Fast Multi-swarm Optimization with Cauchy Mutation and Crossover operation

Qing Zhang[1,2], Changhe Li[1], Y. Liu[3], Lishan Kang[1]

[1]China University of Geosciences, School of Computer, Wuhan, P.R.China, 430074
[2]Huanggang Normal University
[3] The University of Aizu,Aizu-Wakamatsu, Fukushima, Japan, 965-8580
zhangqing@hgnc.net,lch_wfx@yahoo.com.cn ,yliu@u-aizu.ac.jp, kang_whu@yahoo.com

Abstract. The standard Particle Swarm Optimization (PSO) algorithm is a novel evolutionary algorithm in which each particle studies its own previous best solution and the group's previous best to optimize problems. One problem exists in PSO is its tendency of trapping into local optima. In this paper, a multiple swarms technique(FMSO) based on fast particle swarm optimization(FPSO) algorithm is proposed by bringing crossover operation. FPSO is a global search algorithm witch can prevent PSO from trapping into local optima by introducing Cauchy mutation. Though it can get high optimizing precision, the convergence rate is not satisfied, FMSO not only can find satisfied solutions ,but also speeds up the search. By proposing a new information exchanging and sharing mechanism among swarms. By comparing the results on a set of benchmark test functions, FMSO shows a competitive performance with the improved convergence speed and high optimizing precision.

Keywords: Particle swarm optimization, Cauchy mutation, swarm intelligence

1. Introduction

Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart in 1995 [1,2]. It is motivated from the social behavior of organisms, such as bird flocking and fish schooling. Particles "fly" through the search space by following the previous best positions of their neighbors and their own previous best positions. Each particle is represented by a position and a velocity which are updated as follows:

$$X_{id}^{'} = X_{id} + V_{id}^{'} \tag{1}$$

$$\begin{aligned} V_{id}^{'} &= \omega V_{id} + \eta_1 rand\,()(P_{id} - X_{id}) \\ &\quad + \eta_2 rand\,()(P_{gd} - X_{id}) \end{aligned} \tag{2}$$

where X_{id}' and X_{id} represent the current and the previous positions of idth particle, V_{id}

and V_{id}' are the previous and the current velocity of *id*th particle, P_{id} and P_{gd} are the individual's best position and the best position found in the whole swarm so far respectively. $0 \le \omega < 1$ is an inertia weight which determines how much the previous velocity is preserved, η_1 and η_2 are acceleration constants, *rand()* generates random number from interval [0,1].

In PSO, each particle shares the information with its neighbors. PSO combines the cognition component of each particle with the social component of all the particles in a group. Although the speed of convergence is very fast, Once PSO traps into local optimum, it is difficult to jump out of local optimum. Ratnaweera et.al.[3] state that lack of population diversity in PSO algorithms is understood to be a factor in their convergence on local optima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance. A first attempt to model particle swarms using the quantum model(QPSO) was carried out by Sun et.al. [4]. In a quantum model, particles are described by a wave function instead of the standard position and velocity. The quantum Delta potential well model and quantum harmonic oscillators are commonly used in particle physics to describe the stochastic nature of particles. In their studies[5], the variable of *gbest* (the global best particle) and *mbest* (the mean value of all particles' previous best position) is mutated with Cauchy distribution respectively, and the results show that QPSO with *gbest* and *mbest* mutation both performs better than PSO. The work of R. A. Krohling et.al.[6][7] showed that how Gaussian and Cauchy probability distribution can improve the performance of the standard PSO. Recently, evolutionary programming with exponential mutation has also been proposed [8].

In order to prevent PSO from falling in a local optimum, a fast PSO (FPSO) is proposed by introducing a Cauchy mutation operator in this paper. Besides the Cauchy mutation, FPSO chooses the natural selection strategy of evolutionary algorithms as the basic elimination strategy of particles. Although FPSO greatly overcomes the tendency of trapping into local optima of PSO, the convergence rate isn't satisfied. Like distributed genetic algorithm, multiple swarms idea is a very useful for speeding up the search. In this paper, a multiple swarms algorithm(FMSO) based on FPSO is proposed by introducing a crossover operation, the new information exchanging and sharing mechanism of FMSO make it converge fast on the global optimum.

The rest of this paper is organized as follows. Section 2 gives the analysis of PSO, and the detail of FPSO. Section 3 gives a brief review of the multi-population technique and then describe FMSO explicitly. Section 4 describes the experiment setup and presents the experiment results. Finally, Section 5 concludes the paper with a brief summary.

2. Fast Multi-swarm Optimization with Cauchy Mutation and Crossover operatio

2.1 Cauchy mutation

From the mathematic theoretical analysis of the trajectory of a PSO particle [9], the trajectory of a particle X_{id} converges to a weighted mean of P_{id} and P_{gd}. Whenever the particle converges, it will "fly" to the personal best position and the global best particle's position. This information sharing mechanism makes PSO have a very fast speed of convergence. Meanwhile, because of this mechanism, PSO can't guarantee to find the global minimal value of a function. In fact, the particles usually converge to local optima. Without loss of generality, only function minimization is discussed here. Once the particles trap into a local optimum, in which P_{id} can be assumed to be the same as P_{gd}, all the particles converge on P_{gd}. At this condition, the velocity update equation becomes:

$$V_{id}^{'} = \omega V_{id} \tag{3}$$

When the iteration in the equation (3) goes to infinite, the velocity of the particle V_{id} will be close to 0 because of $0 \leq \omega < 1$. After that, the position of the particle X_{id} will not change, so that PSO has no capability of jumping out of the local optimum. It is the reason that PSO often fails on finding the global minimal value.

To overcome the weakness of PSO discussed at the beginning of this section, the Cauchy mutation is incorporated into PSO algorithm. The basic idea is that, the velocity and position of a particle are updated not only according to (1) and (2), but also according to Cauchy mutation as follows:

$$V_{id}^{'} = V_{id} \exp(\delta) \tag{4}$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \delta_{id} \tag{5}$$

where δ and δ_{id} denote Cauchy random numbers

Since the expectation of Cauchy distribution doesn't exist, the variance of Cauchy distribution is infinite so that Cauchy mutation could make a particle have a long jump. By adding the update equations of (4) and (5), FPSO greatly increases the probability of escaping from the local optimum.

2.2 Natural selection strategy

In the standard PSO, all particles are directly updated by their offspring no matter whether they are improved. If a particle moves to a better position, it can be replaced by the updated. However if it moves to a worse position, it is still replaced by its offspring. In fact, the most particles fly to worse positions for most cases, therefore the whole swarm will converge on local optima. Like evolutionary algorithms, FPSO introduces an evolutionary selection strategy in which each particle survives according to a natural

selection rule. Therefore, the particle's position at the next step is not only due to the position update but also the evolutionary selection. Such strategy could greatly reduce the probability of trapping into local optimum.

The evolutionary selection strategy is carried out as follows. Assume the size of the swarm is *m*, pair-wise comparison over the union of parents and offspring (*1,2,…2m*) is made. For each particle, *q* opponents are randomly chosen from all parents and offspring with equal probability. If the fitness of particle *i* is less then its opponent , it will receive a "win". Then select *m* particles that have the more winnings to be the next generation.

The major steps of FPSO are as follows:

Step1: Generate the initial particles by randomly generating the position and velocity for each particle.

Step2: Evaluate each particle's fitness.

Step3: For each particle, if its fitness is smaller than its previous best(P_{id}), update P_{id} .

Step4: For each particle, if its fitness is smaller than the best one (P_{gd}) of all particles, update P_{gd}.

Step5: For each particle ,do

1).Generate a new particle t according to the formula (1) and (2).

2).Generate a new particle t' according to the formula (4) and (5).

3) Compare t with t' ,chose the one with smaller fitness to be the offspring.

Step6: Generate next generation according to the above evolutionary selection strategy.

Step7: if the stop criterion is satisfied, then stop, else goto Step 3.

3. Multiple swarms optimization technique

In order to escape from the local optima and avoid premature convergence, the search for global optimum should be diverse. Many researchers have improved the performance of the PSO by enhancing its ability with a more diverse search. Specifically, some have introduced using multiple swarms, and then exchange information among them. The fast converging behavior of the PSO makes this issue so critical for multimodal problems. Al-Kazemi and Mohan [10] divided the population into two sets to achieve a more diverse, one set moving to the gbest while another moving in opposite direction. After some generations, if the gbest would not improve, the particles would switch their group. Two cooperating swarms was used by Baskar and Suganthan [11] to search concurrently for a solution along with sharing the gbest information of two swarms. The two swarms track the *gbest* if it improves. Each swarm using different update equation: One uses the standard PSO while the other uses the Fitness-to-Distance ratio PSO [12]. Their approach improved the performance in solving single objective optimization problems. Then an improved algorithm was proposed by El-Abd and Kamel [13] through adding a twoway flow of information between two swarms. After running a fixed generations, if the best *p* particles improve, then they willl replace the worst *p* particles in the other swarm. This guarantees exchanging new information from the other swarm's experience for the two

swarms.

In this study, A new learning mechanism is introduced among swarms. At each iteration, the particles not only update themselves according to the best particle of their own swarm, but also learn information from the best particle of other swarms. The information sharing and learning mechanism make swarm extend their search space and speedup the convergence speed. The information sharing and learning mechanism that we call it crossover operation is described as follows:

Step1: for each particle of swarm *k*, randomly select a best particle *p'* from a random swarm.

Strp2: for each dimension *i* of particle *p'*s position *px[i]* and velocity *pv[i]* , if rand()<q_c,
crossover particle *p* with *p'* as follows:

$px[i]=(1-\alpha)*px[i]+\alpha*p'x[i]$.

$pv[i]=rand()*(p'x[i]-px[i])$.

Step3: if all particles of swarm *k* are updated, end the operator, else go to Step 1.

where q_c is crossover rate , α is a random number of (0,1).

4 Experiments and Results

Eight benchmark functions (f_1-f_8) are used in this paper. Table 1 gives the details of these functions. Algorithm parameters are as follows for all experiments: acceleration constants of η_1 and η_2 are both set to be 1.496180, and inertia weight $\omega=0.729844$ as suggested by den Bergh [14], crossover rate q_c is 0.8, running time is 50. In order to be the same number of function evaluations to PSO, a particle will be evaluated two times each generation in FPSO and FMSO. The other parameters are given in the following experiments.

Two groups of experiments are carried out in this section.

Firstly, FMSO is compared with standard PSO and FPSO to show the performance of FMSO algorithm on 10 problems. In the experiments, the number of population is 60 , the tournament size ($q=10$) is chosen for FPSO. 3 swarms are used in FMSO, swarm size is 20, the tournament size ($q=5$) is chosen and crossover rate q_c is 0.6 for FMSO. The other parameters are the same as the above. Fig 1 shows the comparison results of the process with the same evaluations and table 3 show the statistical results for all test problems over 50 runs. By viewing the results of Fig 1 and table 3, we can easily see that FPSO show better performance than PSO on function f_2,f_3, f_6 and f_7. All the results of FMSO are better than PSO. The results of Table 3 demonstrate that FMSO finds the global optima for function f_2, f_3, f_5, f_6, f_7 and f_8, especially for function f_2, f_3 and f_7, the global optima of them are fond for each run over 50 runs. From the comparison results, we can know that Cauchy mutation is helpful for some problems, and the multiple swarms technique works for all test problems. It indicates that FMSO not only speeds up the search, but also improves the optimizing precision.

Table 1.Details of test functions, where n is the dimension of the function, fmin is the minimum value of the function, $S \subseteq Rn$

Test function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	(-5.12,5.12)	0
$f_2(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	(-500,500)	-12569.5
$f_3(x) = 6 \cdot \sum_{i=1}^{5} \lfloor x_i \rfloor$	30	(-5.12, 5.12)	0
$f_4(x) = \sum_{i=1}^{n} i \cdot x_i^4 + U(0,1)$	30	(-1.28,1.28)	0
$f_5(x) = \frac{(\sin^2 \sqrt{x^2 + y^2}) - 0.5}{(1.0 + 0.001(x^2 + y^2))^2} + 0.5$	2	(-100.0,100.0)	0
$f_6(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	30	(-300.0,300.0)	0
$f_7(x) = -20 \cdot \exp(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi \cdot x_i)) + 20 + e$	30	(-30.0,30.0)	0
$f_8(x) = \sum_{i=1}^{n} 100((x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	(-2.048,2.048)	0

Secondly, the aim of group 2 is to analyze the effect of different swarms and swarm size(*m*n*) for a same population size *80*. Function f_1,f_4,f_6,f_8 are chosen to test. the maximum generation is *1000*, *m* and *n* are respectively the number of swarms and swarm size. *4* sets experiments are conducted, Table 2 shows the value of *m* and *n*. All the other parameters are the same as above. By viewing the comparison results of Table 4, the more number swarms is better for function f_4 and f_6 ,however , it isn't the case for function f_1 and f_8. That is to say, although multiple swarms can speed up the convergence rate, not the more the better. it is hard to find a optimal swarm size and number of swarms for general problems. Actually, the optimal swarm size and swarm numbers depend on the distributions of optimal solutions and number of optimal solutions. For functions with a few optimum, small number of swarms might be enough. However, for functions with a lot of optimum, large swarms might be needed.

Table 2 The value of swarms(*m*) and swarm size(*n*),*q* is the tournament size

	Set 1	Set 2	Set 3	Set 4
m	*2*	*4*	*8*	*16*
n(q)	*40(6)*	*20(5)*	*10(4)*	*5(3)*

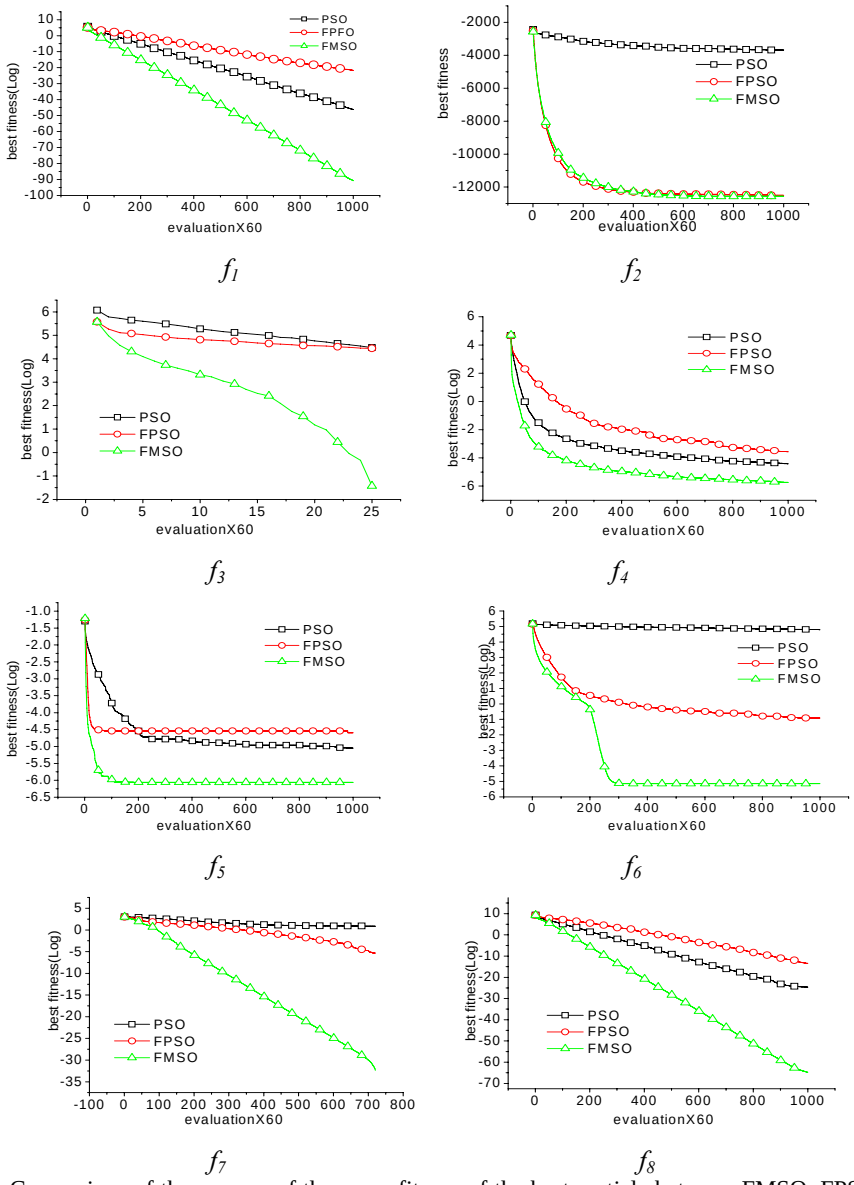

Fig1. Comparison of the process of the mean fitness of the best particle between FMSO, FPSO and PSO for 50 runs, the vertical axis is the function value and the horizontal axis is the number of evaluations.

Table 3 The maximum(*Max*) ,minimum(*Min*) and average(*Avg*) best fitness over *50* runs. *Std* is the standard deviation

F	evaluations		PSO	FPSO	FMSO
f_1	$6*10^4$	*Max*	*3.42138E-20*	*8.8762e-009*	*3.87449e-039*
		Min	*1.24857E-23*	*5.53311e-014*	*1.08821e-042*
		Avg	*5.06623E-21*	*3.73898e-010*	*3.4003e-040*
		std	*8.87525e-021*	*1.48328e-009*	*6.55003e-040*
f_2	$6*10^4$	*Max*	*-2793.79*	*-12086.3*	*-12569.5*
		Min	*-4836.19*	*-12569.5*	*-12569.5*
		Avg	*-3682.48*	*-12506.2*	*-12569.5*
		std	*457.426*	*101.935*	*0*
f_3	*6*250*	*Max*	*0*	*0*	*0*
		Min	*0*	*0*	*0*
		Avg	*0*	*0*	*0*
		std	*0*	*0*	*0*
f_4	$6*10^4$	*Max*	*0.024044*	*0.0575288*	*0.009357*
		Min	*0.00349205*	*0.0101007*	*0.000910975*
		Avg	*0.0120778*	*0.0284553*	*0.00321394*
		std	*0.00527977*	*0.0109074*	*0.00156956*
f_5	$6*10^4$	*Max*	*0.00971591*	*0.126991*	*0.00971591*
		Min	*0*	*0*	*0*
		Avg	*0.00641252*	*0.0100854*	*0.00233185*
		std	*0.00460249*	*0.0177373*	*0.00414948*
f_6	$6*10^4$	*Max*	*140.958*	*2.16826*	*0.0245904*
		Min	*87.6869*	*2.68284e-011*	*0*
		Avg	*120.464*	*0.39966*	*0.00580988*
		std	*11.4045*	*0.528499*	*0.00774408*
f_7	$6*10^4$	*Max*	*6.25549*	*0.000532368*	*0*
		Min	*3.12093e-007*	*5.90058e-007*	*0*
		Avg	*2.25332*	*5.85617e-005*	*0*
		std	*1.07449*	*0.000107675*	*0*
f_8	$6*10^4$	*Max*	*9.60947e-010*	*1.31399e-005*	*4.25245e-028*
		Min	*5.7947e-015*	*2.24269e-009*	*0*
		Avg	*2.07824e-011*	*1.36425e-006*	*7.15891e-029*
		std	*1.34386e-010*	*2.76779e-006*	*8.37964e-029*

5 Conclusions

By analyzing the advantage and disadvantage of the standard PSO, FPSO based on Cauchy mutation and evolutionary selection strategy is proposed in this paper. Although FPSO greatly overcomes the tendency of trapping into local optima of PSO, the convergence rate isn't satisfied, so a multiple swarms algorithm(FMSO) based on FPSO is proposed by introducing a crossover operation, FMSO is tested on *8* benchmark functions. From the experimental results of these functions, it can be seen that the FMSO performed much better than PSO and FPSO on the selected problems.

Only functions with the dimension less than *30* were tested in this paper. Further research will focus on testing the performance of FMSO on higher dimensional problems in order to find whether FMSO would scale up well for the large function optimization problems.

Table 4 Comparison with different swarms and swarm size for FMSO over *50* runs, The maximum(*Max*) ,minimum(*Min*) and average(*Avg*) best fitness over *50* runs. *Std* is the standard deviation

F		2 swarms*40	4 warms*20	8 swarms*10	16swarm*5
f_1	*Max*	*1.97774e-39*	*4.37499e-40*	*3.97464e-40*	*5.18094e-34*
	Min	*3.68037e-44*	*1.63423e-42*	*8.54154e-44*	*6.24203e-36*
	Avg	*1.36448e-40*	*4.47269e-41*	*2.01959e-41*	*1.11419e-34*
	std	*3.89216e-40*	*8.05011e-41*	*5.88923e-41*	*1.04846e-34*
f_4	*Max*	*0.00573433*	*0.0050845*	*0.0050845*	*0.00284168*
	Min	*0.000839845*	*0.000748565*	*0.000748565*	*0.000767533*
	Avg	*0.00304075*	*0.00184794*	*0.00184794*	*0.00168235*
	std	*0.00110113*	*0.000686807*	*0.000686807*	*0.000493363*
f_6	*Max*	*0.0513256*	*0.0319228*	*0.0343841*	*0.00739604*
	Min	*0*	*0*	*0*	*0*
	Avg	*0.00934373*	*0.00285618*	*0.00708914*	*0.000443762*
	std	*0.011266*	*0.00607201*	*0.00913191*	*0.00175646*
f_8	*Max*	*1.57772e-028*	*3.59918e-028*	*6.40949e-29*	*4.47899e-21*
	Min	*0*	*0*	*0*	*1.95173e-23*
	Avg	*8.94864e-30*	*9.30856e-29*	*3.57452e-30*	*3.94318e-22*
	std	*2.75515e-29*	*7.01503e-29*	*1.22276e-29*	*6.52812e-22*

References

[1] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, IEEE International Conference on Neural Networks, pp.1942-1948, 1995.

[2] R. C. Eberhart and J. Kennedy, A New Optimizer Using Particle Swarm Theory, Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp.39-43, 1995.
[3] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transactions on Evolutionary Computation, vol. 8,no. 3, pp. 240–255, 2004.
[4] J. Sun, B. Feng, W. Xu, Particle swarm optimization with particles having quantum behavior, in Proceedings of the IEEE Congress on Evolutionary Computation, Portland, Oregon USA, pp. 325-331, 2004.
[5] Jing Liu, Wenbo Xu, Jun Sun, Quantum-behaved particle swarm optimization with mutation operator. Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence Pages: 237 - 240 , 2005
[6] R. A. Krohling, Gaussian particle swarm with jumps, in Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK,pp. 1226-1231, 2005.
[7] R. A. Krohling, L. dos Santos Coelho, PSO-E: Particle Swarm with Exponential Distribution , in Proceedings of the IEEE Congress on Evolutionary Computation, pp1428- 1433, July 2006.
[8] H. Narihisa, T. Taniguchi, M. Ohta, and K. Katayama, Evolutionary Programming with Exponential Mutation, in Proceedings of the IASTED Artificial Intelligence and soft Computing, Benidorn, Spain, pp. 55-50, 2005.
[9] M. Clerc and J. Kennedy, The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space, IEEE Trans. on Evolutionary Computation, Vol.6,pp:58-73,2002.
[10] B. Al-Kazemi and C. K. Mohan, “Multi-phase discrete particle swarm optimization” In Proc. of 4th Int. Workshop on Frontiers on Evolut.Alg., Research Triangle Park, NC, 2002.
[11] S. Baskar, and P. N. Suganthan, “A novel concurrent particle swarm optimization,” In Proc. of Cong. on Evolut. Comput., Portland, OR, pp. 792-796, 2004.
[12] T. Peram, K. Veeramachaneni, and C. K. Mohan, “Fitness-distanceratio based particle swarm optimization,” In Proc. of IEEE Swarm Intell. Symp., Indianapolis, IN, pp. 88-94, 2003.
[13] M. El-Abd, and M. Kamel, “Information exchange in multiple cooperating swarms,” In Proc. of Cong. on Evolut. Comput., Edinburgh, UK, pp. 138-142, 2005.
[14] F. van den Bergh. An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.