, and (Figure 8) how perturbances help PSO to escape local minima.

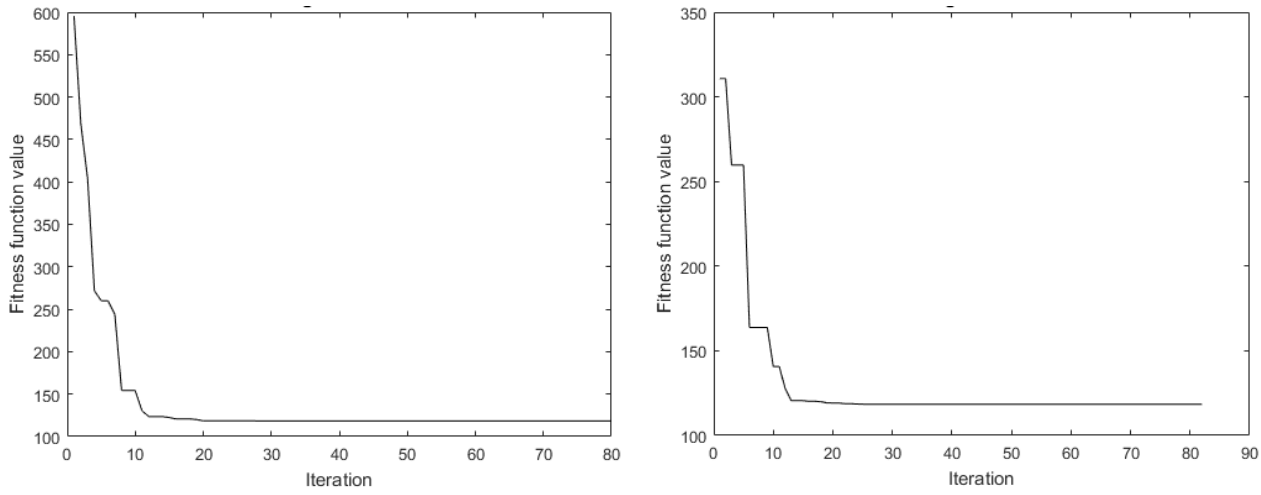
Table 1 Average of the minimum values given by different algorithms with different parameters

	PSO	**Proposed Algorithm with *c1* = *c2* =**						
		2	**10**	**20**	**50**	**100**	**500**	**1000**
No. of global minimum findings (out of 30)	10	9	10	10	12	14	19	21
Average of minimum values	93.87	97.80	90.07	101.47	85.73	77.87	50.73	35.40



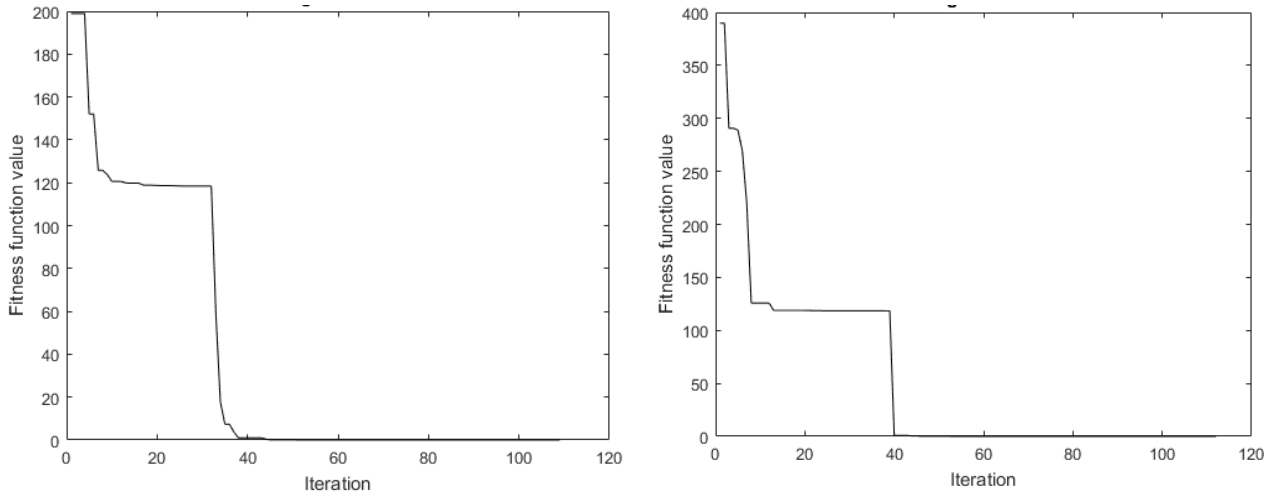
a. Example 1 b. Example 2

Figure 6 Examples that PSO found the global minimum without the help of perturbances



a. Example 1 b. Example 2

Figure 7. Examples that PSO stuck in local minima



a. Example 1 b. Example 2

Figure 8. Examples that PSO found the global minimum with the help of perturbances

Examples in both Figure 7 and 8 show that there is a local minimum around 120 fitness value in the solution space. Figure 6 shows that PSO can find the global minimum without the help of perturbance step. However, both Figure 7 and 8 and also the Table 1 show that still PSO can trap in local minima and therefore need perturbance step to escape them. The following section brings the conclusions of the research based on the hypothesis we made, the method we proposed, and the above results.

5. Conclusion

PSO is a powerful meta-heuristic technique which has been used to solve plenty of complex optimization problems. However, it has a limitation where when it traps in local minima in the solution space, it cannot get out from the trap on its own. Therefore, in this research, we slightly modify the algorithm by adding an extra step to perturb the particles when they trap in local minima. This modification to the algorithm bring a significant improvement to the results discussed in the previous section.

However, even though we identify that PSO likely to trap in local minima when it is used to solve functions which have many local minima and the proposed modifications to the algorithm can solve the problem, the parameter selection is still an issue. Therefore, similar to the other hyper parameters, this value of the learning factors during the perturbance stage have to be selected after a grid search. Values are subject to change with the difficulty of the problem. Therefore, a trial and error attempt to find the best learning factors to perturb the particles when they trap in local minima is necessary. These factors should be able to perturb the particles until at least they produce the initial fitness values.

References

1. Jeenanunta, C. and K.D. Abeyrathn, *Combine Particle Swarm Optimization with Artificial Neural Networks for Short-Term Load Forecasting.* ISJET, 2017. **8**: p. 25.

2. Daş, G.S., *Forecasting the energy demand of Turkey with a NN based on an improved Particle Swarm Optimization.* Neural Computing and Applications, 2017. **28**(1): p. 539-549.
3. Shi, Y. and R.C. Eberhart. *Parameter selection in particle swarm optimization*. 1998. Berlin, Heidelberg: Springer Berlin Heidelberg.
4. Trelea, I.C., *The particle swarm optimization algorithm: convergence analysis and parameter selection.* Information Processing Letters, 2003. **85**(6): p. 317-325.
5. Van den Bergh, F. and A.P. Engelbrecht, *A cooperative approach to particle swarm optimization.* IEEE transactions on evolutionary computation, 2004. **8**(3): p. 225-239.
6. Sun, J., B. Feng, and W. Xu. *Particle swarm optimization with particles having quantum behavior*. in *Evolutionary Computation, 2004. CEC2004. Congress on*. 2004. IEEE.
7. Zhan, Z.-H., et al., *Adaptive particle swarm optimization.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009. **39**(6): p. 1362-1381.
8. Abeyrathna, K.D. and C. Jeenanunta, *Hybrid Particle Swarm Optimization With Genetic Algorithm to Train Artificial Neural Networks for Short-Term Load Forecasting.* International Journal of Swarm Intelligence Research (IJSIR), 2019. **10**(1): p. 1-14.
9. Angeline, P.J. *Using selection to improve particle swarm optimization*. in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 1998. IEEE.
10. Liu, B., et al., *Improved particle swarm optimization combined with chaos.* Chaos, Solitons & Fractals, 2005. **25**(5): p. 1261-1271.
11. Whitley, D., V.S. Gordon, and K. Mathias. *Lamarckian evolution, the Baldwin effect and function optimization*. 1994. Berlin, Heidelberg: Springer Berlin Heidelberg.