Effects of the Number of Swarms on Parallel Multi-Swarm PSO

Şaban Gülcü, and Halife Kodaz

***Abstract*—Particle swarm optimization is a population-based and stochastic optimization technique. It inspired from the social behaviors of bird flocks. Each individual in the population represents a potential solution. The comprehensive learning PSO (CLPSO) is a PSO variant. The *gbest* position is never used in CLPSO. To update a particle's velocity, all other particles' *pbest* information is used. The parallel comprehensive learning particle swarm optimization (PCLPSO) algorithm is based on CLPSO. It has multiple swarms based on the master-slave paradigm and works cooperatively and concurrently. PCLPSO is capable of increasing the quality of the solutions, decreasing the search time and improving the robustness. This article studies the effect of the number of swarms on PCLPSO algorithm.**

***Keywords*—global optimization, parallel algorithm, parallel multi-swarm algorithm, particle swarm optimization.**

I. Introduction

Particle swarm optimization (PSO) is developed by Kennedy and Eberhart in 1995 [1] is a population-based and stochastic optimization technique. It inspired from the social behaviors of bird flocks. Each individual in the population, called particle, represents a potential solution. PSO has been used in many various areas such as synchronous motor design [2] and product design and manufacturing [3].

In recent years, many PSO variants have been proposed. The comprehensive learning PSO (CLPSO) algorithm [4] is one of them. In CLPSO, instead of using a particle's *pbest* information in the original PSO, all other particles' *pbest* information is used to update the particle's velocity. Further, the *gbest* position in PSO is never used in CLPSO. With this strategy, CLPSO searches a larger area and the probability of finding global optimum is increased. The parallel comprehensive learning particle swarm optimization (PCLPSO) algorithm [5] based on CLPSO has multiple swarms based on the master-slave paradigm and works cooperatively and concurrently. PCLPSO is capable of increasing the quality of the solutions, decreasing the search time and improving the robustness. This article studies the effect of the number of swarms on PCLPSO algorithm.

This article is organized as follows: Section 2 gives the background information on the PSO, CLPSO and PCLPSO algorithms. The experimental results are presented in Section 3. Finally, the article is concluded in Section 4.

Manuscript received March 9, 2016.

Ş. Gülcü is with the Computer Engineering Department, Necmettin Erbakan University, Konya, Turkey .

H. Kodaz is with the Computer Engineering Department, Selcuk University, Konya, Turkey .

II. Materials and Methods

A. PSO

Each particle in PSO represents a bird and offers a solution. Each particle has a fitness value according to the result of the fitness function. Particles have velocity information which leads them in the search area. PSO is started with a population generated randomly in the search area. The particles update their velocity and position information by using Equations (1) and (2) respectively to find the most proper solution. To update the position of a particle, *pbest* of the particle and *gbest* of the whole population are used. *pbest* and *gbest* are repeatedly updated during the optimization process. Thus, the global optimum or a solution close to the global optimum is found at the end of the algorithm.

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

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

where V_i^d and X_i^d represent the velocity and the position of the dth dimension of the particle i. The constant w is called inertia weight plays the role to balance between the global search ability and local search ability [6]. c_1 and c_2 are the acceleration coefficients. *rand*1 and *rand*2 which are the random numbers between 0 and 1 affect the stochastic nature of the algorithm [7]. $pbest_i$ is the best position of the particle i. *gbest* is the best position in the entire swarm. The inertia weight w is updated according to Equation (3) during the optimization process.

$$w(t) = w_{\max} - t*(w_{\max} - w_{\min})/T \quad (3)$$

where $w_{\max}$ and $w_{\min}$ are the maximum and minimum inertia weights and usually set to 0.9 and 0.2 respectively [6]. t is the actual iteration number and T is the maximum number of iteration cycles.

B. CLPSO

CLPSO which is a variant of PSO was proposed by Liang, Qin, Suganthan and Baskar [4]. If the *gbest* falls into a local minimum, the population can easily fall into this local minimum. For this reason, CLPSO doesn't use *gbest*. Another property of CLPSO is that a particle uses also the *pbest*s of all other particles. This method is called as the comprehensive learning approach. The velocity of a particle in CLPSO is updated using Equation (4).

$$V_i^d = w * V_i^d + c * rand_i^d * (pbest_{fi(d)}^d - X_i^d) \quad (4)$$

where $\boldsymbol{f_i} = [f_i(1), f_i(2),\ldots, f_i(D)]$ is a list of the random selected particles which can be any particles in the swarm including the particle *i*. They are determined by the *Pc* value, called as learning probability, in Equation (5). $pbest^d_{fi(d)}$ indicates the *pbest* value of the particle which is stored in the list $\boldsymbol{f_i}$ of the particle *i* for the *d*th dimension. How a particle selects the *pbest*s for each dimension is explained in [4].

$$Pc_i = 0.05 + 0.45 * \frac{\left(\exp\left(\frac{10(i-1)}{ps-1}\right)-1\right)}{(\exp(10)-1)} \quad (5)$$

CLPSO uses a parameter *m*, called the refreshing gap. It is used to learn from good exemplars and to escape from local optima. The flowchart of the CLPSO algorithm is shown in [4].

C. PCLPSO

The PCLPSO algorithm based on CLPSO was proposed by Gülcü and Kodaz [5]. PCLPSO enhances the solution quality through its multiswarm and cooperation properties. Also, through its parallel running on a distributed environment, it improves computational efficiency.

In the multiswarm technique, a population is divided into subpopulations. Each subpopulation represents a swarm and each swarm independently runs PCLPSO algorithm. Thus, they explore the search area. PCLPSO has two types of swarms: master-swarm and slave swarm. In the cooperation technique, each swarm periodically shares its own global best position with other swarms. The parallelism property of PCLPSO is that each swarm runs the algorithm on a different computer at the same time. Thus, the computational efficiency is achieved. The topology of PCLPSO is shown in Figure 1. The PCLPSO contains only one master-swarm and several slave-swarms. Each swarm including master cooperatively and synchronously runs the PCLPSO algorithm to find the global optimum. Jade middleware framework [8] is used to fulfill the parallelism. Table 1 shows the cluster specifications. The flowchart of the PCLPSO algorithm is shown in Figure 2.

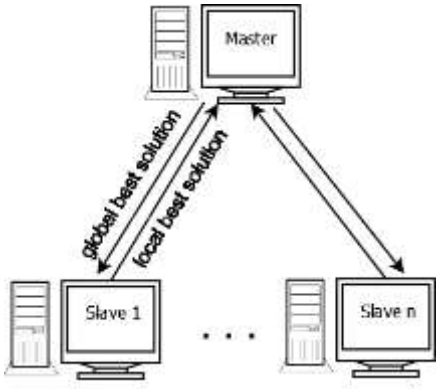

Fig. 1. The communication topology [5].

In the communication topology, the neighbor swarms to share the information are determined. The communications are only between the master swarm and the slave swarms. There isn't any directly communication between slave swarms as shown in Figure 1. Migration process occurs periodically after a certain number of iterations in PCLPSO. Each swarm sends the own local best solution to the master in the PCLPSO's migration process. The master collects the received local best solutions including own local best solution into a pool. It chooses the best solution from the pool. This solution is sent to all slave swarms by the master. Thus, PCLPSO obtains better and more robust solutions.

In the phase of the PCLPSO's replacement strategy that handles with the usage of the received information, a random selected particle in each slave swarm is replaced with the global best solution was sent by the master. A random selected particle in the master swarm is replaced with the global best solution, too. Thus, PCLPSO improves the solution quality.

TABLE I: THE SPECIFICATIONS OF THE CLUSTER SYSTEM

Operating System	Windows XP
Cpu	Pentium i5 3.10 GHz
Memory	2 GB
Java Version	Java SE 1.7
Jade Version	Jade 4.2
Network	Gigabit Ethernet

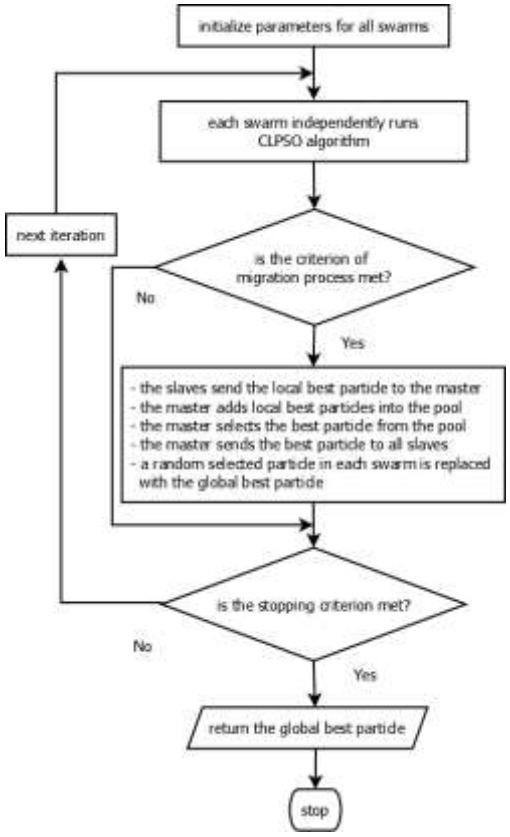

Fig. 2. The flowchart of the PCLPSO algorithm [5].

III. EXPERIMENTAL RESULTS

The experiments performed in this section were designed to study the behavior of PCLPSO by varying the number of swarms. The number of swarms varies with regard to the difficulty of problems. The number of swarms is a critical parameter in PCLPSO and affects the performance of the algorithm. This article studies the effect of the number of swarms on PCLPSO algorithm.

Two unimodal and two multimodal benchmark functions are selected. These functions are well known to the global optimization community and commonly used for the test of optimization algorithms. The formulas of the four functions are given in next subsection. The global optimum values, the search and initialization ranges of these functions are given in Table 2. The total number of particles is set as 60. The number of particles (k) per swarm is calculated using 60/n. Here, n represents the number of swarms.

The experiments are divided into three groups according to the dimensions of functions. In the first group, the functions have ten dimensions. The maximum fitness evaluation (FE) is set as $3x10^4$. In the second group, the functions have 30 dimensions and the FE is set as $2x10^5$. In the third group, the functions have 100 dimensions and the FE is set as $3x10^5$. Table 3 shows the dimension of functions, the maximum number of the fitness evaluation, the number of swarms (n) and the number of particles (k) per swarm. The experiments are carried out on a cluster whose specification is presented in Table 1. The migration period P is set to 20 for all test functions. The inertia weight w linearly decreases from 0.9 to 0.2 during the iterations, the acceleration coefficient c is equal to 1.49445 and the refreshing gap m is equal to five. 30 independent runs are applied for each function. The mean values and standard deviation of the results are presented in Table 4, Table 5 and Table 6.

TABLE II: THE SPECIFICATIONS OF THE BENCHMARK FUNCTIONS

f	Global Min. x*	$f(x^*)$	Search Range	Initial Range
f_1	[0,0,…,0]	0	$[-100, 100]^D$	$[-100, 50]^D$
f_2	[1,1,…,1]	0	$[-2.048, 2.048]^D$	$[-2.048, 2.048]^D$
f_3	[0,0,…,0]	0	$[-32.768, 32.768]^D$	$[-32.768, 16]^D$
f_4	[0,0,…,0]	0	$[-600, 600]^D$	$[-600, 200]^D$

TABLE III: PARAMETERS USED IN EXPERIMENTS

Dimension	FE	n	k
10	$3x10^4$	2	30
		3	20
		4	15
		5	12
		6	10
30	$2x10^5$	2	30
		3	20
		4	15
		5	12
		6	10
100	$3x10^5$	2	30
		3	20
		4	15
		5	12
		6	10

A. Functions

The functions used in the experiments are the following.

Sphere function:

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \tag{6}$$

Rosenbrock function:

$$f_2(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \tag{7}$$

Ackley function:

$$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \tag{6}$$

Griewank function:

$$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{8}$$

Functions f_1 and f_2 are unimodal. Unimodal functions have only one optimum and no local minima. Functions f_3 and f_4 are multimodal. Multimodal functions have only one optimum and many local minima. They are treated as a difficult class of benchmark functions by researchers because the number of local minima of the function grows exponentially as the number of its dimension increases [9-12].

B. Results of the 10-D problem

Table 4 presents the mean, standard deviation and calculation time of the function values for 10-D problem.

TABLE IV: RESULTS OF 10-D PROBLEMS

f	n	Mean	Standard Deviation	Time (ms)
f_1	2	7.38e-05	8.24e-05	167
	3	9.54e-05	1.47e-04	82
	4	6.72e-05	7.23e-05	79
	5	6.85e-05	1.23e-04	76
	6	2.18e-05	3.57e-05	80
f_2	2	5.17e+00	1.19e+00	302
	3	5.42e+00	1.62e+00	157
	4	6.29e+00	1.79e+00	150
	5	6.95e+00	2.44e+00	144
	6	7.33e+00	2.90e+00	150
f_3	2	2.10e-02	1.17e-02	433
	3	1.93e-02	8.98e-03	231
	4	1.64e-02	8.86e-03	219
	5	1.46e-02	7.46e-03	211
	6	2.74e-02	4.08e-02	219
f_4	2	4.75e-02	2.66e-02	559
	3	5.98e-02	2.54e-02	304
	4	5.81e-02	2.76e-02	290
	5	5.65e-02	2.69e-02	279
	6	5.48e-02	4.52e-02	289

A. Results of the 30-D problem

Table 5 presents the mean, standard deviation and calculation time of the function values for 30-D problem.

TABLE V: RESULTS OF 30-D PROBLEMS

f	n	Mean	Standard Deviation	Time (ms)
f_1	2	2.00e-14	3.52e-14	854
	3	6.56e-15	2.01e-14	754
	4	3.46e-15	1.55e-14	665
	5	2.98e-15	1.56e-14	625
	6	2.15e-15	1.17e-14	593
f_2	2	2.27e+01	1.92e+00	1700
	3	2.29e+01	2.02e+00	1504
	4	2.51e+01	1.99e+00	1319
	5	2.39e+01	2.35e+00	1240
	6	2.51e+01	1.89e+00	1172
f_3	2	7.91e-08	2.14e-08	2540
	3	3.76e-08	1.69e-08	2243
	4	1.59e-08	6.24e-09	1982
	5	6.28e-09	2.89e-09	1850
	6	2.35e-09	1.63e-09	1751
f_4	2	3.09e-08	7.99e-08	3399
	3	1.59e-08	3.88e-08	2965
	4	1.07e-08	3.30e-08	2619
	5	2.61e-08	1.29e-07	2464
	6	2.31e-08	7.36e-08	2329

A. *Results of the 100-D problem*

Table 6 presents the mean, standard deviation and calculation time of the function values for 100-D problem.

TABLE VI: RESULTS OF 100-D PROBLEMS

f	n	Mean	Standard Deviation	Time (ms)
f_1	2	4.14e-07	1.47e-06	2964
	3	4.11e-07	2.12e-06	2384
	4	2.61e-07	1.41e-06	2083
	5	2.56e-07	1.40e-06	1879
	6	3.86e-07	2.11e-06	1832
f_2	2	9.56e+01	6.98e-01	5918
	3	9.56e+01	1.02e+00	4709
	4	9.60e+01	7.88e-01	4163
	5	9.62e+01	6.13e-01	3744
	6	9.65e+01	3.70e-01	3648
f_3	2	7.37e-05	1.08e-05	8900
	3	3.11e-05	5.74e-06	7031
	4	1.12e-05	2.65e-06	6218
	5	3.78e-06	9.36e-07	5584
	6	1.07e-06	4.57e-07	5469
f_4	2	1.04e-06	1.65e-06	12281
	3	3.27e-07	3.90e-07	9505
	4	3.32e-07	9.34e-07	8277
	5	2.59e-08	8.71e-08	7416
	6	1.87e-08	8.29e-08	7289

IV. CONCLUSIONS

This article studies the effect of the number of swarms on PCLPSO algorithm. PCLPSO based on the master-slave paradigm has multiple swarms which work cooperatively and concurrently on distributed computers. Each swarm runs the algorithm independently. In the cooperation, the swarms exchange their own local best particle with each other in every migration process. Thus, the diversity of the solutions increases through the multiple swarms and cooperation. PCLPSO runs on a cluster. We used the well-known benchmark functions in the experiments. According to the result of the experiments, the calculation time decreases as the number of swarms is increased. We obtained better results on few functions when the number of swarms is equal to two. Also, we obtained better results on some functions when the number of swarms is equal to six. The number of swarms should be tuned for different problems. Namely, it varies with regard to the difficulty of problems. As future work, we plan to investigate the effects of the migration interval and the number of particles to be exchanged between swarms on the performance of the PCLPSO algorithm.

REFERENCES

[1] J. Kennedy, and R. Eberhart, "Particle swarm optimization," *1995 Ieee International Conference on Neural Networks Proceedings*, vol. 1-6, pp. 1942-1948, 1995. http://dx.doi.org/10.1109/icnn.1995.488968

[2] M. Mutluer, and O. Bilgin, "Design optimization of PMSM by particle swarm optimization and genetic algorithm," *Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on, IEEE*, pp. 1-4, 2012. http://dx.doi.org/10.1109/inista.2012.6247024

[3] A.R. Yıldız, "A novel particle swarm optimization approach for product design and manufacturing," *Int J Adv Manuf Technol*, vol. 40, pp. 617-628, 2009. http://dx.doi.org/10.1007/s00170-008-1453-1

[4] J.J. Liang, A.K. Qin, P.N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *Ieee T Evolut Comput*, vol.10, pp. 281-295, 2006. http://dx.doi.org/10.1109/TEVC.2005.857610

[5] Ş. Gülcü, and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 33-45, 2015. http://dx.doi.org/10.1016/j.engappai.2015.06.013

[6] Y. Shi, and R. Eberhart, "A modified particle swarm optimizer," *Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence*, pp. 69-73, 1998. http://dx.doi.org/10.1109/icec.1998.699146

[7] F. Van Den Bergh, "An analysis of particle swarm optimizers," *University of Pretoria*, 2006.

[8] F.L. Bellifemine, G. Caire and D. Greenwood, *Developing multi-agent systems with JADE*, John Wiley & Sons, 2007. http://dx.doi.org/10.1002/9780470058411

[9] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82-102, 1999. http://dx.doi.org/10.1109/4235.771163

[10] B.-Y. Qu, P.N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, pp. 387-402, 2013. http://dx.doi.org/10.1109/TEVC.2012.2203138

[11] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 150-169, 2010. http://dx.doi.org/10.1109/TEVC.2009.2026270

[12] S.C. Esquivel, and C.A. Coello Coello, "On the use of particle swarm optimization with multimodal functions," *The 2003 Congress on IEEE Evolutionary Computation CEC'03*, pp. 1130-1136, 2003. http://dx.doi.org/10.1109/cec.2003.1299795