A Performance Class-Based Particle Swarm Optimizer

Chia Emmanuel Tungom[1], Maja Gulan[2(✉)], and Ben Niu[1(✉)]

[1] College of Management, Shenzhen University, Shenzhen 518060, China
chemago99@yahoo.com, drniuben@gmail.com
[2] Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia
majica.gulan@gmail.com

Abstract. One of the main concerns with Particle Swarm Optimization (PSO) is to increase or maintain diversity during search in order to avoid premature convergence. In this study, a Performance Class-Based learning PSO (PCB-PSO) algorithm is proposed, that not only increases and maintains swarm diversity but also improves exploration and exploitation while speeding up convergence simultaneously. In the PCB-PSO algorithm, each particle belongs to a class based on its fitness value and particles might change classes at evolutionary stages or search step based on their updated position. The particles are divided into an upper, middle and lower. In the upper class are particles with top fitness values, the middle are those with average while particles in the bottom class are the worst performing in the swarm. The number of particles in each group is predetermined. Each class has a unique learning strategy designed specifically for a given task. The upper class is designed to converge towards the best solution found, Middle class particles exploit the search space while lower class particles explore. The algorithm's strength is its flexibility and robustness as the population of each class allows us to prioritize a desired swarm behavior. The Algorithm is tested on a set of 8 benchmark functions which have generally proven to be difficult to optimize. The algorithm is able to be on par with some cutting edge PSO variants and outperforms other swarm and evolutionary algorithms on a number of functions. On complex multimodal functions, it is able to outperform other PSO variants showing its ability to escape local optima solutions.

Keywords: Particle Swarm Optimization · Learning strategy · Swarm intelligence

1 Introduction

Optimization is one of the key features in obtaining good performance in systems. In fact optimization problems can be found everywhere in real life from transportation to even dieting. PSO is an intelligent optimization algorithm designed

Y. Tan et al. (Eds.): ICSI 2020, LNCS 12145, pp. 176–188, 2020.
https://doi.org/10.1007/978-3-030-53956-6_16

in the mid 1990's by Eberth and Shi [1,2]. The algorithm's working principle is based off-of simulating the collective behavior of bird flock and fish school.

The original PSO algorithm has a topology fully connected network where all particles learn from their personal best historical search position and the global best particle in the swarm. This learning structure is the main reason why the original PSO algorithm is inefficient and ca easily be trapped into a local minima as all particles are guided by one global leader. Several other topological structures have been introduce to enhanced performance e.g. the ring topology, the von Neumann topology, the pyramid topology [3]. These topologies use different ways to update the velocity and position of particles in the swarm. Fully Informed PSO (FIPS) determines the velocity of a particle by looking at its neighborhood topology [4]. Comprehensive learning PSO (CLPSO) tries to solve the problem of premature convergence by using different learning topologies on different dimensions to ensure diversity is maintained [5].

Exploration, the ability of the swarm to search its entire environment (global search) and exploitation, the ability for particles to thoroughly search their neighborhood (local search) are two important features of any PSO or search algorithm. To ensure swarm stability, stability based adaptive inertia weight (SAIW) uses a performance based approach to determine each particles inertia weight [6]. Mixed Swarm Cooperative PSO (MCPSO) achieves exploration and exploitation by dividing particles into exploration and exploitation groups [7]. By leveraging comprehensive learning strategy, Heterogeneous Comprehensive Learning PSO (HCLPSO) enhances its exploration and exploitation [8]. There are several other balanced Algorithms that are specifically designed to ensure both exploration and exploitation [8–10].

In this study, a novel PSO algorithm called PCB-PSO is proposed, with a new learning topology to ensure exploration and exploitation while also ensuring a high convergence speed therefore avoiding premature convergence. In PCB-PSO, particles are divided into three groups, upper class, middle class and lower class based on their fitness values, and a particles' group might change from iteration to iteration. The upper class consist of particles with superior performance while the lower class consist of the poorest performing members. The middle class is made-up of members considered not to be performing poorly and not having superior performance. Particles in the same group have a common learning strategy or topology, which is different from those in the other groups. Lower class particles are designed to enhance exploration while the middle class particles are designed for exploitation. Upper class particles are designed for fast convergence. The intuition here is that if a particle is performing poorly, it has to do more exploration and if its performance is good, it focuses on converging faster, and if its performing neither poorly nor well, then it should exploit its neighborhood. The main contributions of this study can be listed as follows:

1. Introduces a new paradigm of PSO learning that enhances exploration, exploitation and convergence speed simultaneously.
2. Deals with the problem of premature convergence by making some particles continue exploration and exploitation while others converge to a given minima or optima.
3. The problem of swarm diversity is dealt with by continuous exploration of the search space by lower class particles.
4. Introduces flexibility by allowing a given behavior to be prioritized by simply assigning more or all particles to a given class.

The rest of the paper is organized as follows: The original PSO algorithm and some learning topologies are reviewed in Sect. 2. The proposed PCB-PSO algorithm is discussed in Sect. 3 and in Sect. 4, analysis and comparison of the results with other PSO variants on several benchmark functions is discussed. Finally, in Sect. 5 we draw our conclusion from this paper and propose future research directions.

2 Basic PSO Algorithm

PSO is a population based stochastic swarm and evolutionary computational algorithm. In PSO, a population of particles, with each having a position and velocity component, is use to find the solution to an optimization problem. Each particle is a solution and the search space is the set of all solutions to the given problem. The particles or solutions are evolved by updating the velocity and position after every iteration. A particles update is done using its personal best experience and the best experience of the entire swarm. This update is designed to guide the particles towards the global best solution and eventually and eventually towards the optimal solution. The update of velocity and position of each particle are done using Eqs. 1 and 2 respectively. The particles continue to evolve until a termination criterion is met usually the maximum iteration pre-determined before start of search.

$$V_i^{t+1} = wV_i^t + c_1 r_1 (X_{Pb}^t - X_i^t) + c_2 r_2 (X_{Gb}^t - X_i^t) \tag{1}$$

$$X_i^{t+1)} = X_i^t + V_i^{t+1} \tag{2}$$

where w is the inertia weight and wV_i^t the Inertia or momentum component. The inertia component directs a particles' trajectory in the search space for both exploration and exploitation towards unvisited search areas, its choice of value usually depend on the search dimension and varies in the range [0, 1]. Particles trajectories are maintained by the inertia component, which forces particles to

navigate through areas independent of previously successful searches. For a given particle in the swarm, X_i is its current positon, X_i^t its personal best solution at a given iteration time is X_{Pb}^t and the best position X_{Gb}^t of the entire swarm referred to as global best. The social component $c_2 r_2(X_{Gb}^t - X_i^t)$ and cognitive component $c_1 r_1(X_{Pb}^t - X_i^t)$ guide the particles towards the swarms best solution and a particles personal best solution respectively.

c_1 and c_2 are cognitive and social acceleration coefficients respectively usually set to 2 by default in the basic PSO algorithm. r1 and r2 are uniform random numbers generated between 0 and 1. w, c_1 and c_2 play an influential role in determining the swarm's behavior. A choice of each value is problem dependent and determines the convergence speed, exploitation and exploration ability of the swarm. Therefore, the choice of values should be harmonious. In a simple PSO algorithm, w can be between 0.4 and 0.9 and c_1 and c_2 can be set to 2.

3 Proposed PSO Algorithm

In this section, an efficient variant of PSO called PCB-PSO is proposed to simultaneously tackle the problems of premature convergence, exploration, exploitation and diversity while also converging faster to the global minima. The algorithm introduces a new learning topology and the major difference from the base PSO is as follows (1) Particles in the swarm belong to one of three groups based of their fitness value. (2) Each group has a learning strategy or update mechanism unique to it. In PCB-PSO, The best or top performing particles fall in the Upper Class (UC), average performing in the Middle Class (MC) and poorly performing particles fall in the Lower Class (LC). The population size of each class is predetermined and will be discussed in the later section. The intuition here is that, classifying particles into three groups can allow us to design a learning strategy for each group of the swarm to simultaneously explore, exploit and converge while maintaining diversity throughout the search. This is opposed to the base PSO where all the particles learn from a global leader making them to move towards one region of the search space. UC particles, which, have good performance and are most likely to find the optimal solution are designed to enhance convergence speed. LC particles, which are poorly performing in the swarm, roam the search area exploring new solutions and maintaining diversity of the swarm. MC particles, which are performing neither poorly nor well, are designed for exploitation since they move in areas between UC and MC particles (Fig. 1).

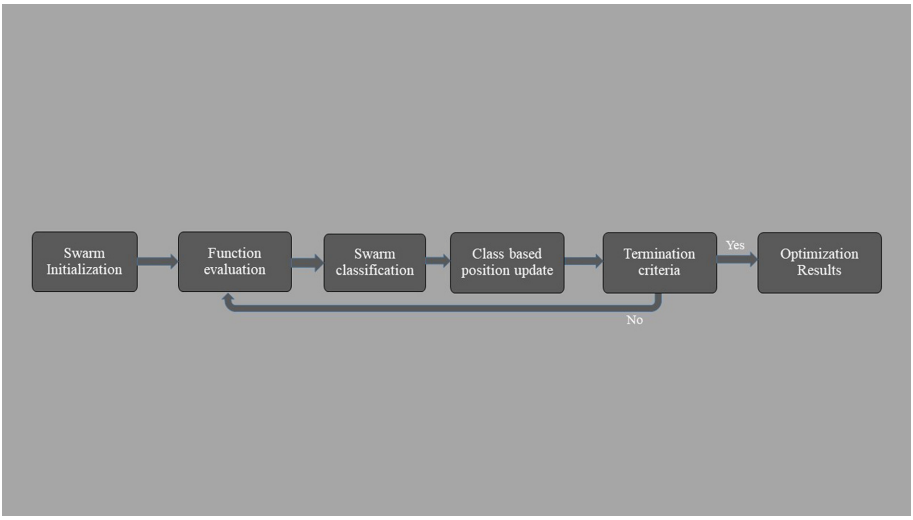

Fig. 1. Shows the sequential search process of the PCB-PSO.

3.1 Swarm Classes

We want to sort the particles into three classes UC, MC and LC with population sizes N_{uc},N_{mc} and N_{lc} respectively. The population of each class is determined in three simple steps. First, a class is chosen and the proportion of the total population N to belong to that class is assigned to determine the population of the given class. Secondly, the remaining population is used and a proportion of it is determined to fall in to the next class. Finally, the last remainder of the population falls into the last class. Note that we can start with any class. For convenience sake, in this paper, we will be starting with UC, MC and then LC in that order. Let us say the proportion of particles we want in UC is 1/3 of N then $S_{uc} = 3$ and Let's assume the proportion of the remaining particles we want for MC is 1/2, then $S_{mc} = 2$. N is used to determine N_{uc} while $N - N_{uc}$ is use to determine N_{mc} as shown in Eqs. 3 and 4 respectively. The remaining particles after selecting N_{uc} and N_{mc} make up N_{lc} (Fig. 2).

$$N_{uc} = \frac{N}{S_{uc}} \tag{3}$$

$$N_{mc} = \frac{(N - N_{uc})}{S_{mc}} \tag{4}$$

$$N_{lc} = |N_{uc} - N_{mc}| \tag{5}$$

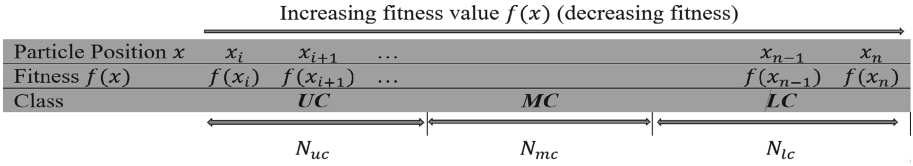

Fig. 2. Illustrates the classification of particles based on fitness with their corresponding classes and population sizes search process of the PCB-PSO.

UC and LC particles have probabilities associated to them. This associated probability is a measure of how likely a particle is chosen to be learned from by another particle in the learning strategy of a given class, which will be discussed later. For UC particles, the probability is proportional to the fitness value meaning the higher the value, the higher the probability while for LC particles; the probability is inversely proportional to the fitness value. MC particles learn from both UC and LC members and so we want the best particles in UC to be less likely to learn from and the best particles in LC to be more likely to learn from so as to keep them exploiting regions between MC and LC. The probability of a UC particle is calculated using Eq. 6 while that of LC is calculated using Eq. 7.

$$P_{UC_i} = \frac{f(X_{UC_i})}{\sum_{j=1}^{n} f(X_{UC_j})} \tag{6}$$

$$P_{LC_i} = \frac{\tilde{f}(X_{LC_i})}{\sum_{j=1}^{n} \tilde{f}(X_{LC_j})} \tag{7}$$

where UC_i is the i^{th} particle of UC and P_{UC_i} is the probability of that particle.LC_i is the i^{th} particle of LC and P_{LC_i} is the probability of that particle.$\tilde{f}$ is calculated such that the appropriate probabilities are derived for LC particles i.e. the lowest fitness values are flipped for all members with the poorest fit particle becoming the fittest and vice versa.

3.2 Update Mechanism

In PSO particles are updated by directing them towards the global best solution of the swarm and the personal best solution of the given particle with a velocity. This might be problematic if the solution found is not actually the best solution in the search space then particles will fail to explore other search regions missing other potentially better solutions. This problem has been solved by introducing different learning strategies and parameter modification. The simplest way is to vary the inertia weight and acceleration coefficients throughout the search to enhance exploration in the early stages and exploitation in the late stages. We leverage this idea but ensure both exploration and exploitation are carried-out throughout the search by introducing a new topology structure. This ensures a thorough search of the solution space making it less likely to miss a global

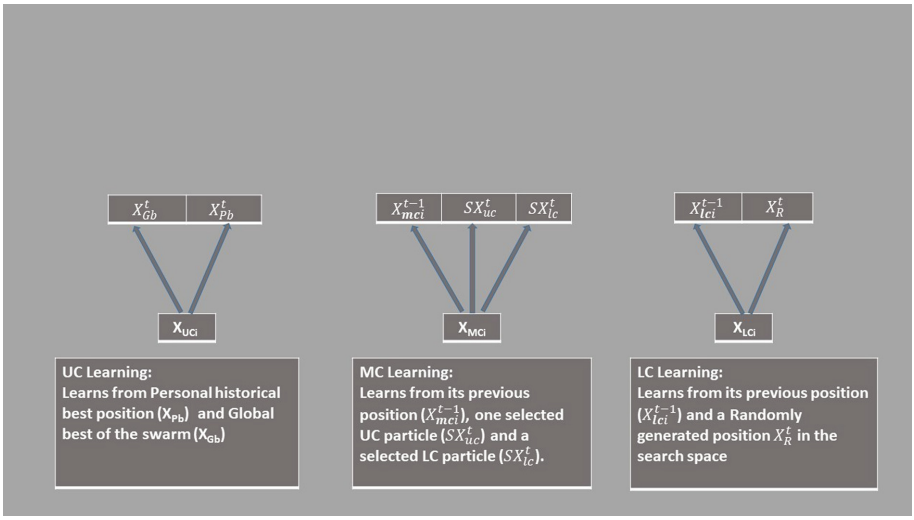

Fig. 3. Illustrates the learning mechanisms of each class with their source of information indicated by arrows.

best solution. The velocity component is not used in this algorithm because in higher dimensions, the inertia weight is required to be less than 0.1 to achieve reasonable results. Instead of using the recommended small value of the inertia weight, we focus on tuning acceleration of the social and cognitive component, which further simplifies the position update equation by eliminating the inertia weight and velocity. Hence, in this algorithm, we do not need to calculate the velocity component of a particle. In PCB-PSO, the update mechanism of each class is design for a given property of exploration, exploitation and convergence of the particles. Three different update mechanisms are designed for the three classes. Each update strategy is unique to a given class.

UC Update Mechanism. These are the best performing particles in the swarm and so are close to the best solution found at any period during the search. The update equation for UC is as shown in Eq. 8

$$X_i^{t+1} = X_i^t + c_1 r_1 (X_{Pb}^t - X_i^t) + c_2 r_2 (X_{Gb}^t - X_i^t) \tag{8}$$

The parameters are same as discussed in Sect. 2 (Fig. 3).

MC Update Mechanism. Particles in this class are designed to exploit the region outside the best solution found which ensures diversity and thorough exploitation of the space. The position update of this class is as shown in Eq. 9.

$$X_i^{t+1} = wV_i^t + c_3 r_3 (X_i^{t-1} - X_i^t) + c_4 r_4 (SX_{uc}^t - SX_{lc}^t) \tag{9}$$

where SX_{uc}^t and SX_{lc}^t are stochastically selected UC and LC particles at time t respectively. Taking the difference between a UC and LC particle allows the particle to fall somewhere outside the best solution already found and using the particles previous position instead of the personal best boosts the stochasticity of the search.

LC Update Mechanism. The update mechanism here is designed so that particles will explore the entire search region. The movement of particles in this class is chaotic which helps them to roam the swarm looking for better solutions. The position update is as shown in Eq. 10.

$$X_i^{t+1} = X_i^t + c_5 r_5 (X_i^{t-1} - X_i^t) + c_6 r_6 (X_R^{t-1} - X_i^t) \tag{10}$$

where X_R^t is a randomly generated position in the search space at iteration t aiding with the chaotic movement of particles in this group across the search space.

To further speed up convergence, at certain short intervals during the search, the population of the entire swarm is set to N_{uc} for UC learning as shown in Algorithm 1.

3.3 Parameter Setting

There are two groups of parameters to be set in this algorithm. The first is the class populations and the second is the acceleration coefficients for each class. The recommended settings for the proposed algorithm is outlined in Table 1. Note that the population size of any group cannot be greater than the entire swarm and the sum of all class population must be equal to the population of the entire swarm. At shown in Algorithm one, Such can be varied to push for quicker convergence at certain periods of the search. A higher swarm population will always enhance better results as opposed to other algorithms where increasing the swarm size after a certain threshold is reached might not help the performance of the algorithm which is because of the classification and behavioral setting of the particles. In PCB-PSO, the threshold is very high especially for problems with higher complexity.

Table 1. Parameters for proposed algorithm with desired and recommended settings.

Parameter	Name	Recommended value	Desired range
N	Population of swarm	[D*5, D*10]	[D*5, D*10]
N_{uc}	UC Population	>N/4	[1, N]
N_{mc}	MC Population	>N/4	[1, N]
N_{lc}	LC Population	>N/4	[1, N]
t	Iteration number	>D*3	>0
c_1	UC social acceleration coefficient	2.5	[0.0, 4.0]
c_2	UC cognitive acceleration coefficient	0.5	[0.0, 4.0]
c_3	MC social acceleration coefficient	1.5	[0.0, 4.0]
c_4	MC cognitive acceleration coefficient	1.5	[0.0, 4.0]
c_5	LC cognitive acceleration coefficient	1	[0.0, 4.0]
c_6	LC random acceleration coefficient	2	[0.0, 4.0]

4 Experiment Simulation and Results

To evaluate and ascertain the capability of the proposed algorithm, it is compared with other existing swarm and evolutionary algorithms on a set of benchmark functions. The PSO variants we compare with have been adopted for real world engineering applications and so can be regarded as state of the art. The algorithms used for this comparison include PSO, Harmony Search (HS), GA, Artificial Bee Colony (ABC), Cultural Algorithm (CA) and other advanced PSO variants.

Table 2. Outline of parameter settings for algorithms used for comparison.

Algorithm	Parameters
PSO	maximum velocity = 0.6, inertia weight=0.9, acceleration constants $c_1 = c_2 = 2$
HS	bandwidth = 0.2, harmony memory accepting rate = 0.95, pitch-adjusting rate = 0.3
GA	crossover probability = 0.7, mutation probability = 0.3
ABC	limit = 0.6×dimension×population
CA	Probability of the knowledge source = 0.35, number of accepted individuals = probability of the knowledge source × population
BBPSO	No parameter setting required
BBPSOV	Logistic map used as the search parameter
Proposed PSO	$N_{uc} = N_{mc} = N_{lc}$, $c_1 = 2.5$, $c_2 = 0.5$, $c_3 = c_4 = 1.5$, c_5 and c_6 are set to 1.0 and 2.0 respectively

4.1 Benchmark Functions

The benchmark functions used for evaluation have widely been used to evaluate swarm intelligence and evolutionary algorithms [11–13]. They include a set of 8 benchmark unimodal and multimodal functions as shown in Table 3. These functions have varied difficulty and have proven to be difficult to find optima solutions in high dimensions (>30). The unimodal functions (F1-F5) have single optimal solutions while the multimodal functions (F6-F8) have multiple. These functions are challenging and are often used to evaluate local exploitation and global exploration ability of a search algorithm.

Table 3. Shows benchmark functions used for performance testing.

Function	Range	Global minima
F1 Dixon-Price	[−10, 10]	0
F2 Sphere	[−5.12, 5.12]	0
F3 Rotated hyper-ellipsoid	[−65.536, 65.536]	0
F4 Sum squares	[−5.12, 5.12]	0
F5 Sum of different powers	[−1, 1]	0
F6 Ackley	[−32, 32]	0
F7 Griewank	[−600, 600]	0
F8 Powell	[−4, 5]	0

4.2 Experiment Setup and Results

For each of the Algorithms, swarm size is set to 50 and the search Dimension for each optimization function is set to 50. The maximum number of iteration is 1000 and each algorithm undertakes 30 independent runs. The parameter settings for each of the algorithms to be used is as shown in Table 2.

The proposed Algorithm shows superior performance in comparison to the other variants on the multimodal functions F6–F8. This shows the agility of the algorithms and its ability to escape the local optima while still managing to push for faster convergence. On unimodal functions, F1–F5 the performance is on par with other PSO variants but outperforms the other swarm and evolutionary algorithms. In unimodal functions, there is only one minima but our algorithm was still set to continuously explore and exploit other regions. If we set the whole population to UC, the algorithm will work towards faster convergence and we will expect achieve better quality results. In all we can see the flexibility and robustness of the algorithm given its built characteristics and swarm behavior in mind (Table 4).

Table 4. Results of algorithms on benchmark functions.

Function	Algorithm	Mean	Std	Min	Max
F1	PSO	2.30E+06	7.71E+05	1.28E+06	5.13E+06
	HS	7.91E+01	1.87E+01	4.54E+01	1.19E+02
	GA	4.12E+05	2.49E+05	1.02E+05	1.01E+06
	ABC	3.19E+06	8.10E+05	1.67E+06	4.97E+06
	CA	4.96E+03	4.51E+03	5.91E+02	1.86E+04
	BBPSO	2.63E+05	3.53E+05	7.86E+01	1.41E+06
	BBPSOV	4.77E+00	3.16E+00	6.70E−01	9.82E+00
	Proposed PSO	3.15E−06	4.96E−06	3.59E−08	2.05E−05
F2	PSO	1.47E+02	2.59E+01	8.46E+01	1.80E+02
	HS	3.91E−01	4.28E−02	3.09E−01	4.61E−01
	GA	4.12E+05	2.49E+05	1.02E+05	8.40E+01
	ABC	1.54E+02	2.35E+01	1.06E+02	1.94E+02
	CA	1.01E+00	1.25E+00	7.09E−02	6.42E+00
	BBPSO	2.35E+01	2.57E+01	1.29E−02	1.05E+02
	BBPSOV	5.21E−08	1.03E−07	9.30E−10	3.22E−07
	Proposed PSO	2.21E−06	7.45E−06	4.11E−09	4.00E−05
F3	PSO	6.03E+05	1.29E+05	3.96E+05	8.79E+05
	HS	1.68E+03	4.51E+02	7.65E+02	2.73E+03
	GA	2.40E+05	4.88E+04	1.53E+05	3.87E+05
	ABC	5.05E+05	7.24E+04	3.88E+05	6.54E+05
	CA	4.85E+03	4.89E+03	5.01E+02	2.33E+04
	BBPSO	1.21E+05	1.13E+05	4.46E+03	3.48E+05
	BBPSOV	1.88E−04	3.94E−04	3.84E−06	1.74E−03
	Proposed PSO	1.24E−03,	1.90E−05	4.19E−06	8.40E−04
F4	PSO	1.48E+04	3.97E+03	7.65E+03	2.47E+04
	HS	1.52E+01	2.16E+00	1.21E+01	2.27E+01
	GA	4.09E+03	1.25E+03	2.44E+03	7.34E+03
	ABC	1.18E+04	1.62E+03	8.60E+03	1.43E+04
	CA	1.46E+02	1.91E+02	5.08E+00	1.01E+03
	BBPSO	2.32E+03	1.53E+03	3.02E+02	5.71E+03
	BBPSOV	4.78E−06	8.15E−06	1.41E−07	3.11E−05
	Proposed PSO	1.50E−06	2.52E−06	1.12E−05	1.12E−05
F5	PSO	3.43E−01	2.60E−01	2.36E−02	9.53E−01
	HS	4.78E−07	6.68E−07	1.08E−08	3.45E−06
	GA	2.39E−02	2.15E−02	2.92E−03	8.04E−02
	ABC	7.59E−01	3.00E−01	1.87E−01	1.50E+00
	CA	6.15E−05	1.38E−04	1.86E−07	6.89E−04
	BBPSO	7.92E−10	3.68E−09	1.27E−15	2.01E−08
	BBPSOV	3.28E−18	1.01E−17	1.20E−22	5.33E−17
	Proposed PSO	9.25E−08	3.23E−07	1.27E−10	1.78E−06

(*continued*)

Table 4. (*continued*)

Function	Algorithm	Mean	Std	Min	Max
F6	PSO	1.95E+01	5.71E−01	1.82E+01	2.05E+01
	HS	3.09E+01	3.97E−01	3.02E+01	4.08E+01
	GA	1.61E+01	8.56E−01	1.44E+01	1.81E+01
	ABC	2.03E+01	2.54E−01	1.98E+01	2.08E+01
	CA	1.10E+01	2.36E+00	5.46E+00	1.74E+01
	BBPSO	1.85E+01	1.41E+00	1.42E+01	2.01E+01
	BBPSOV	3.27E+00	8.23E−01	2.08E+00	5.68E+00
	Proposed PSO	6.94E−05	1.17E−05	3.31E−07	5.31E−05
F7	PSO	5.28E+02	9.93E+01	3.09E+02	6.91E+02
	HS	3.06E+00	4.47E−01	2.08E+00	3.78E+00
	GA	2.56E+02	7.11E+01	1.45E+02	4.39E+02
	ABC	5.59E+02	7.47E+01	3.73E+02	7.23E+02
	CA	5.77E+00	5.48E+00	1.15E+00	2.83E+01
	BBPSO	7.27E+01	8.93E+01	1.04E+00	3.62E+02
	BBPSOV	1.78E−02	2.43E−02	1.14E−06	8.50E−02
	Proposed PSO	3.27E−04	4.37E−04	2.93E−05	1.70E−03
F8	PSO	1.40E+04	4.03E+03	6.44E+03	2.24E+04
	HS	2.25E+01	8.61E+00	8.69E+00	4.77E+01
	GA	2.35E+03	1.10E+03	1.02E+03	6.55E+03
	ABC	1.56E+04	3.57E+03	1.10E+04	2.35E+04
	CA	1.49E+02	1.14E+02	1.99E+01	4.62E+02
	BBPSO	2.67E+03	1.97E+03	1.79E+02	6.93E+03
	BBPSOV	5.79E−02	4.07E−02	1.85E−02	1.76E−01
	Proposed PSO	8.24E−07	9.19E−07	2.35E−08	3.08E−06

5 Conclusion and Future Work

In this study, a novel learning paradigm for PSO is introduce to balance exploration, exploitation and convergence while maintaining diversity of the swarm. The algorithm uses only the social and cognitive components of PSO for learning which further simplifies the algorithm. Particles Learn according to the class they fall in and these gives flexibility and robustness to the algorithm as different learning methods are in cooperated at the same time. The algorithm further introduces dynamism by varying class population at given intervals during the search. The algorithm is tested on a number of benchmark functions, compared with other swarm and evolutionary optimization algorithms including other advance PSO techniques, and see the superiority of the proposed algorithm. In future work, this algorithm will be tested on more functions like the 2013 or 2017 CEC suit. It will also be used to solve a real world problem.

Acknowledgement. This work is partially supported by Shenzhen Philosophy and Social Sciences Plan Project (SZ2019D018), Guangdong Provincial Soft Science Project (2019A101002075)

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
2. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of 6th International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
3. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 1671–1676 (2002)
4. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Trans. Evol. Comput. **8**(3), 204–210 (2004)
5. Liang, J.J., Qin, A.K., Suganthan, P.N., Baska, S.: Comprehensive learning particleswarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
6. Taherkhani, M., Safabakhsh, R.: A novel stability-based adaptive inertia weight for particle swarm optimization. Appl. Soft Comput. **38**, 281–295 (2016)
7. Jie, J., Zang, J., Zheng, H., Hou, B.: Formalized model and analysis of mixed swarm based cooperative particle swarm optimization. Neurocomputing **174**, 542–552 (2016)
8. Zhao, X., Lin, W., Hao, J., Zuo, X., Yuan, J.: Clustering and pattern search for enhancing particle swarm optimization with Euclidean spatial neighborhood search. Neurocomputing **171**, 966–981 (2016)
9. Meng, A., Li, Z., Yin, H., Chen, S., Guo, Z.: Accelerating particle swarm optimization using crisscross search. Inf. Sci. **329**, 52–72 (2016)
10. Yu, K., Wang, X., Wang, Z.: Multiple learning particle swarm optimization with space transformation perturbation and its application in ethylene cracking furnace optimization. Knowl.-Based Syst. **96**, 156–170 (2016)
11. Zhang, L., Srisukkham, W., Neoh, S.C., Lim, C.P., Pandit, D.: Classifier ensemble reduction using a modified firefly algorithm: an empirical evaluation. Expert Syst. Appl. **93**, 395–422 (2018)
12. Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. **27**(4), 1053–1073 (2015). https://doi.org/10.1007/s00521-015-1920-1
13. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl.-Based Syst. **89**, 228–249 (2015)