A Combined Particle Swarm Optimization Algorithm Based on the Previous Global Best and the Global Best Positions

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 combined algorithm to particle swarm based optimization and discusses the results of experimentally comparing the performances of its three versions with the performance of the particle swarm optimizer. In the combined algorithm, each particle flies and is attracted toward a new position according to its previous best position and the point resulted from the combination of the previous global best position and the global best position. The variants of the combined algorithm and the particle swarm optimizer 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 a viable alternative to solve the optimization problems.

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

1. Introduction

The particle swarm algorithm, which is frequently called particle swarm optimizer, is a new evolutionary algorithm, where the population is now 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.

Particle Swarm Optimization (PSO) algorithm was proposed by Eberhart and Kennedy in 1995 [1], and had been applied to evolve weights and structure for artificial neural networks by Shi and Eberhart in 1998 [2], manufacture and milling by Tandon in 2000 [3], reactive power and voltage control by Abido M.A. in 2002 [4] and Jiang Chuanwenet in 2005 [5], and state estimation for electric power distribution systems by Shigenori et al. in 2003 [6]. The convergence and parameterization aspects of the PSO have also been discussed in [7, 8, 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].

This paper aims to introduce a combined algorithm of PSO and discuss the results of experimentally comparing the performance of its versions with the particle swarm optimizer (PSO)[8]. In the combined algorithm, each particle flies and is attracted toward a new position according its own best position and the point resulted from the combination of the the point resulted from the combination of the previous global best position and the global best position.

The rest of the paper is organized as follows: in section 2, the PSO method is described. In section 3, the combined 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

The particles evaluate their positions relative to a goal (fitness) at every iteration, and 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 define each particle as a potential solution to a problem in the *D*-dimensional space, with the i^{th} particle represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle also remembers its best position, designated as X_{p_i}, and its velocity $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ [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 (X_g) reached by all particles until the preceding generation. After finding the two best values, the particle updates its velocity and positions according to equations (1) and (2).

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

$$X_i(t) = cX_i(t-1) + dV_i(t) \qquad \textbf{(2)}$$

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 (X_g). The strength of attraction is given by the average of the own and the social attraction coefficients b_1 and b_2 . The particle position $X_i(t)$ is updated using its current value and the newly computed velocity $V_i(t)$, affected by coefficients *c* and *d*, respectively and they can be set to unity without loss of generality [8]. Randomness useful for good state space exploration is introduced via the vectors of random numbers r_1 and r_2. 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.

Several interesting variations of the PSO algorithm have recently been proposed by researchers in [12], [13], [14], [15], [16], [17]. Many of these PSO improvements are essentially extrinsic to the particle dynamics at the heart of the PSO algorithm and can be applied to augment the new algorithm presented in this paper. By contrast to most other PSO variations, this paper proposes a significant modification to the dynamics of particles in PSO, moving each particle towards a new position according its own best position and the point resulted from the combination between the previous global best position and the global best position instead of the global best position that used in the standard particle swarm. This is in addition to the terms in the original PSO update equations.

3. The Combined 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) (X_g) reached by all particles until the preceding generation. Whereas in the combined PSO 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 (X_c) resulted from the combination of the global best position (X_g) and the previous global best position (X_{2g}) as illustrated in equation (3). Where R_1 and $R_2 \in$ [0,1] are uniform random variables defined as the combination weights.

$$(X_c) = R_1\,(X_g) + R_2\,(X_{2g}) \qquad \textbf{(3)}$$

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 (4) and updates its positions according to equation (2).

$$V_i(t) = aV_i(t-1) + b_1 r_1 \left(X_{p_i} - X_i(t)\right) + b_1 r_1 \left(\left(R_1 X_g + R_2 X_{2g}\right) - X_i(t)\right) \qquad \textbf{(4)}$$

In order to study the effects of the parameters R_1 *and* R_2 in (4) on the performance of the Combined Particle Swarm Optimization algorithm (CPSO), three variants were used in the experiments denoted as CPSO1, CPSO2, and CPSO3.

In the CPSO1 version, the particle updates its velocity according to equation 3 with equal random weight combination between the global best position (X_g) and the previous global best position (X_{2g}). i.e. $R_1 = R_2 = R$.

In the CPSO2 version, the particle updates its velocity according to equation 3 with random weight combination between the global best position (X_g) and the previous global best position (X_{2g}). i.e. R_1 and R_2 are two different random variables.

In the CPSO3 version, the particle updates its velocity according to equation 3 with random linear combination between the global best position (X_g) and the previous global best position (X_{2g}). i.e. $R_2 = (1- R_1)$.

4. Test Functions and Conditions

In order to know how competitive the combined algorithm is and the effects of the combination weights R_1 and R_2 , we decided to compare its three versions against the PSO algorithm that is represented in [8]. Five benchmarking functions were selected to investigate the performance of the three versions CPSO algorithm and PSO. The considered test functions were used in [6], [7] and [8]. The functions, the number of dimensions (D), the admissible range of the variable (x), and the goal values are summarized in Table 1.

Two parameter sets (Eqs. (1), (2) and (4)) a and $b = b_1 = b_2$ were selected to be used in the test based on the suggestions in other literature where these values have been found, empiricaly, to provide good performance [10, 7, 9]. and used in testing the PSO by I.C. Trelea [8].

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

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 [8].

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

5. Optimization Test Experiments

In order to test the performance of the three versions of CPSO and PSO 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 2000. Each optimization experiment was run 20 times with random initial values of x and v in the range $[x_{min}, x_{max}]$ 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. The 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 $[x_{min}, x_{max}]$ and also the velocity was not restricted.

6. 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) x (number of particlsin the swarm)/ (success rate)

Table 2 shows that the CPSO1 and CPSO2 algorithms reached the goal during all the runs for solving the Sphere function (F_0) with both parameter sets. While CPSO3 and PSO algorithms failed to reached the goal during some runs with parameter set 1.

Table 1. Test functions [8]

Name	Formula	Dim. D	Rang [xmin, xmax]	Goal for F
Sphere	$F_0(\vec{x}) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]^D$	0.01
Rosenbrock	$F_1(\vec{x}) = \sum_{i=1}^{D-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right)$	30	$[-30, 30]^D$	100
Rastrigin	$F_2(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos\left(2px_i\right) + 10\right)$	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

Also, as illustrated in figure 1, all the algorithms, except CPSO3 and PSO with the parameter set1 and 15 particles, have reached a function value below the goal of the sphere function with both parameter sets. That means the number of particles affects the convergancy of the CPSO3 and PSO algorithms for such problems type.

None of the algorithms, with the exception of the CPSO3 with both parameter sets and 15 particles and PSO with parameter set2 and 15 particles, had any difficulty reaching the goal of the Rosenbrock function (F_1) during any of the runs. Table 3 shows the expected number of function evaluations for all the algorithms required for solving the Rosenbrock function (F_1) with the CPSO1 algorithm requiring the fewest function evaluations overall. Also, fig. 2 illustrates that the CPSO1 and CPSO2 reached a value below the function goal while CPSO3 and PSO stacked near the function goal.

Table 4 shows that the CPSO1 and CPSO2 algorithms perform admirably on the Rastrigin function (F_2), but the CPSO3 and PSO algorithms are less robust on the same function.

Note that the CPSO1 algorithm is doing very well on this problem, delivering the best overall performance for the Rastrigin function where it reached the goal on approximately 60 iterations and reached the optimal solution in approximately less than 400 iterations as illustrated in fig. 3.

Concerning the effects of the parameter sets on the algorithms performance on Rastrigin function (F_2), there is no significant difference between the algorithms performance with both the parameter sets except the performance of CPSO2 algorithm with parameter set1 is a little mach better than its performance with parameter set2 as shown in figure 3.

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	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
		CPSO3	532	720	535	699	461	500	903	1313	0.45	1	21070	10796
		PSO	769	764	722	731	523	586	1377	1275	0.40	1	28838	11460
	30	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
		CPSO3	311	358	306	360	246	315	499	434	1	1	9323	10739
		PSO	344	395	333	395	266	330	457	572	1	1	10320	11850
	60	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
		CPSO3	224	281	222	285	197	245	254	310	1	1	13431	16854
		PSO	252	314	252	313	214	269	309	368	1	1	15120	18840

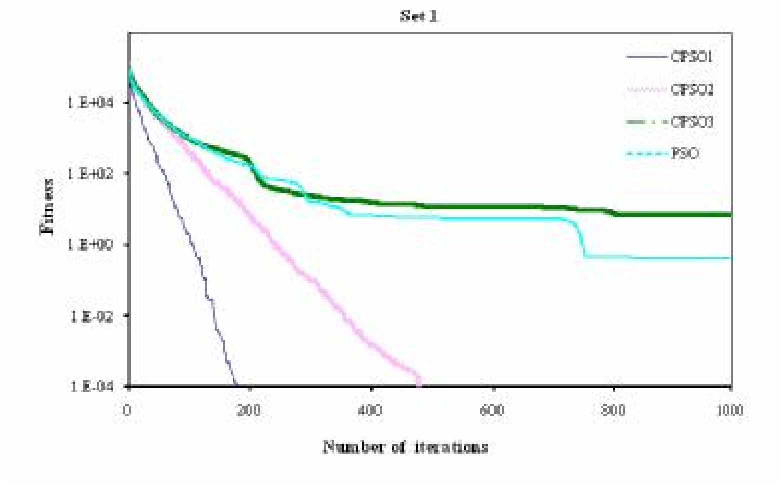


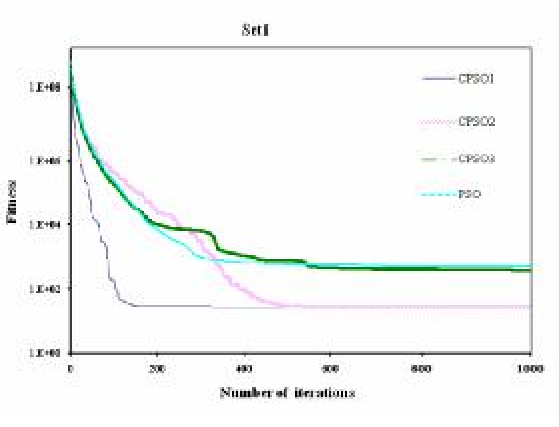
Figure 1. Average best fitness curves for sphere function (Fo)

Griewank’s function (F_3) proves to be hard to solve for all the algorithms except CPSO1 CPSO2 algorithms, as can be seen in Table 5. Only the CPSO1 and CPSO2 algorithms consistently reached the goal during all runs with both parameter sets and they are candidate to reach the optimal solution (see fig. 4) while PSO and CPSO3 did not reach the goal during some runs (see Table 5).

The CPSO3 failed almost completely to reach the goal for solving Griewank function (F_3) with parameter set1 and 15 particles, as can be seen in Table 5 while it reached the goal in almost 11 out of 20 runs. Fig. 4. illustrates the effects of the parameter sets on the performance of CPSO3 and PSO algorithms. Note that both the CPSO3 and PSO algorithms failed almost completely to reached the goal with parameter set1 in 1000 runs but they did with parameter set2 in less than 350 runs, while the CPSO1 and CPSO2 algorithms managed to solve the same function consistently with both the parameter sets. That means CPSO3 and PSO algorithms are very fast in algorithming the solution but making a bridging – zigzagging- near the optimal solution.

Table 3. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations, for the test function F1

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_1	15	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
		CPSO3	587	753	490	666	325	416	1147	1403	0.50	0.80	17601	14118
		PSO	531	1430	523	729	413	452	695	9476	0.50	1	15930	21450
	30	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
		CPSO3	450	477	401	417	212	249	961	1051	1	1	13500	14315
		PSO	614	900	383	408	239	298	3718	4642	1	1	18420	27000
	60	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
		CPSO3	263	356	230	280	170	215	531	775	1	0.85	15768	25138
		PSO	337	611	284	311	189	219	916	4450	1	1	20220	36660



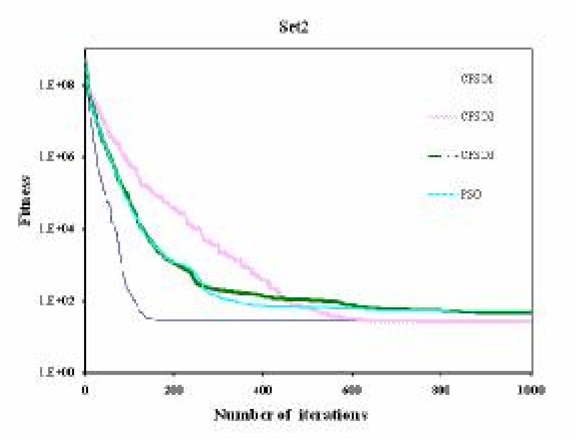


Figure 2. Average best fitness curves for Rosenbrock function (F1)

Table 4. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F2

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_2	15	CPSO1	57	81	51	71	23	47	139	147	1	1	357	1217
		CPSO2	264	548	220	517	157	145	698	1357	0.90	1	4402	8214
		CPSO3	122	169	120	156	80	105	194	309	0.85	1	2145	2671
		PSO	172	299	147	292	102	123	208	299	0.35	0.8	7371	5606
	30	CPSO1	48	68	44	63	32	37	83	145	1	1	1451	2054
		CPSO2	281	443	233	348	121	159	357	955	0.95	1	3862	13283
		CPSO3	106	155	100	148	66	104	192	212	0.90	1	3522	4641
		PSO	140	182	128	174	104	123	208	299	0.90	0.95	4667	5747
	60	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
		CPSO3	91	127	87	120	62	80	136	193	1	1	5457	7614
		PSO	122	166	116	164	84	119	168	214	0.95	1	7705	9960

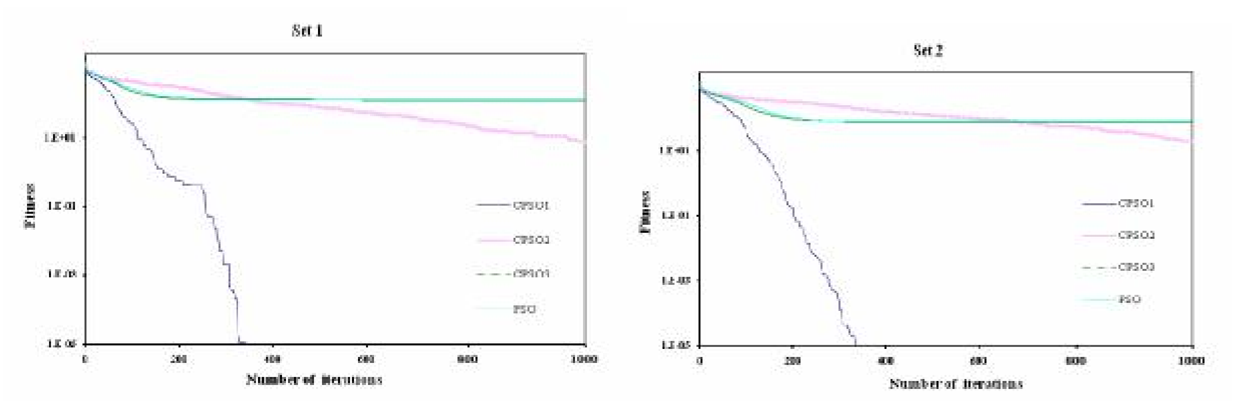


Figure 3. Average best fitness curves for Rastrigin function (F2)

Concerning the Schaffer's function (F_6): Table 6 illustrates that only the CPOS1 and CPSO2 algorithms reached the goal in all runs with both parameter sets, while CPSO3 and PSO algorithms had some difficulties in reaching the goal.

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	CPSO1	132	215	131	205	34	89	191	470	1	1	1982	3218
		CPSO2	364	581	314	522	254	329	731	1403	1	1	5465	8714
		CPSO3	-	508	-	487	0	438	0	702	-	0.55	-	13845
		PSO	689	755	680	608	443	470	1589	1755	0.35	0.6	29529	18875
	30	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
		CPSO3	266	312	265	301	235	270	327	505	0.85	0.90	9386	10387
		PSO	313	365	304	361	257	319	401	455	0.90	0.90	10433	12167
	60	CPSO1	113	152	109	148	72	115	199	198	1	1	6753	9117
		CPSO2	325	421	268	370	188	301	353	718	1	1	19494	25245
		CPSO3	211	251	208	248	178	216	240	286	1	1	12681	15084
		PSO	226	287	224	280	202	266	250	238	0.95	1	14274	17220

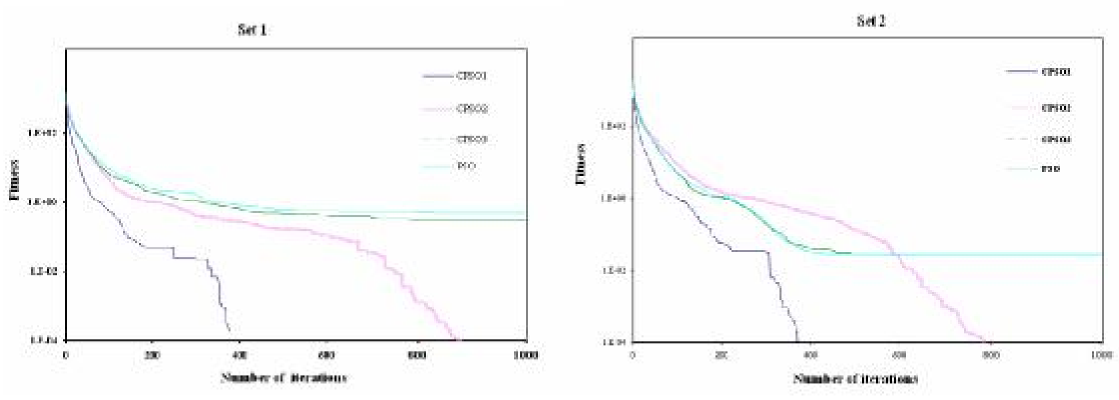


Figure 5. Average best fitness curves for Schaffer function F6

Note that, only the MPOS1 and CPSO2 algorithms reached below the goal and the optimal values in less than 350 iterations on average with both parameter sets while CPSO3 and PSO algorithms had stacked before the goal with both parameter sets in 1000 iterations (see fig. 5).

7. Discussion

Overall, as far as robustness is concerned, the CPSO1 algorithm appears to be the winner, since it achieved a perfect score in all the test cases as represented in boldface (see Tables 2-6).

The CPSO2, algorithm is less robust, followed closely by the CPSO3 and PSO algorithms. The standard PSO algorithm was fairly unreliable on this set of problems.

As a result, the PSO must be executed several times to ensure good results, whereas one run of CPSO1 and the CPSO2 usually sufficient.

Note that in the set1 case, there is a little difference between the performance of the algorithms with parameter set1 and with parameter set2 of where the algorithms' conversances are faster with parameter set1 than with parameter set2.

CPSO3 and PSO are 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.

Regarding convergence speed, CPSO1 is always the fastest followed by CPSO2, whereas the CPSO3 or PSO are always the slowest. Especially on the all functions, CPSO1 has a very fast convergence (2-5 times faster than PSO). 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, CPSO1 is clearly the best performing algorithm in this study. It finds the lowest fitness value for most of the problems, which emphasized in boldface, see figures 1-5.

Regarding the parameter sets: 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 set1 than with parameter set2. That means parameter set2 slows the algorithms and don’t make a bredging phenomina while parameter set1 accelerate the algorithms but somewhile make a bredging phenomina.

Looking at the number of function evaluations, the CPSO1was in the lead, followed by the CPSO2 algorithm, as shown in boldface (see Tables 1-5)

Considering, the above mentioned point that CPSO1 had no difficulty in reaching the goal and all its solutions are below their corresponding goals more than the other algorithms. So, we can conclude that CPSO1 is more superior to the other algorithms. That means we can consider it as a best alternative algorithm for solving optimization problems.

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					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F6	15	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
		CPSO3	286	307	177	273	64	114	704	613	0.40	0.55	10720	8366
		PSO	583	1203	138	126	63	91	3706	5853	0.45	0.4	19433	45113
	30	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
		CPSO3	348	401	271	234	73	111	1252	1769	0.65	0.60	16047	20025
		PSO	161	350	120	157	74	102	595	1264	0.75	0.60	6440	17500
	60	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
		CPSO3	402	305	217	146	72	75	1727	1378	0.85	0.95	28385	19257
		PSO	169	319	91	119	40	83	354	2361	0.90	0.95	11267	20147

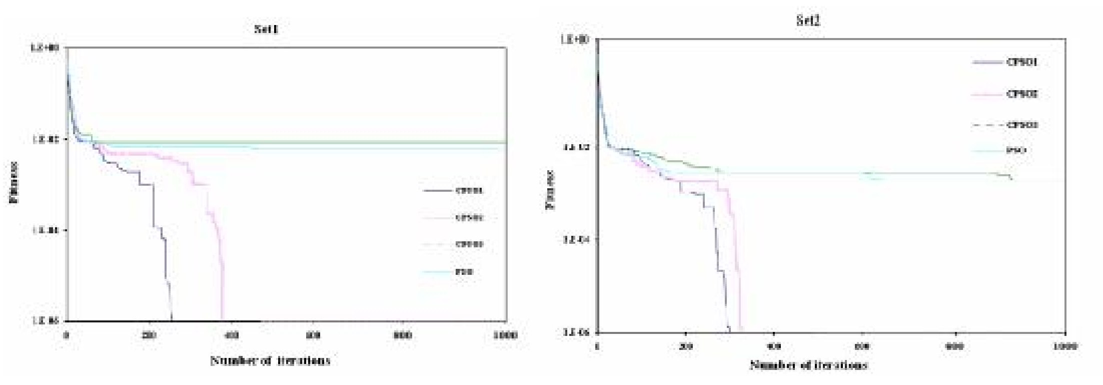


Figure5. Average best fitness curves for Schaffer function F6

8. Conclusion

This paper has proposed a new variation of the particle swarm optimization algorithm called a combined PSO, introducing a new term into the velocity component update equation: each particle is moved toward a new position according its best previous position and the point resulted from the combination of the best previous global position and the former best previous global position. The implementation of this idea is simple, based on storing the provious positions. The new algorithm outperfoms PSO on many benchmark functions, being less susceptible to premature convergence, and less likely to be stuck in local optima.

In this study, the CPSO1 has shown its worth on tested problems, and it outperformed CPSO2, CPSO3 and PSO on all the numerical benchmark problems as well. Among the tested algorithms, the CPSO1 can rightfully be regarded as an excellent first choice, when faced with a new optimization problem to solve.

To conclude, the performance of CPSO1 is outstanding in comparison to the other algorithms tested. It is simple, robust, converges fast, and finds the optimum in almost 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 CPSO, testing the new algorithm on other benchmark problems, and evaluating its performance relative to Evolutionary Algorithms.

9. References

[1] R. Eberhart and J. Kennedy, “A new 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”, Electri Power and Energy Syst 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] R. C. Eberhart and Y. Shi, “Comparing Inertia Weights and Constriction Factors In Particle Swarm Optimization” in: Proc. CEC, San Diego, CA, (2000) 84–88.

[8] Ioan Cristian Trelea, “The Particle Swarm Optimization Algorithm: Convergence Analysis And Parameter Selection”, Information Processing Letters 85 (2003) 317–325.

[9] R. Eberhart and Y. Shi, “Comparison between Genetic Algorithms and Particle Swarm Optimization”, The 7th Annual Conference on Evolutionary Programming, 1998, San Diego, USA.

[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] Van den Bergh, F., Engelbrecht, A. P., “Effects of Swarm Size on Cooperative Particle Swarm Optimizers”, Genetic and Evolutionary Computation Conference, San Francisco, USA, 2001.
[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] Jacques Riget and Jakob S. Vesterstorm, “A Diversity-Guided Particle Swarm Optimizer - The ARPSO” , EVALife Technical Report no. 2002-02.
[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.