# Dynamic Cognitive-Social Particle Swarm Optimization

Khelil Kassoul[1,3]
[1]*Geneva School of Economics and Management, GSEM*
*University of Geneva*
Geneva, Switzerland
Khelil.Kassoul@etu.unige.ch

Samir Brahim Belhaouari
[2]*College of Science and Engineering*
*Division of Informatic & Computing Technology*
*Hamad Bin Khalifa University*
Doha, Qatar
sbelhaouari@hbku.edu.qa

Naoufel Cheikhrouhou
[3]*Geneva School of Business Administration*
*University of Applied Sciences Western Switzerland (HES-SO),*
Geneva, Switzerland
naoufel.cheikhrouhou@hesge.ch

2021 7th International Conference on Automation, Robotics and Applications (ICARA) | 978-1-6654-0469-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICARA51699.2021.9376550

***Abstract*—Particle Swarm Optimization (PSO) is a heuristic optimization algorithm based on the modeling of the behavior of fishes and birds flock. This paper proposes a better version of PSO, named Dynamic Cognitive-Social PSO "DCS-PSO", for global minima search by introducing optimal and dynamic cognitive and social scaling parameters without taking into consideration the inertia term. Furthermore, the velocity of each particle is controlled systematically at each iteration to avoid local minimum traps and to converge quickly and reliably towards the global minimum. The proposed algorithm is more suitable for high dimensional optimization problems and it has gotten over the limitations of classical Particle Swarm Optimization. Several experiments have been carried out, using the proposed DCS-PSO, to optimize thirteen benchmark functions and an important improvement has been observed, not only in terms of reaching the best global solutions but also in terms of convergence speed, compared to the existing benchmark approaches.**

***Keywords—dynamic parameters, particle swarm optimization, velocity, convergence***

## I. INTRODUCTION

The Particle Swarm Optimization (PSO) is an intelligent algorithm first introduced by [1]. PSO is inspired by the social collective behavior of fishes (or birds) flocking. Imagine the following scheme: a group of fishes searches randomly for food in an area in which there is only one piece. The fishes know that food is far away on every move but Not all know where food is. The effective strategy for finding food is certainly to follow the fish nearest to the food.

PSO is a heuristic global optimization method (also an algorithm), which learned from this scheme. Each potential solution in the search space is a "fish" called "particle". Each particle learns from its own experience and from other particles during the repeated flights. The learning from own experience may be denoted as *cognitive learning*. In this case, the particle memories the *personal best* position i.e. the best solution visited so far by itself. The learning from others may be denoted as *social learning* and the particle memories in this case the best solution visited by any particle of the swarm which called as *global best* position [2].

The PSO is an iterative algorithm, initializing with a set of random solutions, called a swarm of particles, and then searches for optima by updating iterations. Each particle could be a potential solution and it has several parameters such as the velocity, the best position visited by the particle so far, and the current position [3]. In a *d*-dimensional search space, the position and the velocity of each particle is represented by *d*-dimensional vectors, which guide the moving of the particles according to the individual and the collective knowledge. The velocity and position of the $i^{th}$ particle at iteration $t$ are represented as follows:

$$v_i(t) = (v_{i1}, v_{i2}, \dots, v_{id}) \quad (1)$$

$$x_i(t) = (x_{i1}, x_{i2}, \dots, x_{id}) \quad (2)$$

In the original version of PSO, the velocity and position, at every iteration *t*, are updated as follows:

$$v_i(t+1) = v_i(t) + r_1 c_1 \big(p_i(t) - x_i(t)\big) + r_2 c_2 \left(p_g(t) - x_i(t)\right) \quad (3)$$

$$x_i(t+1) = v_i(t+1) + x_i(t) \quad (4)$$

where, $v_i(t)$ is the velocity of $i^{th}$ particle at iteration $t$, $p_i(t)$ refers to the *personal best* position of the $i^{th}$ particle, $p_g(t)$ refers to the *global best* position of the entire swarm, $r_1$ and $r_2$ are two random values in the interval [0,1] and the two acceleration constants $c_1$ *and* $c_2$ are known respectively as *cognitive* and *social* scaling parameters. They are generally fixed as 2.0. However, [4] in their study, $c_1$ *and* $c_2$ are function in terms of iteration *t* as follows:

$$c_1 = c_2 = 2 + \frac{t_{max} - t}{5 t_{max}}, \quad (5)$$

where $t_{max}$ represents the maximum number of iterations.

Since the introduction of the standard PSO, several modifications are introduced and other versions have been proposed. [5, 6], because the basic PSO [1] has its limitations (e.g. in the field of plasmonics [7]). The idea of regulating of the velocity has been presented in [8]. The authors study the effects of velocity limitations and use different maximum values to controlling the velocity.

To regulate the velocity, most common methods consist in multiplying by factors when updating the value of velocity.

The key input in this research is the integration of dynamic factors to regulate the velocity at each iteration, thus for two reasons: One is to avoid local minimum traps and to converge quickly and reliably towards the global minimum whereas the other one is to achieve a balance between exploitation and exploration. The proposed algorithm achieves better results

than the standard PSO algorithm and it is competitive when compared with other benchmark methods.

## II. Particle Swarm Optimization Versions

The success of an optimization algorithm depends generally on bringing about a compromise between global search and local search throughout the whole search space. Considering of this, an *inertia weight*, *w*, of a particle is brought into the original equation and the velocity is updated as follow:

$$v_i(t+1) = wv_i(t) + c_1 r_1\big(p_i(t) - x_i(t)\big) + c_2 r_2\left(p_g(t) - x_i(t)\right) \quad (6)$$

According to [9], most of the common PSO algorithms developed until now include the *inertia weight* coefficient, and it is commonly mentioned as the Standard PSO (SPSO). Table I summaries several existing *inertia weight* strategies reported in the literature.

Reference [20] proposes the introduction of a *constriction factor* $\chi$ to insure the convergence of the PSO. In this case, the velocity is updated according to the following equation:

$$v_i(t+1) = \chi[v_i(t) + \varphi_1 r_1\big(p_i(t) - x_i(t)\big) + \varphi_2 r_2\left(p_g(t) - x_i(t)\right)] \quad (7)$$

where $\chi$ is given by:

$$\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad (8)$$

and $\varphi = \varphi_1 + \varphi_2\ ,\ \varphi > 4$.

Thus, the equation 7 can be written as:

$$v_i(t+1) = \chi v_i(t) + \chi\varphi_1 r_1\big(p_i(t) - x_i(t)\big) + \chi\varphi_2 r_2\left(p_g(t) - x_i(t)\right)] \quad (9)$$

We can note that equation 9 becomes analogous to equation 6 with:

$$w = \chi, c_1 = \chi\varphi_1 \; and \; c_2 = \chi\varphi_2 \quad (10)$$

A comparison between the *constriction factor* and the *inertia weight* is proposed in the study of [12]. It has been concluded that the use of the constriction factor gives better quality solutions. When this factor is used, generally a value of $4.1$ for $\varphi$ is taken, and thus $\chi \approx 0.7298$, with: $\varphi_1 = \varphi_2 = 2.05 \; and \; c_1 = c_2 = 1.49445$. Note that $\chi$ can be taken as a function. However, [21] proposed a *constriction factor* based on the cosine defined in the following equation:

$$\chi = \frac{\cos\left(2\frac{\pi}{t_{max}}(t - t_{max})\right) + 2.428571}{4} \quad (11)$$

where $t_{max}$ is the maximum iteration number.

TABLE I. Different Inertia Weight Strategies

| N. | Inertia weight formula | Related description | Reference |
|---|---|---|---|
| 1 | $w = constant$ | Constant *inertia weight* | [10] |
| 2 | $w = 0.5 + \frac{rnd()}{2}$ | Random *w* | [11] |
| 3 | $w = w_{max} - \frac{w_{max} - w_{min}}{t_{max}} \cdot t$ $w_{max}$, $w_{min}$: intial and final values of *w* | Linear decreasing inertia weight | [12] |
| 4 | $w = 1.1 + \frac{p_g}{Avg(p_i)}$ $t_{max}$: maximum number of iterations | Adaptive global-local best inertia weight | [13] |
| 5 | $w = w_{min} + (w_{max} - w_{min})\left(\frac{t_{max} - t}{t_{max}}\right)^n$ $p$: global best position, $p_i$: personal best position | Linear time varying inertia weight | [14] |
| 6 | $w = w_{min} - (w_{max} - w_{min}) \cdot \lambda^{(t-1)}$ $\lambda = 0.95$ | Simulated annealing inertia weight | [15] |
| 7 | $w = w_{min} - (w_{max} - w_{min}) \cdot \frac{rank_i}{total\ population}$ $Rank_i$: position of the $i^{th}$ particle | Adaptive particle rank | [16] |
| 8 | $w = w_{min} \cdot u^{itert}$ $u$: constant value in the range [1.0001, 1.005] | Nonlinear time varying inertia weight | [17] |
| 9 | $w = w_{max} \cdot z + (w_{min} - w_{max})\frac{t_{max}}{t_{max} - t}$ $Z$: chaotic term. | Chaotic inertia weight | [18] |
| 10 | $w(t+1) = w_i(t) - (w_i(t) - 0.4) \cdot exp\left(-\left|(p_g - p_i) \cdot \frac{t}{t_{max}}\right|\right)$ | Adaptive nonlinear inertia weight | [19] |

## III. The Proposed Method

The proposed method in this paper, named Dynamic Cognitive-Social PSO "*DCS-PSO*", is characterized by the non-use of the *inertia* term and thus focusing on the *social* and *cognitive* parts of the PSO along with controlling the speed values. The idea of this strategy is based on the fact that the particles in the swarm should follow those which have shown an improvement in their respective position in the local search space. So, a leaping strategy is used in order to escape converging prematurely towards a local minimum. A new *cognitive* and *social* functions $c_1(t)$ and $c_2(t)$ are introduced. The function $c_1(t)$ is incorporated using a logarithm function defined in equation (12). The position of each particle is limited by the range of $[x_{min}, x_{max}]$, where $x_{min}$ and $x_{max}$ are respectively the lower and the upper bounds of variables.

$$c_1(t) = a(t)^{\alpha(x_{max} - x_{min})} \cdot log(\|p_i(t) - x_i(t)\| + \epsilon) \quad (12)$$

where, $\epsilon$ is a very small positive value, $a(t)$ is a *dynamic cognitive* parameter and $\alpha(x_{max} - x_{min})$ is a *dynamic regulator exponent* defined as follow:

$$\alpha(x_{max} - x_{min}) = \begin{cases} 5\,, & |x_{max} - x_{min}| \le 10 \\ 1\,, & |x_{max} - x_{min}| > 10 \end{cases} \quad (13)$$

Four initial values of $a(0)$ are tested: $(\chi\pi)^{\frac{1}{4}}$, $\cos(\sqrt{\chi\pi})$, $\sin(\sqrt{\chi\pi})$ and $e$, while the *social* function is given as $c_2(t) = (\chi\pi)^{\alpha}$.

Every time, when a particle is stuck at a local minimum, the velocity is near to zero, the particle could make a lead using the logarithm function and thus skipping a possible local optimum. To allow larger leaps at the start of iterations to explore the whole search space and small leaps for a refinement in specific solution-space regions, the idea of linear decreasing *cognitive* parameter, with a dynamic damping coefficient $d(t)$ and a damping coefficient $k$, is used as follow:

$$a(\text{t+1}) = d(t) \cdot a(t) \tag{14}$$

$$d(t+1) = k \cdot d(t) \tag{15}$$

where the initial values of $d(0)$ and $k$ are taken respectively 0.9995 and 0.99999, while $a(0)$ will take the four different values given above.

The range of the velocity $[v_{min}, v_{max}]$ is selected to be:

$$\begin{cases} v_{max} = 0.9\lceil x_{max} - x_{min} \rceil \\ v_{min} = -0.1\lceil (x_{max} - x_{min}) \rceil \end{cases} \tag{16}$$

So, the velocity equation of the proposed DCS-PSO will be updated as:

$$v_i(t+1) = \begin{cases} v_{max} & if\ h(t) > v_{max} \\ v_{min} & if\ h(t) < v_{min} \\ h(t) & otherwise \end{cases} \tag{17}$$

where

$$h(t) = r_1 c_1(t)\big(p_i(t) - x_i(t)\big) + r_2 c_2(t)\left(p_g(t) - x_i(t)\right)$$

The strategy of the proposed DCS-PSO is described in algorithm 1.

ALGORITHM 1. PSEUDOCODE OF DCS-PSO ALGORITHM

**1. Initialization**
- ***Step 1:*** Set the initial parameters
- ***Step 2:*** Initialize randomly the velocity and the position of the particles
- ***Step 3:*** Set the actual position of particles equal to the first best position values
- ***Step 4:*** Evaluate the fitness values of particles and determine the global best position.

**2. Update of velocity and position of particles**
- ***Step 5:*** Update the velocity of particles by equation (16)
- ***Step 6:*** Apply velocity limits
- ***Step 7:*** Update the position of particles by equation (4)
- ***Step 8:*** Apply position limits

**3. Update of personal and global best positions**
- ***Step 9:*** Update best and global positions
- ***Step 10:*** Update dynamic parameter and coefficients

**4. Convergence procedure**
- ***Step 11*:** Repeat step 5 to 11 until convergence or until reacting the maximum number of iterations.

## IV. RESULTS

### A. Comparison of DCS-PSO with PSO variants

In this section, DCS-PSO is compared with existing variants of PSO [22,23] on 8 numerical functions taken from the report CEC2005 [24]. Table II gives a detailed description of the 8 benchmark functions including the search spaces and the global minimums. The dimension of all those functions are fixed to 30. The swarm population size, the maximum number of iterations, and the independent runs are respectively chosen as 30, 10000 and 20. The PSO algorithm will stop when the maximum number of iterations is reached. For the comparison to be judicious, we consider, like in the study of [22], that results below $10^{-30}$ are assumed to be 0.

The data presented in Table III, with the exception of the DCS-PSO algorithm, are taken directly from the literature [22]. The comparison is based on mean fitness values, where, it can be seen that DCS-PSO has a high-quality solution in the search space compared to other PSO variants. The results in terms of precision of the solution are clearly better. It can be noted that the results for the four different initial values of a(t) tested are almost close. However, the use of a(0) = e remains very interesting and gives the best result.

### B. Comparison of DCS-PSO with other optimization algorithms

To test the performance of the proposed method, other seven common benchmark functions are tested [25]. Table II summaries the description of these functions. From Table IV, it can be observed that the proposed algorithm has performed significantly better over the standard PSO for all functions $f_9$- $f_{15}$. For functions $f_9$ et $f_{14}$, the results are comparable with these of DE [28] and P-PSO [27]. The result obtained for $f_{12}$ are better over P-PSO [27] and w-PSO [27]. The results for $f_{10}$, $f_{11}$, $f_{13}$ and $f_{15}$ were compared only with the standard PSO due to the lack of results in the literature.

The results given in Table IV demonstrates that the DCS-PSO algorithm has achieved better results and are usually very close to the global minimums of the functions tested. This is certainly due, on the one hand, to the strategy of leaping which allow to not falling into the local minima (see Fig. 1). Indeed, if a particle found stuck i.e. its personal best position and its current position have the same values; what does mean that its velocity is close to zero. In this case, the logarithmic function allows the particle to do a jumping (take a big value) and thus escape converging towards a local minimum. Note that equation (17) ensures that the value of velocity always remains admissible. On the other hand, to the advantage of the cognitive and damping parameters to achieve on a compromise between the capacity of local and global search by improving the power of PSO to overcome of the local extremums with fast convergence speed near to the global value. To verify the fast convergence, DCS-PSO is compared with the standard PSO (Fig. 2). From this figure, we can note a great advantage of DCS-PSO in offering a faster convergence and we can observe that few iterations are needed to reach best solutions, regardless the dimension [29].

TABLE II. DIFFERENT BENCHMARK FUNCTIONS USED [24, 25]

| Name of function | Equation | Range Space | $f(x^*)$ | Features |
|---|---|---|---|---|
| Sphere | $f_1(x) = \sum_{i=1}^{n} x_i^2$ | $[-100,100]$ | 0 | Multimodal, scalable, separable, differentiable |
| Rosenbrock | $f_2(x) = \sum_{i=1}^{n}(b(x_{i+1} - x_i^2)^2 + (x_i - a)^2),\ b = 100,\ a = 1$ | $[-2.048,2.048]$ | 0 | Unimodal, scalable, non-separable, differentiable |
| Schwefel | $f_3(x) = 418.9829 \cdot n - \sum_{i=1}^{N} x_i sin(\sqrt{x_i})$ | $[-500,500]$ | 0 | Unimodal, scalable, partially-separable, differentiable |
| Weierstrass | $f_4(x) = \sum_{i=1}^{n}\left(\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k(x_i + 0.5))]\right) - n\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k 0.5)]$ $a = 0.5\,, b = 3\,, k_{max} = 20$ | $[-0.5,0.5]$ | 0 | Multimodal, scalable, separable, differentiable |
| Shifted Schwefel 1.2 | $f_5(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} z_j\right)^2 - 450$ | $[-100,100]$ | -450 | Shifted, unimodal, scalable, non-separable |
| Shifted Rastrigin | $f_6(x) = \sum_{i=1}^{n}(z_i^2 - 10cos(2\pi z_i) + 10) - 330$ | $[-5,5]$ | -330 | Shifted, multimodal, scalable, separable |
| Shifted Rotated Rastrigin | $f_7(x) = \sum_{i=1}^{n}(z_i^2 - 10cos(2\pi z_i) + 10) - 330$ | $[-5,5]$ | -330 | Shifted, rotated, multimodal, scalable, non-separable |
| Shifted Rotated Weierstrass | $f_8(x) = \sum_{i=1}^{n}\left(\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k(z_i + 0.5))]\right) - n\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k 0.5)]$ $a = 0.5\,, b = 3\,, k_{max} = 20$ | $[-0.5,0.5]$ | 0 | Shifted, rotated, multimodal, scalable, non-separable |
| Ackley | $f_9(x) = -20exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) - exp\left(\frac{1}{N}\sum_{i=1}^{N} cos2\pi x_i\right) + 20 + e$ | $[-32,32]$ | 0 | Multimodal, non-convex, differentiable |
| Schwefel 2.20 | $f_{10}(x) = \sum_{i=1}^{n}\lvert x_i\rvert$ | $[-100,100]$ | 0 | Multimodal, non-convex, differentiable |
| Periodic | $f_{11}(x) = 1 + \sum_{i=1}^{n}\sin(x)^2 - 0.1e^{\sum_{i=1}^{n} x_i^2}$ | $[-10,10]$ | 0 | Multimodal, non-separable, differentiable, non-convex |
| SumSquare | $f_{12}(x) = \sum_{i=1}^{n} i x_i^2$ | $[-10,10]$ | 0 | Unimodal, convex |
| Alpine | $f_{13}(x) = \sum_{i=1}^{n}\lvert x_i sin(x_i) + 0.1x_i\rvert$ | $[-10,10]$ | 0 | Multimodal, non-convex, differentiable, non-separable, |
| Griewank | $f_{14}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | $[-600,600]$ | 0 | Unimodal, non-convex |
| Exponential | $f_{15}(x) = -exp(-0.5\sum_{i=1}^{n} x_i^2)$ | $[-1,1]$ | -1 | Unimodal, convex, differentiable, non-separable |

$z = x - o$ for $f_5$ and $f_6$ and $z = (x - o) \cdot M$ for $f_7$ and $f_8$. $o = (o_1, o_2, \ldots, o_n)$ is the (shifted) gloabal optimum and $M$ is the orthogonal square matrix.

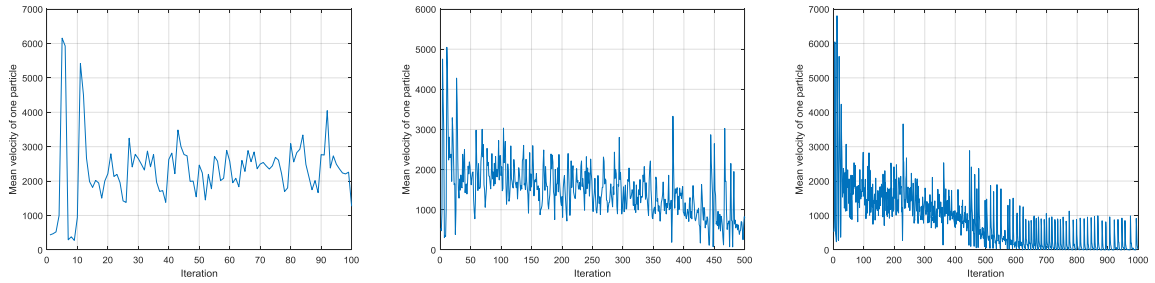
TABLE III. DCS-PSO PERFORMANCE COMPARISON WITH THE 9 PSO VARIANTS

| Function | ScPSO | CLPSO | UPSO | SG-PSO | SP-PSO | Local PSO-cf | Global PSO-cf | Local PSO-w | Global PSO-w | Proposed method: DCS-PSO (a) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | $a = (\chi\pi)^{\frac{1}{4}}$ | $a = \cos(\sqrt{\chi\pi})$ | $a = sin(\sqrt{\chi\pi})$ | $a = e$ |
| $f_1$ | **0.00e+00** | **0.00e+00** | 9.07e+03 | **0.00e+00** | 1.48e−06 | 1.34e+04 | 1.00e+03 | 1.79e+02 | 1.00e+03 | **0.00e+00** | **0.00e+00** | **0.00e+00** | **0.00e+00** |
| $f_2$ | 3.10e+01 | 4.81e+01 | 1.03e+03 | 2.08e+02 | 2.20e+01 | 1.72e+03 | 1.07e+02 | 1.27e+02 | 1.07e+02 | **0.00e+00** | 8.11e+01 | 3.23e+00 | **0.00e+00** |
| $f_3$ | 2.21e+03 | 5.46e+03 | 7.98e+03 | 4.20e+03 | 4.03e+03 | 7.36e+03 | 4.46e+03 | 6.41e+03 | 4.46e+03 | **3.81e-04** | 4.73e+03 | **3.81e-04** | **3.81e-04** |
| $f_4$ | 3.30e−01 | 2.42e−01 | 2.64e+01 | 1.60e+00 | **0.00e+00** | 2.70e+01 | 1.83e+00 | 4.37e+00 | 1.83e+00 | **0.00e+00** | **0.00e+00** | **0.00e+00** | **0.00e+00** |
| $f_5$ | 3.97e+04 | 5.49e+06 | 7.85e+06 | 5.70e+06 | 1.73e+05 | 1.01e+07 | 6.00e+06 | 5.89e+06 | 6.00e+06 | 1.29e+05 | 2.29e+05 | 1.65e+05 | **1.60 e+04** |
| $f_6$ | 3.75e+01 | 2.16e+02 | 2.48e+02 | 1.57e+02 | 1.29e+02 | 4.01e+02 | 1.38e+02 | 2.93e+02 | 1.38e+02 | 5.21e+00 | 7.04e+00 | 5.52e+00 | **4.63e+00** |
| $f_7$ | 1.74e+02 | 1.70e+02 | 3.90e+02 | 2.31e+02 | 1.72e+02 | 5.72e+02 | 2.28e+02 | 3.83e+02 | 2.28e+02 | -3.07e+01 | -2.86e+01 | -3.78e+01 | **-5.25e+01** |
| $f_8$ | **2.73e+01** | 3.27e+01 | 3.98e+01 | 2.77e+01 | 2.89e+01 | 3.65e+01 | 2.99e+01 | 3.37e+01 | 2.99e+01 | 4.01e+01 | 4.00e+01 | 3.94e+01 | 3.26e+01 |

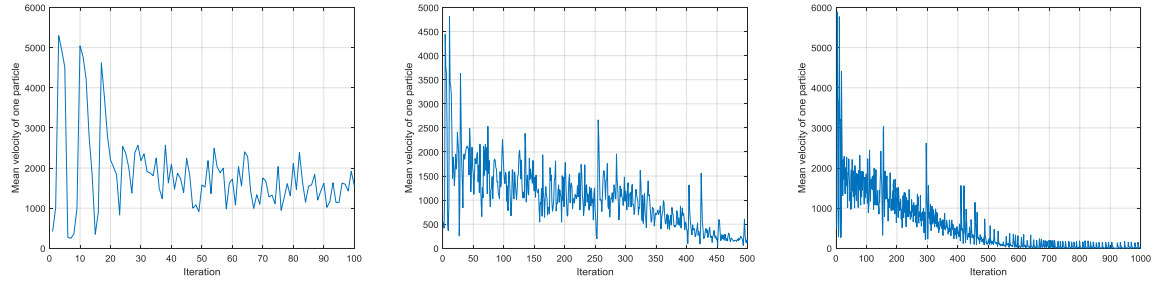
TABLE IV. DCS-PSO PERFORMANCE EVALUATION

| Function | PSO | GA [26] | BitABC [26] | P-PSO [27] | w-PSO [27] | DE [28] | FEP [28] | Proposed method: DCS-PSO (a) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | $a = (\chi\pi)^{\frac{1}{4}}$ | $a = \cos(\sqrt{\chi\pi})$ | $a = sin(\sqrt{\chi\pi})$ | $a = e$ |
| $f_9$ | 1.63e+01 | 1.22e-02 | 1.22e-04 | 4.22e-15 | 1.05e-06 | 9.70e-08 | 1.80e-02 | 4.99e-14 | 7.08e-01 | 5.00e-14 | **8.88e-16** |
| $f_{10}$ | 4.58e+02 | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. | 3.82e-13 | 4.99e+00 | 3.26e-13 | **0.00e+00** |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_{11}$ | 1.75e+00 | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. | **0.90e+00** | **0.90e+00** | **0.90e+00** | **0.90e+00** |
| $f_{12}$ | 1.87e+03 | N.A. | N.A. | 7.12e+01 | 1.59e-12 | N.A. | N.A. | **0.00e+00** | 5.00e+00 | **0.00e+00** | **0.00e+00** |
| $f_{13}$ | 1.44e+01 | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. | 9.63e-11 | 7.01e-07 | **2.86e-11** | 4.27e-02 |
| $f_{14}$ | 1.37e+02 | 7.39e-02 | 1.45e-05 | 7.02e-02 | 6.84e-02 | **0.00e+00** | 1.60e-02 | 9.65e+00 | 2.32e+01 | 1.12e+01 | **0.00e+00** |
| $f_{15}$ | -4.60e-01 | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. | -9.99 e-01 | -9.29e-01 | -9.51e-01 | **-1** |




(a) Without the dynamic parameter a(t)




(b) With the dynamic parameter a(t)

Fig. 1. Influence of the dynamic parameter strategy ($f_{10}$)

(a) $f_{15}$ D = 30 D = 100 D = 500

(b) $f_{13}$ D = 30 D = 100 D = 500

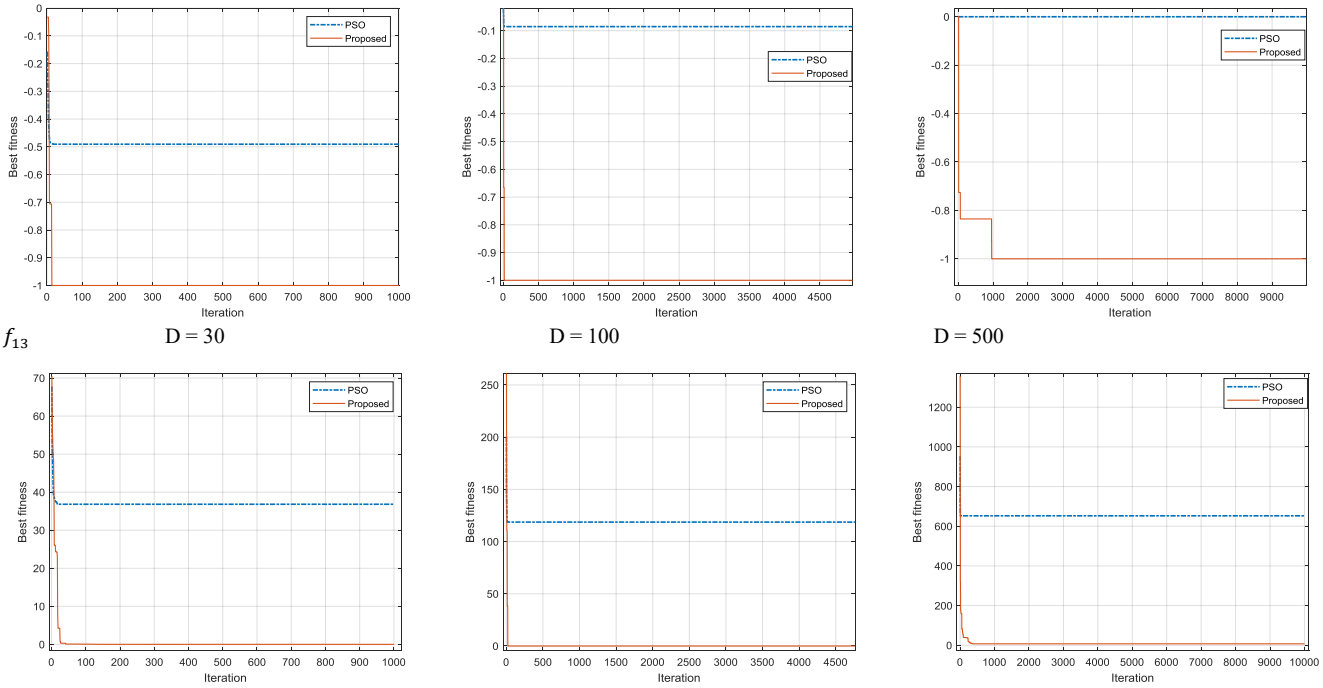



Fig. 2. Evolution of the solution with the number of iterations for different dimension sizes (D=30, 100 and 500)

## V. CONCLUSION

Dynamic Cognitive-Social Particle swarm optimization is an effective algorithm for optimizing several benchmark functions regardless the dimension. The algorithm has shown great reliability in getting a *satisfactory* solution when it was tested on different known common functions in the literature review. The Dynamic parameters along with controlling the recursive velocity equation that was integrated in the algorithm make the convergence to the global solutions faster.

In order to better model the movement of bird or fish swarm, further investigation in finding better dynamic parameters along with clustering the set of particles will improve DCS-PSO in reaching the global solution faster.

## REFERENCES


[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neuran Networks, 1995, pp. 1942–1948.

[2] Bansal, J.C.: Particle swarm optimization. Evolutionary and Swarm Intelligence Algorithms, pp. 11–23. Springer, New York (2019)

[3] Taherkhani, M., & Safabakhsh, R. (2016). A novel stability-based adaptive inertia weight for particle swarm optimization. *Applied Soft Computing*, *38*(4), 281–295.

[4] T. Yalcinoz and K. Rudion, "Economic Load Dispatch Using an Improved Particle Swarm Optimization based on functional constriction factor and functional inertia weight," 2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe), Genova, Italy, 2019, pp. 1-5.

[5] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in Proc. IEEE Congress on Evolutionary Computation *(CEC'99)*, Washington, DC, 1999, pp. 1945-1950.

[6] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 39, pp. 1362-1381, 2009.

[7] S. Kessentini, D. Barchiesi, T. Grosges, and M. L. de la Chapelle, "Particle swarm optimization and evolutionary methods for plasmonic biomedical applications," in *Proc. IEEE Congress on Evolutionary Computation (CEC'11)*, New Orleans, LA, 2011, pp. 2315-2320.

[8] Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS '95), pp. 39–43.

[9] Freitas, D.; Lopes, L.; Morgado-Dias, F. Particle Swarm Optimisation: A Historical Review Up to the Current Developments. Entropy 2020, 22, 362.

[10] Y. Shi and R. Eberhart., "A modified particle swarm optimizer", In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 69–73. IEEE, 2002.

[11] R.C. Eberhart, S. Yuhui, Tracking and optimizing dynamic systems with parti-cle swarms, in: IEEE Congress on Evolutionary Computation, Seoul, 2001, pp.94–100.

[12] Eberhart, R.C.; Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In Proceedings of the Congress on Evolutionary Computation (CEC), La Jolla, CA, USA, 16–19 July 2000; Volume 1, pp. 84–88.

[13] M.S. Arumugam, M.V.C. Rao, On the improved performances of the par-ticle swarm optimization algorithms with adaptive parameters, cross-overoperators and root mean square (RMS) variants for computing optimal controlof a class of hybrid systems, Appl. Soft Comput. 8 (2008) 324–336.

[14] Chatterjee, A.; Siarry, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Comput. Oper. Res. 2006, 33, 859–871.10.1016/j.cor.2004.08.012.

[15] W. Al-Hassan, MB Fayek, and SI Shaheen, "Psosa: An optimized particle swarm technique for solving the urban planning problem", In Computer Engineering and Systems, The 2006 International Conference on, pages 401–405. IEEE, 2007.

[16] B.K. Panigrahi, V.R. Pandi, S. Das, Adaptive particle swarm optimization approach for static and dynamic economic load dispatch, Energy Conversion and Management 49 (2008) 1407–1415.

[17] B. Jiao, Z. Lian, X. Gu, A dynamic inertia weight particle swarm optimization algorithm, Chaos, Solitons & Fractals 37 (2008) 698–705.

[18] Y. Feng, G. Teng, A. Wang, Y.M. Yao, Chaotic inertia weight in particle swarm optimization, in: Second International Conference on Innovative Computing, Information and Control (ICICIC 07), 2007, pp. 475–1475.

[19] Y. Feng, Y.M. Yao, A. Wang, Comparing with chaotic inertia weights in particle swarm optimization, in: International Conference on Machine Learning and Cybernetics, August 2007, 2007, pp. 329–333.

[20] M. Clerc (1999): The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. *Congress on Evolutionary Computation*, Washington, D. C., pp. 1951–1957.

[21] Lu Y, Liang M, Ye Z, Cao L (2015) Improved particle swarm optimization algorithm and its application in text feature selection. Appl Soft Comput 35:629–636.

[22] Koyuncu H, Ceylan R (2019) A PSO based approach: scout particle swarm algorithm for continuous global optimization problems. J Comput Des Eng 6(2):129–142.

[23] Shin, Y. B., & Kita, E. (2014). Search performance improvement of Particle Swarm Optimization by second best particle information. Applied Mathematics and Computation, 246, 346–354.

[24] P.N. Suganthan, N. Hansen, et al., Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGALReport, 2005.

[25] Hussain K, Salleh M, Cheng S and Naseem R 2017 Common Benchmark Functions for Metaheuristic Evaluation: A Review Int. Journal on Informatics Visualization 1 218-23.

[26] D. Jia, X. Duan, and M. K. Khan, "Binary artificial bee colony optimization using bitwise operation," Computers and Industrial Engineering, vol. 76, pp. 360–365, 2014.

[27] Agrawal A et al (2018) Particle swarm optimization with probabilistic inertia weight. In: Harmony search and nature inspired optimization algorithms, ICHSA 2018, pp 239–248.

[28] Mirjalili S , Lewis A . The whale optimization algorithm. Adv Eng Software 2016;95:51–67 .

[29] A. U. Rehman, A. Islam and S. B. Belhaouari, "Multi-Cluster Jumping Particle Swarm Optimization for Fast Convergence," in *IEEE Access*, vol. 8, pp.189382-189394,2020, doi: 10.1109/ACCESS.2020.3031003.