# Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization

S. Z. Zhao[1], J. J. Liang[1], P. N. Suganthan[1], *Snr Member, IEEE* and M. F. Tasgetiren[2]

***Abstract*—In this paper, the performance of dynamic multi-swarm particle swarm optimizer (DMS-PSO) on the set of benchmark functions provided for the CEC2008 Special Session on Large Scale optimization is reported. Different from the existing multi-swarm PSOs and local versions of PSO, the sub-swarms are dynamic and the sub-swarms' size is very small. The whole population is divided into a large number sub-swarms, these sub-swarms are regrouped frequently by using various regrouping schedules and information is exchanged among the particles in the whole swarm. The Quasi-Newton method is combined to improve its local searching ability.**

## I. INTRODUCTION

Particle swarm optimizer (PSO) emulates flocking behavior of birds and herding behavior of animals to solve optimization problems. The PSO was introduced by Kennedy and Eberhart in 1995 [1][2]. Many single objective bound constrained optimization problems can be expressed as:

$$\begin{aligned} &Min\ f(x),\ x=[x_1,x_2,...,x_D] \\ &x\in[x_{\min},x_{\max}] \end{aligned} \tag{1}$$

where $D$ is the number of parameters to be optimized. The $x_{\min}$ and $x_{\max}$ are the upper and lower bounds of the search space. In PSO, each potential solution is regarded as a particle. All particles fly through the $D$ dimensional parameter space of the problem while learning from the historical information collected during the search process. The particles have a tendency to fly towards better search regions over the course of search process. The velocity $V_i^d$ and position $X_i^d$ updates of the $d^{th}$ dimension of the $i^{th}$ particle are presented below:

$$\begin{aligned} V_i^d &= w*V_i^d + c_1*rand1_i^d*(pbest_i^d - X_i^d) \\ &\quad + c_2*rand2_i^d*(gbest^d - x_i^d) \end{aligned} \tag{2}$$

$$X_i^d = X_i^d + V_i^d \tag{3}$$

where $c_1$ and $c_2$ are the acceleration constants, $rand1_i^d$ and $rand2_i^d$ are two uniformly distributed random numbers in [0,1]. $X_i=(X_1,X_2,...,X_D)$ is the position of the $i^{th}$ particle; $pbest_i=(pbest_i^1,pbest_i^2,...,pbest_i^D)$ is the best previous position yielding the best fitness value for the $i^{th}$ particle; $gbest=(gbest^1,gbest^2,...,gbest^D)$ is the best position discovered by the whole population; $V_i=(v_i^1,v_i^2,...,v_i^D)$ represents the rate of position change (velocity) for particle $i$. $w$ is the inertia weight used to balance between the global and local search abilities.

In the PSO domain, there are two main variants: global PSO and local PSO. In the local version of the PSO, each particle's velocity is adjusted according to its personal best position **pbest** and the best position **lbest** achieved so far within its neighborhood. The global PSO learns from the personal best position **pbest** and the best position **gbest** achieved so far by the whole population. The velocity update of the local PSO is:

$$\begin{aligned} V_i^d &= w*V_i^d + c_1*rand1_i^d*(pbest_i^d - X_i^d) \\ &\quad + c_2*rand2_i^d*(lbest_i^d - x_i^d) \end{aligned} \tag{4}$$

where $lbest=(lbest_i^1,lbest_i^2,...,lbest_i^D)$ is the best position achieved within $i^{th}$ particle's neighborhood.

Focusing on improving the local variants of the PSO, different neighborhood structures were proposed and discussed [3][4][5][6][7]. Except these local PSO variants, some variants that use multi-swarm [8], subpopulation [9] can also be regarded as the local PSO variants if we view the sub-groups as special neighborhood structures. In the existing local versions of PSO with different neighborhood structures and the multi-swarm PSOs, the swarms are predefined or dynamically adjusted according to the distance. Hence, the freedom of sub-swarms is limited. In [11], a dynamic multi-swarm particle swarm optimizer (DMS-PSO) was proposed whose neighborhood topology is dynamic and randomized. DMS-PSO gives a better performance on multimodal problems than some other PSO variants, but the local search performance is not satisfactory. In this paper, we

Manuscript received December 14, 2007. Authors with Nanyang Technological University were supported by the A*Star (Agency for Science, Technology and Research) under the grant # 052 101 0020.

[1]Authors affiliation: Nanyang Technological University, School of Electrical and Electronic Engineering, Singapore, 639798 (phone: 65-67905404; fax: 65-67933318; e-mails: ZH0032NG@ntu.edu.sg , epnsugan@ntu.edu.sg).

[2]Mehmet Fatih Tasgetiren is with the Department of Operations Management and Business Statistics, Muscat, Sultanate of Oman (corresponding author; phone: +968-92992057; e-mail: mfatih@squ.edu.om



978-1-4244-1823-7/08/$25.00©2008 IEEE

improved this DMS-PSO by combining a classic local search algorithm, Quasi-Newton method, and test this DMS-PSO variant on the test functions provided in CEC2008 large scale optimization special session and competition.

## II. DMS-PSO WITH LOCAL SEARCH

The dynamic multi-swarm particle swarm optimizer was constructed based on the local version of PSO with a new neighborhood topology [11]. Many existing evolutionary algorithms require larger populations, while PSO needs a comparatively smaller population size. A population with three to five particles can achieve satisfactory results for simple problems. According to many reported results on the local version of PSO [3][4], PSO with small neighborhoods performs better on complex problems. Hence, in order to slow down convergence speed and to increase diversity to achieve better results on multimodal problems, in the DMS-PSO, small neighborhoods are used. The population is divided into small sized swarms. Each sub-swarm uses its own members to search for better regions in the search space.

Since the small sized swarms are searching using their own best historical information, they are easy to converge to a local optimum because of PSO's speedy convergence behavior. Further, unlike a co-evolutionary PSO, we allow maximum information exchange among the particles to enhance the diversity of the particles. Hence, a randomized regrouping schedule is introduced to make the particles have a dynamically changing neighborhood structures. Every $R$ generations, the population is regrouped randomly and starts searching using a new configuration of small swarms. Here $R$ is called regrouping period. In this way, the information obtained by each swarm is exchanged among the swarms. Simultaneously the diversity of the population is increased. The new neighborhood structure has more freedom when compared with the classical neighborhood structure. It is not surprising that it performs better on complex multimodal problems.

For example, suppose we have three swarms with three particles in each swarm. First, the nine particles are divided into three swarms randomly. Then the three swarms use their own particles to search for better solutions. In this period, they may converge to near a local optimum. Then the whole population is regrouped into new swarms. The new swarms begin their search. This process is continued until a stop criterion is satisfied. With the randomly regrouping schedule, particles from different swarms are grouped in a new configuration so that each small swarms search space is enlarged and better solutions are possible to be found by the new small swarms. This procedure is shown in Figure 1.

In this paper, we introduce two concepts to modify the DMS-PSO for seeking better performance. In steady step, we just use the basic updating equation, when updating the positions of the particles, half of the dimensions are kept the same as its best historical position, $pbest$, to make better use of the particles' historical information to improve its global search ability.

A larger diversity and a faster convergence velocity are always a trade-off problem. Since we achieve a larger diversity using DMS-PSO, at the same time, we lose the fast convergence velocity. In order to alleviate this weakness and give a better search in the better local areas, a local search is added into DMS-PSO:

1) Every $L$ generations, sort the groups according to their fitness value and refine the $lbest$ positions of the best 25% groups using the Quasi-Newton method.

2) In the end of the search, the best solution achieved so far is refined using Quasi-Newton method

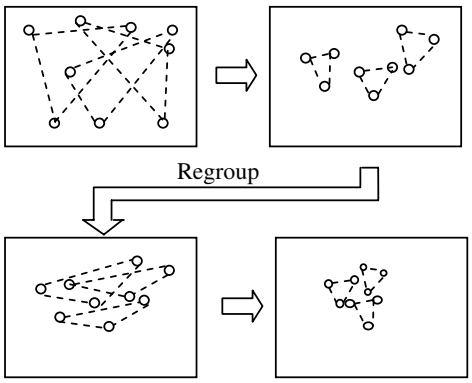


Fig 1. DMS-PSO's Search

In order to achieve better results on multi-modal problems, DMS-PSO is designed to make the particles have a large diversity, and consequently the convergence speed will be slow. Even after the global region is found, the particles will not converge very fast in order to avoid premature convergence. How to maintain the diversity and get the good result at the same time is a problem. Thus in DMS-PSO, a local search phase is added. Combining with the local search schedule, the DMS-PSO is modified to DMS-PSO with local search (DMS-L-PSO). In our algorithm, in order to constrain the particles within the range, we calculate the fitness value of a particle and update its *pbest* only if the particle is in the range. Since all exemplars are within the range, the particle will eventually return to the search range.

Every L generations, the **pbest**s of five randomly chosen particles will be used as the starting points and the BFGS Quasi-Newton method is employed to do the local search. Then the five refined solutions will be obtained. We calculate the Euclidean distance to all the **pbest**s for each refined solution and replace the nearest ones with the refined solutions if the refined solution is better. In the end of the search, every 5*$L$ generations, the best solution achieved so far is refined using the BFGS Quasi-Newton method.

An illustration for the local search phase for a swarm of 10 particles is given in Figure 2. Five **pbest**s " ● " $\mathbf{pbest}_2$, $\mathbf{pbest}_4$, $\mathbf{pbest}_6$, $\mathbf{pbest}_8$ and $\mathbf{pbest}_{10}$ are randomly chosen as the start points for the local search and 3 local optima $\mathbf{x}_1$*,$\mathbf{x}_2$* and $\mathbf{x}_3$* are achieved after the local search. The nearest three **pbest**s " ○ " $\mathbf{pbest}_1$, $\mathbf{pbest}_4$, $\mathbf{pbest}_8$ are replaced by $\mathbf{x}_1$*,$\mathbf{x}_2$* and $\mathbf{x}_3$* " ★ " respectively provided the refined solutions are better.

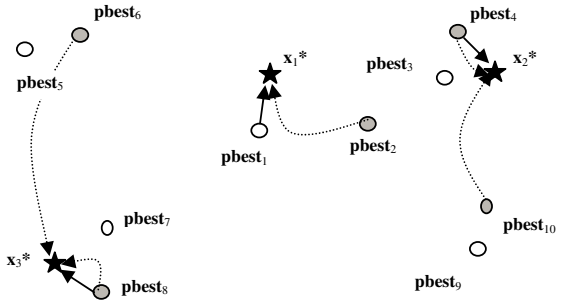


Fig 2. Illustration of local search phase for a population with 10 particles

Quasi-Newton methods build up curvature information in each generation to formulate a quadratic model problem as below:

$$\min \ \mathbf{x}^T\mathbf{H}\mathbf{x}+\mathbf{c}^T\mathbf{x}+b \tag{5}$$

Here **H** is the Hessian matrix, a positive definite symmetric matrix, c is a constant vector and b is a constant. If x* is the optimal solution of this model, x* will satisfy

$$\nabla f(\mathbf{x}^*)=\mathbf{H}\mathbf{x}^*+\mathbf{c}=0 \tag{6}$$

Then

$$\mathbf{x}=-\mathbf{H}^{-1}\mathbf{c} \tag{7}$$

Quasi-Newton methods approximate **H** using the observed behavior of $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ to build up curvature information with an updating method to avoid calculating **H** numerically. An effective formula of Broyden [4], Fletch [5], Goldfarb [6] and Shanno [7] (BFGS) is employed in the BFGS Quasi-Newton method.

$$\mathbf{H}_{k+1}=\mathbf{H}_k+\frac{\mathbf{q}_k\mathbf{q}_k^{\ T}}{\mathbf{q}_k^{\ T}\mathbf{s}_k}-\frac{\mathbf{H}_k^T\mathbf{s}_k^T\mathbf{s}_k\mathbf{H}_k}{\mathbf{s}_k^T\mathbf{H}_k\mathbf{s}_k} \tag{8}$$

where

$$\mathbf{s}_k=\mathbf{x}_{k+1}-\mathbf{x}_k$$
$$\mathbf{q}_k=\nabla f(\mathbf{x}_{k+1})-\nabla f(\mathbf{x}_k)$$

$\mathbf{H}_0$ is set to the identity matrix **I**. The inverse Hessian $\mathbf{H}^{-1}$ can also be approximated using DFP formula of Davidon [8], Fletcher and Powell [9]. At each generation, a line search is performed in the direction

$$\mathbf{d}=-\mathbf{H}_k^{-1}\cdot\nabla f(\mathbf{x}_k) \tag{9}$$

$$\mathbf{x}_{k+1}=\mathbf{x}_k+\boldsymbol{\alpha}_k\mathbf{d} \tag{10}$$

To simplify, the function "*fminunc*" in Matlab 7.1 is employed to realize the BFGS Quasi-Newton local search. Assuming the gradient information is not available for all problems, in all experiments， no gradient information is supplied to the Quasi-Newton method. Figure 3 is Sub-flowchart for DMS-L-PSO (local search phase).

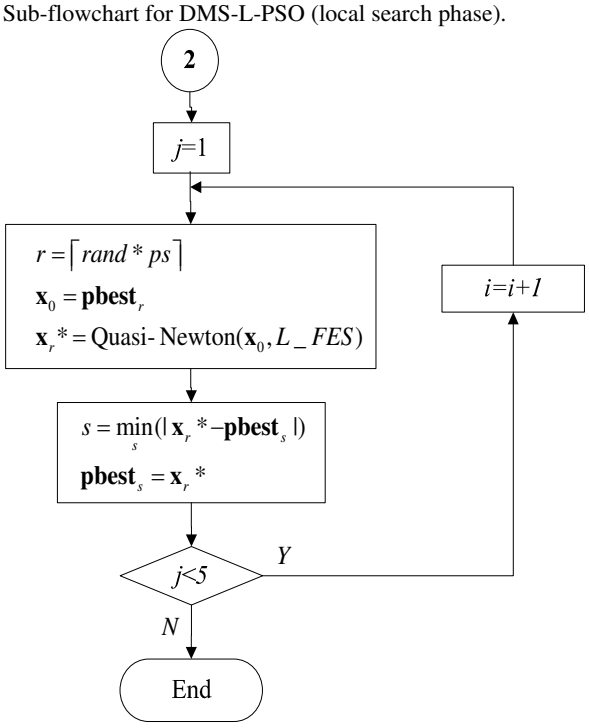


Fig 3. Sub-flowchart 2 for DMS-L-PSO (local search phase)

Every generation, the distances among the **pbest** of all particles are checked, if the maximum distance for each dimension is smaller than a predefined value or some other convergence criteria are satisfied (e.g. 90 percent of maximum fitness evaluations have been used), the convergence phase is started. In the convergence phase, all particles form a single swarm to become a global PSO version. The flowchart of the convergence phase is presented in Figure 4. The flowchart of DMS-L-PSO is given in Figure 5.

## III. EXPERIMENTS

The new proposed set of test problems includes 7 CEC'08 Test Functions with different problems, and two of them are Unimodal problems and other five are multimodal problems [10]. There is the summary of those 7 test functions:

Unimodal Functions (2):
F1: Shifted Sphere Function
F2: Shifted Schwefel's Problem 2.21
Multimodal Functions (5):
F3: Shifted Rosenbrock's Function

F4: Shifted Rastrigin's Function
F5: Shifted Griewank's Function
F6: Shifted Ackley's Function
F7: FastFractal "DoubleDip" Function

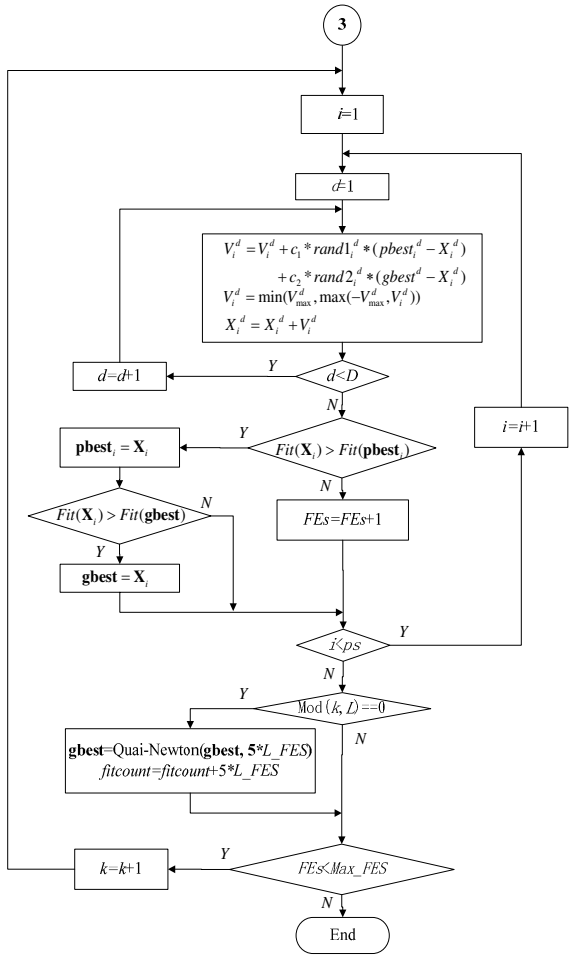


Fig 4. Sub-flowchart 3 for DMS-L-PSO (convergence phase)

Experiments were conducted on all seven minimization problems with 100-, 500- and 1000 Dimensions. To solve these problems, the number of sub-swarms is set at 30, 150 and 300 for each kind of dimensions respectively. Each sub-swarm has 3 particles. Hence, the population size is 90, 450 and 900 respectively. $\omega$ =0.729, $c_1 = c_2$ =1.49445, $R$=10, $L$=100, $L\_FES$ =200. $Max\_FES$ is set at 500,000 for 100-$D$, 2,500,000 for 500-$D$ and 5,000,000 for 1000-$D$. $V\max$ restricts particles' velocities, where $V\max$ is equal to 20% of the search range.

For each problem, the DMS-L-PSO with the local search is run 25 times. The error value $(f(x) - f(x^*))$ was recorded after $1/100 * Max\_FES$ , $1/10 * Max\_FES$ and at termination due to $Max\_FES$ for each run. For each function, we sorted the error values in 25 runs from the smallest (best) to the largest (worst). We present the following: 1st (best), 7th, 13th (median), 19th, 25th (worst) function values, Mean and STD for the 25 runs. Convergence Graphs for each problem of 1000-Dimension are also presented. The graph shows the median performance of each 25 runs with termination by the $Max\_FES$. The semi-log graphs should show $\log 10(f(x) - f(x^*))$ vs $FES$ for each problem.

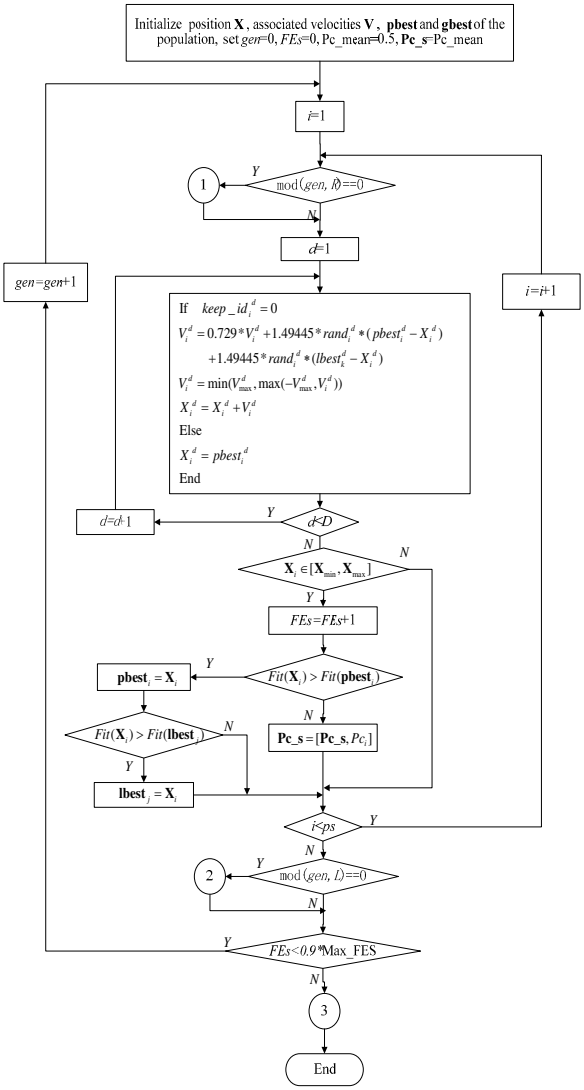


Fig 5. The flowchart of DMS-L-PSO

The computer system configuration is:
**System:** Windows XP (SP1)
**CPU:** Pentium(R) 4 3.00GHz
**RAM:** 1 G
**Language:** Matlab 7.1

From the results of 100 dimensions, we can observe that among those 7 test functions, problems 1, 5 and 6 are comparatively easy for our algorithm. Problems 3 and 4 appear to be the hardest problems. As we do not know the exact optimal solution for problem 7, it is impossible to comment on the quality of solution for problem 7. From the Convergence Graphs in Figure 6, we can also have the same observations.

We can observe almost a similar performance from the results of 500 dimensions as for the 100 dimensions. Problem 3 has the largest function error values, while problems 1 and 5 are comparatively easiest. In Figure 8, we can see the obvious slower convergence speed as in results of 100-dimensions. It is obvious that DMS_L_PSO still performs very well in finding the optimal solutions for problems 1 and 5 in 1000-dimensions.

TABLE I
ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH D=100, MAX_FES=500000

| FEs \ Problems | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 5.00e+3 | $1^{th}$ (Best) | 1.0735e+05 | 8.2211e+01 | 1.9158e+10 | 1.1471e+03 | 8.1223e+02 | 1.9012e+01 | -8.8791e+02 |
| | $7^{th}$ | 1.0912e+05 | 8.9001e+01 | 2.0583e+10 | 1.1680e+03 | 8.6456e+02 | 1.9166e+01 | -8.7100e+02 |
| | $13^{th}$ (Median) | 1.1462e+05 | 9.0541e+01 | 2.1761e+10 | 1.2008e+03 | 8.4553e+02 | 1.9251e+01 | -8.5527e+02 |
| | $19^{th}$ | 1.1515e+05 | 9.1592e+01 | 2.2877e+10 | 1.2275e+03 | 8.8612e+02 | 1.9875e+01 | -8.4785e+02 |
| | $25^{th}$ (Worst) | 1.1810e+05 | 9.8193e+01 | 2.6311e+10 | 1.2688e+03 | 9.3996e+02 | 2.0548e+01 | -8.3522e+02 |
| | Mean | 1.1287e+05 | 9.0372e+01 | 2.2405e+10 | 1.2057e+03 | 8.9275e+02 | 1.9900e+01 | -8.5833e+02 |
| | Std | 4.4762e+03 | 4.1231e+00 | 2.4128e+09 | 3.7057e+01 | 6.6774e+01 | 9.1722e-01 | 1.7137e+01 |
| 5.00e+4 | $1^{th}$ (Best) | 2.0912e+03 | 3.3263e+01 | 3.3234e+07 | 6.3967e+02 | 1.7100e_01 | 6.2364e+00 | -1.0014e+03 |
| | $7^{th}$ | 2.4128e+03 | 3.4974e+01 | 3.6745e+07 | 6.7594e+02 | 1.7456e+01 | 6.3568e+00 | -9.8311e+02 |
| | $13^{th}$ (Median) | 2.5546e+03 | 3.5552e+01 | 4.5233e+07 | 6.8801e+02 | 1.7813e+01 | 6.4040e+00 | -9.7933e+02 |
| | $19^{th}$ | 2.6923e+03 | 3.7241e+01 | 5.0561e+07 | 6.9894e+02 | 1.7921e+01 | 7.7145e+00 | -9.7527e+02 |
| | $25^{th}$ (Worst) | 3.0256e+03 | 4.1142e+01 | 5.5451e+07 | 7.3081e+02 | 1.8049e+01 | 7.8192e+00 | -9.5798e+02 |
| | Mean | 2.5587e+03 | 3.6361e+01 | 4.3544e+07 | 6.8775e+02 | 1.7931e+01 | 7.1116e+00 | -9.7774e+02 |
| | Std | 3.4345e+02 | 1.9792e+00 | 6.9457e+06 | 2.0394e+01 | 1.6674e-01 | 1.0007e+00 | 1.3783e+01 |
| 5.00e+5 | $1^{th}$ (Best) | 0 | 1.9313e+00 | 1.2569e+02 | 1.4134e+02 | 0 | 0 | -1.1581e+03 |
| | $7^{th}$ | 0 | 3.1831e+00 | 2.3065e+02 | 1.6622e+02 | 0 | 0 | -1.1505e+03 |
| | $13^{th}$ (Median) | 0 | 3.7632e+00 | 2.5792e+02 | 1.8312e+02 | 0 | 0 | -1.1437e+03 |
| | $19^{th}$ | 0 | 3.9864e+00 | 3.2920e_02 | 2.0105e+02 | 0 | 0 | -1.1419e+03 |
| | $25^{th}$ (Worst) | 0 | 4.9041e+00 | 4.7901e+02 | 2.1094e+02 | 0 | 0 | -1.1292e+03 |
| | Mean | 0 | 3.6452e+00 | 2.8301e+02 | 1.8294e+02 | 0 | 0 | -1.1448e+03 |
| | Std | 0 | 7.3022e-01 | 9.4020e+02 | 2.1638e+01 | 0 | 0 | 8.4836e+00 |

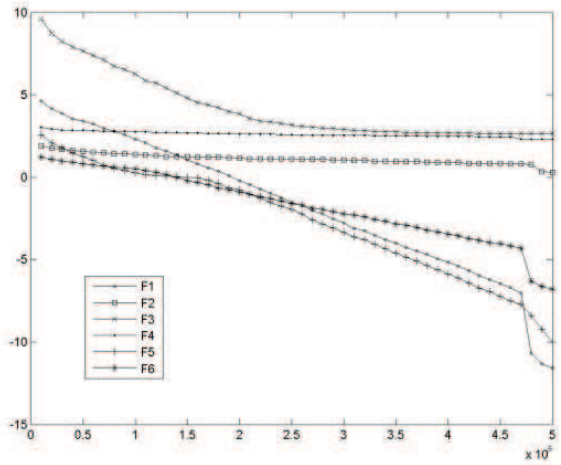


Fig 6. Convergence Graphs for problems 1-6 with D=100

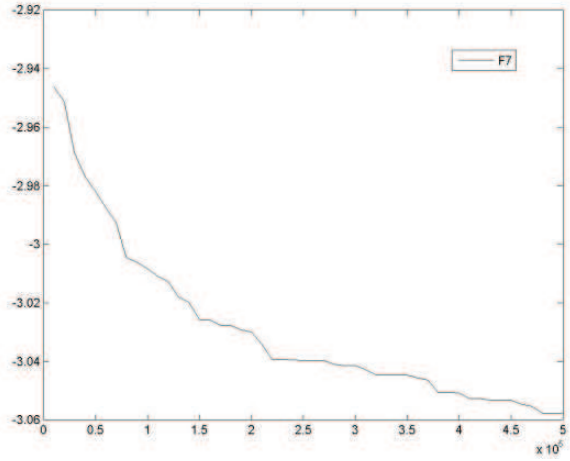


Fig 7. Convergence Graph for problem 7 with D=100

TABLE II
ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH D=500, MAX_FES=2500000

| FEs \ Prob | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 2.50e+4 | $1^{th}$ (Best) | 1.4203e+06 | 1.1030 e+02 | 8.2255e+11 | 8.7512e+03 | 1.1954e+04 | 2.0963e+01 | -3.4961e+03 |
| | $7^{th}$ | 1.4385e+06 | 1.4654 e+02 | 8.6103e+11 | 8.8655e+03 | 1.2362e+04 | 2.0978e+01 | -3.4387e+03 |
| | $13^{th}$ (Median) | 1.4532e+06 | 1.4857 e+02 | 8.7534e+11 | 8.9127e+03 | 1.2493e+04 | 2.1008e+01 | -3.4198e+03 |
| | $19^{th}$ | 1.4712e+06 | 1.4988 e+02 | 8.8800e+11 | 8.9669e+03 | 1.2629e+04 | 2.1024e+01 | -3.3945e+03 |
| | $25^{th}$ (Worst) | 1.4853e+06 | 1.5175e+02 | 9.3494e+11 | 9.0760e+03 | 1.2925e+04 | 2.1061e+01 | -3.3607e+03 |
| | Mean | 1.4529e+06 | 1.4636 e+02 | 8.7066e+11 | 8.9212e+03 | 1.2530e+04 | 2.1006e+01 | -3.4198e+03 |
| | Std | 3.2486e+04 | 9.0477 e+00 | 3.2122e+10 | 8.9283e+01 | 2.4108e+02 | 2.6115e-02 | 3.3293e+01 |
| 2.50e+5 | $1^{th}$ (Best) | 1.6322e+05 | 7.9682 e+01 | 2.5945e+10 | 5.8057e+03 | 1.3566e+03 | 1.5193e+01 | -3.7574e+03 |
| | $7^{th}$ | 1.6474e+05 | 9.7450 e+01 | 2.9035e+10 | 5.8363e+03 | 1.4034e+03 | 1.5384e+01 | -3.7219e+03 |
| | $13^{th}$ (Median) | 1.6559e+05 | 9.8631 e+01 | 3.0633e+10 | 5.9062e+03 | 1.4097e+03 | 1.5464e+01 | -3.7108e+03 |
| | $19^{th}$ | 1.6844e+05 | 9.9248 e+01 | 3.1912e+10 | 5.9192e+03 | 1.4645e+03 | 1.5543e+01 | -3.6966e+03 |
| | $25^{th}$ (Worst) | 1.6928e+05 | 9.9758 e+01 | 3.4386e+10 | 5.9952e+03 | 1.4991e+03 | 1.5750e+01 | -3.6748e+03 |
| | Mean | 1.6600e+05 | 9.7498 e+01 | 3.0624e+10 | 5.8933e+03 | 1.4295e+03 | 1.5482e+01 | -3.7116e+03 |
| | Std | 3.0110e+03 | 4.4193 e+00 | 2.0435e+09 | 4.8894e+01 | 4.5005e+01 | 1.1843e-01 | 2.1624e+01 |
| 2.50e+5 | $1^{th}$ (Best) | 0 | 6.1804 e+01 | 4.0391e+07 | 1.4508e+03 | 0 | 1.7357e+00 | -4.2216e+03 |
| | $7^{th}$ | 0 | 6.8660 e+01 | 4.1084e+07 | 1.5170e+03 | 0 | 1.9369e+00 | -4.2098e+03 |
| | $13^{th}$ (Median) | 0 | 6.9415 e+01 | 4.4443e+07 | 1.5703e+03 | 0 | 1.9987e+00 | -4.2010e+03 |
| | $19^{th}$ | 0 | 6.9572 e+01 | 5.1467e+07 | 1.6743e+03 | 0 | 2.0413e+00 | -4.1916e+03 |
| | $25^{th}$ (Worst) | 0 | 7.1363 e+01 | 5.9443e+07 | 1.8400e+03 | 0 | 2.1579e+00 | -4.1702e+03 |
| | Mean | 0 | 6.8934 e+01 | 4.6712e+07 | 1.6088e+03 | 0 | 2.0022e+00 | -4.1999e+03 |
| | Std | 0 | 2.0144 e+00 | 5.8714e+06 | 1.0429e+02 | 0 | 9.6612e-02 | 1.2949e+01 |

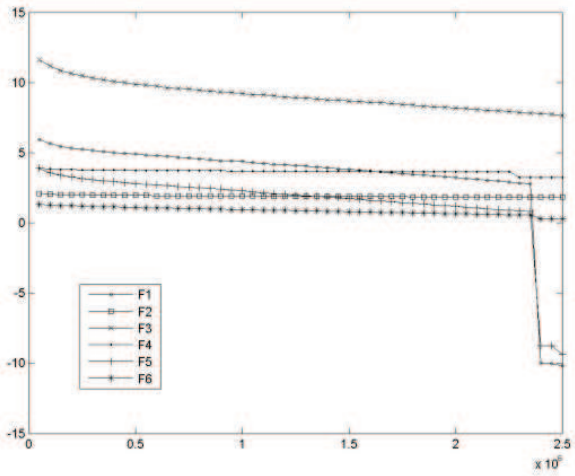


Fig 8. Convergence Graphs for problems 1-6 with D=500

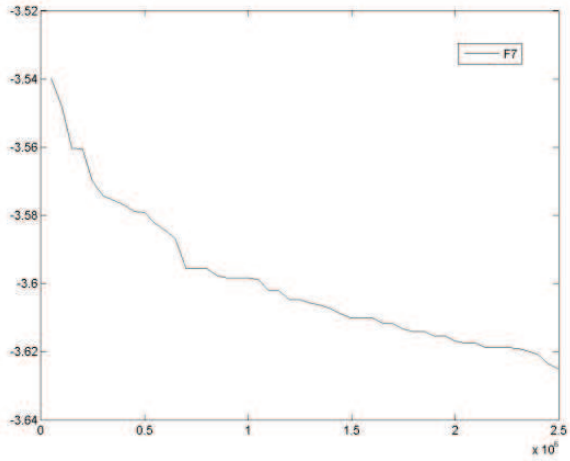


Fig 9. Convergence Graph for problem 7 with D=500

TABLE III
ERROR VALUES ACHIEVED FOR PROBLEMS 1-7, WITH D=1000, MAX_FES=5000000

| FEs \ Prob | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 5.00e+4 | $1^{th}$ (Best) | 4.2415e+05 | 1.7589e+02 | 3.8097e+12 | 2.1308e+04 | 5.9778e+03 | 2.1384e+01 | -6.5544e+03 |
| | $7^{th}$ | 4.6974e+05 | 1.7983 e+02 | 4.0226e+12 | 2.1630e+04 | 6.6861e+03 | 2.1398e+01 | -6.4748e+03 |
| | $13^{th}$ (Median) | 4.7568e+05 | 1.8104 e+02 | 4.1970e+12 | 2.1752e+04 | 6.9514e+03 | 2.1406e+01 | -6.4463e+03 |
| | $19^{th}$ | 4.8863e+05 | 1.8169 e+02 | 4.3502e+12 | 2.1876e+04 | 7.6330e+03 | 2.1415e+01 | -6.4050e+03 |
| | $25^{th}$ (Worst) | 4.8956e+05 | 1.8322 e+02 | 4.6643e+12 | 2.2321e+04 | 8.3880e+03 | 2.1428e+01 | -6.3887e+03 |
| | Mean | 4.6455e+05 | 1.8063 e+02 | 3.3139e+12 | 2.1794e+04 | 7.1136e+03 | 2.1407e+01 | -6.4433e+03 |
| | Std | 2.8067e+04 | 1.7392 e+00 | 1.5800e+12 | 2.4889e+02 | 7.8121e+02 | 1.1615e-02 | 4.8307e+01 |
| 5.00e+5 | $1^{th}$ (Best) | 0 | 1.1441 e+02 | 3.6874e+11 | 1.4185e+04 | 0 | 1.9084e+01 | -6.7421e+03 |
| | $7^{th}$ | 0 | 1.1611 e+02 | 3.7995e+11 | 1.4294e+04 | 0 | 1.9121e+01 | -6.7201e+03 |
| | $13^{th}$ (Median) | 0 | 1.1696 e+02 | 3.9064e+11 | 1.4350e+04 | 0 | 1.9152e+01 | -6.7124e+03 |
| | $19^{th}$ | 0 | 1.1772 e+02 | 4.1284e+11 | 1.4398e+04 | 0 | 1.9182e+01 | -6.7022e+03 |
| | $25^{th}$ (Worst) | 0 | 1.1850 e+02 | 4.2571e+11 | 1.4453e+04 | 0 | 1.9235e+01 | -6.6619e+03 |
| | Mean | 0 | 1.1684 e+02 | 2.9407e+11 | 1.4341e+04 | 0 | 1.9156e+01 | -6.7071e+03 |
| | Std | 0 | 1.0742 e+00 | 1.6855e+11 | 7.3572e+01 | 0 | 4.0612e-02 | 2.0703e+01 |
| 5.00e+6 | $1^{th}$ (Best) | 0 | 8.9961 e+01 | 8.2789e+09 | 3.5492e+03 | 0 | 7.5752e+00 | -7.5372e+03 |
| | $7^{th}$ | 0 | 9.1155 e+01 | 8.9417e+09 | 3.7366e+03 | 0 | 7.6952e+00 | -7.5254e+03 |
| | $13^{th}$ (Median) | 0 | 9.1674 e+01 | 9.0483e+09 | 3.8244e+03 | 0 | 7.7573e+00 | -7.5119e+03 |
| | $19^{th}$ | 0 | 9.2074 e+01 | 9.2050e+09 | 3.9971e+03 | 0 | 7.8228e+00 | -7.4938e+03 |
| | $25^{th}$ (Worst) | 0 | 9.2816 e+01 | 9.4508e+09 | 4.1066e+03 | 0 | 7.8966e+00 | -7.4851e+03 |
| | Mean | 0 | 9.1519 e+01 | 8.9849e+09 | 3.8399e+03 | 0 | 7.7566e+00 | -7.5086e+03 |
| | Std | 0 | 7.1364 e-01 | 4.3870e+08 | 1.7091e+02 | 0 | 8.9212e-02 | 1.6399e+01 |

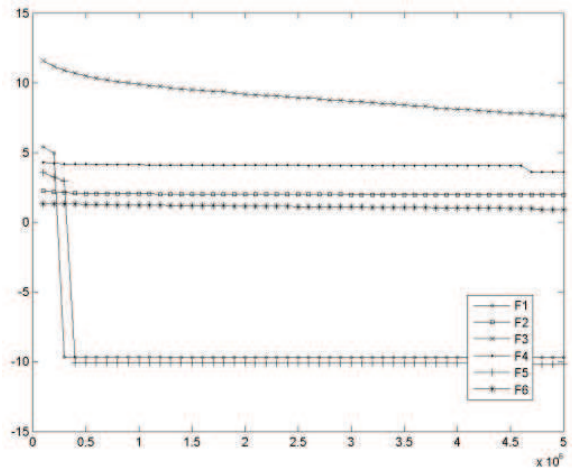


Fig 10. Convergence Graphs for problems 1-6 with D=1000

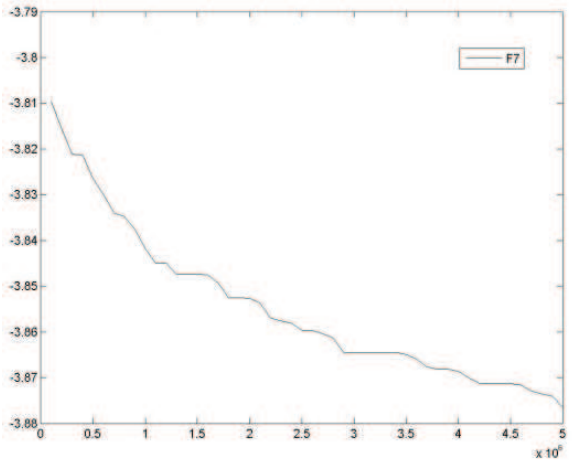


Fig 11. Convergence Graph for problem 7 with D=1000

## IV. CONCLUSIONS

This paper presents a dynamic multi-swarm particle swarm optimizer with local search for solving the CEC 2008 large scale global optimization problems. In order to achieve a better diversity, more freedom is given to the DMS-PSO, as the population is divided into many small sub-swarms, and these sub-swarms are regrouped frequently to exchange the information among all the particles. Through combining the exploitation and exploration together, this neighborhood structure gives better performance on complex problems. A local search is combined with the algorithm to improve the overall algorithm's local searching ability. The DMS-L-PSO is tested on a set of benchmark functions and the results show that the proposed algorithm can find reasonable solutions for all of the problems.

## 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. pp. 39-43, 1995.

[2] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization ". *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ. pp. 1942-1948, 1995.

[3] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance". *Proc. of IEEE Congress on Evolutionary Computation (CEC 1999),* Piscataway, NJ. pp. 1931-1938, 1999.

[4] Broyden, C.G., "The Convergence of a Class of Double-rank Minimization Algorithms," Journal of the Institute of Mathematics and Applications., Vol. 6, pp 76-90, 1970.

[5] Fletcher, R., "A New Approach to Variable Metric Algorithms," Computer Journal, Vol. 13, pp 317-322, 1970.

[6] Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," Mathematics of Computing, Vol. 24, pp 23-26, 1970.

[7] Shanno, D.F., "Conditioning of Quasi-Newton Methods for Function Minimization," Mathematics of Computing, Vol. 24, pp 647-656, 1970.

[8] Davidon, W.C., "Variable Metric Method for Minimization," A.E.C. Research and Development Report, ANL-5990, 1959, M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.

[9] Fletcher, R. and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, pp 163-168, 1963.

[10] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, Z. Yang, "Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization", http://www.ntu.edu.sg/home/EPNSugan, http://nical.ustc.edu.cn/cec08ss.php, Nov 27, 2007.

[11] J. J. Liang, and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer," *Proc. of IEEE Int. Swarm Intelligence Symposium*, pp. 124-129, 2005.