A Fast Particle Swarm Optimization Algorithm with Cauchy Mutation and Natural Selection Strategy

Changhe Li[1], Yong Liu[2], Aimin Zhou[3], Lishan Kang[1], Hui Wang[1]

[1]China University of Geosciences, School of Computer, Wuhan, P.R.China, 430074
[2]The University of Aizu,Aizu-Wakamatsu, Fukushima,Japan, 965-8580
[3]Department of Computer Science,University of Essex Wivenhoe Park, Colchester,CO4 3SQ
lch_wfx@yahoo.com.cn, yliu@u-aizu.ac.jp, azhou@essex.ac.uk, kang_whu@yahoo.com, wanghui_cug@yahoo.com.cn

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 fast particle swarm optimization (FPSO) algorithm is proposed by combining PSO and the Cauchy mutation and an evolutionary selection strategy. The idea is to introduce the Cauchy mutation into PSO in the hope of preventing PSO from trapping into a local optimum through long jumps made by the Cauchy mutation. FPSO has been compared with another improved PSO called AMPSO [12] on a set of benchmark functions. The results show that FPSO is much faster than AMPSO on all the test functions.

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]. PSO 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:

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

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

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 idth 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. The updating equations (1) and (2) show that PSO combines the cognition component of each particle with the social component of all the particles in a group. The social component suggests that individuals ignore their own experience and adjust their behavior according to the previous best particle in the neighborhood of the group. On the other hand, the cognition component treats individuals as isolated beings and adjusts their behavior only according to their own experience.

Although the speed of convergence is very fast, many experiments have shown that once PSO traps into local optimum, it is difficult for PSO to jump out of the 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. Because the expectation of Cauchy distribution does not exist, the variance of Cauchy distribution is infinite. Some researches [9][10] have indicated that the Cauchy mutation operator is good at the global search for its long jump ability. This paper shows that the Cauchy mutation is helpful in PSO as well. Besides the Cauchy mutation, FPSO chooses the natural selection strategy of evolutionary algorithms as the basic elimination strategy of particles. FPSO combines PSO with Cauchy mutation and evolutionary selection strategy. It has the fast convergence speed characteristic of PSO, and greatly overcomes the tendency of trapping into local optima of PSO.

The rest of this paper is organized as follows. Section 2 gives the analysis of PSO. The detail of FPSO is introduced in Section 3. Section 4 describes the experiment setup and presents the experiment results. Finally, Section 5 concludes the paper with a brief summary.

2. Fast Particle Swarm Optimization Algorithm with Cauchy Mutation and Natural Selection Strategy

2.1 Cauchy mutation

From the mathematic theoretical analysis of the trajectory of a PSO particle [11], 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. According to the update equation, the personal best position of the particle will gradually move closer to the global best position. Therefore, all the particles will converge onto 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 $0 \le \omega < 1$. 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. In standard PSO, the position of a particle is updated according to equations (1) and (2). That is, for each particle，there is nowhere to move but following the direction of the best particle, and the flying direction is nearly determinate through the generation. From the above analysis of PSO, the particles incline to converge on a 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 ($1,2,\ldots 2m$) 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 detail of the selection framework are as follows:

Step1: For each particle of parent and offspring, assign $win[i]=0$.

Step2: Randomly select q particles (opponents) for each particle in parent and offspring.

Step3: For each particle, compare it with its q opponents. For particle i, if the fitness of its opponent j is larger than particle i , then $win[i]++$.

Step4: Select m particles that have the more winnings to be the next generation.

2.3 Algorithm framework

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}) fitness, update P_{id} .

Step4: For each particle, if its fitness is smaller than the best one (P_{gd}) of all the 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 the next generation according to the above evolutionary selection strategy.

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

3 Experiments and Results

Twelve benchmark functions (f_1-f_{12}) are used in this paper. Function f_1-f_9 are chosen from [12], and function f_{10}-f_{12} from[9]. Functions f_1-f_4 are unimodal functions while functions f_5-f_{12} have many local optima. Generally speaking, multimodal functions are often regarded as the most difficult in function optimization. Table 1 gives the details of these functions.

Algorithm parameters are as follows: 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 [13]. In FPSO, a particle will be evaluated two times each generation, in order to be the same number of function evaluations to AMPSO [12] and PSO at each generation, the swarm size of FPSO is half of the size of AMPSO and PSO for all experiments. And the other parameters are given in the following experiments.

Two groups of experiments are conducted in this section.

Firstly, the proposed algorithm FPSO is compared with another improved PSO called AMPSO [12] on nine problems. In the experiments, the number of particles is 20 (40 in AMPSO) and other parameters are the same as in [12]. For AMPSO, it stops when no improvement can be made. And then, the mean fitness obtained by AMPSO is calculated as the target value of FPSO, and FPSO will stop when it surpasses this target value. The mean and standard deviation of fitness values achieved and generations spent are shown in Table 2 based on 30 independent runs. The results show that FPSO could find the target values achieved by AMPSO in shorter function evaluations for all test functions except for f_3, The function f_3 is an easy function with integer object function values. Both FPSO and AMPSO could find the global minimum in one or a few generations. It indicates that FPSO has the global search capability while AMPSO could only found the global minimum for functions f_1, f_2, and f_3, but failed on reaching the global minimum for the rest of functions including all multimodal functions. It can be known that FPSO not only has faster convergence, but also has better global search.

Secondly, the dynamics of FPSO and PSO are discussed, and the contribution of Cauchy mutation in FPSO is studied as well. For all experiments in the second group, the number of runs is *50* times, the particles size *m* is *50* for FPSO (100 for PSO), and the tournament size ($q=10$) is chosen for FPSO. The left side in Fig 1 and Fig 2 shows how the best fitness of the particle evolves in the search process. It suggests that PSO could only improve the best fitness for simple unimodal functions but hardly decrease the best fitness for the multimodal functions. The results of f_3,f_4,f_7,f_8,f_{11} and f_{12} are not provided here due to space limitation. Table 3 shows the mean best values and the standard deviation got by FPSO and PSO for 50 runs.

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

Test function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	3	(-5.12,5.12)	0
$f_2(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$	2	(-2.048,2.048)	0
$f_3(x) = 6 \cdot \sum_{i=1}^{5} \lfloor x_i \rfloor$	5	(-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
$f_9(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	(-5.12,5.12)	0
$f_{10}(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	(-500,500)	-12569.5
$f_{11}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	(-5,5)	-1.0316285
$f_{12}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	(-2.0,2.0)	3

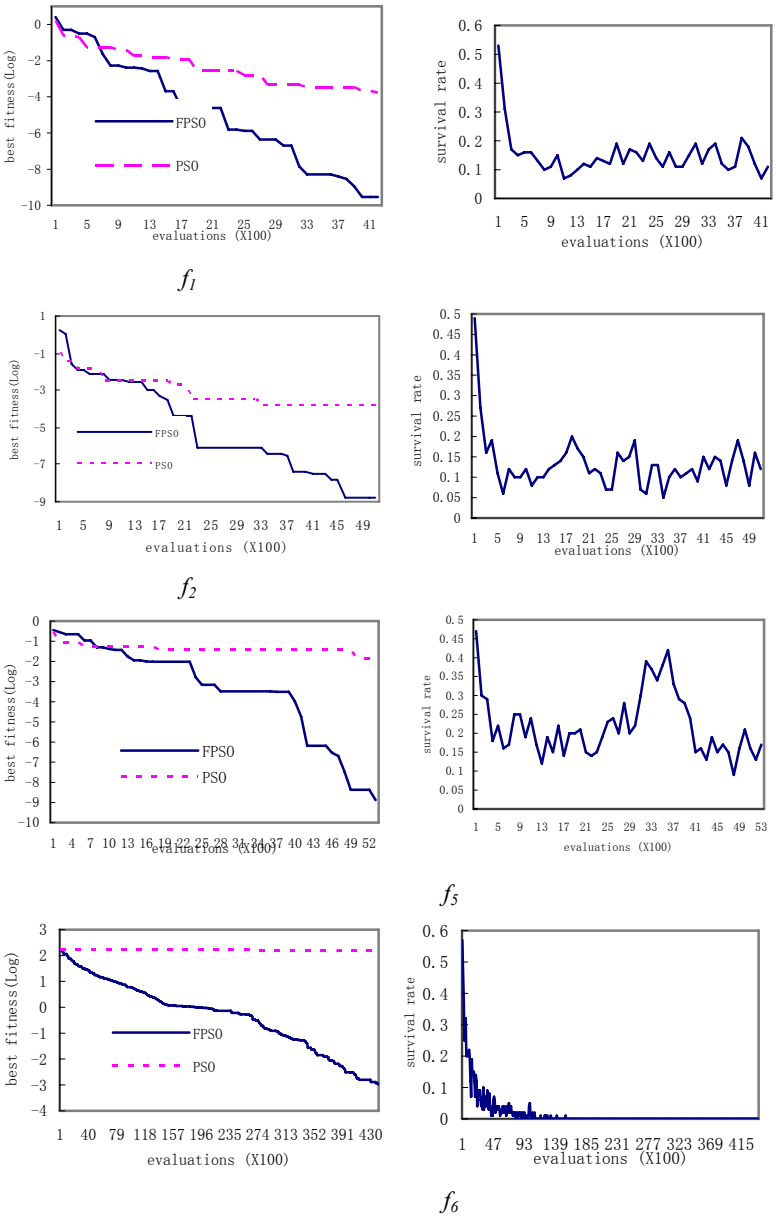
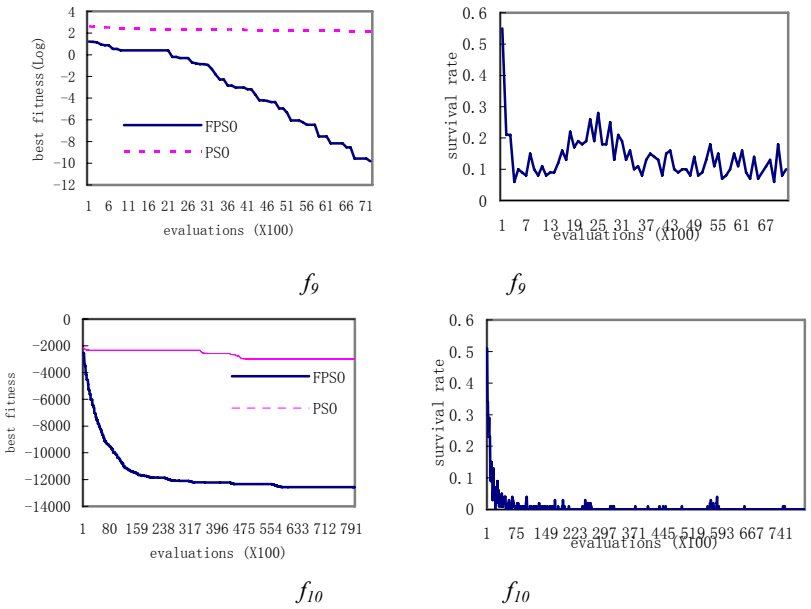


Fig 1 Evolution process of the best fitness in FPSO and PSO, and the survival rate in FPSO



f_9 f_9 f_{10} f_{10}

Fig 2 Evolution process of the best fitness in FPSO and PSO, and the survival rate in FPSO

Table 2. Comparison between FPSO and AMPSO, all results have been averaged over 30 runs, where f is the test function

f	Target Value		Mean Evaluations	
	FPSO	**AMPSO**	**FPSO**	**AMPSO**
f_1	*0.00017748±1.6323e-5*	*0.000300±0*	*656*	*1088.8*
f_2	*2.83656e-5±2.5674e-6*	*0.000049±0.000107*	*858.4*	*64471.2*
f_3	*0.0±0.0*	*0.0±0.0*	*124*	*40*
f_4	*16.9737±0.57268*	*41.469003±1.922524*	*1801.2*	*32499.6*
f_5	*0.01501±0.00079774*	*0.024139±0.004212*	*446.4*	*312049*
f_6	*63.3212±0.634179*	*106.118084±4.398166*	*660*	*79524.4*
f_7	*19.4785±0.0242021*	*19.660458±0.056733*	*1444*	*33084.8*
f_8	*2058.51±24.6575*	*2198.368745±86.302836*	*2804*	*22173.2*
f_9	*214.475±1.5299*	*224.765413±35.262816*	*2632*	*18287.2*

The right side in Fig 1 and Fig 2 shows the survival rate of Cauchy mutation in FPSO through the evolution process. The survival rate is the ratio of the number of successful Cauchy mutation to the total number of Cauchy mutations. The higher survival rate, the more particles generated from Cauchy mutation survive. The survival rate remains at a certain level through the whole search process for some test functions, while it decreased for other test functions after the objective values of these functions close to the global minimum had been found. It can be explained that when the particles are far away from the global minimum, long jump made by Cauchy

mutation was helpful to find the particles with smaller objective values. Once the particles were getting closer to the global optimum, FPSO would have to depend on the small search steps decided by the equations (1) and (2) rather than the long jump made by Cauchy mutation in order to fine tune the solutions. By viewing the results of Table 3, we can easily see that the mean best values got by FPSO are obvious better than PSO for function f_1,f_2,f_5,f_6,f_7 ,f_{10},f_{11},f_{12} , and slightly better for function f_4. For function f_3 and f_9, the performance of FPSO are nearly the same to PSO. Only for function f_8, the convergence speed of FPSO is slower than PSO, however if given more number of generation, FPSO will converge on the global optima. For most test cases, FPSO is more efficient than the standard PSO.

Table 3 Comparison between FPSO and PSO on f1-f12, where "Mean Best" is mean best function values found in the last run, and "Std Dev" indicates the standard deviation.

f	Number of evaluations	**FPSO**		**PSO**	
		Mean Best	Std Dev	Mean Best	Std Dev
f_1	*1.5e4*	*2.299e-17*	*9.784e-18*	*4224.77*	*201.038*
f_2	*2.5e3*	*0*	*0*	*1.145e-13*	*6.509e-14*
f_3	*300*	*0*	*0*	*0*	*0*
f_4	*1.5e4*	*3.668e-3*	*2.072e-4*	*4.63e-3*	*2.100e-4*
f_5	*1.5e4*	*0*	*0*	*0.005*	*6.848e-4*
f_6	*1.5e4*	*0.082*	*0.031*	*101.41*	*1.523*
f_7	*1.5e4*	*1.356e-09*	*4.141e-10*	*1.306*	*0.148*
f_8	*1.5e4*	*7.53016e-14*	*4.53682e-14*	*4.59643e-26*	*1.83991e-26*
f_9	*1.5e4*	*0*	*0*	*0*	*0*
f_{10}	*1.5e4*	*-12563.6*	*3.437*	*-4005.02*	*54.012*
f_{11}	*4000*	*-1.03163*	*0*	*-1.03162*	*4.05433e-6*
f_{12}	*4000*	*3*	*5.01747e-10*	*3.00004*	*6.87691e-6*

4 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 PSO has a fast convergence rate, it is likely to trap into the local optimum, and can't guarantee converge to the global optimum. FPSO introduces Cauchy mutations into the position and velocity update equations in order to increase the probability of jumping out of the local optimum. FPSO was tested on *12* benchmark functions. From the experimental results of these functions, it can be seen that the FPSO performed much better than AMPSO on the selected problems. Although FPSO needs more time to perform Cauchy mutation, it decreases the fitness evaluations remarkably comparing to AMPSO.

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

optimization problems.

5 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] X. Yao and Y. Liu. Fast evolutionary programming, Proc. of the Fifth Annual Conference on Evolutionary Programming (EP'96), San Diego, CA, USA, 29/2-2/3/96. (1996).451-460, the MIT Press.

[10] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster. IEEE Trans. Evolutionary Computation 82–102, 1999.

[11] 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.

[12] G. Pampara, N. Franken, A. P. Engelbrecht, Combining Particle Swarm Optimisation with angle modulation to solve binary problems. In proceedings of the IEEE Congress on Evolutionary Computation, pages 89- 96, Sept. 2005.

[13] F. van den Bergh. An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.