

[Egyptian Informatics Journal (2011) 12, 1](http://dx.doi.org/10.1016/j.eij.2011.02.004)–[8](http://dx.doi.org/10.1016/j.eij.2011.02.004)

Cairo University

Egyptian Informatics Journal

[www.elsevier.com/locate/eij](http://www.elsevier.com/locate/eij) [www.sciencedirect.com](http://www.sciencedirect.com/science/journal/11108665)

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

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 algo- rithm and the results of different variations of particle swarm optimizer are experimentally com- pared. 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.

© 2011 Faculty of Computers and Information, Cairo University.

KEYWORDS

Particle swarm optimization; Convergence;

Evolutionary computation

Production and hosting by Elsevier B.V. All rights reserved.

1. Introduction

Particle Swarm Optimization (PSO) is a new evolutionary com- putation technique motivated from the simulation of social

behavior and originally designed and developed by Eberhart

and Kennedy [[1–3]](#_bookmark21). The population in the PSO is called a swarm and each individual is called a particle [[4]](#_bookmark22). 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 ben-

E-mail address: [msherbiny@ksu.edu.sa](mailto:msherbiny@ksu.edu.sa)

1110-8665 © 2011 Faculty of Computers and Information, Cairo University. Production and hosting by Elsevier B.V. All rights reserved.

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

doi:[10.1016/j.eij.2011.02.004](http://dx.doi.org/10.1016/j.eij.2011.02.004)

Production and hosting by Elsevier

efits from the experience of its own and that of the other mem- bers of the swarm during the search for food.

PSO algorithm has a number of desirable properties, includ- ing 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]](#_bookmark23), short- term load forecasting [[6]](#_bookmark28), soft-sensor [[7]](#_bookmark30), to estimate the voltage stability of the electric power distribution systems [[8,9]](#_bookmark32) to direct the orbits of discrete chaotic dynamical systems towards desired target region [[10]](#_bookmark24) and to solve the permutation flowshop sequencing problem [[11]](#_bookmark24).

PSO has been successfully used as an alternative to other evolutionary algorithms in the optimization of D-dimen- sional real functions. Particles move in a coordinated way through the D-dimensional search space towards the opti- mum function. Their movement is influenced not only by

cient is to prevent the velocity to grow out of bounds, with the advantage that, theoretically, velocity clamping is no long- er required. As a result of this study, the velocity Eq. [(1)](#_bookmark3) changes to Eq. [(3)](#_bookmark0).

*vid*(*t* + 1)= v(*vid*(*t*)+ *b*1*r*1(*p* (*t*)— *xid*(*t*)) + *b*2*r*2(*p* (*t*)— *xid*(*t*)))

each particle’s own previous experience, but also by a social compulsion to move towards the best position found by its

*id id*

(3)

neighbors. To implement these behaviors, each particle is de- fined by its position and velocity in the search space. In each iteration, changes resulting from both influences in the par- ticle’s trajectory are made to its velocity. The particle’s posi- tion is then updated accordingly to the calculated velocity. PSO, its main variants and the structural model behind it are extensively discussed in [[12]](#_bookmark24). Some work is done that al- ters basic particle motion with some success, such as by El\_Sherbiny [[13,14]](#_bookmark24), Li-Yong Wan and Wei Li, [[15]](#_bookmark24), Hui Wang et al. [[16]](#_bookmark24), Jianhua Liu et al. [[17]](#_bookmark25), and Yi Jiang and Qingling Yue [[18]](#_bookmark26) . 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 dis- cuss the results of experimentally comparing the performance of its versions with the standard particle swarm optimizer (PSO) [[19,13,14]](#_bookmark27).

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 func- tions 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.

1. Particle swarm optimization

Let us assume an *n*-dimensional search space, *S* ffi*n* and de- notes 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 *xi* = (*xi*1, *xi*2, *.. .*, *xin*) ∈ *S*,

(b) its current own best position *pi* = (*pi*1, *pi*2, *.. .*, *pin*), (c) the global best position, *pg* = (*pg*1, *pg*2, *.. .*, *pgn*), i.e., the position of the best particle that gives the best fitness in the entire pop- ulation; and (d) its current velocity *i* = (*i*1, *i*2, *.. .*, *in*), which rep-

resents its position change. For each particle in the swarm in the next iteration step, the velocity is updated using [(1)](#_bookmark3), hence the position is updated using Eq. (2).

*vid*(*t* + 1)= x*vid*(*t*)+ *b*1*r*1(*pid* (*t*)— *xid*(*t*))

+ *b*2*r*2(*pid* (*t*)— *xid*(*t*)) (1)

*xid*(*t* + 1)= *xid*(*t*)+ *vid*(*t* + 1) (2)

The original PSO formulas developed by Kennedy and Eber- hart [[3]](#_bookmark21) were combined by Shi and Eberhart [[23]](#_bookmark34) with the intro- duction of an inertia parameter, x, that was shown empirically to improve the overall performance of PSO. Clerc and Ken- nedy provided a theoretical analysis of particle trajectories to ensure convergence to a stable point [[12]](#_bookmark24),

*b r P* (*t*)+ *b r P*

Several interesting variations of the PSO algorithm have re- cently been proposed by researchers in [[13–18]](#_bookmark24). Many of these PSO improvements are essentially extrinsic to the par- ticle 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]](#_bookmark24). In the CPSO algorithm the particle updates its velocity according to Eq. [(4)](#_bookmark0) instead of Eq. (2), where (*Xg*) is the global best position, (*X*2*g*) is the previous global best position and *R*1, *R*2 ∈ *U*[0, 1].

*Vi*(*t*)= *aVi*(*t* — 1)+ *b*1*r*1(*Xpi* — *Xi*(*t*)) + *b*1*r*1 (*R*1*Xg* + *R*2*X*2*g*)— *Xi*(*t*)

ÿ

(4)

Another variation of PSO is the Modified Particle Swarm Optimizer (MPSO) [[14]](#_bookmark24). In the MPSO algorithm, the particle updates its velocity according to Eq. [(5)](#_bookmark1) instead of Eq. [(4)](#_bookmark0) where constriction coefficient v is presented.

*Vi*(*t*)= v(*Vi*(*t* — 1)+ *b*1*r*1(*Xpi* — *Xi*(*t*)) + *b*2*r*2((*R*1*Xg* + *R*2*X*2*g*)— *Xi*(*t*)))

(5)

1. PSWV algorithm

The Particle Swarm without velocity (PSWV) equation optimi- zation Algorithm is built based on the random linear combina- tion between the local best position and the global best position, which is represented by Eq. [(6)](#_bookmark2). The particles fly be- tween its own previous best position and the global best posi- tion. 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.

*xi*(*t* + 1)= *c*1*r*1*pi*(*t*)+ *c*2*r*2*pg*(*t*) (6)

where *r*1 and *r*2 are defined as the combination weights, *r*1, *r*2 ∈ *U*[0, 1], and *b*1, *b*2 ∈ [0, 1] are defined as the own and the social attraction coefficients. *pi*(*t*) is the particle best position and *pg*(*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)](#_bookmark3) be replaced with Eq. [(6)](#_bookmark2). The main steps of PSWV algorithm is illustrated in [Fig. 1](#_bookmark6).

1. Convergence analysis of PSWV algorithm

For simplicity Eq. [(6)](#_bookmark2) can be rewritten as follows:

*xi*(*t* + 1)= k1*Pi*(*t*)+ k2*Pg* (7)

v = 1 1 *i*

2 2 *g*

*b*1*r*1 + *b*2*r*2

The main result of this work is the introduction of the constric- tion coefficient and different classes of constriction models. The objective of this theoretically derived constriction coeffi-

where *c*1*r*1 = k1 and *c*2*r*2 = k2. Consider that k1 + k2 = 1, Eq.

[(7)](#_bookmark4) can be translated into Eq. [(8)](#_bookmark5)

*xi*(*t* + 1)= k*Pi*(*t*)+ (1 — k)*Pg* (8)

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

lim *xi*(*t* + 1)= lim (k*txi*(0)+ (1 — k*t*)*Pg*)

*t*→+∞ *t*→+∞

lim *xi*(*t* + 1)= lim k*txi*(0)+ lim (1 — k*t*)*Pg*



*t*→+∞

*t*→+∞

*t*→+∞

lim *xi*(*t* + 1)= lim k*txi*(0)+ lim (—k*tPg*)+ lim *Pg*

*t*→+∞

*t*→+∞

*t*→+∞

*t*→+∞

 *t* lim *xi*(*t* + 1)= *Pg* (13)

→+∞

From Eq. [(13)](#_bookmark8) we can conclude that *xi*(*t* + 1) will converge to

*Pg*.

1. Test functions and test conditions





Figure 1 Main steps of PSWV algorithm.

For simplicity we can consider that *Pi*(*t*)= *xi*(*t*). So, in the ini- tial equation conditions *Pi*(0) = *xi*(0) that means that at the start iteration Eq. [(8)](#_bookmark5) would take the following form (Eq. [(9)](#_bookmark7)).

*xi*(1)= k*xi*(0)+ (1 — k)*Pg* (9)

at *t* = 1 Eq. [(9)](#_bookmark7) will be

*xi*(2)= k*xi*(1)+ (1 — k)*Pg* (10)

Substituting *xi*(1) from (8) and Eq. (10) yield Eq. [(11)](#_bookmark9) as follows:

*xi*(2)= k[k*xi*(0)+ (1 — k)*Pg*]+ (1 — k)*Pg*

*x* (2)= k *x* (0)+ k(1 — k)*P* + (1 — k)*P*

*i*

*i*

*g*

*g*

2

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 rep- resented in [[19]](#_bookmark27), two versions of Modified PSO algorithm (MPSO1 and MPSO2) that is represented in [[13]](#_bookmark24), and two ver- sions of Combined PSO algorithm (CPSO1 and CPSO2) that is represented in [[14]](#_bookmark24). Five benchmarking functions were selected to examine the performance of such Algorithms. The consid- ered benchmark functions were used in [[20,21,19]](#_bookmark29). The bench- mark functions and its dimensions are illustrated in [Table 1](#_bookmark10), while, its admissible range of the variable (*x*), the goal values, and the optimal solution are summarized in [Table 2](#_bookmark11).

Two parameter sets (Eqs. [(1)–(3)](#_bookmark3)) v = *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, empir- ically, to provide good performance [[11,12,21]](#_bookmark24), and used in testing the PSO by Trelea [[19]](#_bookmark27).

*Parameter set* 1 (*a* = 0.6 and *b* = 1.7) was selected by the author in the algorithm convergence domain after a large num- ber of simulation experiments [[21]](#_bookmark31). *c*1 and *c*2 in Eq. [(3)](#_bookmark0) are set

to be 1/1.7 for PSWV1 algorithm.

*xi*(2)= k2*xi*(0)+ (k + 1)(1 — k)*Pg* (11)

Also, at *t* = 2 Eq. [(8)](#_bookmark5) will be

*xi*(3)= k3*xi*(0)+ (k2 + k + 1)(1 — k)*Pg*

So, recurrence relation can be obtained as follows:

*xi*(*t* + 1)= k*txi*(0)+ (k*t*—1 + k*t*—2 + *...* + k + 1)(1 — k)*Pgxi*(*t* + 1)

= k*tx* (0)+ (1 — k ) (1 — k)*P*

*t*

*i*

*g*

*Parameter set* 2 (*a* = 0.729 and *b* = 1.494) was recom- mended by Clerc [[22]](#_bookmark33) and also tested in [[21]](#_bookmark31) giving the best re- sults published so far known to the author. *c*1 and *c*2 in Eq. [(4)](#_bookmark0) 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]](#_bookmark27).

1. Optimization test experiments

(1 — k)

*xi*(*t* + 1)= k*txi*(0)+ (1 — k*t*)*Pg* (12)

where 0 6 k 6 1

In order to test the performance of the PSWV and the other algorithms [[13,14,19]](#_bookmark24) two sets of experiments were used with

Table 1 Benchmark functions.

Function

Sphere *F*0(→*x*)= *x*

1

P

*D*

2

*i*=1 *i*

Rosenbrock *F* (→*x*

)= (

2

P

*D*—

1

2

2

2

*i*=1

100

(*x*

*i*

+1

— *x* ) + (

*i*

*x*

*i*

— 1) )

Rastrigin *F* (→*x*

)= (

Griewank *F*3(→*x*)=

P

*D*

1

*i*=1 *i*

*x* — 10 cos

2

(

2 *x* )+ 10

Q

*i*

)

4000

Dim

30

30

30

30

Range [*x*min, *x*max]

*D*

[—100, 100]

*D*

[—30, 30]

*D*

[—5.12, 5.12]

*D*

[—600, 600]

Goal for *F*

0.01

100

100

0.1

Schaffer’s *F* (→*x*)=

0.5 —

P

*D*

2

*i*=1 *i i*=1

(sin *x* +*x* ) —0.5

,

*x* — cos

ﬃﬃﬃﬃﬃﬃﬃﬃ**ﬃ**

2 2

1 2

Q

*D*

( )+

6

(

1+0.001

(*x* +

2 2

,ﬃ

2

*xi*

*i*

1

2

2

1 2

*x* ))

[—100, 100]

2

10

—5

Optimal value of all functions is 0.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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.

Table 2 Result of test functions *F*0, *F*1, and *F*2.

No. of Par. N Algorithm Sphere function *F*0

Rosenbrock function *F*1

version

Rastrigin function *F*2

Average Success Ex. No. Average Success Ex. No. Average Success Ex. No. of number of rate (%) of Fn. number of rate (%) of Fn. number of rate (%) Fn. evaluation iterations evaluation iterations evaluation iterations

In the first set of experiments, the maximum iteration num- ber 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](#_bookmark10). Population sizes of *N* = 15, 30 and 60 particles were tested. Average number, success rate of required iterations, and expected number of function evalu- ations, for each test function are calculated and presented in [Tables](#_bookmark12) [3–7](#_bookmark12).

In the second set of experiments, each optimization experi- ment was run 20 times for 1000 iterations with population sizes of *N* = 30 particles. Averages of the best values in each itera- tion were calculated and plotted in [Figs. 2–6](#_bookmark15).

1. 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]](#_bookmark27). [Tables 3–7](#_bookmark12) present the following informa- tion: Average number of iterations required to reach a func- tion 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 fol- lowing formula.

Ex. *#* of Fn. evaluation = (average number of iterations)

× (number of particles in the swarm)

/(success rate)

[Tables 3 and 4](#_bookmark12) show that while SPOS1 algorithm with 15 par- ticles failed to reach the goal during some runs for solving the Sphere (*F*0) and Rosenbrock (*F*1) functions, the other algo- rithms 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 algo- rithms succeeded to reach the goal in a few numbers of itera- tions than the remaining algorithms.

Also, as illustrated in [Fig. 2](#_bookmark15), 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 ex- pected number of function evaluation needed for the two ver- sions 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

No. of Par. N

Result of test functions *F*3 and *F*6.

Algorithm

version

Griewank function *F*3

Schaffer’s function *F*6

Average number

of iterations

Success

rate (%)

Ex. No. of fn. Average number

evaluation

of iterations

Success Ex. No. of fn.

rate (%) 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 Average number Success Ex. No. of Fn. | | | | |  | Table 5 The result of test function *F*2.  No. of Par. N Algorithm Average number Success Ex. No. of | | | | |
|  | version | of iterations | rate | evaluation |  |  | version | of iterations | rate | Fn. evaluation |
| 15 | PSWV1 | 9 | 1 | 132 |  | 15 | PSWV1 | 8 | 1 | 113 |
|  | PSWV2 | 11 | 1 | 164 |  |  | PSWV2 | 9 | 1 | 137 |
|  | MPSO1 | 34 | 1 | 517 |  |  | MPSO1 | 23 | 1 | 344 |
|  | MPSO2 | 59 | 1 | 888 |  |  | MPSO2 | 38 | 1 | 576 |
|  | CPSO1 | 88 | 1 | 1318 |  |  | CPSO1 | 57 | 1 | 857 |
|  | CPSO2 | 112 | 1 | 1673 |  |  | CPSO2 | 81 | 1 | 1217 |
|  | PSO1 | 531 | 0.50 | 15,930 |  |  | PSO1 | 172 | 0.35 | 7371 |
|  | PSO2 | 1430 | 1 | 21,450 |  |  | PSO2 | 299 | 0.80 | 5606 |
| 30 | PSWV1 | 9 | 1 | 266 |  | 30 | PSWV1 | 7 | 1 | 203 |
|  | PSWV2 | 11 | 1 | 321 |  |  | PSWV2 | 9 | 1 | 260 |
|  | MPSO1 | 32 | 1 | 953 |  |  | MPSO1 | 22 | 1 | 667 |
|  | MPSO2 | 55 | 1 | 1645 |  |  | MPSO2 | 37 | 1 | 1111 |
|  | CPSO1 | 84 | 1 | 2520 |  |  | CPSO1 | 48 | 1 | 1451 |
|  | CPSO2 | 105 | 1 | 3149 |  |  | CPSO2 | 68 | 1 | 2054 |
|  | PSO1 | 614 | 1 | 18,420 |  |  | PSO1 | 140 | 0.90 | 4667 |
|  | PSO2 | 900 | 1 | 27,000 |  |  | PSO2 | 182 | 0.95 | 5747 |
| 6 | PSWV1 | 8 | 1 | 504 |  | 60 | PSWV1 | 7 | 1 | 420 |
|  | PSWV2 | 10 | 1 | 624 |  |  | PSWV2 | 9 | 1 | 513 |
|  | MPSO1 | 30 | 1 | 1789 |  |  | MPSO1 | 21 | 1 | 1271 |
|  | MPSO2 | 53 | 1 | 3145 |  |  | MPSO2 | 36 | 1 | 2180 |
|  | CPSO1 | 77 | 1 | 4623 |  |  | CPSO1 | 60 | 1 | 3603 |
|  | CPSO2 | 102 | 1 | 6138 |  |  | CPSO2 | 69 | 1 | 4152 |
|  | PSO1 | 337 | 1 | 20,220 |  |  | PSO1 | 122 | 0.95 | 7705 |
|  | PSO2 | 611 | 1 | 36,660 |  |  | PSO2 | 166 | 1 | 9960 |
|  |  |  |  |  |  |  |  |  |  |  |

**1.E+06**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
|  |  |  |  |  |

**1.E+04**

**1.E+02**

**Fitness**

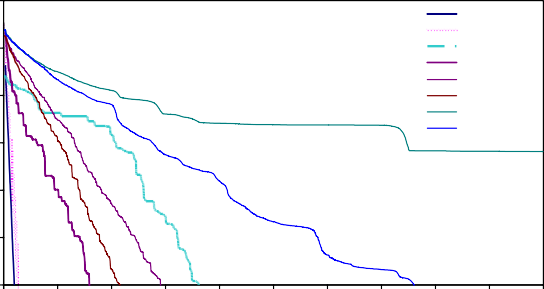
**1.E+00**

**1.E-02**

**1.E-04**

**1.E-06**

***F0***



**PSWV1 PSWV2 MPSO2 MPSO1 CPSO2 CPSO1 PSO1 PSO2**

**0 100 200 300 400 500 600 700 800 900 1000**

**Number of Iterations**

Figure 2 Average best fitness curves for Sphere function (*F*0).

**1.E+09**

**1.E+07**

**1.E+05**

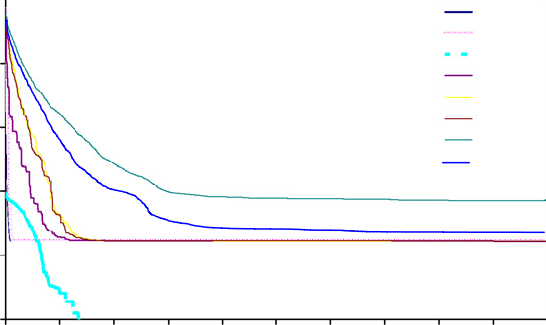
**Fitness**

**1.E+03**

**1.E+01**

**1.E-01**

***F1***



**PSWV1 PSWV2 MPSO2 MPSO1 CPSO2 CPSO1 PSO1 PSO2**

**0 100 200 300 400 500 600 700 800 900 1000**

**Number of Iterations**

Figure 3 Average best fitness curves for Rosenbrock function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
|  |  |  |  |  |

(*F*1).

**1.E+03**

**1.E+02**

**1.E+01**

**1.E+00**

**Fitness**

**1.E-01**

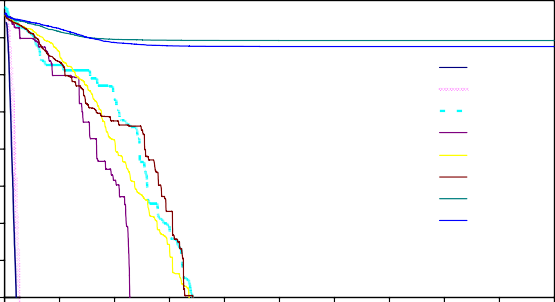
**1.E-02**

**1.E-03**

**1.E-04**

**1.E-05**

***F2***



**PSWV1 PSWV2 MPSO2 MPSO1 CPSO2 CPSO1 PSO1 PSO2**

**0 100 200 300 400 500 600 700 800 900 1000**

**Number of Iterations**

Figure 4 Average best fitness curves of functions *F*0, *F*1, and *F*2.

algorithms tested in this paper in searching for the solution of such kind of problems (see [Table 4](#_bookmark14)).

Also, [Fig. 3](#_bookmark17) illustrates that the PSWV1, PSWV2 reached values below the function goal in different numbers of itera-

tions and below that value reached by the remaining algo- rithms. This means that solution quality of PSWV algorithm is better than that of other algorithms for such kind of problems.

[Table 5](#_bookmark13) 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 diffi- culties in reaching the goal in some runs.

[Table 5](#_bookmark13) and [Fig. 4](#_bookmark18) illustrate that the PSWV1 and PSWV1 algorithms are doing very well on such problem type, deliver- ing 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 solu- tion in approximately less than 100 iterations. While all ver- sions 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 10*e* + 02. This means that the PSWV algorithm is superior to all algorithms as shown in [Fig. 4](#_bookmark18).

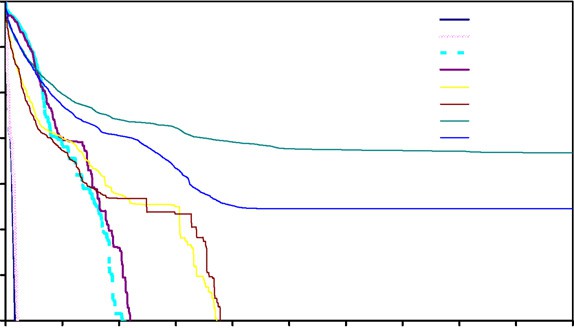
Griewank’s function (*F*3) proves to be hard to solve with the two versions of PSO [[19]](#_bookmark27) algorithm where it had some dif- ficulties in reaching its goal. While PSO had some difficulties in reaching its goal the other algorithms consistently reached the goal in all runs as can be seen in [Table 6](#_bookmark16).

[Fig. 5](#_bookmark19) 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](#_bookmark16) illustrates that while the two versions of PSWV, MPSO, and CPSO algo- rithms reached the goal in all runs, the two versions of stan- dard PSO algorithm have some difficulties in reaching the goal. [Fig. 6](#_bookmark20) 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

***F3***

**1.E+03**



**PSWV1 PSWV2 MPSO2 MPSO1 CPSO2 CPSO1 PSO1 PSO2**

**1.E+02**

**1.E+01**

**1.E+00**

**Fitness**

**1.E-01**

**1.E-02**

**1.E-03**

**1.E-04**

**0 100 200 300 400 500 600 700 800 900 1000**

**Number of Iterations**

Figure 5 Average best fitness curves of functions *F*3 and *F*6.

***F6***

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.

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 repre- sented in boldface (see [Tables 3–7](#_bookmark12)). As a result, the PSO [[19]](#_bookmark27) must be executed several times to ensure good results, whereas one run of PSWV and MPSO [[13]](#_bookmark24) are usually sufficient.

Note that, there is a little difference between the perfor- mance of the algorithms with parameter set1 and with param- eter 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](#_bookmark12)). 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 computa- tionally 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](#_bookmark15).

Looking at the number of function evaluations, the PSWV was in the lead, followed by the MPSO [[13]](#_bookmark24) algorithm and PSO

[[19]](#_bookmark27) algorithm comes in the last as shown in [Tables 3–7](#_bookmark12). Considering the above mentioned point that the two ver-

sions 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 alterna- tive algorithm for solving optimization problems.

9. Conclusion

This paper has proposed a new particle swarm- inspired opti- mization algorithm (PSWV). In this algorithm the new posi- tion of each particle is calculating directly from the combination of its own best position and the global best posi- tion. The implementation of this idea is simple, based on stor- ing the previous positions. The PSWV algorithm outperforms

all the algorithms under study on many benchmark functions,

**1.E+00**

**1.E-01**

**1.E-02**

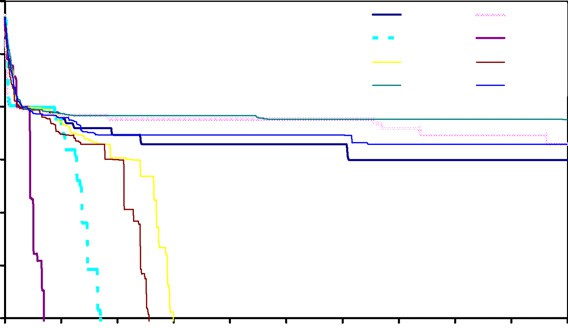
**Fitness**

**1.E-03**

**1.E-04**

**1.E-05**

**1.E-06**



**PSWV1 PSWV2**

**MPSO1 MPSO1**

**CPSO2 CPSO1**

**PSO1 PSO2**

**0 100 200 300 400 500 600 700 800 900 1000**

**Number of Iterations**

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]](#_bookmark24), CPSO [[14]](#_bookmark24), and PSO [[19]](#_bookmark27) 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]](#_bookmark24), CPSO [[14]](#_bookmark24), and PSO [[19]](#_bookmark27) algo- rithms. It is simple, robust, converges fast, and finds the opti- mum in almost every run. In addition, it has few parameters to set, and the same settings can be used for many different

Figure 6 Average best fitness curves for Schaffer’s function *F*6.

problems.

Future work includes further experimentation with param- eters of PSWV, testing the new algorithm on other benchmark problems, and evaluating its performance relative to evolution- ary algorithms.

Acknowledgement

The author is grateful for the valuable comments and sugges- tions 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, Aus- tralia), 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 compu- tation (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 place- ment 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.

1. Liu Bo, Wang Ling, Jin Yi-Hui, Tang Fang, Huang De-Xian. Directing orbits of chaotic systems by particle swarm optimiza- tion. Chaos, Solitons Fractals 2006;29:454–61.
2. 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.
3. 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.
4. 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.
5. El-Sherbiny MM. A modified algorithm for particle swarm optimization with constriction coefficient. Int J Comp Inf (IJCI) 2009;2:17–30.
6. 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 cyber- netics, Kunming, 12–15 July 2008; 2008. p. 4090–5.
7. 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.
8. Liu Jianhua, Fan Xiaoping, Qu Zhihua. An improved particle swarm optimization with mutation based on similarity. Third Int Conf Nat Comput 2007:824–8.
9. Jiang Yi, Yue Qingling. An improved particle swarm optimization with new select mechanism. Int Workshop Knowledge Discov Data Mining 2008:383–6.
10. Trelea Ioan Cristian. The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 2003;85:317–25.
11. Gazi Veysel, Passino Kevin M. Stability analysis of social foraging swarms. IEEE Trans Syst Man Cybern B Cybern 2004;34(1): 539–57.
12. Boeringer Daniel W, Werner Douglas H. Particle swarm optimi- zation versus genetic algorithms for phased array synthesis. IEEE Trans Antenna Propagation 2004;52(3):771–9.
13. Clerc M. The Swarm And The Queen: Towards A Deterministic And Adaptive Particle Swarm Optimization. Proc ICEC, Wash- ington, DC 1999:1951–7.
14. Shi Y, Eberhart R. A combined particle swarm optimizer. Proc IEEE World Congress Comput Intell 1998:69–73.