EFFECTS OF THE DIFFERENT MIGRATION PERIODS ON PARALLEL MULTI-SWARM PSO

Şaban Gülcü[1] and Halife Kodaz[2]

[1]Department of Computer Engineering,
Necmettin Erbakan University, Konya, Turkey
sgulcu@konya.edu.tr
[2]Department of Computer Engineering, Selcuk University, Konya, Turkey
hkodaz@selcuk.edu.tr

ABSTRACT

In recent years, there has been an increasing interest in parallel computing. In parallel computing, multiple computing resources are used simultaneously in solving a problem. There are multiple processors that will work concurrently and the program is divided into different tasks to be simultaneously solved. Recently, a considerable literature has grown up around the theme of metaheuristic algorithms. Particle swarm optimization (PSO) algorithm is a popular metaheuristic algorithm. The parallel comprehensive learning particle swarm optimization (PCLPSO) algorithm based on PSO has multiple swarms based on the master-slave paradigm and works cooperatively and concurrently. The migration period is an important parameter in PCLPSO and affects the efficiency of the algorithm. We used the well-known benchmark functions in the experiments and analysed the performance of PCLPSO using different migration periods.

KEYWORDS

Particle Swarm Optimization, Migration Period, Parallel Algorithm, Global Optimization

1. INTRODUCTION

In recent years, there has been an increasing interest in parallel computing. Software applications developed by using conventional methods run on a computer with limited resources as serial computing. Software executed by a processor on a computer consists of a collection of instructions. Each instruction is processed after another. An instruction is only processed at a time. But in parallel computing, multiple computing resources are used simultaneously in solving a problem. There are multiple processors that will work concurrently and the program is divided into different tasks to be simultaneously solved. Each task is divided into different instructions. The instructions are processed on different processors at the same time. Thus, performance increases and computer programs run in a shorter time. Parallel computing has been used in many different fields such as cloud computing [1], physics [2] and nanotechnology [3].

Jan Zizka et al. (Eds) : CCSEIT, AIAP, DMDB, MoWiN, CoSIT, CRIS, SIGL, ICBB, CNSA-2016
pp. 13–22, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60602

Recently, a considerable literature has grown up around the theme of metaheuristic algorithms. Particle swarm optimization (PSO) algorithm is developed by Kennedy and Eberhart in 1995 [4] is a popular metaheuristic algorithm. It is a population-based and stochastic optimization technique. It inspired from the social behaviours of bird flocks. Each individual in the population, called particle, represents a potential solution. In recent years, many algorithms based on PSO have been developed such as the comprehensive learning PSO (CLPSO) algorithm [5] and the parallel comprehensive learning particle swarm optimization (PCLPSO) algorithm [6]. In recent years, devising parallel models of algorithms has been a healthy field for developing more efficient optimization procedures [14-17]. Parallelism is an approach not only to reduce the resolution time but also to improve the quality of the provided solutions. In CLPSO, instead of using a particle's best information in the original PSO, all other particles' historical best information is used to update the particle's velocity. Further, the global best position of population 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 PCLPSO algorithm based on CLPSO has multiple swarms based on the master-slave paradigm and works cooperatively and concurrently. Through PCLPSO, the solution quality and the global search ability are improved. This article studies the effect of the different migration periods on PCLPSO algorithm.

This article has been organized in the following way: Section 2 is concerned with the methodologies used for this study. Section 3 presents the experimental results and the findings of the research. Finally, the article is concluded in Section 4.

2. MATERIALS & METHODS

2.1. PSO

Each particle in PSO represents a bird and offers a solution. Each particle has a fitness value calculated by fitness function. Particles have velocity information and position information updated during the optimization process. Each particle searches the food in the search area using the velocity and position information. PSO aims to find the global optimum or a solution close to the global optimum and therefore is launched with a random population. The particles update their velocity and position information by using Equations (1) and (2) respectively. 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 [7]. c_1 and c_2 are the acceleration coefficients. *rand*1 and *rand*2 are the two random numbers between 0 and 1. They affect the stochastic nature of the algorithm [8]. $pbest_i$ 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 [7]. t is the actual iteration number and T is the maximum number of iteration cycles.

2.2. CLPSO

CLPSO based on PSO was proposed by Liang, Qin, Suganthan and Baskar [5]. PSO has some deficiencies. For instance, 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). $pbest_{fi(d)}^d$ indicates the *pbest* value of the particle which is stored in the list $\boldsymbol{f}_i$ of the particle i for the dth dimension. How a particle selects the *pbest*s for each dimension is explained in [5].

$$V_i^d = w * V_i^d + c * rand_i^d * (pbest_{fi(d)}^d - X_i^d) \quad (5)$$

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 given in [5].

2.3. PCLPSO

Although PSO has many advantages, the main deficiency of PSO is the premature convergence [8]. PCLPSO handles to overcome this deficiency like many PSO variants. The PCLPSO algorithm based on CLPSO was proposed by Gülcü and Kodaz [6]. The solution quality is enhanced through multiswarm and cooperation properties. Also, computational efficiency is improved because PCLPSO runs parallel on a distributed environment.

A population is split into subpopulations. Each subpopulation represents a swarm and each swarm independently runs PCLPSO algorithm. Thus, they seek the search area. There are 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 is that each swarm runs the algorithm on a different computer at the same time to achieve computational efficiency. The topology is shown in Figure 1. Each swarm runs cooperatively and synchronously the PCLPSO algorithm to find the global optimum. PCLPSO uses Jade middleware framework [9] to establish the parallelism. The cluster specifications are so: windows XP operating system, pentium i5 3.10 GHz, 2 GB memory, java se 1.7, Jade 4.2 and gigabit ethernet. The flowchart of the PCLPSO algorithm is given in [6].

In the communication topology, there isn't any directly communication between slave swarms as shown in Figure 1. Migration process occurs periodically after a certain number of cycles. Each swarm sends the own local best solution to the master in the PCLPSO's migration process. The master collects the local best solutions into a pool, called *ElitePool*. It chooses the best solution

from the *ElitePool*. This solution is sent to all slave swarms by the master. Thus, PCLPSO obtains better and more robust solutions.

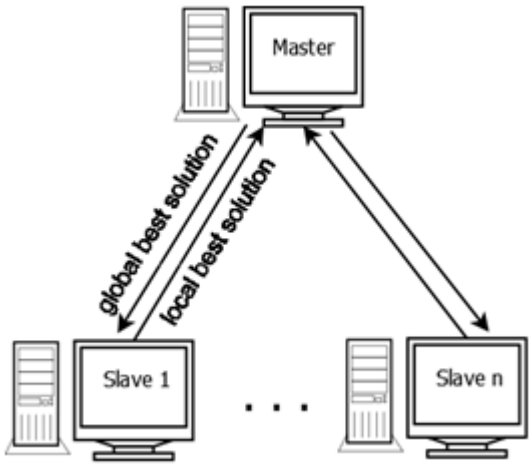

Figure 1. The communication topology [6]

3. EXPERIMENTAL RESULTS

The experiments performed in this section were designed to study the behaviour of PCLPSO by varying the migration period. The migration period is an important parameter in PCLPSO and affects the efficiency of the algorithm. This article studies the effect of the migration period on PCLPSO algorithm.

Two unimodal and two multimodal benchmark functions which are well known to the global optimization community and commonly used for the test of optimization algorithms are selected. The formulas of the four functions are given in next subsection. The properties of these functions are given in Table 1. The number of particles per swarm is 15. According to the dimensions of functions, the experiments are split into three groups. The properties of these groups are given in Table 2. The term FE in the table refers the maximum fitness evaluation.

The experiments are carried out on a cluster whose specifications are windows XP operating system, pentium i5 3.10 GHz, 2 GB memory, java se 1.7, Jade 4.2 and gigabit ethernet. 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 tests are carried out for each function. The results are given in next subsections.

Table 1. Type, Global Minimum, Function Value, Search and Initialization Ranges of the Benchmark Functions

f	Global Minimum x^*	Function Value $f(x^*)$	Search Range	Initialization 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 2. Parameters used in experiments

Dimension	FE	Number of swarms	Number of particles
10	$3x10^4$	4	15
30	$2x10^5$	4	15
100	$3x10^5$	4	15

3.1. 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:

$$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \tag{8}$$

Griewank function:

$$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{9}$$

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 [10-13].

3.2. Results of the 10-D problems

Table 3 presents the mean of the function values for 10-D problems according to the different migration periods. Table 4 presents the calculation time of the functions for 10-D problems. In [6], the importance of the migration period is emphasized: if the information is very often exchanged, then the solution quality may be better, but the computational efficiency deteriorates. If the migration interval is longer, the computational efficiency is better, but the solution quality may be worse. It is apparent from these tables that the computational efficiency is better when the migration interval is equal to 100 as expected. But the best values of functions f_1-f_4 are obtained when the migration intervals are equal to 11, 2, 6 and 1, respectively.

Table 3. The mean values for 10-D problems.

P	f_1	f_2	f_3	f_4
1	2.45e-03	7.20e+00	9.02e-02	**1.11e-01**
2	4.71e-03	**6.70e+00**	8.23e-02	1.38e-01
3	5.66e-03	7.51e+00	2.51e-01	1.49e-01
4	3.28e-03	7.03e+00	1.23e-01	1.43e-01
5	3.70e-03	7.68e+00	6.57e-02	1.28e-01
6	4.03e-03	7.94e+00	**6.49e-02**	1.29e-01
7	3.24e-03	7.32e+00	8.10e-02	1.36e-01
8	2.15e-03	7.17e+00	9.52e-02	1.38e-01
9	4.23e-03	7.90e+00	9.71e-02	1.40e-01
10	3.87e-03	8.98e+00	7.67e-02	1.25e-01
11	**1.97e-03**	7.17e+00	1.08e-01	1.28e-01
12	3.69e-03	7.78e+00	1.46e-01	1.43e-01
13	3.86e-03	8.26e+00	1.42e-01	1.03e-01
14	2.99e-03	7.16e+00	1.09e-01	1.24e-01
15	3.41e-03	8.49e+00	9.15e-02	1.18e-01
16	3.55e-03	8.79e+00	1.96e-01	1.27e-01
17	3.21e-03	7.47e+00	2.64e-01	1.32e-01
18	4.13e-03	8.16e+00	1.64e-01	1.37e-01
19	4.02e-03	7.08e+00	9.34e-02	1.51e-01
20	3.97e-03	6.84e+00	1.31e-01	1.29e-01
50	2.96e-03	7.92e+00	1.26e-01	1.30e-01
100	3.63e-03	6.90e+00	2.55e-01	1.25e-01

Table 4. The calculation time (ms) for 10-D problems

P	f_1	f_2	f_3	f_4
1	4463	4484	10359	15532
2	2230	2246	5184	7769
3	1484	1497	3450	5163
4	1122	1131	2600	3889
5	901	907	2081	3112
6	750	755	1732	2588
7	643	648	1482	2214
8	564	567	1303	1932
9	501	507	1158	1723
10	458	462	1051	1563
11	413	417	946	1406
12	377	380	864	1285
13	351	353	804	1187
14	322	326	736	1094
15	305	307	694	1031
16	289	290	657	975
17	271	272	614	911
18	252	254	573	846
19	243	244	551	815
20	234	236	530	784
50	101	102	220	319
100	**56**	**57**	**116**	**163**

Table 5. The mean values for 30-D problems.

P	f_1	f_2	f_3	f_4
1	**1.04e-09**	2.50e+01	1.49e-05	3.16e-06
2	1.63e-09	2.38e+01	1.07e-05	6.14e-07
3	3.17e-09	2.42e+01	1.48e-05	1.48e-06
4	2.09e-09	2.25e+01	1.39e-05	1.09e-06
5	1.12e-09	2.37e+01	1.77e-05	2.05e-06
6	2.40e-09	2.40e+01	1.35e-05	1.92e-05
7	4.05e-09	2.48e+01	1.08e-05	1.55e-06
8	1.37e-09	2.31e+01	1.51e-05	6.05e-07
9	2.15e-09	2.39e+01	1.31e-05	1.29e-05
10	1.51e-09	**2.17e+01**	1.65e-05	1.42e-06
11	1.52e-09	2.77e+01	**8.89e-06**	2.65e-06
12	1.93e-09	3.14e+01	1.35e-05	**3.12e-07**
13	1.32e-09	2.22e+01	1.20e-05	8.73e-07
14	2.33e-09	2.59e+01	1.01e-05	7.74e-07
15	4.27e-09	2.24e+01	1.07e-05	5.33e-07
16	1.85e-09	2.53e+01	1.60e-05	1.99e-06
17	1.78e-09	2.49e+01	1.40e-05	7.22e-07
18	2.12e-09	2.53e+01	1.54e-05	4.80e-06
19	3.17e-09	2.29e+01	1.37e-05	7.04e-07
20	1.95e-09	2.52e+01	1.72e-05	1.71e-06
50	2.63e-09	2.49e+01	1.73e-05	9.63e-06
100	3.21e-09	2.29e+01	1.24e-05	2.37e-06

Table 6. The calculation time (ms) for 30-D problems

P	f_1	f_2	f_3	f_4
1	501775	507009	1172359	1974659
2	250866	253531	586153	987428
3	167366	169169	390778	658059
4	125544	126884	293141	493516
5	100431	101484	234481	394644
6	83734	84672	195575	329072
7	71819	72603	167788	282494
8	62819	63466	146556	246678
9	55866	56447	130387	219428
10	50291	50831	117366	197528
11	45772	46262	106825	179744
12	41866	42291	97666	164303
13	38700	39109	90266	151909
14	35984	36359	83941	141228
15	33578	33928	78306	131734
16	31519	31891	73522	123447
17	29675	29981	69169	116419
18	28031	28309	65288	109822
19	26503	26784	61772	103956
20	25150	25500	58628	98703
50	10106	10216	23447	39425
100	**5709**	**5859**	**12519**	**20522**

Table 7. The mean values for 100-D problems.

P	f_1	f_2	f_3	f_4
1	7.04e-03	1.41e+02	3.46e-01	7.03e-03
2	1.95e-02	1.50e+02	1.03e+00	9.48e-03
3	1.69e-02	2.23e+02	2.27e-02	1.22e-02
4	2.05e-02	1.46e+02	1.80e-02	2.76e-02
5	9.95e-03	1.54e+02	**8.78e-03**	4.91e-02
6	1.63e-02	9.65e+01	1.84e-02	1.53e-02
7	8.94e-03	1.81e+02	1.74e-02	6.66e-03
8	3.65e-02	9.89e+01	5.83e-01	2.43e-02
9	1.67e-02	9.64e+01	4.70e-01	1.90e-02
10	2.15e-02	1.49e+02	1.03e+00	5.71e-03
11	**4.19e-03**	1.30e+02	1.91e-02	1.08e-02
12	1.34e-02	2.04e+02	1.41e-02	7.42e-03
13	1.16e-02	9.75e+01	1.29e-02	7.88e-03
14	6.72e-02	9.82e+01	1.09e+00	1.97e-01
15	5.70e-01	9.34e+01	1.56e-02	**3.93e-03**
16	1.31e-02	1.80e+02	2.03e-02	5.04e-03
17	4.63e-02	**9.10e+01**	2.94e-02	8.61e-03
18	3.27e-02	1.48e+02	1.03e+00	1.99e-02
19	1.32e-02	9.74e+01	2.56e-02	2.58e-02
20	1.68e-02	1.01e+02	1.40e-02	1.83e-02
50	3.02e-02	9.60e+01	1.78e-02	1.46e-02
100	9.58e-03	1.52e+02	7.94e-01	2.89e-02

Table 8. The calculation time (ms) for 100-D problems.

P	f_1	f_2	f_3	f_4
1	3865343	3907344	8432688	14837734
2	1934047	1952688	4217828	7418000
3	1288563	1301906	2809469	4943000
4	966735	976625	2108328	3708093
5	773703	782109	1686797	2966766
6	644532	651125	1405250	2471500
7	552656	558188	1204468	2118610
8	484078	488656	1054594	1855094
9	430563	434047	937250	1648172
10	387219	391531	845172	1484891
11	351922	356875	766891	1348719
12	322094	325265	702344	1235453
13	297469	300390	648907	1141359
14	276625	279359	603359	1060469
15	258031	260594	563062	989703
16	241766	244140	527125	926532
17	227922	230109	496844	873141
18	214859	216891	468234	823079
19	203953	206016	444921	781469
20	193954	195843	422985	742984
50	78093	78797	170172	297797
100	**39468**	**39844**	**85391**	**149562**

3.3. Results of the 30-D problems

Table 5 presents the mean of the function values for 30-D problems according to the different migration periods. The best mean values of functions f1-f4 are obtained when the migration periods are equal to 1, 10, 11 and 12, respectively. Table 6 presents the calculation time of the function values for 30-D problems.

3.4. Results of the 100-D problems

Table 7 presents the mean of the function values for 100-D problems according to the different migration periods. The best mean values of functions f1-f4 are obtained when the migration periods are equal to 11, 17, 5 and 15, respectively. Table 8 presents the calculation time of the functions for 100-D problems.

4. CONCLUSIONS

The purpose of the current study was to determine the effect of the migration period 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. In the experiments, the performance of PCLPSO is analysed using different migration periods. This study has shown that the calculation time decreases when the migration interval is longer. We obtained better results on some functions when the migration period is around 10. The migration period should be tuned for different problems. Namely, it varies with regard to the difficulty of problems. As future work, we plan to investigate the number of particles to be exchanged between swarms on the performance of the PCLPSO algorithm.

ACKNOWLEDGEMENTS

This research was supported by Scientific Research Projects Office of Necmettin Erbakan University (Project No: 162518001-136).

REFERENCES

[1] M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.-G. Talbi, A.Y. Zomaya, D. Tuyttens, A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, Journal of Parallel and Distributed Computing, 71 (2011) 1497-1508.

[2] Z. Guo, J. Mi, P. Grant, An implicit parallel multigrid computing scheme to solve coupled thermal-solute phase-field equations for dendrite evolution, Journal of Computational Physics, 231 (2012) 1781-1796.

[3] J. Pang, A.R. Lebeck, C. Dwyer, Modeling and simulation of a nanoscale optical computing system, Journal of Parallel and Distributed Computing, 74 (2014) 2470-2483.

[4] J. Kennedy, R. Eberhart, Particle swarm optimization, 1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6, (1995) 1942-1948.

[5] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, Ieee T Evolut Comput, 10 (2006) 281-295.

[6] Ş. Gülcü, H. Kodaz, A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, Engineering Applications of Artificial Intelligence, 45 (2015) 33-45.

[7] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, IEEE, 1998, pp. 69-73.

[8] F. Van Den Bergh, An analysis of particle swarm optimizers, in, University of Pretoria, 2006.

[9] F.L. Bellifemine, G. Caire, D. Greenwood, Developing multi-agent systems with JADE, John Wiley & Sons, 2007.

[10] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, Evolutionary Computation, IEEE Transactions on, 3 (1999) 82-102.

[11] B.-Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, Evolutionary Computation, IEEE Transactions on, 17 (2013) 387-402.

[12] X. Li, Niching without niching parameters: particle swarm optimization using a ring topology, Evolutionary Computation, IEEE Transactions on, 14 (2010) 150-169.

[13] S.C. Esquivel, C.A. Coello Coello, On the use of particle swarm optimization with multimodal functions, in: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, IEEE, 2003, pp. 1130-1136.

[14] E. Alba, Parallel metaheuristics: a new class of algorithms, John Wiley & Sons, 2005.

[15] G.-W. Zhang, Z.-H. Zhan, K.-J. Du, Y. Lin, W.-N. Chen, J.-J. Li, J. Zhang, Parallel particle swarm optimization using message passing interface, in: Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, Volume 1, Springer, 2015, pp. 55-64.

[16] M. Pedemonte, S. Nesmachnow, H. Cancela, A survey on parallel ant colony optimization, Applied Soft Computing, 11 (2011) 5181-5197.

[17] E B. Li, K. Wada, Communication latency tolerant parallel algorithm for particle swarm optimization, Parallel Computing, 37 (2011) 1-10.