Egyptian Informatics Journal (2011) **12**, 1–8

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

Particle swarm inspired optimization algorithm without velocity equation

Mahmoud Mostafa El-Sherbiny

Dept. of Quantitative Analysis, Faculty of Business Administration, King Saud University, Saudi Arabia

Received 7 September 2010; accepted 5 January 2011
Available online 31 March 2011

KEYWORDS

Particle swarm optimization;
Convergence;
Evolutionary computation

Abstract This paper introduces Particle Swarm Without Velocity equation optimization algorithm (PSWV) that significantly reduces the number of iterations required to reach good solutions for optimization problems. PSWV algorithm uses a set of particles as in particle swarm optimization algorithm but a different mechanism for finding the next position for each particle is used in order to reach a good solution in a minimum number of iterations. In PSWV algorithm, the new position of each particle is determined directly from the result of linear combination between its own best position and the swarm best position without using velocity equation. The results of PSWV algorithm and the results of different variations of particle swarm optimizer are experimentally compared. The performance of PSWV algorithm and the solution quality prove that PSWV is highly competitive and can be considered as a viable alternative to solve optimization problems.

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 [1–3]. The population in the PSO is called a swarm and each individual is called a particle [4]. 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 solve capacitor placement problem [5], short-term load forecasting [6], soft-sensor [7], to estimate the voltage stability of the electric power distribution systems [8,9] to direct the orbits of discrete chaotic dynamical systems towards desired target region [10] and to solve the permutation flowshop sequencing problem [11].

E-mail address: msherbiny@ksu.edu.sa

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2011.02.004

Production and hosting by Elsevier

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 function. Their movement is influenced not only by each particle's own previous experience, but also by a social compulsion to move towards the best position found by its neighbors. To implement these behaviors, 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. PSO, its main variants and the structural model behind it are extensively discussed in [12]. Some work is done that alters basic particle motion with some success, such as by El_Sherbiny [13,14], Li-Yong Wan and Wei Li, [15], Hui Wang et al. [16], Jianhua Liu et al. [17], and Yi Jiang and Qingling Yue [18] . But, the possibility for improvement in this area is still open.

This paper aims to introduce a Particle Swarm Without Velocity equation optimization algorithm (PSWV) and to discuss the results of experimentally comparing the performance of its versions with the standard particle swarm optimizer (PSO) [19,13,14].

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

2. Particle swarm optimization

Let us assume an n-dimensional search space, $S \subset \Re_n$ and denotes the size of the swarm population. Each particle represents a candidate solution and has the following attributes: (a) its current position in the search space $x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in S$, (b) its current own best position $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})$, (c) the global best position, $p_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$, i.e., the position of the best particle that gives the best fitness in the entire population; and (d) its current velocity $_i = (_{i1}, _{i2}, \ldots, _{in})$, which represents its position change. For each particle in the swarm in the next iteration step, the velocity is updated using (1), hence the position is updated using Eq. (2).

$$v_{id}(t+1) = \omega v_{id}(t) + b_1 r_1 (p_{id}(t) - x_{id}(t)) + b_2 r_2 (p_{id}(t) - x_{id}(t)) \quad (1)$$

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

The original PSO formulas developed by Kennedy and Eberhart [3] were combined by Shi and Eberhart [23] 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 trajectories to ensure convergence to a stable point [12],

$$\chi = \frac{b_1 r_1 P_i(t) + b_2 r_2 P_g}{b_1 r_1 + b_2 r_2}$$

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 Eq. (1) changes to Eq. (3).

$$v_{id}(t+1) = \chi(v_{id}(t) + b_1 r_1 (p_{id}(t) - x_{id}(t)) + b_2 r_2 (p_{id}(t) - x_{id}(t))) \quad (3)$$

Several interesting variations of the PSO algorithm have recently been proposed by researchers in [13–18]. 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 algorithm (CPSO) that presented by El_Sherbiny [13]. In the CPSO algorithm the particle updates its velocity according to Eq. (4) instead of Eq. (2), where (X_g) is the global best position, (X_{2g}) is the previous global best position and R_1, $R_2 \in U[0,1]$.

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

Another variation of PSO is the Modified Particle Swarm Optimizer (MPSO) [14]. In the MPSO algorithm, the particle updates its velocity according to Eq. (5) instead of Eq. (4) where constriction coefficient χ is presented.

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

3. PSWV algorithm

The Particle Swarm without velocity (PSWV) equation optimization Algorithm is built based on the random linear combination between the local best position and the global best position, which is represented by Eq. (6). The particles fly between its own previous best position and the global best position. It means that the new position of each particle is allocated in the area between its own current local best and the current global best.

$$x_i(t+1) = c_1 r_1 p_i(t) + c_2 r_2 p_g(t) \quad (6)$$

where r_1 and r_2 are defined as the combination weights, r_1, $r_2 \in U[0,1]$, and b_1, $b_2 \in [0,1]$ are defined as the own and the social attraction coefficients. $p_i(t)$ is the particle best position and $p_g(t)$ is the global best position at generation t.

In the PSWV algorithm, we do not need the velocity equation, which implies that Eqs. (1) and (2) be replaced with Eq. (6). The main steps of PSWV algorithm is illustrated in Fig. 1.

4. Convergence analysis of PSWV algorithm

For simplicity Eq. (6) can be rewritten as follows:

$$x_i(t+1) = \lambda_1 P_i(t) + \lambda_2 P_g \quad (7)$$

where $c_1 r_1 = \lambda_1$ and $c_2 r_2 = \lambda_2$. Consider that $\lambda_1 + \lambda_2 = 1$, Eq. (7) can be translated into Eq. (8)

$$x_i(t+1) = \lambda P_i(t) + (1-\lambda) P_g \quad (8)$$

Initialize swarm:

Random positions

Score each particle

Define P_i (local best) and P_g (Global best).

Repeat for each iteration

Repeat for each particle

Update positions of i^{th} particle along each dimension d according to Equation (6).

$$x_i(t+1) = c_1 r_1 p_i(t) + c_2 r_2 p_g(t) \quad (6)$$

Clip positions (if required) to preserve it within their limits.

Score solution represented by new particle positions.

Is this position the best found by this particle so far? Update P_i,

Is this the best found by any particle so far? Update P_g,

Solution is final global best P_g

Figure 1 Main steps of PSWV algorithm.

For simplicity we can consider that $P_i(t) = x_i(t)$. So, in the initial equation conditions $P_i(0) = x_i(0)$ that means that at the start iteration Eq. (8) would take the following form (Eq. (9)).

$$x_i(1) = \lambda x_i(0) + (1-\lambda)P_g \quad (9)$$

at $t = 1$ Eq. (9) will be

$$x_i(2) = \lambda x_i(1) + (1-\lambda)P_g \quad (10)$$

Substituting $x_i(1)$ from (8) and Eq. (10) yield Eq. (11) as follows:

$$\begin{aligned} x_i(2) &= \lambda[\lambda x_i(0) + (1-\lambda)P_g] + (1-\lambda)P_g \\ x_i(2) &= \lambda^2 x_i(0) + \lambda(1-\lambda)P_g + (1-\lambda)P_g \\ x_i(2) &= \lambda^2 x_i(0) + (\lambda+1)(1-\lambda)P_g \end{aligned} \quad (11)$$

Also, at $t = 2$ Eq. (8) will be

$$x_i(3) = \lambda^3 x_i(0) + (\lambda^2+\lambda+1)(1-\lambda)P_g$$

So, recurrence relation can be obtained as follows:

$$\begin{aligned} x_i(t+1) &= \lambda^t x_i(0) + (\lambda^{t-1} + \lambda^{t-2} + \ldots + \lambda + 1)(1-\lambda)P_g x_i(t+1) \\ &= \lambda^t x_i(0) + \frac{(1-\lambda^t)}{(1-\lambda)}(1-\lambda)P_g \\ x_i(t+1) &= \lambda^t x_i(0) + (1-\lambda^t)P_g \end{aligned} \quad (12)$$

where $0 \leqslant \lambda \leqslant 1$

We will calculate the limits of $x_i(t+1)$ at t tend to infinity as follows:

$$\begin{aligned} \lim_{t\to+\infty} x_i(t+1) &= \lim_{t\to+\infty} (\lambda^t x_i(0) + (1-\lambda^t)P_g) \\ \lim_{t\to+\infty} x_i(t+1) &= \lim_{t\to+\infty} \lambda^t x_i(0) + \lim_{t\to+\infty} (1-\lambda^t)P_g \\ \lim_{t\to+\infty} x_i(t+1) &= \lim_{t\to+\infty} \lambda^t x_i(0) + \lim_{t\to+\infty} (-\lambda^t P_g) + \lim_{t\to+\infty} P_g \\ \lim_{t\to+\infty} x_i(t+1) &= P_g \end{aligned} \quad (13)$$

From Eq. (13) we can conclude that $x_i(t+1)$ will converge to P_g.

5. Test functions and test conditions

In order to know how competitive the PSWV algorithm is, we decided to compare its two versions (PSWV1 and PSWV2) against many versions of particle swarm algorithms as follows: Two versions of PSO algorithm (PSO1 and PSO2) that is represented in [19], two versions of Modified PSO algorithm (MPSO1 and MPSO2) that is represented in [13], and two versions of Combined PSO algorithm (CPSO1 and CPSO2) that is represented in [14]. Five benchmarking functions were selected to examine the performance of such Algorithms. The considered benchmark functions were used in [20,21,19]. The benchmark functions and its dimensions are illustrated in Table 1, while, its admissible range of the variable (x), the goal values, and the optimal solution are summarized in Table 2.

Two parameter sets (Eqs. (1)–(3)) $\chi = a$ 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, empirically, to provide good performance [11,12,21], and used in testing the PSO by Trelea [19].

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 [21]. c_1 and c_2 in Eq. (3) are set to be 1/1.7 for PSWV1 algorithm.

Parameter set 2 ($a = 0.729$ and $b = 1.494$) was recommended by Clerc [22] and also tested in [21] giving the best results published so far known to the author. c_1 and c_2 in Eq. (4) are set to be 1/1.494 for PSWV2 algorithm.

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 PSWV and the other algorithms [13,14,19] two sets of experiments were used with

Table 1 Benchmark functions.

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

Optimal value of all functions is 0.

Table 2 Result of test functions F_0, F_1, and F_2.

No. of Par. N	Algorithm version	Sphere function F_0			Rosenbrock function F_1			Rastrigin function F_2		
		Average number of iterations	Success rate (%)	Ex. No. of Fn. evaluation	Average number of iterations	Success rate (%)	Ex. No. of Fn. evaluation	Average number of iterations	Success rate (%)	Ex. No. of Fn. evaluation
15	**PSWV1**	**14**	**1**	**213**	**9**	**1**	**132**	**8**	**1**	**113**
	PSWV2	18	1	266	11	1	164	9	1	137
	MPSO1	56	1	844	34	1	517	23	1	344
	MPSO2	94	1	1416	59	1	888	38	1	576
	CPSO1	125	1	1874	88	1	1318	57	1	857
	CPSO2	168	1	2516	112	1	1673	81	1	1217
	PSO1	769	0.4	28,838	531	0.50	15,930	172	0.35	7371
	PSO2	764	1	11,460	1430	1	21,450	299	0.80	5606
30	**PSWV1**	**14**	**1**	**419**	**9**	**1**	**266**	**7**	**1**	**203**
	PSWV2	17	1	515	11	1	321	9	1	260
	MPSO1	53	1	1575	32	1	953	22	1	667
	MPSO2	88	1	2650	55	1	1645	37	1	1111
	CPSO1	131	1	3917	84	1	2520	48	1	1451
	CPSO2	180	1	5396	105	1	3149	68	1	2054
	PSO1	344	1	10,320	614	1	18,420	140	0.90	4667
	PSO2	395	1	11,850	900	1	27,000	182	0.95	5747
60	**PSWV1**	**14**	**1**	**819**	**8**	**1**	**504**	**7**	**1**	**420**
	PSWV2	17	1	999	10	1	624	9	1	513
	MPSO1	49	1	2961	30	1	1789	21	1	1271
	MPSO2	85	1	5080	53	1	3145	36	1	2180
	CPSO1	118	1	7083	77	1	4623	60	1	3603
	CPSO2	157	1	9423	102	1	6138	69	1	4152
	PSO1	252	1	15,120	337	1	20,220	122	0.95	7705
	PSO2	314	1	18,840	611	1	36,660	166	1	9960

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 in the range $[x_{\min}, x_{\max}]$ indicated in Table 1. Population sizes of $N = 15$, 30 and 60 particles were tested. Average number, success rate of required iterations, and expected number of function evaluations, for each test function are calculated and presented in Tables 3–7.

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 Figs. 2–6.

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 [19]. Tables 3–7 present the following information: Average number of iterations required to reach a function value below the goal, "success rate", 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.

$$\text{Ex. \# of Fn. evaluation} = (\text{average number of iterations}) \times (\text{number of particles in the swarm}) / (\text{success rate})$$

Tables 3 and 4 show that while SPOS1 algorithm with 15 particles failed to reach the goal during some runs for solving the Sphere (F_0) and Rosenbrock (F_1) functions, the other algorithms succeeded to reach the goal during all runs. This means the number of particles affects the convergence of the standard PSO algorithm for such type of problems while such effect is not with the PSWV algorithm. PSWV1 and PSWV1 algorithms succeeded to reach the goal in a few numbers of iterations than the remaining algorithms.

Also, as illustrated in Fig. 2, the two versions of PSWV algorithm (PSWV1 and PSWV2) are the candidates to reach the optimal solution in a few numbers of iterations than the remaining algorithms. This means the convergence speed of PSWV is faster than the other algorithms and it is more robust than the others.

While all the algorithms have no difficulties in reaching the goal of Rosenbrock function (F_1), the PSO1 algorithm with 15 particles had difficulties for solving such function. Also, the expected number of function evaluation needed for the two versions of PSWV algorithm is less than the expected number of function evaluation needed for the other algorithms. This means the speed of PSWV algorithm is faster than the other

Table 3 Result of test functions F_3 and F_6.

No. of Par. N	Algorithm version	Griewank function F_3			Schaffer's function F_6		
		Average number of iterations	Success rate (%)	Ex. No. of fn. evaluation	Average number of iterations	Success rate (%)	Ex. No. of fn. evaluation
15	PSWV1	13	1	194	487	0.95	7696
	PSWV2	19	1	278	583	0.85	10,285
	MPSO1	53	1	790	148	1	2216
	MPSO2	89	1	1332	142	1	2132
	CPSO1	132	1	1982	198	1	2971
	CPSO2	215	1	3218	232	1	3476
	PSO1	689	0.35	29,529	583	0.45	19,433
	PSO2	755	0.60	18,875	1203	0.40	45,113
30	PSWV1	13	1	384	263	1	7877
	PSWV2	19	1	582	400	0.95	12,620
	MPSO1	47	1	1417	93	1	2791
	MPSO2	83	1	2479	128	1	3826
	CPSO1	131	1	3929	122	1	3668
	CPSO2	168	1	5046	148	1	4434
	PSO1	313	0.90	10,433	161	0.75	6440
	PSO2	365	0.90	12,167	350	0.60	17,500
60	PSWV1	12	1	744	105	1	6291
	PSWV2	17	1	1044	221	0.95	13,981
	MPSO1	45	1	2700	60	1	3616
	MPSO2	76	1	4554	83	1	4954
	CPSO1	113	1	6780	93	1	5559
	CPSO2	152	1	8392	112	1	6708
	PSO1	226	0.95	14,274	169	0.90	11,267
	PSO2	287	1	17,220	319	0.95	20,147

Table 4 The result of test function F_1.

No. of Par. N	Algorithm version	Average number of iterations	Success rate	Ex. No. of Fn. evaluation
15	PSWV1	9	1	132
	PSWV2	11	1	164
	MPSO1	34	1	517
	MPSO2	59	1	888
	CPSO1	88	1	1318
	CPSO2	112	1	1673
	PSO1	531	0.50	15,930
	PSO2	1430	1	21,450
30	PSWV1	9	1	266
	PSWV2	11	1	321
	MPSO1	32	1	953
	MPSO2	55	1	1645
	CPSO1	84	1	2520
	CPSO2	105	1	3149
	PSO1	614	1	18,420
	PSO2	900	1	27,000
6	PSWV1	8	1	504
	PSWV2	10	1	624
	MPSO1	30	1	1789
	MPSO2	53	1	3145
	CPSO1	77	1	4623
	CPSO2	102	1	6138
	PSO1	337	1	20,220
	PSO2	611	1	36,660

Table 5 The result of test function F_2.

No. of Par. N	Algorithm version	Average number of iterations	Success rate	Ex. No. of Fn. evaluation
15	PSWV1	8	1	113
	PSWV2	9	1	137
	MPSO1	23	1	344
	MPSO2	38	1	576
	CPSO1	57	1	857
	CPSO2	81	1	1217
	PSO1	172	0.35	7371
	PSO2	299	0.80	5606
30	PSWV1	7	1	203
	PSWV2	9	1	260
	MPSO1	22	1	667
	MPSO2	37	1	1111
	CPSO1	48	1	1451
	CPSO2	68	1	2054
	PSO1	140	0.90	4667
	PSO2	182	0.95	5747
60	PSWV1	7	1	420
	PSWV2	9	1	513
	MPSO1	21	1	1271
	MPSO2	36	1	2180
	CPSO1	60	1	3603
	CPSO2	69	1	4152
	PSO1	122	0.95	7705
	PSO2	166	1	9960

Table 6 The result of test function F_3.

# of Par. N	Algorithm version	Average number of iterations	Success rate	Ex. # of Fn. evaluation
15	PSWV1	13	1	194
	PSWV2	19	1	278
	MPSO1	53	1	790
	MPSO2	89	1	1332
	CPSO1	132	1	1982
	CPSO2	215	1	3218
	PSO1	689	0.35	29,529
	PSO2	755	0.60	18,875
30	PSWV1	13	1	384
	PSWV2	19	1	582
	MPSO1	47	1	1417
	MPSO2	83	1	2479
	CPSO1	131	1	3929
	CPSO2	168	1	5046
	PSO1	313	0.90	10,433
	PSO2	365	0.90	12,167
60	PSWV1	12	1	744
	PSWV2	17	1	1044
	MPSO1	45	1	2700
	MPSO2	76	1	4554
	CPSO1	113	1	6780
	CPSO2	152	1	8392
	PSO1	226	0.95	14,274
	PSO2	287	1	17,220

Table 7 The result of test function F6.

# of Par. N	Algorithm version	Average number of iterations	Success rate	Ex. # of Fn. evaluation
15	PSWV1	487	0.95	7696
	PSWV2	583	0.85	10,285
	MPSO1	148	1	2216
	MPSO2	142	1	2132
	CPSO1	198	1	2971
	CPSO2	232	1	3476
	PSO1	583	0.45	19,433
	PSO2	1203	0.40	45,113
30	PSWV1	263	1	7877
	PSWV2	400	0.95	12,620
	MPSO1	93	1	2791
	MPSO2	128	1	3826
	CPSO1	122	1	3668
	CPSO2	148	1	4434
	PSO1	161	0.75	6440
	PSO2	350	0.60	17,500
60	PSWV1	105	1	6291
	PSWV2	221	0.95	13,981
	MPSO1	60	1	3616
	MPSO2	83	1	4954
	CPSO1	93	1	5559
	CPSO2	112	1	6708
	PSO1	169	0.90	11,267
	PSO2	319	0.95	20,147

algorithms tested in this paper in searching for the solution of such kind of problems (see Table 4).

Also, Fig. 3 illustrates that the PSWV1, PSWV2 reached values below the function goal in different numbers of iterations and below that value reached by the remaining algorithms. This means that solution quality of PSWV algorithm is better than that of other algorithms for such kind of problems.

Table 5 shows that the two versions of PSWV algorithm perform admirably on the Rastrigin function (F_2), while the two versions of SPO algorithm are less robust on the same function for such type of problems where they had some difficulties in reaching the goal in some runs.

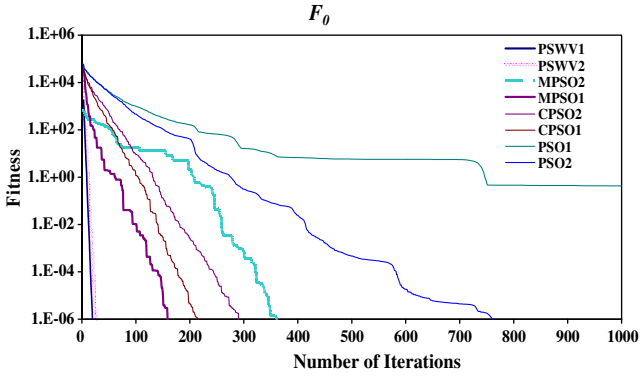

Figure 2 Average best fitness curves for Sphere function (F_0).

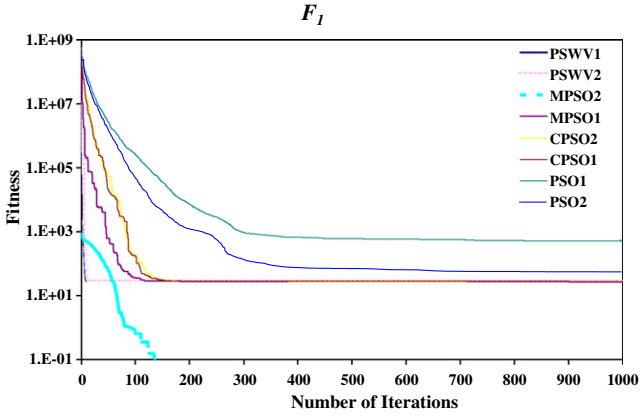

Figure 3 Average best fitness curves for Rosenbrock function (F_1).

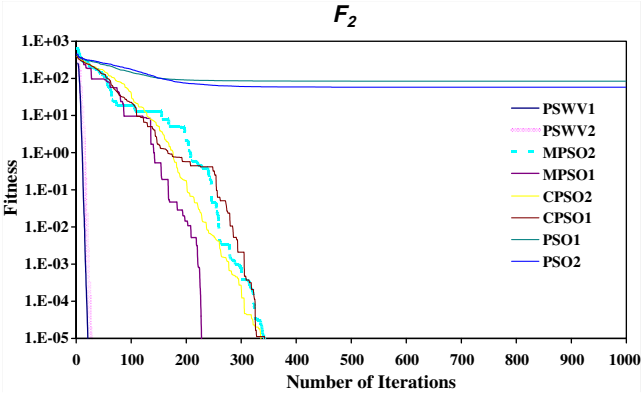

Figure 4 Average best fitness curves of functions F_0, F_1, and F_2.

Table 5 and Fig. 4 illustrate that the PSWV1 and PSWV1 algorithms are doing very well on such problem type, delivering the best overall performance for the Rastrigin function (F_2), where they reached the goal in approximately less than 10 iterations and they are candidates to reach the optimal solution in approximately less than 100 iterations. While all versions of CPSO and MPSO algorithms are candidates to reach it in approximately less than 200, 400 iterations PSO1 and PSO2 are stacked at a solution value far from the optimal solution by $10e + 02$. This means that the PSWV algorithm is superior to all algorithms as shown in Fig. 4.

Griewank's function (F_3) proves to be hard to solve with the two versions of PSO [19] algorithm where it had some difficulties in reaching its goal. While PSO had some difficulties in reaching its goal the other algorithms consistently reached the goal in all runs as can be seen in Table 6.

Fig. 5 illustrates that the versions of PSWV, MPSO, and CPSO algorithms are candidates to reach the optimal solution of Griewank function (F_3) while both versions of PSO did not reach the goal during some runs.

Concerning the Schaffer's function (F_6), Table 6 illustrates that while the two versions of PSWV, MPSO, and CPSO algorithms reached the goal in all runs, the two versions of standard PSO algorithm have some difficulties in reaching the goal.

Fig. 6 illustrates that: using the average of 30 runs for 1000 iterations of each algorithm, the versions of PSWV algorithm reached a solution value (10^{-3}) below the goal in different number of iterations and be candidates to reach the goal while the two versions of SPO algorithm are stacked at value 10^{-2} greater than the goal. PSWV1 reached a value below the goal in 13 iterations on average while the two versions of PSO needed more than 200 iterations and the four versions of MPSO and CPSO algorithms come in between them.

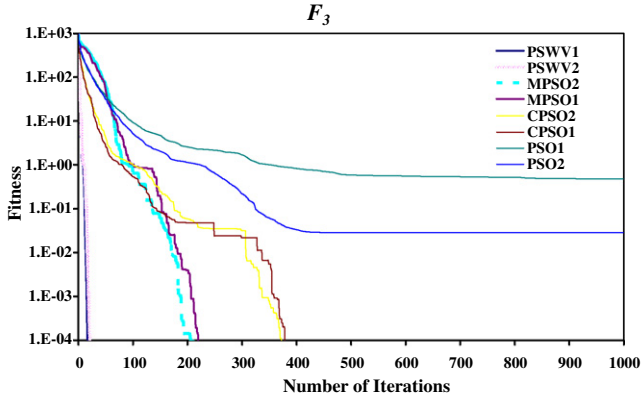

Figure 5 Average best fitness curves of functions F_3 and F_6.

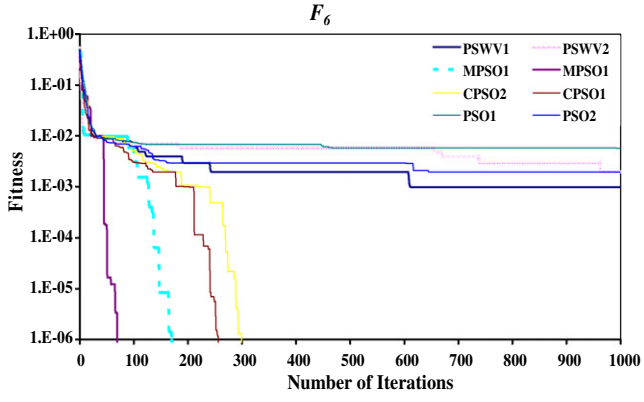

Figure 6 Average best fitness curves for Schaffer's function F_6.

8. Discussion

Overall, as far as robustness is concerned, the PSWV algorithm appears to be the winner, since its two versions (PSWV1 and PSWV2) achieved perfect scores in most test cases as represented in boldface (see Tables 3–7). As a result, the PSO [19] must be executed several times to ensure good results, whereas one run of PSWV and MPSO [13] are usually sufficient.

Note that, there is a little difference between the performance of the algorithms with parameter set1 and with parameter set 2 where all the algorithms' convergence under study with parameter set1 are faster than algorithms' conversance with parameter set 2 (see Tables 3–7). Regarding convergence speed, PSWV1 is always the fastest followed by PSWV2, whereas the PSO1 and PSO2 are always the slowest. Especially on the all functions, PSWV1 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, PSWV is clearly the best performing algorithms under study. It finds the lowest fitness value for most of the problems, see Figs. 2–6.

Looking at the number of function evaluations, the PSWV was in the lead, followed by the MPSO [13] algorithm and PSO [19] algorithm comes in the last as shown in Tables 3–7.

Considering the above mentioned point that the two versions of PSWV have no difficulty in reaching the goal and all its solutions are below their corresponding goals more than other algorithms under study can conclude that PSWV is more superior and robust. So, we can consider it as the best alternative algorithm for solving optimization problems.

9. Conclusion

This paper has proposed a new particle swarm- inspired optimization algorithm (PSWV). In this algorithm the new position of each particle is calculating directly from the combination of its own best position and the global best position. The implementation of this idea is simple, based on storing the previous positions. The PSWV algorithm outperforms all the algorithms under study on many benchmark functions, being less susceptible to premature convergence, and less likely to be stuck in local optima.

In this study, the PSWV has shown its worth on tested problems, and it outperformed MPSO [13], CPSO [14], and PSO [19] algorithms on all the numerical benchmark problems as well. Among the tested algorithms, the PSWV can rightfully be regarded as an excellent first choice, when faced with a new optimization problem to solve.

To conclude, the performance of PSWV is outstanding in comparison to MPSO [13], CPSO [14], and PSO [19] algorithms. 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 PSWV, testing the new algorithm on other benchmark problems, and evaluating its performance relative to evolutionary algorithms.

Acknowledgement

The author is grateful for the valuable comments and suggestions made by the reviewers, which helped in improving the contents of the paper.

References

[1] Eberhart R, Kennedy J. A modified optimizer using particle swarm theory. In: Proc. 6th Int. Symp. Micro Machine and Human Science (Nagoya, Japan); 1995. p. 39–43.
[2] Eberhart RC, Kennedy J. A new optimizer using particles swarm theory. In: Proc. sixth international symposium on micro machine and human science (Nagoya, Japan). IEEE Service Center: Piscataway, NJ; 1995. p. 39–43.
[3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc. IEEE international conference on neural networks (Perth, Australia), vol. IV. IEEE Service Center: Piscataway, NJ; 1995. p. 1942–8.
[4] Kennedy J. The particle swarm: social adaptation of knowledge. In: Proc. IEEE international conference on evolutionary computation (Indianapolis, Indiana). IEEE Service Center: Piscataway, NJ; 1997, p. 303–8.
[5] Naing win O. A comparison study on particle swarm and evolutionary particle swarm optimization using capacitor placement problem. 2nd IEEE international conf. on power and energy, Dec. 2008; 2008. p. 1249–2.
[6] Wang B, Tai Neng-ling, Zhai Hai-qing, Ye Jian, Zhuc Jia-dong, Qic Liang-bo. A new ARMAX model based on evolutionary algorithm and particle swarm optimization for short-term load forecasting. Electr Power Syst Res 2008;78:1679–85.
[7] Wang Hui, Qian Feng. An improved particle swarm optimizer with behavior-distance models and its application in soft-sensor. 7th World Congress Intell Control Automation 2008:4473–8.
[8] Shigenori N, Takamu G, Toshiku Y, Yoshikazu F. A hybrid particle swarm optimization for distribution state estimation. IEEE Trans Power Syst 2003;18:60–8.
[9] EL-Dib Amgad A, Youssef Hosam M, EL-Metwally MM, Osman Z. Maximum loadability of power systems using hybrid particle swarm optimization. Electr Power Syst Res 2006;76: 485–92.
[10] Liu Bo, Wang Ling, Jin Yi-Hui, Tang Fang, Huang De-Xian. Directing orbits of chaotic systems by particle swarm optimization. Chaos, Solitons Fractals 2006;29:454–61.
[11] Tasgetiren M, Liang Yun-Chia, Sevkli Mehmet, Gencyilmaz Gunes. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 2007;1930–47.
[12] Clerc M, Kennedy J. The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans Evol Comput 2002;6:58–73.
[13] El_Sherbiny MM. A combined particle swarm optimization algorithm based on the previous global best and the global best positions. Int J Comput Inf (IJCI) 2007;1:13–26.
[14] El-Sherbiny MM. A modified algorithm for particle swarm optimization with constriction coefficient. Int J Comp Inf (IJCI) 2009;2:17–30.
[15] Wan Li-Yong, Li Wei. An improved particle swarm optimization algorithm with rank-based selection. In: Proceedings of the seventh international conference on machine learning and cybernetics, Kunming, 12–15 July 2008; 2008. p. 4090–5.
[16] Wang Hui, Liu Yong, Wu Zhijian, Sun Hui, Zeng Sanyou, Kang Lishan. An improved Particle Swarm Optimization with adaptive jumps. IEEE World Congress Comput Intell 2008:392–7.
[17] Liu Jianhua, Fan Xiaoping, Qu Zhihua. An improved particle swarm optimization with mutation based on similarity. Third Int Conf Nat Comput 2007:824–8.
[18] Jiang Yi, Yue Qingling. An improved particle swarm optimization with new select mechanism. Int Workshop Knowledge Discov Data Mining 2008:383–6.
[19] Trelea Ioan Cristian. The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 2003;85:317–25.
[20] Gazi Veysel, Passino Kevin M. Stability analysis of social foraging swarms. IEEE Trans Syst Man Cybern B Cybern 2004;34(1): 539–57.
[21] Boeringer Daniel W, Werner Douglas H. Particle swarm optimization versus genetic algorithms for phased array synthesis. IEEE Trans Antenna Propagation 2004;52(3):771–9.
[22] Clerc M. The Swarm And The Queen: Towards A Deterministic And Adaptive Particle Swarm Optimization. Proc ICEC, Washington, DC 1999:1951–7.
[23] Shi Y, Eberhart R. A combined particle swarm optimizer. Proc IEEE World Congress Comput Intell 1998:69–73.