01 Jan 2006

Empirical Study of an Unconstrained Modified Particle Swarm Optimization

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Phillip W. Moore

Recommended Citation

G. K. Venayagamoorthy and P. W. Moore, "Empirical Study of an Unconstrained Modified Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Evolutionary Computation, 2006*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2006.
The definitive version is available at https://doi.org/10.1109/CEC.2006.1688483

2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Empirical Study of an Unconstrained Modified Particle Swarm Optimization

Phillip W. Moore, *Student Member, IEEE* and Ganesh K. Venayagamoorthy, *Senior Member, IEEE*

***Abstract*— In this paper, an unconstrained modified particle swarm optimization (UMPSO) algorithm is introduced and studied empirically. Four well known benchmark functions, with asymmetric initial position values, are used as testing functions for the UMPSO algorithm. The UMPSO is a variation of the canonical PSO in which the velocity and position is unconstrained, an additional strategic component is added, and the social component term has been modified. The strategy component is used instead of varying parameters or mutation to enhance diversity in the swarm during the search. The UMPSO algorithm is then compared to results obtained from the constrained canonical PSO (CPSO) and the unconstrained canonical PSO (UPSO). The results show that UMPSO algorithm with no maximum velocity and position, and no minimum velocity and position value that performs better than the CPSO and the UPSO for the Sphere, Rosenbrock, Rastrigrin, and Griewank benchmark functions.**

I. Introduction

THERE are many algorithms used in optimization that utilize population search techniques. The genetic algorithm, genetic programming, differential evolution, and other evolutionary computational algorithms are all motivated by evolution as seen in nature. The particle swarm optimization (PSO) algorithm, however, is motivated from the imitation of social behavior individually and in groups. The PSO, developed by James Kennedy and Russell Eberhart [1, 2], is modeled after movement patterns and social interactions as seen in flocks of birds and schools of fish. Instead of using selection, crossover, and mutation operators to manipulate individuals in the population, each individual is flown through a hyper dimensional search space, containing the solutions for given problems, and coordinates their flying experiences through the population.

In the PSO algorithm, each particle performs an evaluation of its position, called the fitness evaluation, through each time step in the search space. The best performance of each particle and respective location is then stored into memory and called the pbest of the particle. Then the best of the pbest in the population, called the gbest, is stored into memory. The concept of PSO lies in the acceleration of each particle towards its pbest and the gbest locations at each time step, and is modified by its own inertia.

In this paper, an unconstrained modified version of the canonical PSO is presented (UMPSO). The UMPSO is a modified version of the canonical PSO with no constraints of maximum velocity, maximum position, minimum velocity, and minimum position. The canonical PSO is modified by using a constriction factor and a new term, called the strategy component, to increase the performance of the PSO algorithm's velocity update equation. The strategy component is used instead of varying parameters or mutation to enhance diversity in the swarm during the search. The social component term of the velocity update equation is also modified such that the term is solely based on the global best position of the swarm.

The rest of the sections of this paper are organized as follows: Section II explains the different PSO algorithms (Constrained/Unconstrained Canonical PSO and UMPSO). Section III describes the benchmark functions used for testing the two improved PSO algorithms. Section IV describes the experimental settings. Section V presents the results obtained from the constrained canonical PSO (CPSO), the unconstrained canonical PSO (UPSO), and the UMPSO. Finally, some discussion and conclusions are given in Sections VI and VII, respectively.

II. Particle Swarm Algorithms

The canonical PSO and a modified PSO are studied in this paper. The CPSO and UPSO results are compared with the results obtained from the UMPSO. The canonical PSO, first proposed by James Kennedy and Russell Eberhart [1], sets maximum velocity, maximum position, minimum velocity, and minimum position limits. In flocks of birds and schools of fish, these restraints are not given. In real life, birds and fish are able to move freely throughout their environment. Though there are boundaries such as air, water, and speed, evolution may theoretically propel birds or fish beyond these boundaries if need be. The UMPSO algorithm reflects these observations and places no constraints on particles' (birds, fish, etc.) velocities and positions.

A. Constrained/Unconstrained Canonical Particle Swarm

In the canonical PSO algorithm, the velocity of a particle, i, in dimension, d, is obtained using (1). The subscript i from (1) takes on values from 1 to I, where I is the maximum number of particles in the population. Each particle is assigned to a positional point in a D-dimensional search space. In (1), $v_{id}(t+1)$ is the new velocity for the i^{th}

This work is supported by the National Science Foundation CAREER grant ECS #0348221.

Phillip. W. Moore and Ganesh K. Venayagamoorthy with the Real-Time Power and Intelligent Systems (RTPIS) Laboratory, Department of Electrical and Computer Engineering, University of Missouri – Rolla, 1870 Miner Circle, Rolla, MO 65409, USA (email: pwmpn2@umr.edu, gkumar@ieee.org)

0-7803-9487-9/06/$20.00/©2006 IEEE

particle at dimension *d*. The first term of (1), $v_{id}(t)$ is the current velocity for the i^{th} particle at dimension *d*. The values for *w*, c_1, and c_2 represent the inertia constant, cognitive acceleration constant, and social acceleration constant, respectively. The terms rand1( ) and rand2( ) are uniform random numbers between 0 and 1. The random values change for each particle. The term $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ is the current personal best fitness for the i^{th} particle, where D is the maximum number of dimensions.

$$v_{id}(t+1) = \left\{ \begin{array}{l} w \times v_{id}(t) + \\ c_1 \times \text{rand1}(\,) \times \left[p_{id}(t) - x_{id}(t) \right] + \\ c_2 \times \text{rand2}(\,) \times \left[g_d(t) - x_{id}(t) \right] \end{array} \right\} \quad (1)$$

There are different topologies that can be chosen for the canonical PSO. The main two are the star and the ring topologies. The star topology allows for each particle to communicate to the swarm's best particle (*g*). In the canonical PSO used in this paper, the star topology is used, and the swarm's best particle is $g = (g_1, g_2, \ldots, g_D)$. The ring topology allows for each particle to communicate to the best particle in its neighborhood (*l*). The ring topology is not studied in this paper.

For the canonical PSO, the positions of the particles are updated using (2). In (2), $x_{id}(t+1)$ is the new position of the i^{th} particle at dimension *d*, and the first term, $x_{id}(t)$, is the current position of the i^{th} particle at dimension *d*. The summation of $x_{id}(t)$ and $v_{id}(t+1)$ yields the new position of the i^{th} particle.

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

The CPSO and UPSO both use (1) and (2) for the velocity and position updates. For a CPSO the velocities and positions have a respective maximum and minimum value. If a velocity or position exceeds these values, the velocity or position is set to the respective maximum or minimum value. For a UPSO, there is no maximum or minimal value for the velocities and positions to exceed.

B. Unconstrained Modified Particle Swarm

The UMPSO is a modified version of the UPSO such that there is a constriction factor, a strategic component, and a modified social component of the UPSO. The modified social component is impacted only by the global best position of the swarm. The constriction factor for PSO was brought about by Clerc's constriction factor [3]. Clerc's constriction factor is normally used to ensure convergence. A detailed discussion of the constriction factor is beyond the scope of this paper, but a simplified method of incorporation with the UMPSO is shown in (3) and (4) where the values of c_3 and c_4 are set to 2.05 and the constriction factor, *K*, is set to 0.729. With c_3 and c_4 set to 2.05, When the values of c_3 and c_4 are multiplied by the constriction factor of 0.729 to obtain c1 and c_2, the values of c_1 and c_2 become 1.49.

The strategic component that is added to the PSO is used, instead of using varying parameters or mutation, to enhance diversity. The strategy component is equal to the inner product of an N-dimensional number *randN* (N is the dimension of the solution space and each dimension of *randN* is uniformly distributed random numbers in the range from 0 to 1) and the difference between the *gbest* and *pbest*. For example, the vector size for pbest will equal N (the number of dimensions). The vector for the swarm's gbest will also have a vector size of N. Therefore the difference of these terms will also produce a vector of size N. The dot product of a vector of random numbers of vector size N and the difference of the two terms will produce a scalar value. This scalar value is referred to as the strategy component. The strategy component will have the same value for each dimension of a single particle. This strategy component prevents pseudo-convergence or undesired maturity by increasing search diversity at an early stage when *pbest* and *gbest* are very different. As the *pbest* and *gbest* values converge to a solution, the difference between these values will converge to zero. One may question whether the convergence of PSO algorithm is affected by adding the social component. The PSO algorithm is not affected in practice, because the simulations always converge. In theory, the social component does not break the social behavior based on which PSO is developed. The equation for the positional update of UMPSO is the same as the canonical PSO update equation given in (2). Using this equation, the coordinates of gbest is enforced in the value of the velocity rather than moving towards the gbest from the current position.

$$v_{id}(t+1) = \left\{ \begin{array}{l} K \times \left\{ \begin{array}{l} v_{id}(t) + \\ c_1 \times \text{rand1} \times \left[p_{id}(t) - x_{id}(t) \right] + \\ c_2 \times \text{rand2} \times g_d(t) \end{array} \right\} + \\ \\ \text{randN} \bullet \left[p_i(t) - g(t) \right] \end{array} \right\} \quad (3)$$

$$K = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4 \times \varphi} \right|}, \varphi = c3 + c4, \varphi > 4 \quad (4)$$

III. Benchmark Functions

In this paper, four benchmark functions have been used for optimization problems [4, 5, 6]. The functions are the Sphere function (5), Rosenbrock's function (6), Rastrigrin's function (7), and Griewank's function (8). These four benchmark functions test the performance of the CPSO, UPSO, and UMPSO.

$$f_1(x) = \sum_i^n x_i^2 \quad (5)$$

$$f_2(x) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i\right)^2 + \left(x_i - 1\right)^2 \quad (6)$$

$$f_3(x) = \sum_{i=1}^{n} \left[x_i^{\,2} - 10 \times \cos\left(2 \times \pi \times x_i\right) + 10 \right] \quad (7)$$

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

IV. Experimental Settings

For our experiment in optimization of four benchmark functions, the asymmetric initialization method is used for population initialization [4, 5, 7]. Table I lists the position initialization ranges for the four benchmark functions. The four benchmark functions are minimized for the experiments carried out. The constraints for the CPSO algorithm are given in Table II for each benchmark function. There is no velocity and position constraint for UPSO and UMPSO, as explained in Section II, for all four benchmark functions. For each algorithm and benchmark function, a population size of 20, 40, and 80 are examined. Also, for each algorithm, benchmark function, and population size, a dimension size of 10, 20, 30, 50, and 100 are examined. The number of iterations ran for each dimension is given in Table III.

TABLE I
ASYMMETRIC INITIALIZATION OF POSITIONS

Function	Asymmetric Initialization Range
f_1	(50, 100)
f_2	(15, 30)
f_3	(2.56, 5.12)
f_4	(300, 600)

TABLE II
VELOCITY AND POSITION CONSTRAINTS FOR CPSO

Function	Minimum/Maximum Velocity and Position
f_1	-100 / 100
f_2	-100 / 100
f_3	-10 / 10
f_4	-600 / 600

TABLE III
MAXIMUM ITERATIONS

Dimension	Iterations
10	1000
20	1500
30	2000
50	5000
100	5000

V. Results

The CPSO, UPSO, and UMPSO algorithms are all used to minimize four benchmark functions (Sphere, Rosenbrock, Rastrigrin, and Griewank). The parameters for CPSO and UPSO are: $w = 0.8$, $c_1 = 2$, and $c_2 = 2$. The parameters for UMPSO are: $K = 0.729$, $c_1 = 1.49$, and $c_2 = 1.49$. When applying the same w, c_1, and c_2 parameters to the UMPSO, the results did not turn out as well. When applying the constriction factor to the CPSO and UPSO (instead of using w, c_1, and c_2 parameters) the newly obtained results compared to the CPSO and UPSO were worse. This is why these three variations for the CPSO, UPSO, and UMPSO were not used, and this is why I have selected the given parameters for CPSO, UPSO, and UMPSO. Tables IV – VII show the results of CPSO, UPSO, and UMPSO algorithms for the four benchmark functions. All of the results gathered in the tables are the averaged minimal fitness values from the benchmark functions over 50 trials. All of the results gathered for each benchmark function, population size, dimensions size, and PSO algorithms are averaged over 50 test runs.

A. Sphere Function

The results obtained from optimizing the Sphere function (5) show that the CPSO and UPSO algorithms perform worse as the dimension of the function increases and as the population decreases. For the UMPSO algorithm, as the dimension of the function increases up to 50 the performance of the algorithm increases. As the dimension of the Sphere function increases to 100 the performance of the UMPSO algorithm slightly decreases. The increase in population size for the UMPSO has little effect on the UMPSO algorithm's performance. The performance of UMPSO was better than UPSO and CPSO for every population size and dimension size combinations. Table IV shows the results of each algorithm for the various population and dimension sizes.

B. Rosenbrock Function

The results obtained from optimizing the Rosenbrock function (6) show that the CPSO and UPSO algorithms perform worse as the dimension size of the function increases and as the population size decreases. For the UMPSO algorithm, as the dimension size increases, the performance of the UMPSO decreases. An increase in population size has little effect on the UMPSO algorithm's performance. The UPSO is only able to obtain better results than the UMPSO for a high population size and small dimension size. For the rest of the results, a clear majority, the performance of the UMPSO beats the performances of the CPSO and UPSO. The CPSO outperformed the UPSO for most of the population and dimension size variations. Table V shows the results of each algorithm for the various population and dimension sizes.

C. Rastrigrin Function

The results from optimizing the Rastrigrin function (7) are the same as the results from the Sphere function (5) for CPSO and UPSO. As the population size increases and the

dimension size decreases, the performance of CPSO and UPSO gets better. The UMPSO algorithm, however, is able to find the minimal solution for every variation of population sizes and dimension sizes. The UMPSO outperforms the CPSO which outperforms the UPSO for every variation of population sizes and dimension sizes. Table VI shows the results of each algorithm for the various population and dimension sizes.

D. Griewank Function

The results obtained from optimizing the Griewank function (8) varies for the different dimension sizes. This is due to the fact that as the dimension size increases, so does the number of iterations. As the population size increases, the performance of CPSO and UPSO get better.

TABLE IV
EMPIRICAL EVALUATION WITH THE SPHERE FUNCTION

Pop	Dim	Iter	CPSO	UPSO	UMPSO
20	10	1000	14.3375 ± 9.0613	1.8337e+003 ± 2.2648e+003	2.9479e-045 ± 1.8670e-044
	20	1500	168.9215 ± 84.2691	4.1081e+004 ± 1.3630e+004	2.9349e-070 ± 2.0723e-069
	30	2000	604.6584 ± 190.9694	8.5829e+004 ± 2.0132e+004	3.3207e-102 ± 2.0754e-101
	50	5000	1.0118e+003 ± 266.0150	1.9436e+005 ± 2.1353e+004	2.3641e-264 ± 0
	100	5000	1.0024e+004 ± 1.5531e+003	4.6945e+005 ± 2.7273e+004	1.5618e-255 ± 0
40	10	1000	2.7040 ± 1.4629	18.0193 ± 59.7917	8.3325e-044 ± 5.4542e-043
	20	1500	61.1763 ± 24.3178	1.5950e+004 ± 8.8459e+003	4.5850e-071 ± 2.4707e-070
	30	2000	186.2706 ± 56.2312	5.2100e+004 ± 1.5665e+004	1.0353e-100 ± 5.5202e-100
	50	5000	244.4477 ± 82.9629	1.3965e+005 ± 2.3585e+004	1.6726e-262 ± 0
	100	5000	3.6192e+003 ± 406.4337	4.0929e+005 ± 2.9349e+004	1.1442e-264 ± 0
80	10	1000	0.6159 ± 0.3504	3.9297e-006 ± 1.6520e-005	2.0069e-045 ± 8.4195e-045
	20	1500	13.4140 ± 6.5374	2.1178e+003 ± 2.0065e+003	1.6170e-067 ± 1.1433e-066
	30	2000	41.1833 ± 14.1383	1.7909e+004 ± 7.9992e+003	1.7909e-099 ± 1.2563e-098
	50	5000	64.0966 ± 14.5559	9.2552e+004 ± 1.8420e+004	2.3809e-261 ± 0
	100	5000	1.5826e+003 ± 327.8356	3.2109e+005 ± 3.4460e+004	1.1396e-259 ± 0

TABLE V
EMPIRICAL EVALUATION WITH THE ROSENBROCK FUNCTION

Pop	Dim	Iter	CPSO	UPSO	UMPSO
20	10	1000	3.3501e+003 ± 1.9417e+003	2.8888e+003 ± 1.1333e+003	7.1207 ± 1.0808
	20	1500	1.1217e+004 ± 1.1787e+003	9.4947 ± 1.0699e+003	15.6063 ± 1.4108
	30	2000	1.8477e+004 ± 1.5896e+003	1.6315e+004 ± 1.3921e+003	24.5977 ± 1.5131
	50	5000	3.0136e+004 ± 1.5237e+003	3.0570e+004 ± 2.5609e+003	41.8011 ± 2.6084
	100	5000	7.8885e+004 ± 5.0767e+003	6.6296e+004 ± 3.8205e+003	86.3193 ± 4.8965
40	10	1000	3.2405e+003 ± 1.6834e+003	8.3996 ± 27.3871	6.6161 ± 1.2925
	20	1500	9.5915e+003 ± 569.4242	8.2952e+003 ± 562.7265	15.8352 ± 1.7466
	30	2000	1.5240e+004 ± 784.0405	1.3732e+004 ± 539.9592	24.4118 ± 1.9519
	50	5000	2.6365e+004 ± 1.4284e+003	2.5163e+004 ± 939.1171	42.2297 ± 3.1453
	100	5000	6.6149e+004 ± 4.5768e+003	5.5273e+004 ± 1.7811e+003	86.7361 ± 5.3449
80	10	1000	901.7533 ± 1.6309e+003	1.0466e-006 ± 4.2976e-006	6.6194 ± 1.6081
	20	1500	8.7857e+003 ± 469.8712	3.9806e+003 ± 2.0879e+003	15.2034 ± 1.5940
	30	2000	1.4472e+004 ± 591.1318	1.2229e+004 ± 820.4599	24.8515 ± 2.6669
	50	5000	2.4241e+004 ± 731.7662	2.2841e+004 ± 478.4604	42.9488 ± 3.7506
	100	5000	5.6412e+004 ± 1.6794e+003	4.9655e+004 ± 962.9508	86.5200 ± 5.4722

TABLE VI
EMPIRICAL EVALUATION WITH THE RASTRIGRIN FUNCTION

Pop	Dim	Iter	CPSO	UPSO	UMPSO
20	10	1000	29.0770 ± 9.4898	91.1710 ± 45.6996	0 ± 0
	20	1500	110.3294 ± 16.5217	277.2081 ± 52.6843	0 ± 0
	30	2000	194.2152 ± 39.7327	471.2959 ± 54.6243	0 ± 0
	50	5000	429.6415 ± 48.0173	922.4154 ± 39.3056	0 ± 0
	100	5000	1.5043e+003 ± 157.6142	2.0448e+003 ± 66.9770	0 ± 0
40	10	1000	22.4251 ± 7.9207	74.3478 ± 33.2963	0 ± 0
	20	1500	77.0191 ± 16.1267	248.2973 ± 45.8245	0 ± 0
	30	2000	199.2235 ± 54.3471	422.4963 ± 64.5098	0 ± 0
	50	5000	337.2394 ± 60.3831	848.9662 ± 51.5138	0 ± 0
	100	5000	1.3114e+003 ± 151.2171	1.9368e+003 ± 58.6183	0 ± 0

80	10	1000	17.8075 ± 5.0798	64.5741 ± 33.6009	0 ± 0
	20	1500	56.2418 ± 16.2717	217.5272 ± 53.2187	0 ± 0
	30	2000	150.9791 ± 39.4416	390.2007 ± 55.2521	0 ± 0
	50	5000	304.5458 ± 46.2564	752.6109 ± 53.9969	0 ± 0
	100	5000	1.1106e+003 ± 120.5360	1.7652e+003 ± 73.7778	0 ± 0

TABLE VII
EMPIRICAL EVALUATION WITH THE GRIEWANK FUNCTION

Pop	Dim	Iter	CPSO	UPSO	UMPSO
20	10	1000	0.6694 ± 0.3948	28.1349 ± 23.2931	0 ± 0
	20	1500	2.2554 ± 1.4409	354.8366 ± 120.7057	0 ± 0
	30	2000	4.2407 ± 1.1067	818.8691 ± 185.5850	0 ± 0
	50	5000	8.8352 ± 2.7666	1.8120e+003 ± 198.2503	0 ± 0
	100	5000	89.1876 ± 10.4594	4.3247e+003 ± 224.8858	0 ± 0
40	10	1000	0.1843 ± 0.1302	1.6749 ± 5.2513	0 ± 0
	20	1500	0.8211 ± 0.2331	124.0973 ± 60.2632	0 ± 0
	30	2000	1.5794± 0.3662	424.9801 ± 136.0264	0 ± 0
	50	5000	2.5239 ± 0.4401	1.3421e+003 ± 162.0179	0 ± 0
	100	5000	41.2122 ± 5.1474	3.6847e+003 ± 247.2692	0 ± 0
80	10	1000	0.0349 ± 0.0199	0.0211 ± 0.1046	0 ± 0
	20	1500	0.3607 ± 0.5838	23.3257 ± 16.9792	0 ± 0
	30	2000	0.5637 ± 0.1867	178.0840 ± 69.8172	0 ± 0
	50	5000	0.6774 ± 0.1082	800.5959 ± 163.1089	0 ± 0
	100	5000	14.4755 ± 2.9136	3.0104e+003 ± 272.1686	0 ± 0

The UMPSO algorithm is able to find the minimal solution for every variation of population sizes and dimension sizes. The UMPSO outperforms the CPSO for every population and dimension size variations. The UPSO only outperforms the CPSO with a large population size and small dimension size. Table VII above shows the results of each algorithm for the various population and dimension sizes.

VI. DISCUSSION

For the results of the UMPSO algorithm for the Rastrigrin and Griewank functions, a performance measure of 0 is obtained. This means that the performance measure is less than 1×10^{-323} since this is studied in the Matlab software.

It is important to note that the parameters (K, c_1, c_2, population, and positional update equation) for the CPSO, UPSO and UMPSO algorithms are static. For each benchmark function, population size, and dimension size, the UMPSO algorithm is able to outperform CPSO which is able to outperform UPSO (except for 5% of the time with the CPSO algorithm and 3% of the time with the UPSO algorithm in which the two algorithms performed better than the UMPSO). An advantage of UMPSO is that a small population size is adequate to obtain the same results with that of a larger population size.

VII. CONCLUSION

For a PSO to be a robust algorithm, it is important for it to perform well under many different problems. In this paper, the UMPSO algorithm is applied to four different benchmark functions and it obtains great results for each population and dimension size variations. Although many new PSO algorithms have come out with dynamic and/or adaptive parameters [8, 9, 10] to improve the performance of PSO, they will still need constraints for velocities and positions. These constraints will need to be custom fit for each optimization problem. The UMPSO algorithm, however, is custom fit for every application since there are no constraints. The robustness of this algorithm remains to be verified and refined. The application of the UMPSO algorithm may further be improved by enhancing the PSO parameters dynamically during search process.

REFERENCES

[1] R. C. Eberhart, J. Kennedy, "A New Optimizer Using Particles Swarm Theory", *Proc. Sixth International Symposium on Micro Machine and Human Science*, (Nagoya, Japan) IEEE Service Center, Piscataway, NJ, October 4 – 6, 1995, pp. 39 – 43.

[2] R. C. Eberhart, Y. Shi. "Comparison Between Genetic Algorithms and Particle Swarm Optimization", *Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447*, pp. 611 – 616.

[3] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization", *Proc. CEC 1999*, Washington, DC, 1999, pp. 1951 – 1957.

[4] J. Sun, W. Xu, B. Feng, "A Global Search Strategy of Quantum-Behaved Particle Swarm Optimization", *2004 IEEE Conference on Cybernetics and Intelligent Systems*, Vol. 1, December 1 – 3, 2004, pp. 111 – 116.

[5] R. C. Eberhart, Y. Shi, "Empirical Study of Particle Swarm Optimization", *Proc. of the 1999 Congress on Evolutionary Computation*, Vol. 3, July 6 – 9, 19 99, pp. 1945 – 1950.

[6] J. J. Liange, P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer", *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*, June 8 – 10, 2005, pp. 124 – 129.

[7] Y. Shi, R. C. Eberhart, "Fuzzy Adaptive Particle Swarm Optimization", *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, May 27 – 30, 2001, pp. 101 – 106.

[8] X. Xie, W. Zhang, Z. Yang, "Hybrid Particle Swarm Optimizer with Mass Extinction", *IEEE 2002 International Conference on*

Communications, Circuits and Systems, and West Sino Expositions, Vol. 2, June 29 – July 1, 2002, pp. 1170 – 1173.

[9] J. Sun, W. Xu, B. Feng, "Adaptive Parameter Control for Quantum-Behaved Particle Swarm Optimization on Individual Level", *2005 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, October 10 – 12, 2005, pp. 3049 – 3054.

[10] G. K. Venayagamoorthy, "Adaptive Critics for Dynamic Particle Swarm Optimization", *Proceedings of the 2004 IEEE International Symposium on Intelligent Control*, 2004, pp. 380 – 384.