IJCI. Vol. 2 – No. 1, January 2009

A Modified Algorithm for Particle Swarm Optimization with Constriction Coefficient

Mahmoud M. El-Sherbiny
Operations Research Dept, Institute of Statistical Studies and Research (ISSR),
Cairo University, Egypt.
Email: m_sherbiny@yahoo.com

Abstract *This paper introduces a modified algorithm to partical swarm based optimization that significantly reduces the number of iterations required to reach good solutions and discusses the results of experimentally comparing its performance with the performance of several variants of the standard particle swarm optimizer. The variants of the modified algorithm and the most common variants of the particle swarm optimizers are tested using a set of multimodal functions commonly used as benchmark optimization problems in evolutionary computation. Results indicate that the algorithm is highly competitive and can be considered as aviable alternative to solve the optimization problems.*

Keywords*: Particle swarm optimization; Convergence; Evolutionary computation; constriction coefficient*

1. Introduction

Particle Swarm Optimization (PSO) is a new evolutionary computation technique motivated from the simulation of social behavior and originally designed and developed by Eberhart and Kennedy [22,23,24]. The population in the PSO is called a swarm and each individual is called a particle [1]. It is inspired by the behavior of bird flocking and fish schooling. A large number of birds or fish flock synchronously, change direction suddenly, and scatter and regroup together. Each particle benefits from the experience of its own and that of the other members of the swarm during the search for food.

PSO algorithm has a number of desirable properties, including simplicity of implementation, scalability in dimension, and good empirical performance. So. It is an attractive choice for solving nonlinear programming problems. PSO algorithm had been applied to evolve weights and structure for artificial neural networks by Shi and Eberhart in 1998 [2], manufacture and milling [3], reactive power and voltage control by Abido, M.A. in 2002 [4,5], to estimate the voltage stability of the electric power distribution systems [6, 7] direct the orbits of discrete chaotic dynamical systems towards desired target region [8] and to solve the permutation flowshop sequencing problem [9]

PSO has been successfully used as an alternative to other evolutionary algorithms in the optimization of D-dimensional real functions. Particles move in a coordinated way through the D-dimensional search space towards the optimum of the function. Their movement is influenced not only by each particle own previous experience, but also by a social compulsion to move towards the best position found by its neighbours. To implement these behaviours, each particle is defined by its position and velocity in the search space. In each iteration, changes resulting from both influences in the particle's trajectory are made to its velocity. The particle's position is then updated accordingly to the calculated velocity. The PSO, its main variants and the structural model behind it are extensively discussed in [10]. Some work has been done that alters basic particle motion with some success, but the possibility for improvement in this area is still open [27].

This paper aims to introduce a modified algorithm for particle swarm optimization (MPSO) and discuss the results of experimentally comparing the performance of its versions with the standard particle swarm optimizer (SPSO)[21] and the combined PSO [25].

The rest of the paper is organized as follows: in section 2, the PSO method is described. In section 3, the modified algorithm and its versions are exposed. Test functions and test conditions are presented in sections 4 and 5. In section 6, optimization test experiments are illustrated. In section 7, the experimental results are reported, and are discussed in section 8. Finally, conclusion is reported in section 9.

2. Particle Swarm Optimization

In PSO, particles evaluate their positions relative to a goal (fitness) at every iteration, and the particles in a local neighborhood share memories of their "best" positions, then use those memories to adjust their own velocities and positions as shown in equations (1) and (2). The PSO formula defines each particle as a potential solution to a problem in the D-dimensional space, with the ith particle represented as Xi = (xi1, xi2, …, xiD). Each particle also remembers its best position, designated as X_{p_i}, and its velocity Vi = (vi1, vi2, …, viD) [11].

In each generation (iteration) t, the velocity of each particle is updated, being pulled in the direction of its own best position (X_{p_i}) and the best of all positions (Xg) reached by all particles until the preceding generation. After finding the two best values, the particle updates its velocity and position according to equations (1) and (2).

$$V_i(t) = aV_i(t-1) + b_1 r_1\left(X_{p_i} - X_i(t)\right) + b_2 r_2\left(X_g - X_i(t)\right) \quad (1)$$

$$X_i(t) = cX_i(t-1) + dV_i(t) \quad (2)$$

At iteration t, the velocity $V_i(t-1)$ is updated based on its current value affected by a momentum factor a and on a term which attracts the particle towards previously found best positions: its own previous best position (X_{p_i}) and globally best position in the whole swarm (Xg). The strength of attraction is given by the average of the own and the social attraction coefficients b1 and b2 . The particle position $X_i(t)$ is updated using its current value and the modified ly computed velocity $V_i(t)$, affected by coefficients c and d, respectively and they can be set to unity without loss of generality [21]. Randomness that useful for good state space exploration is introduced via the vectors of random numbers r1 and r2. They are usually selected as uniform random numbers in the range [0, 1].

The original PSO formula developed by Kennedy and Eberhart [1] were combined by Shi and Eberhart [2] with the introduction of an inertia parameter, ω , that was shown empirically to improve the overall performance of PSO. Clerc and Kennedy provided a theoretical analysis of particle tajectories to ensure convergence to a stable point [26],

$$\chi = \frac{b_1 r_1 X_p + b_2 r_2 X_g}{b_1 r_1 + b_2 r_2}$$

The main result of this work is the introduction of the constriction coefficient and different classes of constriction models. The objective of this theoretically derived constriction coefficient is to prevent the velocity to grow out of bounds, with the advantage that, theoretically, velocity clamping is no longer required. As a result of this study, the velocity equation (1) changes to equation 3.

$$V_i(t) = \chi\left(V_i(t-1) + b_1 r_1\left(X_{p_i} - X_i(t)\right) + b_2 r_2\left(X_g - X_i(t)\right)\right) \quad (3)$$

Several interesting variations of the PSO algorithm have recently been proposed by researchers in [12 - 17]. Many of these PSO improvements are essentially extrinsic to the particle dynamics at the heart of the PSO algorithm. One of these variations is the combined particle swarm optimization (CPSO) algorithm that presented by M. El_Sherbiny [25] and can be applied to augment of the modified algorithm presented in this paper. This paper proposes a modification to the dynamics of particles in PSO, presenting the constriction coefficient to the velocity update equation of the CPSO algorithm.

3. The Modified Algorithm

In the standard PSO algorithm, in each generation t, the velocity of each particle is updated, being pulled in the direction of its own previous best position (X_{p_i}) and the best of all positions (global position) (Xg) reached by all particles until the preceding generation. Whereas in the CPSO algorithm, in each generation t, the velocity $V_i(t-1)$ of particle i is updated based on its own best position (X_{p_i}) and the point (Xc) resulted from the combination of the

global best position (Xg) and the previous global best position (X2g) as illustrated in equation (3). Where R1 and R2 are defied as the combination weights.

$$(X_c) = R_1 (X_g) + R_2 (X_{2g}) \qquad (4)$$

In other words, after finding a new global best position for the (X_g) its old position will be assigned to (X_{2g}) and the particle updates its velocity according to equation (5) and updates its positions according to equation (2).

$$V_i(t) = aV_i(t-1) + b_1 r_1 \left(X_{p_i} - X_i(t)\right) + b_2 r_2 \left(\left(R_1 X_g + R_2 X_{2g}\right) - X_i(t)\right) \qquad (5)$$

In the MPSO algorithm, particle updates its velocity according to equation (6) instead of equation (5) where constartion coefficient χ is presented.

$$V_i(t) = \chi\left(V_i(t-1) + b_1 r_1 \left(X_{p_i} - X_i(t)\right) + b_2 r_2 \left(\left(R_1 X_g + R_2 X_{2g}\right) - X_i(t)\right)\right) \qquad (6)$$

In order to study the effects of constriction coefficiant χ and the parameters R1 and R2 in (6) on the performance of the MPSO algorithm, two variants were used in the experiments denoted as MPSO1, and MPSO2.

In the MPSO1 version, the particle updates its velocity according to equation 6 with equal random weight combination between the global best position (Xg) and the previous global best position (X2g). i.e. R1 = R2 =R.

In the MPSO2 version, the particle updates its velocity according to equation 6 with random weight combination between the global best position (Xg) and the previous global best position (X2g). i.e. R1 and R2 are two different random variables.

4. Test Functions

In order to know how competitive the MPSO algorithm is and the effects of the constriction coefficient χ and combination weights R1 and R2 , two versions of the MPSO algorithm (MPSO1 and MPSO2) will be compared against two counterparts' versions of the CPSO algorithm (CPSO1 and CPSO2) [25] and the SPSO algorithm [21]. Five benchmarking functions were selected to investigate the performance of the above-mentioned algorithms' versions. The considered test functions were used in [6,7, 21]. The functions, the number of dimensions (D), the admissible range of the variable (x), and the goal values are summarized in Table 1.

Table 1. Test functions [21]

Name	Formula	Dim D	Range [xmin, xmax]	Goal for F
Sphere	$F_0(\vec{x}) = \sum_{i=1}^{D} x_i^2$	30	[-100, 100]D	0.01
Rosenbrock	$F_1(\vec{x}) = \sum_{i=1}^{D-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	30	[-30, 30]D	100
Rastrigin	$F_2(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	30	[-5.12, 5.12]D	100
Griewank	$F_3(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600, 600]D	0.1
Schaffer's	$F_6(\vec{x}) = 0.5 - \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	2	[-100, 100]2	10-5

The optimal solution for all functions is equal to 0

5. Test Conditions

Two parameter sets (Eqs. (1, 2, 5 and 6) $\chi = a$ and $b = b1 = b2$ were selected to be used in the test based on the suggestions in other literature where these values have been found, empirically, to provide good performance [7, 9, 10] and used in testing the SPSO by I.C. Trelea [21].

Parameter set 1 ($a = 0.6$ and $b = 1.7$) was selected by the author [7] in the algorithm convergence domain after a large number of simulation experiments.

Parameter set 2 ($a = 0.729$ and $b = 1.494$) was recommended by Clerc [18] and also tested in [7] giving the best results published so far known to the author. All elements of c and d were set to 1 as used in [21].

A more detailed study of convergence characteristics for different values of these parameters exists in [19].

6. Optimization Test Experiments

In order to test the performance of the three versions of CPSO and SPSO algorithms two sets of experiments were used with the above mentioned test conditions and the two parameter sets.

In the first set of experiments, the maximum iteration number was fixed to 20000. Each optimization experiment was run 20 times with random initial values of x and v in the range [xmin, xmax] indicated in Table 1. Population sizes of N = 15, 30 and 60 particles were tested. The number of iterations required to reach the goal was recorded. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for each test function are calculated and presented in Tables 2-6.

In the second set of experiments, Each optimization experiment was run 20 times for 1000 iterations with population sizes of N = 30 particles. Averages of the best values in each iteration were calculated and plotted in figures 1- 5.

During the optimization process the particles were allowed to "fly" outside the region defined by [xmin, xmax] and also the velocity was not restricted.

7. The Experimental Results

This section compares the various algorithms to determine their relative rankings using both robustness and convergence speed as criteria. A "robust" algorithm is one that manages to reach the goal consistently (during all runs) in the performed experiments [20]. Tables 2–6 present the following information: Average number, median, minimum, maximum number of iterations required to reach a function value below the goal. Also, success rate of required iterations, and expected number of function evaluations. The "success rate" column lists the number of runs (out of 20) that managed to attain a function value below the goal in less than 2000 iterations, while the "Ex. # of Fn. Evaluation" column presents the expected number of function evaluations needed on average to reach the goal, calculated only for the succeeded runs using the following formula.

Ex. # of Fn. Evaluation = (Average number of iterations) × (number of particls in the swarm) / (success rate)

Table 2 shows that while SPSO algorithm failed to reached the goal during some runs for solving the Sphere function (F0) with parameter set1 and 15 particles, all the algorithms reached the goal during all the runs with both parameter sets.

Table 2. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F_0

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F_0	15	MPSO1	**56**	**94**	**56**	**96**	**47**	**68**	**63**	**107**	1	1	**844**	**1416**
		MPSO2	107	173	107	173	98	159	117	196	1	1	1605	2595
		CPSO1	125	168	126	173	69	102	161	216	1	1	1874	2516
		CPSO2	320	471	316	457	249	358	373	613	1	1	4804	7065
		SPSO	769	764	722	731	523	586	1377	1275	0.40	1	28838	11460
	30	MPSO1	**53**	**88**	**53**	**89**	**48**	**71**	**58**	**106**	1	1	**1575**	**2650**
		MPSO2	90	146	91	146	84	134	96	158	1	1	2709	4371
		CPSO1	131	180	127	179	103	137	161	221	1	1	3917	5396
		CPSO2	300	404	296	396	259	296	352	528	1	1	9006	12126
		SPSO	344	395	333	395	266	330	457	572	1	1	10320	11850
	60	MPSO1	**49**	**85**	**50**	**85**	**46**	**75**	**52**	**92**	1	1	**2961**	**5080**
		MPSO2	80	129	80	129	76	120	87	145	1	1	4814	7761
		CPSO1	118	157	120	159	90	123	132	185	1	1	7083	9423
		CPSO2	264	346	254	346	221	301	302	389	1	1	15816	20760
		SPSO	252	314	252	313	214	269	309	368	1	1	15120	18840

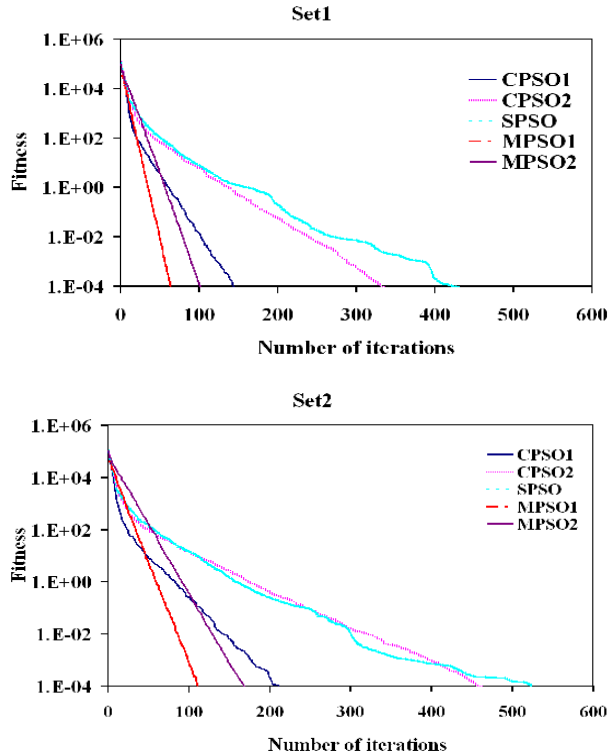

Fig. 1. Average best fitness curves for sphere function (*Fo*)

That means the number of particles affects the convergence of the SPSO algorithm for such problems type while such effect is not obviously clear with the other algorithms for such kind of problems.

Also, as illustrated in fig. 1, all the algorithms have reached a function value below the goal of the Sphere function with both parameter sets and they different are candidate to reach the optimal solution but in number of iterations. The two versions of MPSO algorithm are the candidates to reach the optimal solution in a few number of iteration than its counterpart's versions of the CPSO algorithm. That means the constriction coefficient affects the convergence speed of the versions of MPSO to be faster than its counterpart versions of CPSO algorithm.

Table 3. illustrates the same problem of SPSO algorithm with F1. It can't reach the goal of sphere functions in all runs with parameter set1.

Table 3. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F_1

Fun.	**# of Part. N**	**Algorithm**	**Number of algorithm iterations to achieve the goal**								**Success Rate**		**Ex. # of Fn. Evaluation**	
			Average		**Median**		**Minimum**		**Maximum**					
			Set 1	**Set 2**	**Set 1**	**Set 2**	**Set 1**	**Set 2**	**Set 1**	**Set 2**	**Set 1**	**Set 2**	**Set1**	**Set2**
F_1	15	MPSO1	**34**	**59**	**34**	**59**	**29**	**48**	**39**	**68**	1	1	**517**	**888**
		MPSO2	79	124	78	119	65	96	122	166	1	1	1179	1853
		CPSO1	88	112	89	105	62	63	119	160	1	1	1318	1673
		CPSO2	288	484	276	463	199	281	516	760	1	1	4324	7260
		SPSO	531	1430	523	729	413	452	695	9476	0.50	1	15930	21450
	30	MPSO1	**32**	**55**	**32**	**55**	**29**	**42**	**34**	**64**	1	1	**953**	**1645**
		MPSO2	65	109	64	108	57	89	77	134	1	1	1950	3278
		CPSO1	84	105	81	105	67	74	110	127	1	1	2520	3149
		CPSO2	257	392	258	395	204	253	336	771	1	1	7701	11771
		SPSO	614	900	383	408	239	298	3718	4642	1	1	18420	27000
	60	MPSO1	**30**	**53**	**30**	**52**	**26**	**46**	**33**	**62**	1	1	**1789**	**3154**
		MPSO2	57	97	54	96	47	81	73	126	1	1	3423	5820
		CPSO1	77	102	76	101	64	80	89	124	1	1	4623	6138
		CPSO2	227	309	218	298	174	234	468	430	1	1	13590	18561
		SPSO	337	611	284	311	189	219	916	4450	1	1	20220	36660

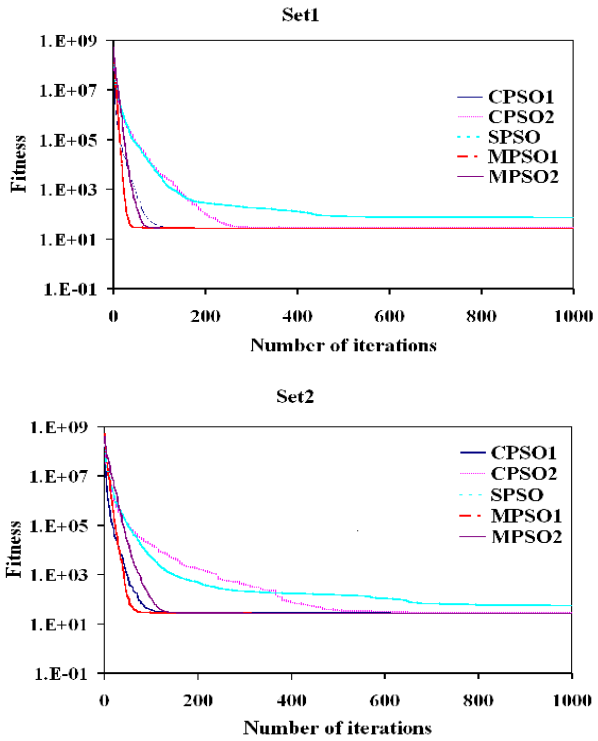

Fig. 2. Average best fitness curves for Rosenbrock function (F_1)

While SPSO with parameter set1 and 15 has some difficulties in reaching the goal of Rosenbrock function (F1), none of the other algorithms had such difficulties with both parameter sets for solving the same function. Also, the expected number of function evaluation needed for the two versions of MPSO algorithm is less than expected number of function evaluation needed for the other algorithms. That means the constriction coefficient accelerates the speed of MPSO algorithm to be faster than the other algorithms tested in this paper in searching for the solution of such kind of problems (see Table 3).

Also, fig. 2 illustrates that the MPSO1 , MPSO2, CPSO1 and CPSO2 reached values below the function goal in different number of iterations and below that value reached by SPSO algorithm. That means solution quality of all algorithms except the SPSO is the same for such kind of problems. But, the versions of MPSO algorithm are reached values below the goal faster than its counterparts of CPSO algorithm. That means, the constriction coefficient accelerates the speed and maintains the solution quality of MPSO the same time.

Table 4 shows that the MPSO1, MPSO2 , and CPSO1 algorithms perform admirably on the Rastrigin function (F2), but the CPSO2 and SPSO algorithms are less robust on the same function for such type of problems where they had some difficulties in reaching the goal in some runs.

Table 4 and fig. 3 illustrate that the MPSO1 and CPSO1 algorithms are doing very well on this problem, delivering the best overall performance for the Rastrigin function (F2), where they reached the goal in approximately 35 and 57 iterations respectively and they are candidates to reach the optimal solution in approximately less than 200 and 400 iterations respectively with both parameter sets. Although the good performance of CPSO1 algorithm the MPSO1 is a superior of it. This superiority because of the effect of constriction coefficient on the performance of MPSO1 algorithm.

Concerning the effects of the parameter sets on the algorithms performance on Rastrigin function (F2), there is no significant difference between the algorithms performance with both the parameter sets except the performance of MPSO2 algorithm with parameter set1 is much better than its performance with parameter set2 as shown in fig. 3.

Griewank's function (F3) proves to be hard to solve for all the algorithms. All the algorithms consistently reached the goal in all runs except SPSO and CPSO2 algorithms had some difficulties in reaching the goal for

Griewank's function with both parameter sets, as can be seen in Table 5. Also, we can realize that the performance of the two versions of MPSO algorithm is best than its counterparts of CPSO algorithm as seen in Table 5.

Table 4. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F_2

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set 1	Set 2	Set 1	Set 2	Set 1	Set 2	Set 1	Set2	Set1	Set 2	Set1	Set2
F_2	15	MPSO1	**23**	**38**	**22**	**37**	**16**	**25**	**30**	**55**	1	1	**344**	**576**
		MPSO2	157	177	122	166	66	52	438	351	1	1	2361	2658
		CPSO1	57	81	51	71	23	47	139	147	1	1	857	1217
		CPSO2	264	548	220	517	157	145	698	1357	0.90	1	4402	8214
		SPSO	172	299	147	292	102	123	208	299	0.35	0.8	7371	5606
	30	MPSO1	**22**	**37**	**21**	**37**	**17**	**24**	**30**	**51**	1	1	**667**	**1111**
		MPSO2	102	166	95	149	47	95	174	316	1	1	3072	4988
		CPSO1	48	68	44	63	32	37	83	145	1	1	1451	2054
		CPSO2	281	443	233	348	121	159	857	955	1	1	8862	13283
		SPSO	140	182	128	174	104	123	208	299	0.90	0.95	4667	5747
	60	MPSO1	**21**	**36**	**21**	**35**	**16**	**24**	**30**	**52**	1	1	**1271**	**2180**
		MPSO2	120	127	88	119	35	75	627	223	1	1	7203	7595
		CPSO1	60	69	55	62	39	34	132	127	1	1	3603	4152
		CPSO2	265	517	209	414	144	215	633	1526	1	1	15906	31026
		SPSO	122	166	116	164	84	119	168	214	0.95	1	7705	9960

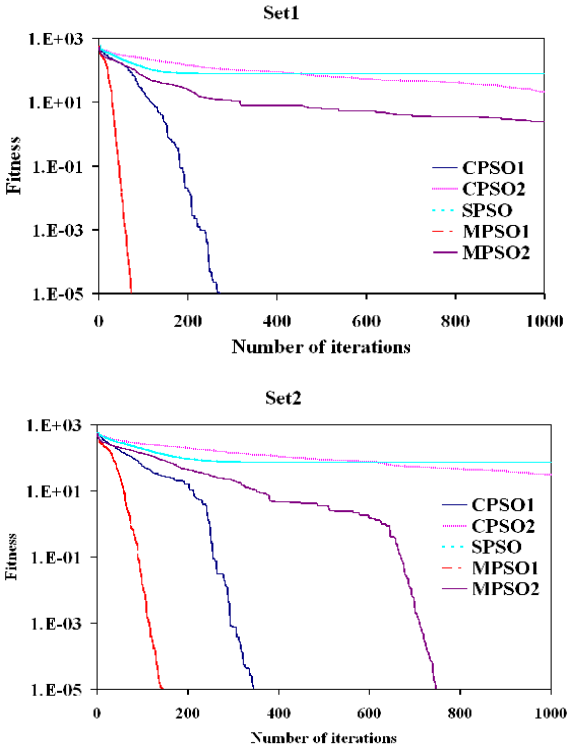

Fig. 3. Average best fitness curves for Rastrigin function F_2

Fig. 4 illustrates that both versions of the MPSO and CPSO1 algorithms with both parameter sets are candidates to reach the optimal solution while SPSO and CPSO2 did not reach the goal during some runs. MPSO1 with parameter set2 is candidate to reach the optimal solution while it is not with parameter set1. (see Fig 4.). Although the good performance of both versions MPSO in solving Griewank function its counterparts of CPSO algorithm are not. That appears the effect of the constriction coefficient on accelerating the speed of MPSO algorithm to be faster than the other algorithms tested in this paper in searching for the solution of such kind of problems.

Fig. 4 illustrates that the MPSO1 and CPSO1 algorithms with both parameter sets are candidates to reach the optimal solution while SPSO and CPSO did not reach the goal during some runs. CPSO1 algorithm with parameter set2 is candidate to reach the optimal while it is not with parameter set1. (see fig. 4).

Table 5. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F_3

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F_3	15	MPSO1	**53**	**89**	**51**	**86**	**47**	**61**	**71**	**120**	1	1	**790**	**1332**
		MPSO2	129	204	120	192	84	142	208	411	1	1	1930	3059
		CPSO1	132	215	131	205	84	89	191	470	1	1	1982	3218
		CPSO2	364	581	314	522	254	329	731	1403	1	1	5465	8714
		SPSO	689	755	580	608	443	470	1589	1755	0.35	0.6	29529	18875
	30	MPSO1	**47**	**83**	**47**	**85**	**42**	**63**	**54**	**108**	1	1	**1417**	**2479**

		MPSO2	117	165	114	144	77	129	176	317	1	1	3496	4936
		CPSO1	131	168	130	165	79	57	229	293	1	1	3929	5046
		CPSO2	342	440	293	431	227	330	720	667	1	0.95	10257	13910
		SPSO	313	365	304	361	257	319	401	455	0.90	0.90	10433	12167
	60	MPSO1	**45**	**76**	**45**	**76**	**40**	**65**	**50**	**100**	1	1	**2700**	**4554**
		MPSO2	90	140	77	118	67	110	174	221	1	1	5380	8392
		CPSO1	113	152	109	148	72	115	199	198	1	1	6753	9117
		CPSO2	325	421	268	370	188	301	853	718	1	1	19494	25245
		SPSO	226	287	224	280	202	266	250	238	0.95	1	14274	17220

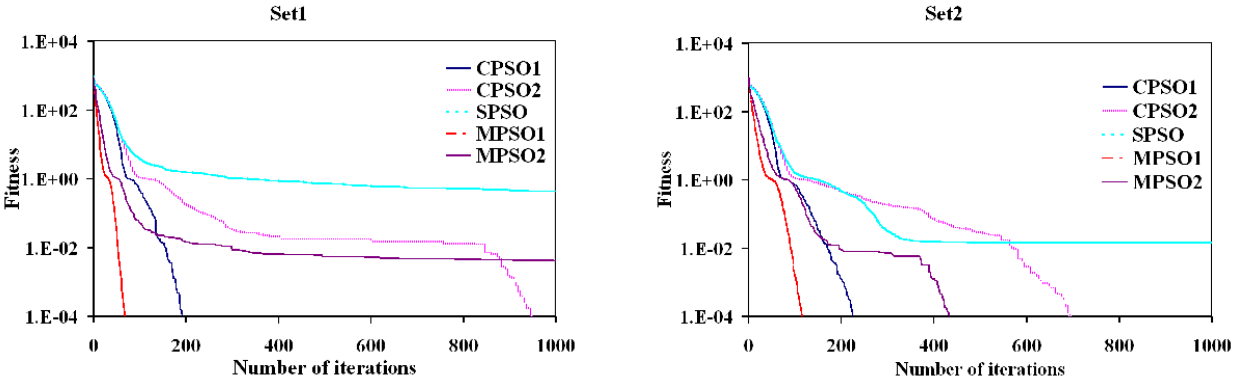

Fig. 4. Average best fitness curves for Griewank function F_3

Concerning the Schaffer's function (F6): Table 6 illustrates that while all versions of the MPOS and CPSO algorithms reached the goal in all runs with both parameter sets, SPSO algorithm had some difficulties in reaching the goal.

Fig. 5 illustrates that: using the average of 30 runs for 1000 iterations of each algorithm, all the algorithms reached a solution value (10-6) below the goal in different number of iterations and be candidate to reach the goal accept the MPSO2 and SPSO algorithms. MPSO2 with parameter set1 stacked at value 10-3 greater than the goal while SPSO algorithm with both parameter sets stacked at value 10-3 greater than the goal. MPSO1 reached a value below the goal in 170 iterations on average while CPSO1 and CPSO2 needed more than 200 iterations.

Table 6. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F6

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set 1	Set2	Set 1	Set 2	Set 1	Set 2	Set1	Set2	Set1	Set2	Set1	Set2
F6	15	MPSO1	**148**	**142**	**129**	**110**	**40**	**49**	**344**	**669**	1	1	**2216**	**2132**
		MPSO2	242	254	120	197	63	54	961	773	1	1	3635	3805
		CPSO1	198	232	183	201	57	63	473	498	1	1	2971	3476
		CPSO2	231	254	207	225	72	99	522	631	1	1	3461	3813
		SPSO	583	1203	138	126	63	91	3706	5853	0.45	0.4	19433	45113
	30	MPSO1	**93**	**128**	**72**	**115**	**45**	**60**	**210**	**250**	1	1	**2791**	**3826**
		MPSO2	159	154	78	118	34	53	639	383	1	1	4784	4617
		CPSO1	122	148	98	134	55	57	307	397	1	1	3668	4434
		CPSO2	144	163	126	163	57	87	439	298	1	1	4322	4893
		SPSO	161	350	120	157	74	102	595	1264	0.75	0.60	6440	17500
	60	MPSO1	**60**	**83**	**57**	**77**	**35**	**54**	**108**	**181**	1	1	**3616**	**4954**
		MPSO2	89	104	76	94	35	65	195	190	1	1	5345	6227
		CPSO1	93	112	76	110	42	73	206	193	1	1	5559	6708
		CPSO2	112	147	94	128	52	53	232	422	1	1	6726	8841
		SPSO	169	319	91	119	40	83	854	2361	0.90	0.95	11267	20147

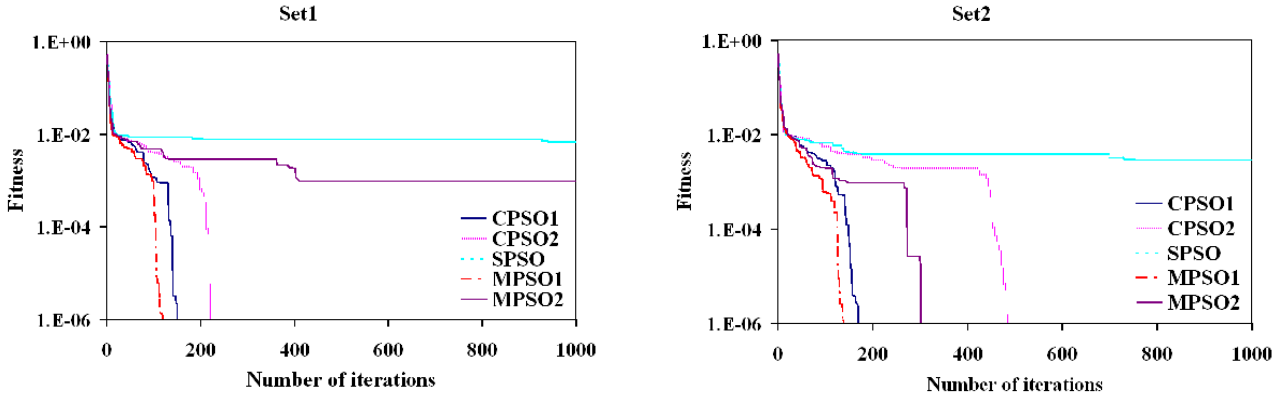

Fig. 5. Average best fitness curves for Schaffer's function F_6

8. Discussion

Overall, as far as robustness is concerned, the MPSO algorithm appears to be the winner, since its two versions (MPSO1 and MPSO2) achieved perfect scores in all the test cases as represented in boldface (see Tables 2-6). So, we can conclued that the constriction coefficient accelerates the convergence and the solution quality of MPSO algorithm. The CPSO1, algorithm is less robust, followed closely by the CPSO2 algorithm. The standard SPSO algorithm was fairly unreliable on this set of problems.

As a result, in many cases the SPSO must be executed several times to ensure good results, whereas one run of MPSO1, MPSO2, CPSO1 and the CPSO2 usually is sufficient.

Regarding convergence speed, MPSO1 is always the fastest followed by MPSO2, and CPSO1 whereas the CPSO2 or SPSO are always the slowest. MPSO1 has very fast convergence (4-6 times faster than SPSO). This may be of practical relevance for some real-world problems where the evaluation is computationally expensive and the search space is relatively simple and of low dimensionality. Overall, MPSO1 and MPSO2 are clearly the best performing algorithms in this study. It finds the lowest fitness value for all the tested problems, see fig. 1-5.

Regarding the parameter sets: SPSO is more sensitive to parameter changes than the other algorithms. When changing the problem, one probably needs to change parameters as well to sustain optimal performance. In general, the performance of all algorithms are best with parameter set1 than the performance with parameter set2, while all of them need less number of iterations to reach the specified goal with parameter set1. That means parameter set2 slows the algorithms and don't make a bridging phenomena while parameter set1 accelerates the algorithms but some while makes a bridging phenomena.

Looking at the number of function evaluations, the MPSO1 was in the lead, followed by the MPSO2 algorithm, as shown in boldface (see Tables 2-6). That means the constriction coefficient speed the algorithm convergence.

Considering, the above-mentioned points the two versions of MPSO algorithm had no difficulty in reaching the functions' goals and all its solutions are below their corresponding goals more than the other algorithms (see fig. 1-5). So, we can conclude that Modified algorithm is more superior to the other algorithms and can be considered as a viable alternative algorithm for solving optimization problems.

9. Conclusion

This paper has proposed a new variation of the particle swarm optimization algorithm called a modified PSO, introducing a constriction coefficient into the velocity and update equation of the combined particle swarm algorithm where a modified term has introduced into the velocity component and update equation. The implementation of this idea is simple, based on storing the previous positions. The MPSO algorithm outperforms the CPSO algorithm [25] and SPSO algorithm [21] on many benchmark functions, being less susceptible to premature convergence, and less likely to be stuck in local optima.

In this study, the MPSO1 and MPSO2 have shown its worth on real-world problems, and it outperformed CPSO and SPSO on all the numerical benchmark problems as well. Among the tested algorithms, the MPSO1 can rightfully be regarded as an excellent first choice, when faced with an optimization problem to solve.

To conclude, the constriction coefficient in MPSO algorithm accelerates the convergence to a stable point in comparison to the other algorithms tested in this paper. The algorithm is simple, robust, converges fast, and finds the optimum in every run. In addition, it has few parameters to set, and the same settings can be used for many different problems.

Future work includes further experimentation with parameters of MPSO, testing the algorithm on other benchmark problems, and evaluating its performance relative to evolutionary algorithms. Investigate the affect of different values of the constriction coefficient on the MPSO algorithm.

10. References

[1] R. Eberhart and J. Kennedy, A modified optimizer using particle swarm theory, in Proc. 6th Int. Symp. Micro Machine and Human Science, Nagoya, Japan, (1995) 39–43.

[2] Y. Shi and R. Eberhart, A Combined Particle Swarm Optimizer. In: Proceedings of IEEE World Congress on Computational Intelligence, (1998) 69–73.

[3] V. Tandon, Closing The Gap Between CAD/CAM and Optimized CNC and Milling, Master thesis, Purdue School of Engineering and Technology, Indiana University Purdue University Indianapolis, 2000.

[4] Abido, M.A. Optimal Power Flow Using Particle Swarm Optimization. Electric Power and Energy Sys. 2002;24:563–71.

[5] Jiang Chuanwen, Etorre Bompard, A Hybrid Method Of Chaotic Particle Swarm Optimization And Linear Interior For Reactive Power Optimization, Mathematics and Computers in Simulation 68 (2005) 57–65.

[6] N. Shigenori, G. Takamu, Y. Toshiku, F. Yoshikazu, A Hybrid Particle Swarm Optimization For Distribution State Estimation, IEEE Transactions on Power Systems 18 (2003) 60– 68.

[7] Amgad A. EL-Dib, Hosam M. Youssef, M.M. EL-Metwally, Z. Osman, Maximum loadability of power systems using hybrid particle swarm optimization, Electric Power Systems Research 76 (2006) 485–492.

[8] Bo Liu, Ling Wang, Yi-Hui Jin, Fang Tang, De-Xian Huang, Directing orbits of chaotic systems by particle swarm optimization, Chaos, Solitons and Fractals 29 (2006) 454–461.

[9] M. Tasgetiren, Yun-Chia Liang, Mehmet Sevkli, Gunes Gencyilmaz , A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European Journal of Operational Research, (2006) in process.

[10] M. Clerc and J. Kennedy, The Particle Swarm: Explosion, Stability, And Convergence In A Multi-Dimensional

Complex Space, IEEE Trans. Evol. Comput. 6,(2002) 58–73.
[11] A. Carlisle and G. Dozier, Adapting Particle Swarm Optimization to Dynamic Environments, Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, (2000) 429-434.
[12] S. Tsumoto et al. (Eds.), A Guaranteed Global Convergence Particle Swarm Optimizer, RSCTC, (2004) 762–767. Springer-Verlag Berlin 2004.
[13] Bo Liu, Ling Wang, Yi-Hui Jin, Fang Tang, De-Xian Huang, Improved particle swarm optimization combined with chaos, Chaos, Solitons and Fractals 25 (2005) 1261–1271
[14] M. Lovbjerg and T. Krink, Extending Particle Swarm Optimizers with Self-Organized Criticality, Proceedings of Fourth Congress on Evolutionary Computation, (2002) 1588-1593.
[15] Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang, Hybrid Particle Swarm Optimizer with Mass Extinction, International Conf. on Communication, Circuits and Systems (ICCCAS), Chengdu, China, 2002.
[16] Xiao-Feng Xie, Wen-Jun Zhang and Zhi-Lian Yang, A Dissipative Particle Swarm Optimization, IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 2002.
[17] F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (2006) 937–971.
[18] M. Clerc, The Swarm And The Queen: Towards A Deterministic And Adaptive Particle Swarm Optimization, in: Proc. ICEC, Washington, DC, (1999) 1951–1957.
[19] F. van den Bergh, "An Analysis Of Particle Swarm Optimizers," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, 2002.
[20] Frans van den Bergh and Andries P. Engelbrecht, A Cooperative Algorithm To Particle Swarm Optimization, IEEE Transactions on Evolutionary Computation, 8(3), (2004) 225-239.
[21] Ioan Cristian Trelea, The Particle Swarm Optimization Algorithm: Convergence Analysis And Parameter Selection, Information Processing Letters 85 (2003) 317–325.
[22] Eberhart, R. C., and Kennedy, J. (1995). A New Optimizer Using Particles Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.
[23] Kennedy, J., and Eberhart, R. C. (1995). Particle Swarm Optimization, Proc. IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1942-1948.
[24] Kennedy, J. (1997), The Particle Swarm: Social Adaptation of Knowledge, Proc. IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ, 303-308.
[25] M. M. El_Sherbiny, A Combined Particle Swarm Optimization Algorithm Based on the Previous Global Best and the Global Best Positions, International Journal of Computers and Information, 1,(2007) 13-26.
[26] M. Clerc, J. Kennedy, The Particle Swarm-Explosion, Stability, and Convergence In A Multidimensional Complex Space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[27] Kennedy, J.: Bare Bones Particle Swarms. In: Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana (2003) 80–87.