# A Self-adaptive Dynamic Particle Swarm Optimizer

J. J. Liang, L. Guo, R. Liu
School of Electrical Engineering
Zhengzhou University
Zhengzhou, China 450001
liangjing@zzu.edu.cn

B. Y. Qu
School of Electric and Information Engineering
Zhongyuan University of Technology
Zhengzhou, China 450007
qby1984@hotmail.com

***Abstract*—A self-adaptive dynamic multi-swarm particle swarm optimizer (sDMS-PSO) is proposed. In PSO, three parameters should be given experimentally or empirically. While in the sDMS-PSO a self-adaptive strategy of parameters is embedded. One or more parameters are assigned to different swarms adaptively. In a single swarm, through specified iterations, the parameters achieving the maximum number of renewal of the local best solutions are recorded. Then the information of competitive arguments is shared among all of the swarms through generating new parameters using the saved part. Multiple swarms detect the arguments in various groups in parallel during the evolutionary process which accelerates the learning speed. What's more, sharing the information of the best parameters leads to faster convergence. A local search method of the quasi-Newton is included to enhance the ability of exploitation. The sDMS-PSO is tested on the set of benchmark functions provided by CEC2015. The results of the experiment are showed in the paper.**

***Keywords—self-adaptive; dynamic; multi-swarm; PSO***

## I. INTRODUCTION

Particle swarm optimizer(PSO) is inspired by the foraging behavior of bird flocks and is proposed by Kennedy and Eberhart in 1995 [1]. As a population based global optimization method, each particle, viewed as an individual in the population, flies through the search space to explore the global best solution. Particles change their velocity according to the "cognitive" and "social" experience. That is to say, the particles' velocity is influenced by the personal best position $pbest$ and the global best position $gbest$. Then the positions of particles are updated at every generation. The two equations below express the process clearly.

$$V_i^d = \omega * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest_i^d - X_i^d) \quad (1)$$

$$X_i^d = X_i^d + V_i^d \quad (2)$$

$X_i = (X_i^1, X_i^2, X_i^3, ...., X_i^D)$ is the position of the $i$th particle for a D dimensional problem. $V_i = (V_i^1, V_i^2, V_i^3, ..., V_i^D)$ is its evolutionary velocity correspondingly. $\omega$ is the inertia weight which plays an important role to balance the global and local search abilities. $c_1$ and $c_2$ are the acceleration constants which are important parameters for velocity update. These three parameters can be fine tuned to control the population's convergence speed. $rand1_i^d$ and $rand2_i^d$ represent two uniformly distributed random numbers in the range of [0,1].

Supported by the National Natural Science Foundation of China (Grant 61473266, 61305080 and U1304602), China Postdoctoral Science Foundation (No.2014M552013)

Depending on the method of selecting leading particles, PSO algorithms can be classified into global and local versions. As (1) and (2) show, a particle's velocity and position share the information of $pbest$, $gbest$ and its previous velocity which is called the global version. While in the local one, particles' velocity update rule comes to be (3):

$$V_i^d = \omega * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (lbest_i^d - X_i^d) \quad (3)$$

where $lbest_i = (lbest_i^1, lbest_i^2, lbest_i^3, ..., lbest_i^D)$ is the best position achieved within its neighborhood which consist of a certain $lbest$ topology structure. Here, the particle is pulled by $lbest$ and $pbest$, thus the topology structure counts much during the evolutionary process. Kennedy et. al. list some commonly used structures [2]. Hu and Mendes also discuss the performance of PSO with neighborhood topology in it [3][4]. In this case, the whole population is usually divided into multiple subpopulations or swarms. But the swarms are predefined or adjusted according to the distance. So, the swarms' freedom is limited. To overcome the drawback of the existing local versions of PSO and avoid getting trapped into local optima, a dynamic multi-swarm particle swarm optimizer (DMS-PSO) is proposed [5], in which swarms are regrouped frequently to form a dynamic population. Then the DMS-PSO is improved by incorporating a local search algorithm of quasi-Newton method [6]. The similar variant called dynamically varying subswarms is used to solve dynamic economic dispatch [7]. These two DMS-PSOs release the swarms' freedom while the parameters PSO needs are set as constants empirically or experimentally. Montalvo et. al. use a self-adaptive learning strategy of parameters which provides better performance compared with the PSOs [8] without it. And further, nonlinear self-adaptive parameters for the PSO may have a faster convergence than the linear one [9]. Elsayed et. al. make the PSO adaptively select its parameters applying (4) and (5) [10]:

$$V_i^\tau = 0.5 r_1 (\tau_i + r_2 (pbest\tau_i - \tau_i) + r_3 (gbest\tau - \tau_i)) \quad (4)$$

$$\tau_i = \tau_i + V_i^\tau \quad (5)$$

978-1-4799-7492-4/15/$31.00 

where $\tau_i$ denotes one parameter of the three for individual *i*. Its update velocity $V_i^\tau$ is calculated as the sum of its previous and new velocity values. $pbest\tau_i$ is $\tau$ 's value of the personal best individual for individual *i*. $gbest\tau$ is the $\tau$ 's value of the global best individual obtained so far, and $r_1$, $r_2$ and $r_3$ are uniform random numbers between zero and one. The self-adaptive mix of PSO has a better performance when solving constrained optimization problems.

Considering what's mentioned above, in this paper we propose a self-adaptive strategy of parameters for DMS-PSO, which avoids the fussy process of preliminary parameter adjustment experiments and retaining competitive parameters at the same time. Then the algorithm is tested on the 10D, 30D, 50D and 100D problems from the set of benchmark functions provided by CEC2015 and the results are presented.

## II. SELF-ADAPTIVE DMS-PSO (SDMS-PSO)

### A. DMS-PSO

In the DMS-PSO [5], the population is divided into multiple swarms. Particles search in the local region following the local best individuals and the personal best position. Thus, using different topology structures to group the population leads to diverse performance and features. On the other hand, swarms with small size and fixed individuals tend to trap into local optimum. Clearly, regrouping these swarms frequently can break the deadlock considering the fact that iterative optimization in small-scale swarms slows down the population's convergence speed and achieves better results on multimodal problems. In this way, swarms are dynamic and exchange information obtained interactively. Moreover, the population's diversity is increased simultaneously. Regrouping period, denoted as *R*, means the population is reorganized at every *R* generations. The search process of DMS-PSO is shown in Fig.1.

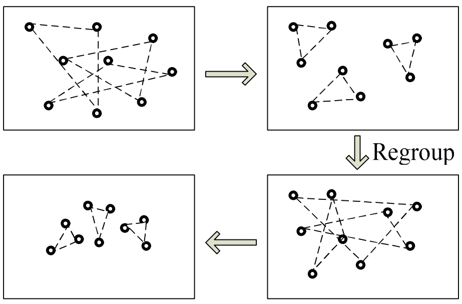


Fig. 1. Search process of DMS-PSO

The DMS-PSO's local search ability is not so satisfactory. A quasi-Newton method was introduced in [6] to strengthen the property which constructs a variant of DMS-PSO with local search (DMS-L-PSO). In our sDMS-PSO, the quasi-Newton method is also added.

### B. sDMS-PSO

Parameters of the original DMS-PSO version are set as specified constants. In fact, various evolutionary phrases have different requirements for arguments. As for $c_1$, it's expected to be larger at the early search stage and smaller at the later. $c_1$ and $\omega$ are in the same situation. $c_2$ is just the opposite. Through the variation, the "exploration" and "exploitation" of the algorithm are balanced which contributes to higher speed of convergence. Hence, adjusting the values of them constantly during the period of optimization is necessary and useful. Montalvo, Xiao and Elsayed use different adaptive strategies to settle theoretical or practical problems [8][9][10]. In this paper, to make the best of the characteristic of multi-swarms and investigate the effects of different adaptive strategies on the algorithm, we adopt two new adaptive methods of tuning parameters denoted by sDMS-PSO1 and sDMS-PSO2. The details are elaborated below.

#### 1) sDMS-PSO1

*a)* Set the bounds of $\omega$. Shi and Eberhart [11] have observed that the optimal solution can be improved by varying the value of $\omega$ from 0.9 to 0.4 for most problems. Here $[0.4, 0.9]$ is specified as the rage of $\omega$ . $c_1$ and $c_2$ are fixed to 1.49445 as the original DMS-PSO does. Generate a set of values for each swarm randomly within the intervals mentioned above.

*b)* In each swarm, find $lbest$ at every generation and update $pbest$ according to (2) and (3) simultaneously. For a group of parameters newly assigned, we set the local iterations as *LP*. In addition, record the number of renewal of $lbest$ .

*c)* An external archive is used to save the values of $\omega$ corresponding to the $lbest$ which has the maximum number of update. The length of the storage space is *LA*. Note that if the pre-defined length is reached, new parameters are generated using the saved part. In concrete terms, take the median as the mean value and 0.1 as variance to produce data following Gaussian distribution. When a eligible parameter enters the archive, delete the earliest one to keep the length changeless. Another advantage of doing this is to eliminate the accumulated effect in the previous phase and reserve the latest information.

*d)* To avoid the parameters archived falling into a small range and degenerating gradually, randomly generated values are considered when the sum of numbers of $lbest$ s' update is smaller than *LP*. This guarantees the sufficiency of using the parametric ranges.

*e)* The local search is performed to $lbest$s every *L* generations where *L* is local refining period. To balance the computational cost and better results, $lbest$s are taken in a certain proportion .

The flowchart of sDMS-PSO1 is given in Table I.

TABLE I. THE FLOWCHART OF SDMS-PSO

| |
|---|
| *m*: Each swarm's scale |
| *n*: the number of swarms |
| *R*: Regrouping period |
| *LP*: Learning period |
| *LA*: Length of archives |

$L$: local refining period
$Max\_FEs$ : Maxfitness evaluations, stop criterion
$L\_FEs$ : Max fitness evaluations using in the local search
Initialize the population with $m*n$ particles (position and velocity)
$c1 = 1.49445,\ c2 = 1.49445$
Group the population into $n$ swarms randomly, with $m$ particles in each swarm.
$FEs$=0; $gen$=0
Set $parameter\_set = \varnothing$
While $FEs < 0.95 * Max\_FEs$
$gen = gen + 1$ ;
Generate initial parameters for each swarm:
If length( $parameter\_set$ ) < $LA$ or sum( $success\_num$ )<=$LP$
$iwt = 0.5 * rand(1,n) + 0.4$
Else
$iwt = normrnd(median(parameter\_set), 0.1, 1, n)$
End
$success\_num = zeros(1,n)$;
For $j = 1 : LP$
For $i = 1 : m*n$
Find $lbest_i$
Evolve every particle according to (3) and (2)
If $X_i \in [X_{\min}, X_{\max}]^D$
Calculate the fitness value and
$FEs = FEs + 1$
Update $pbest_i$ and $lbest_i$
If $pbest_i < lbest_i$ $lbest_i = pbest_i$
$success\_num$ ( $group\_num$ )++;
End
(individual $i$ belongs to $group\_num$ th swarm)
End End End
find the maximum updating number and index of $lbest$ :
[$num$, $id$]=max( $success\_num$ )
Save the parameters assigned to $id$th swarm:
$parameter\_set = [parameter\_set;\ iwt(id)]$
If mod ($gen$, $L$)==0,
Sort the $lbest$ according to their fitness value and refine the first [0.25n] best $lbest$ using quasi-Newton method.
$FEs = FEs + [0.25n] * L\_FEs$
Update the corresponding $pbest$
End
If mod ($gen$, $R$)==0,
Regroup the swarms randomly,
End
End
While $FEs < Max\_FEs$
The population evolves as a whole.
At every generation, the parameters behave the same way.
For $i = 1 : m*n$
Evolve every particle according to (1) and (2).
Update $pbest$ and the global best value
End
End

As shown in Table I, the process of self-adaption and how to save and make use of the previously learnt parameter values is clear. At every $LP$ generations, the value of $\omega$ s generated randomly is archived if the maximum number of update of $lbest$ is gained in a certain swarm until the length of $parameter\_set$ is $LA$. Then new values are generated taking the median of $parameter\_set$ as the mean value and 0.1 as variance to produce data following Gaussian distribution. When a eligible parameter enters the archive, delete the earliest one to keep the length changeless.

### 2) *sDMS-PSO2*

sDMS-PSO2 is the same as sDMS-PSO1 except that $\omega$ and $c_2$ are adaptive parameters. The bound of $c_2$ is $[0.5, 2.5]$ . At the early stage, a random value of $c_2$ is produced for each swarm. Then the way of storage and adjustment is consistent with $\omega$ in Table I. For $c_1$ , the fixed value of 1.49445 is selected.

The self-adaptive strategies realize the parallel learning among the swarms which speeds up the process of parameter adjustment. Combining with the dynamic characteristics, selecting the values leading to fast renewal as target candidates and sharing the meaningful information obtained among all of the swarms make the two sDMS-PSOs efficient algorithms in theory.

## III. Experiments

The CEC'15 Learning-Based Benchmark Suite [12] includes 15 functions: two unimodal functions, three simple multimodal functions, three hybrid functions and seven composition functions. Experiments are conducted on all the fifteen 10D, 30D, 50D and 100D problems. All test functions are minimization problems defined as following:

$$Min\, f(\mathrm{x}),\ \mathrm{x} = [\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_D]^T \tag{6}$$

The number of swarms is changed as the dimensionality of problems varies. Each swarm's population size is 3. $R = 10$ , $LP = 10$ , $LA = 8$ , $L = 100$ , $L\_FEs = 200$ . The maximum number of fitness evaluations, $Max\_FEs$ , is set at 10000*D. To verify the validity of sDMS-PSO1 and sDMS-PSO2, the results from DMS-PSO are given for comparison. In DMS-PSO, the values of $\omega$ , $c_1$ and $c_2$ are predefined as 0.729, 1.49445 and 1.49445 correspondingly. For each function, sDMS-PSO1, sDMS-PSO2 and DMS-PSO are performed 51 single runs in the search space $[-100, 100]^D$ .

Table II to Table V list the mean and standard deviation values of sDMS-PSO1, sDMS-PSO2 and DMS-PSO for 10D problems for all the 15 functions. While Table III gives the results of 30D problems for the 15 functions. The minima of mean values are marked in bold for each function.

First of all, the performance of sDMS-PSO1 and DMS-PSO is evaluated. From Table II, we can see that sDMS-PSO1 achieves better results for the two unimodal functions. As for the three simple multimodal functions and the three hybrid functions, DMS-PSO outperforms sDMS-PSO1 as a whole. Although function 3 and function 6 don't obtain better mean and standard deviation values at the same time, other four functions gain dominated values from DMS-PSO. The competitiveness of DMS-PSO turns weak on the remaining seven functions because only four functions prevail with respect to three functions in sDMS-PSO1.

In Table III, sDMS-PSO1 displays its advantages for 30D problems. Each algorithm wins once on unimodal functions so it's not determinate which one is better. DMS-PSO still shows its effectiveness for the three simple multimodal functions, but sDMS-PSO1 works better on the three hybrid functions and the seven composition functions except function 13. Clearly, sDMS-PSO1 has more obvious advantage in high-dimension problems of hybrid and composition functions compared with both DMS-PSO and its properties in 10D functions. What's more, in terms of function 3 in Table I and in Table II, the difference of mean values is small, but the difference of standard deviation values is distinct. Thus sDMS-PSO1 shows greater robustness here. In conclusion, sDMS-PSO1 owns special characteristics and is superior to DMS-PSO.

Compared with sDMS-PSO2, the results of the two unimodal functions and the three hybrid functions from sDMS-PSO1 for 10D problems are preferable. In Table III, the mean values of the first two functions in column 2 are still smaller than those in column 4. No more conclusions can be drawn about one algorithm precedes the other on a certain category of functions. Table II and III tell that there are 16 functions achieving better values which demonstrates sDMS-PSO1 acts slightly better.

Then sDMS-PSO2 is analyzed in contrast to DMS-PSO. To 10D problems, like sDMS-PSO1, sDMS-PSO2's performance is better on function 1 and function 2, but is poorer on function 3-8. For composition functions of 30D, sDMS-PSO2 behaves outstandingly as sDMS-PSO2 does. In total, 13 superior results are gained. That is to say, sDMS-PSO2 is slightly worse than the original DMS-PSO.

TABLE II. RESULTS FOR 10D

| | sDMS-PSO1 | | sDMS-PSO2 | | DMS-PSO | |
|---|---|---|---|---|---|---|
| *Func.* | *Mean* | *Std* | *Mean* | *Std* | *Mean* | *Std* |
| 1 | **34.20724** | 111.515 | 36.1381 | 139.5449 | 89.80773 | 340.5227 |
| 2 | **111.5708** | 293.4532 | 136.1176 | 318.6136 | 271.9523 | 890.8338 |
| 3 | 19.99886 | 0.005563 | 19.99795 | 0.013196 | **19.77884** | 1.111647 |
| 4 | 4.760195 | 1.932512 | 4.491043 | 1.835116 | **3.823764** | 1.279945 |
| 5 | 130.9601 | 86.34699 | 140.7709 | 76.28062 | **107.1255** | 78.44631 |
| 6 | 276.9249 | 182.9651 | 303.7457 | 220.1335 | **239.4987** | 190.5045 |
| 7 | 0.650468 | 0.378079 | 0.673045 | 0.363671 | **0.551591** | 0.32503 |
| 8 | 61.80629 | 67.25348 | 68.55298 | 100.4694 | **41.02377** | 54.12122 |
| 9 | 100.181 | 0.037846 | **100.1714** | 0.037182 | 100.1958 | 0.060345 |
| 10 | 439.5543 | 131.626 | 397.4726 | 140.4715 | **264.4555** | 67.83025 |
| 11 | 167.4859 | 147.8587 | 159.8951 | 147.4999 | **62.08599** | 119.517 |
| 12 | 101.4498 | 0.389331 | **101.4346** | 0.363386 | 111.6785 | 0.465598 |
| 13 | 28.18082 | 2.150674 | 29.02137 | 1.958054 | **0.012004** | 0.020344 |
| 14 | **878.7974** | 848.9997 | 932.6242 | 1123.66 | 4407.516 | 1562.185 |
| 15 | 100 | 1.32E-13 | 100 | 1.46E-13 | 100 | 0 |

TABLE III. RESULTS FOR 30D

| | sDMS-PSO1 | | sDMS-PSO2 | | DMS-PSO | |
|---|---|---|---|---|---|---|
| *Func.* | *Mean* | *Std* | *Mean* | *Std* | *Mean* | *Std* |
| 1 | **0.014634** | 0.085067 | 0.019258 | 0.084187 | 0.048557 | 0.33439 |
| 2 | 265.8853 | 941.9696 | 287.5755 | 1088.74 | **178.6159** | 265.7517 |
| 3 | 19.99999 | 2.79E-05 | 19.99998 | 4.15E-05 | **19.99997** | 4.34E-05 |
| 4 | 41.43707 | 8.369055 | 42.54908 | 9.305021 | **33.39938** | 5.597478 |
| 5 | 2629.81 | 386.8353 | 2517.995 | 357.9756 | **2359.752** | 328.0307 |
| 6 | 1649.45 | 662.1082 | **1452.358** | 543.6508 | 1676.841 | 696.3675 |
| 7 | **8.782088** | 1.684147 | 9.528008 | 1.932865 | 9.426973 | 0.63972 |
| 8 | 1676.354 | 1143.393 | **1632.067** | 1334.253 | 1745.557 | 1948.861 |
| 9 | 102.9289 | 0.161441 | **102.907** | 0.186418 | 106.819 | 0.289499 |
| 10 | **5595.516** | 3833.932 | 6686.975 | 4574.415 | 7194.374 | 4326.938 |
| 11 | **316.1506** | 8.861513 | 319.8683 | 8.879635 | 399.3995 | 21.8722 |
| 12 | 105.3203 | 0.363941 | **105.1179** | 0.489711 | 109.6884 | 0.422291 |
| 13 | 103.2476 | 5.121294 | 101.4592 | 4.059766 | **0.011931** | 0.001498 |
| 14 | 20058.53 | 3002.365 | **20044.5** | 2593.841 | 32254.33 | 16768.97 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 15 | **100** | 9.14E-14 | 100 | 1.19E-13 | 100 | 1.15E-13 |

TABLE IV. RESULTS FOR 50D

| | sDMS-PSO1 | | sDMS-PSO2 | | DMS-PSO | |
|---|---|---|---|---|---|---|
| *Func.* | *Mean* | *Std* | *Mean* | *Std* | *Mean* | *Std* |
| 1 | 364.2076 | 380.7954 | **277.8535** | 288.1221 | 347.5432 | 320.8353 |
| 2 | 223.8713 | 424.6297 | 196.4153 | 474.8915 | **168.046** | 470.7742 |
| 3 | 20 | 5.66E-06 | **19.99999** | 1.99E-05 | 20 | 7.42E-06 |
| 4 | 93.87714 | 16.72806 | 91.22393 | 19.73074 | **76.68976** | 11.32125 |
| 5 | 5202.99 | 534.7014 | 4816.432 | 617.1488 | **4476.286** | 410.6065 |
| 6 | 3448.385 | 1656.579 | **3180.962** | 857.4502 | 3191.129 | 925.5362 |
| 7 | **30.27664** | 18.12424 | 34.7904 | 20.92537 | 53.85464 | 10.89095 |
| 8 | 3167.139 | 2306.093 | **2517.253** | 1504.457 | 2748.126 | 1565.928 |
| 9 | **102.3552** | 0.145117 | 104.7368 | 0.298007 | 105.7585 | 0.253438 |
| 10 | 20233.86 | 5981.666 | 17850.7 | 9575.609 | **17782.85** | 8717.771 |
| 11 | 407.7897 | 144.7658 | **388.3224** | 100.6328 | 467.5587 | 64.22384 |
| 12 | 116.822 | 0.726465 | **108.1174** | 0.479943 | 108.7235 | 0.665097 |
| 13 | **0.032012** | 0.004711 | 193.3053 | 5.992563 | 189.9 | 5.315153 |
| 14 | 45691.54 | 86.77902 | 27799.55 | 23583.85 | **22582.26** | 19358.99 |
| 15 | **100** | 1.50E-13 | 100.2492 | 1.256217 | 100 | 8.07E-13 |

For 50D problems, the results don't present any regularity in terms of the classified functions except that two thirds of single multimodal functions achieve smaller mean values from DMS-PSO. What's surprising is sDMS-PSO1 loses its all advantages on unimodal and simple multimodal functions. In general, sDMS-PSO2 shows better properties than sDMS-PSO1 and DMS-PSO in Table IV. That is to say, as the dimensionality increases, the superiority of sDMS-PSO2 becomes more remarkable. This trend extends to Table V. From results for 100D below, we can see that DMS-PSO is suited for unimodal functions and single multimodal functions. In detail, the mean and standard deviation values of DMS-PSO are almost the best in the first five rows. The two adaptive algorithms perform better on hybrid functions and composition functions. Here the stability of sDMS-PSO1 is the best by comparing the values of standard deviation.

TABLE V. RESULTS FOR 100D

| | sDMS-PSO1 | | sDMS-PSO2 | | DMS-PSO | |
|---|---|---|---|---|---|---|
| *Func.* | *Mean* | *Std* | *Mean* | *Std* | *Mean* | *Std* |
| 1 | 62649.7 | 40133.48 | 58355.27 | 34506.5 | **57086.8** | 31526.72 |
| 2 | 64.8107 | 195.9357 | 137.0888 | 767.8204 | **41.22913** | 72.45018 |
| 3 | 20 | 4.78E-06 | **19.99999** | 4.86E-06 | 19.99999 | 1.48E-05 |
| 4 | 265.9849 | 49.60402 | 258.8446 | 52.06916 | **232.098** | 23.27178 |
| 5 | 13456.18 | 1325.527 | 12850.58 | 1131.216 | **12548.23** | 998.9978 |
| 6 | 6358.167 | 1620.9 | **6023.607** | 1898.189 | 6482.01 | 2189.244 |
| 7 | **151.8084** | 27.83255 | 160.6599 | 22.28325 | 161.7601 | 6.245019 |
| 8 | 5755.764 | 3112.785 | **5652.538** | 4217.457 | 7149.291 | 5048.039 |
| 9 | 108.256 | 0.697084 | **108.242** | 0.623002 | 111.2172 | 0.548438 |
| 10 | 26468.76 | 11104.37 | **25891.07** | 8335.294 | 32619.93 | 15513.46 |
| 11 | **489.6216** | 136.7353 | 566.5952 | 180.0579 | 1434.906 | 198.9748 |
| 12 | **115.1231** | 0.656553 | 116.4336 | 0.734812 | 116.5452 | 0.664325 |
| 13 | 0.082923 | 0.017012 | 399.1072 | 12.70102 | **0.023811** | 0.236574 |
| 14 | 1525.369 | 27.37192 | **1473.636** | 2833.218 | 1538.438 | 1757.882 |
| 15 | **100.0301** | 0.123647 | 102.62 | 4.505895 | 100.2011 | 2.012032 |

Table II to Table V present the comparison of three optimizers. It's clear that the higher the dimensionality is, the stronger advantages self-adaptive algorithms show. For high-dimension problems, adaptive strategies plays an important role. Further, there are two adaptive parameters in sDMS-PSO2 which leads more ideal results than sDMS-PSO1 where only one parameter is adjusted adaptively. In a word, sDMS-PSO2 outperforms sDMS-PSO1.

To make it easy to read the results of sDMS-PSO2 for readers, here Table VI to Table IX list the best, worst, median, mean and standard deviation values of function error values for sDMS-PSO2.

TABLE VI. RESULTS OF sDMS-PSO2 FOR 10D

| sDMS-PSO2 | | | | | |
|---|---|---|---|---|---|
| *Func.* | *Best* | *Worst* | *Median* | *Std* | *Mean* |
| 1 | 7.41E-05 | 802.8353 | 0.008393 | 36.1381 | 139.5449 |
| 2 | 3.57E-05 | 1790.277 | 0.809562 | 136.1176 | 318.6136 |
| 3 | 19.90568 | 20 | 19.99999 | 19.99795 | 0.013196 |
| 4 | 0.994959 | 9.949586 | 3.979836 | 4.491043 | 1.835116 |
| 5 | 0.312272 | 354.5025 | 132.0357 | 140.7709 | 76.28062 |
| 6 | 12.16389 | 1163.166 | 277.6233 | 303.7457 | 220.1335 |
| 7 | 0.129219 | 1.120479 | 0.754096 | 0.673045 | 0.363671 |
| 8 | 1.241164 | 633.9666 | 35.30247 | 68.55298 | 100.4694 |
| 9 | 100.0858 | 100.2892 | 100.1673 | 100.1714 | 0.037182 |
| 10 | 245.3615 | 1197.927 | 375.0355 | 397.4726 | 140.4715 |
| 11 | 2.340018 | 300.6685 | 300.0747 | 159.8951 | 147.4999 |
| 12 | 100.6219 | 102.142 | 101.4875 | 101.4346 | 0.363386 |
| 13 | 25.40522 | 33.53151 | 28.40382 | 29.02137 | 1.958054 |
| 14 | 100 | 6102.312 | 100 | 932.6242 | 1123.66 |
| 15 | 100 | 100 | 100 | 100 | 1.46E-13 |

TABLE VII. RESULTS OF sDMS-PSO2 FOR 30D

| sDMS-PSO2 | | | | | |
|---|---|---|---|---|---|
| *Func.* | *Best* | *Worst* | *Median* | *Std* | *Mean* |
| 1 | 0.000513 | 0.547308 | 0.00192 | 0.019258 | 0.084187 |
| 2 | 0.000807 | 7202.31 | 0.89323 | 287.5755 | 1088.74 |
| 3 | 19.9998 | 20 | 19.99999 | 19.99998 | 4.15E-05 |
| 4 | 24.87397 | 69.64697 | 42.7832 | 42.54908 | 9.305021 |
| 5 | 1587.52 | 3066.936 | 2534.288 | 2517.995 | 357.9756 |
| 6 | 564.0676 | 2900.608 | 1359.43 | 1452.358 | 543.6508 |
| 7 | 5.829585 | 12.9769 | 9.756328 | 9.528008 | 1.932865 |
| 8 | 538.468 | 7286.164 | 1219.086 | 1632.067 | 1334.253 |
| 9 | 102.5592 | 103.3682 | 102.9067 | 102.907 | 0.186418 |
| 10 | 2613.849 | 19374.3 | 4480.148 | 6686.975 | 4574.415 |
| 11 | 306.3833 | 350.101 | 317.5699 | 319.8683 | 8.879635 |
| 12 | 103.4556 | 106.0548 | 105.1649 | 105.1179 | 0.489711 |
| 13 | 89.6766 | 109.9383 | 101.928 | 101.4592 | 4.059766 |
| 14 | 17469.59 | 26542.39 | 18945.79 | 20044.5 | 2593.841 |
| 15 | 100 | 100 | 100 | 100 | 1.19E-13 |

TABLE VIII. RESULTS OF sDMS-PSO2 FOR 50D

| sDMS-PSO2 | | | | | |
|---|---|---|---|---|---|
| *Func.* | *Best* | *Worst* | *Median* | *Std* | *Mean* |
| 1 | 1.132302 | 1211.45 | 232.8195 | 277.8535 | 288.1221 |
| 2 | 0.033883 | 1910.65 | 0.322652 | 196.4153 | 474.8915 |
| 3 | 19.99987 | 20 | 20 | 19.99999 | 1.99E-05 |
| 4 | 57.70754 | 164.1679 | 89.54619 | 91.22393 | 19.73074 |
| 5 | 3546.194 | 5979.363 | 4987.092 | 4816.432 | 617.1488 |
| 6 | 1682.457 | 5374.138 | 3161.703 | 3180.962 | 857.4502 |
| 7 | 9.799536 | 80.74706 | 25.81203 | 34.7904 | 20.92537 |
| 8 | 770.0091 | 6597.503 | 1947.758 | 2517.253 | 1504.457 |
| 9 | 104.2007 | 105.4504 | 104.7207 | 104.7368 | 0.298007 |
| 10 | 6579.781 | 38372.56 | 14994.28 | 17850.7 | 9575.609 |
| 11 | 307.4718 | 798.1342 | 369.9089 | 388.3224 | 100.6328 |
| 12 | 106.8755 | 109.5405 | 108.0712 | 108.1174 | 0.479943 |
| 13 | 179.1174 | 205.4598 | 194.1283 | 193.3053 | 5.992563 |
| 14 | 14887.22 | 74274.56 | 14978.26 | 27799.55 | 23583.85 |
| 15 | 100 | 107.1711 | 100 | 100.2492 | 1.256217 |

TABLE IX. RESULTS OF sDMS-PSO2 FOR 100D

| sDMS-PSO2 | | | | | |
|---|---|---|---|---|---|
| *Func.* | *Best* | *Worst* | *Median* | *Std* | *Mean* |
| 1 | 5583.565 | 138906.6 | 50044.55 | 58355.27 | 34506.5 |
| 2 | 0.001996 | 5492.496 | 0.03335 | 137.0888 | 767.8204 |
| 3 | 19.99998 | 20 | 20 | 19.99999 | 4.86E-06 |
| 4 | 156.2083 | 359.1786 | 252.7188 | 258.8446 | 52.06916 |
| 5 | 9970.477 | 15334.79 | 12888.48 | 12850.58 | 1131.216 |
| 6 | 3516.573 | 16635.82 | 5756.323 | 6023.607 | 1898.189 |
| 7 | 98.83138 | 180.7747 | 169.2174 | 160.6599 | 22.28325 |
| 8 | 2340.669 | 22470.18 | 4056.443 | 5652.538 | 4217.457 |
| 9 | 106.8164 | 109.4791 | 108.1881 | 108.242 | 0.623002 |
| 10 | 12118.47 | 51795.49 | 25523.9 | 25891.07 | 8335.294 |
| 11 | 339.4558 | 1124.251 | 523.1389 | 566.5952 | 180.0579 |
| 12 | 115.0946 | 118.5217 | 116.3841 | 116.4336 | 0.734812 |
| 13 | 374.1879 | 437.2364 | 398.4124 | 399.1072 | 12.70102 |
| 14 | 1473.636 | 2833.218 | 1548.805 | 1713.834 | 310.8086 |
| 15 | 100 | 117.4506 | 100 | 102.62 | 4.505895 |

To display the convergence process of the three algorithms, function 1 is selected and the average Euclidean distance of all the individuals from the *gbest* at every generation is calculated . Take 10D for example. Fig. 2 shows the difference among the three algorithms.

As we concern, the larger the average Euclidean distance from *gbest* is, the better the distribution of individuals is. Thus ideal solutions can be found. From Fig. 2, the convergence of sDMS-PSO1 is the lowest which indicates the "exploration" of the optimizer is strong. Through the analysis of Table II and III, it's certain that sDMS-PSO1 achieves the best value for function 1. It's consistent with what Fig. 2 shows. Further, in sDMS-PSO1, the individual is pulled by *lbest* and *pbest* without the information of *gbest* which slows down the convergence speed. In sDMS-PSO2, $c2$ becomes changeable. The adaptive strategy of more parameters accelerates convergence, but its speed is too fast. That is to say, the algorithm traps into local optima easily. In addition, the curve of DMS-PSO slopes down more quickly at early stage. Later the trend drops down. So the algorithm performs well.

The computational complexity for $D=10$ , $D=30$ , $D=50$ and $D=100$ is shown in TABLE X-XII. Values of the variables are computed according to [12]. Though sDMS-

PSO2 has two adaptive parameters, its computational complexity is lower than sDMS-PSO1 and DMS-PSO presents the lowest complexity in summary. The exception is the computational complexity of sDMS-PSO1 for 10D is smaller than that of DMS-PSO and sDMS-PSO2. So sDMS-PSO1 has a faster convergence speed on 10D and a lower one on 30D, 50D and 100D. Combining with the data in Table III-V, we know sDMS-PSO1 gains better solutions at the cost of higher computational overhead.

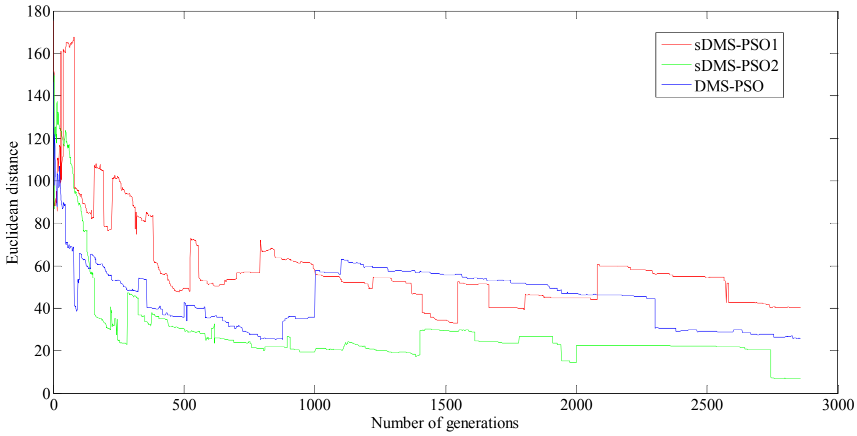


Fig. 2. The average Euclidean distance from *gbest* during convergence process

TABLE X. COMPUTATIONAL COMPLEXITY OF sDMS-PSO1

| | $T0$ | $T1$ | $\hat{T}2$ | $(\hat{T}2-T1)/T0$ |
|---|---|---|---|---|
| *D*=10 | 0.641512 | 9.240873 | 35.2289 | 40.5106 |
| *D*=30 | 0.641512 | 11.924684 | 48.0049 | 56.2425 |
| *D*=50 | 0.641512 | 20.526764 | 167.974795 | 229.8445 |
| *D*=100 | 0.641512 | 40.045548 | 218.451307 | 278.1019 |

TABLE XI. COMPUTATIONAL COMPLEXITY OF sDMS-PSO2

| | $T0$ | $T1$ | $\hat{T}2$ | $(\hat{T}2-T1)/T0$ |
|---|---|---|---|---|
| *D*=10 | 0.641512 | 9.240873 | 37.0011 | 43.2731 |
| *D*=30 | 0.641512 | 11.924684 | 48.1516 | 56.4711 |
| *D*=50 | 0.641512 | 20.526764 | 163.170928 | 222.3562 |
| *D*=100 | 0.641512 | 40.045548 | 212.046295 | 268.1177 |

TABLE XII. COMPUTATIONAL COMPLEXITY OF DMS-PSO

| | $T0$ | $T1$ | $\hat{T}2$ | $(\hat{T}2-T1)/T0$ |
|---|---|---|---|---|
| *D*=10 | 0.641512 | 9.240873 | 35.7837 | 41.3754 |
| *D*=30 | 0.641512 | 11.924684 | 46.7273 | 54.2509 |
| *D*=50 | 0.641512 | 20.526764 | 146.856877 | 196.9256 |
| *D*=100 | 0.641512 | 40.045548 | 189.683797 | 233.2587 |

## IV. CONLUSION

A self-adaptive dynamic multi-swarm particle swarm optimizer is proposed in the paper along with a variant of sDMS-PSO2. The two self-adaptive algorithms conveys the main ideas of self-adaptation. DMS-PSO is for comparison. Their advantages are described in detail. Results obtained are presented in part III and the effectiveness and efficiency of them compared with the DMS-PSO are demonstrated. We still make our best to improve the performance of them so that self-adaptive strategies are used adequately.

## *Acknowledgment*


The work is supported by National Natural Science Foundation of China (61473266, 61305080 and U1304602) and China Postdoctoral Science Foundation (No.2014M552013).


## *References*


[1] R.C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", Proc. of the Sixth Int.Symposium on Micromachine and Human Science, Nagoya, Japan. vol. 1, pp. 39-43, Oct. 1995.

[2] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in Proc. of the IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, May 2002.

[3] X. Hu, and R.C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization". In: Proceedings of IEEE congress on evolutionary computation, Hawaii, vol. 2, pp. 1677-1681, Dept. 2002.

[4] R. Mendes, J. Kennedy and J. Neves, "The fully informed particle swarm: simpler," maybe better. IEEE Trans Evol Comput, vol. 8, pp. 204-210, June 2004.

[5] J.J. Liang and P.N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," IEEE International Swarm Intelligence Symposium, pp. 124-129, June 2005.

[6] J.J. Liang and P.N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," Evolutionary Computation, 2005. The 2005 IEEE Congress on, Edinburgh, Scotland, vol. 1, pp. 522-528, Sept. 2005.

[7] A. Chowdhury, H. Zafar, B.K. Panigrahi, K.R. Krishnanand, A. Mohapatra and Z. Cui, "Dynamic economic dispatch using Lbest-PSO with dynamically varying sub-swarms," Memetic Computing, vol. 6,pp. 85-95, 2014.

[8] I. Montalvo, J. Izquierdo, R. Pérez-García and M. Herrera, "Improved performance of PSO with self-adaptive parameters for computing the optimal design of water supply systems," Engineering applications of artificial intelligence, vol. 23, pp. 727-735, 2010.

[9] R.Y. Xiao and J.H. Yu, “A newly self-adaptive strategy for the PSO,” Natural Computation, 2008. ICNC’s 08. Fourth International Conference on. IEEE, Jinan. vol. 1, pp. 396-400, Oct. 2008.

[10] S.M. Elsayed, R.A. Sarker and E. Mezura-Montes, “Self-adaptive mix of particle swarm methodologies for constrained optimization,” Information Sciences, 2014, vol. 277, pp. 216-233.

[11] Y.H. Shi and R.C. Eberhart, “A modified particle swarm optimization”, IEEE World Congress on Computational Intelligence., Anchorage, Alaska, pp. 69-73, May 1998.

[12] J.J. Liang, B.Y. Qu, P.N. Suganthan and Q. Chen, “Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter Single Objective Optimization,” Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2014.