International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1(A)**, January **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **201–218**

MODIFIED PARTICLE SWARM OPTIMIZATION WITH TIME VARYING VELOCITY VECTOR

Radha Thangaraj[1], Millie Pant[2], Ajith Abraham[3,4] and Vaclav Snasel[3]

[1]Faculty of Science, Technology and Communication
University of Luxembourg
No. 6, rue Coudenhove-Kalergi, L-1359 Luxembourg-Kirchberg
t.radha@ieee.org

[2]Department of Paper Technology
Indian Institute of Technology Roorkee
Saharanpur 247 001, India
millifpt@iitr.ernet.in

[3]Faculty of Electrical Engineering and Computer Science
VSB - Technical University of Ostrava
17, listopadu 15/2172, Ostrava - Poruba, Czech Republic
ajith.abraham@ieee.org; vaclav.snasel@vsb.cz

[4]Machine Intelligent Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence (SNIRE)
P. O. Box 2259 Auburn, Washington 98071, USA

Received June 2010; revised October 2010

Abstract. *Particle Swarm Optimization (PSO) is a population-based computational intelligence paradigm; it originated as a simulation of simplified social model of birds in a flock. The PSO algorithm is easy to implement and has been proven to be very competitive for solving diverse global optimization problems including both test and application problems in comparison to conventional methods and other meta-heuristics. In the present study, a new velocity vector is introduced in the BPSO algorithms and is analyzed on thirty six benchmark problems and three real life problems taken from the literature. The numerical results show that the incorporation of the proposed velocity vector helps in improving the performance of BPSO in terms of final objective function value, number of function evaluations and convergence rate.*
Keywords: Particle swarm optimization, Velocity update equation, Premature convergence

1. **Introduction.** Evolutionary Algorithms (EAs) [1] are a broad class of stochastic optimization algorithms inspired by biology and, in particular, by those biological processes that allow populations of organisms to adapt to their surrounding environments, genetic inheritance and survival of the fittest. EAs have certain prominent advantages over other types of numerical methods and the two are the most important [2] beings:

- Their application to problems that consist of discontinuous, non-differentiable and non-convex objective functions and/or constraints.
- Their efficiency in escaping the local optima.

EAs have been applied to a wide range of benchmark functions (single and multi-objective) and real life problems [3-9]. Some common EAs are Genetic Algorithms (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO) and Differential Evolution (DE), etc. In the present research paper, we have concentrated our work to PSO.

Particle Swarm Optimization (PSO) is relatively a newer addition to a class of population based search technique for solving numerical optimization problems. Metaphorically, PSO imitates the collective and cooperative behavior of species moving in groups. Some classic examples are a swarm of birds, school of fish, cooperative behavior of ants and bees, etc. In classical (or original PSO), developed by Kennedy and Eberhart [10], each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors (or colleagues).

For a D-dimensional search space, the position of the ith particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle maintains a memory of its previous best position $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The best one among all the particles in the population is represented as $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$. The velocity of each particle is represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. In each iteration, the P vector of the particle with best fitness in the local neighborhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined by using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector given by:

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

The first part of Equation (1) represents the inertia of the previous velocity; the second part is the cognition part and it tells us about the personal experience of the particle; the third part represents the cooperation among particles and is named as the social component [11]. Acceleration constants c_1, c_2 and inertia weight w [12] are the predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range of $[0, 1]$. The working of PSO in space is given in Figure 1.

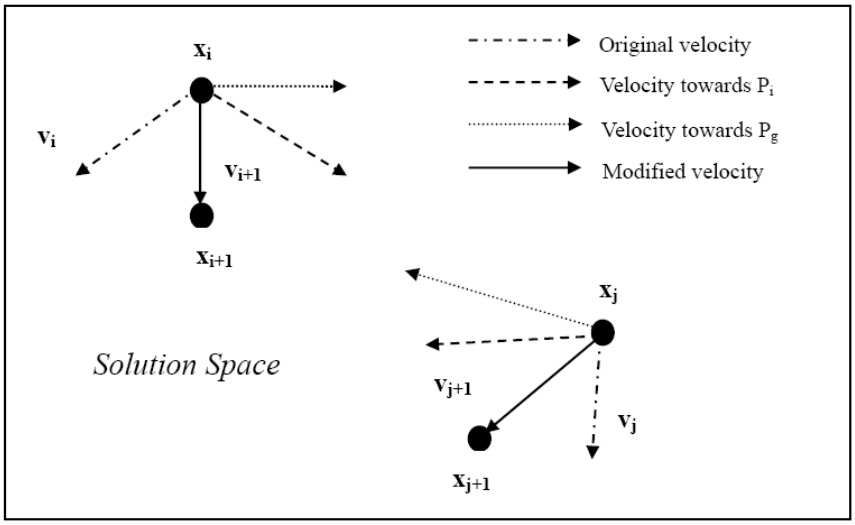


FIGURE 1. Searching mechanism of PSO

Many recent versions of PSO propose modifications in the basic control parameters, like acceleration coefficients, inertia weight, velocity clamping and swarm size [13-17]. From these empirical studies, it can be concluded that PSO is sensitive to control parameters, but few studies are involved in the basic mechanism. In the Basic PSO (BPSO), velocity is an important parameter and is dynamically adjusted according to the historical behavior of the particle and its companions. The present study proposes a Modified Particle Swarm Optimization algorithm (MPSO) with new velocity vector, based on the maximum

distance between any two points in the solution space, distance between the global best particle and the personal best particle, objective function value of global best particle, objective function value of current particle, the current iteration number and the maximum number of iterations. Numerical results show MPSO is quite a permissible algorithm for solving global optimization problems.

The rest of the paper is organized as follows: in Section 2, we have given the literature review; Section 3 describes the MPSO algorithm. The experimental setup, parameter settings and benchmark problems are reported in Section 4. The experimental results of benchmark problems are analyzed in Section 5 and in Section 6, the real life problems and their results are given. Finally, this paper concludes with Section 7.

2. **Previous Works.** Based on the velocity update Equation (1), each particle's new position is influenced by its personal best position and the best position in its neighborhood. Kennedy and Mendes [18] introduced a new velocity equation in which each particle is influenced by the success of all its neighbors, and not on the performance of only one individual. Thompson et al. [19] implemented an alternative approach where different velocity update equations are used for cluster centers and particles within clusters. In their method, each particle is adjusted on the basis of the distance from its own personal best position, the corresponding cluster's best position, and the current position of the cluster center. Blackwell and Bentley [20] developed the charged PSO based on an analogy of electrostatic energy with charged particles. The charged PSO changes the velocity equation by adding particle acceleration to standard velocity update Equation (1). The Fitness-Distance-Ratio PSO (FDR PSO) is introduced by Peram et al. [21], in which a new term is added to the velocity update equation; each particle learns from the experience of the neighboring particles that have a better fitness than itself. Wei et al. [22] introduced a disturbance in velocity or position to prevent the premature phenomenon in basic PSO algorithm. Krohling [23] proposed a velocity update vector with the use of absolute value of the Gaussian probability distribution.

Yang and Simon [24] proposed NPSO algorithm, in which each particle adjusts its position according to its own previous worst solution and its group's previous worst solution to find the optimum value. That is the velocity update equation of NPSO depends upon the particle's personal worst position and the global worst position whereas in classical PSO the velocity update equation depends on the particle's personal best position and the global best position. Another version is PSO-E, a new PSO version proposed by Krohling and Coelho [25] in which the exponential distribution is used for generating the weighting coefficients of velocity update equation of basic PSO. θ-PSO algorithm is a recently proposed PSO algorithm by Zhong et al. [26], based on the phase angle vector but not on the velocity vector. In θ-PSO, an increment of phase angle vector $\Delta\theta$ replaces velocity vector v and the positions are adjusted by the mapping of phase angles.

3. **Modified Particle Swarm Optimization (MPSO).** The proposed MPSO algorithm is a simple and modified version of Basic Particle Swarm Optimization Algorithm (BPSO). It introduces a new velocity vector, based on maximum distance between any two points in the solution space, distance between the global best particle and the personal best particle, objective function value of global best particle, objective function value of current particle, the current iteration number and the maximum number of iterations.

In the proposed algorithm, we have fixed a probability P_v having a certain threshold value provided by the user. In every iteration, if the uniformly distributed random number $U(0,1)$ is less than P_v then the velocity vector is generated by using Equation (3)

otherwise the velocity vector follows the standard PSO algorithm, i.e., the velocity vector is generated by using Equation (1).

The proposed velocity vector is defined as:

$$v_{id} = \alpha * \alpha_1 * \alpha_2 * \alpha_3 * (p_{gd} - p_{id}) \tag{3}$$

where α is an adjustable coefficient. $\alpha_1 = (MAXITE - ITE)/MAXITE$, $MAXITE$ represents the maximum number of iterations and ITE represents the current iteration number.

$$\alpha_2 = (d_{\max} - d_{gi})/d_{\max},$$

$d_{\max}$ represents the maximum distance between two points in the solution space and d_{gi} represents the distance between the global best particle and the ith particle.

The maximum distance $d_{\max}$ between two points in the solution space (a, b) is computed as:

$$d_{\max} = \sqrt{\sum_{i=1}^{D} (b_i - a_i)^2}$$

where $a = (a_1, a_2, \ldots, a_D)$, $b = (b_1, b_2, \ldots, b_D)$.

The distance between two particles x_p and x_q can be calculated as follows:

$$d_{pq} = \sqrt{\sum_{i=1}^{D} (x_{pi} - x_{qi})^2},$$

D represents the dimension of swarm particle.

$$\alpha_3 = f(P_g)/f(X_i),$$

where $f(P_g)$ is a fitness function value of the global best particle P_g, $f(X_i)$ is a fitness function value of the ith particle X_i.

The pseudo code of the proposed MPSO algorithm is given in Figure 2.

Initialize the population
Do
  Linearly decrease w *from* 0.9 *to* 0.4 *and set* $c_1 = c_2 = 2.0$
  For $i = 1$ *to population size* $= M$
    For $d = 1$ *to dimension* D
      Set P_v *and Generate* $U(0, 1)$
        If $(U(0, 1) < P_v)$ *then*
          $v_{id} = \alpha * \alpha_1 * \alpha_2 * \alpha_3 * (p_{gd} - p_{id})$
        Else
          $v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})$
        End if
        $x_{id} = x_{id} + v_{id}$
    End for
    If $(f(X_i) < f(P_i))$ $P_i = X_i$
      If $(f(P_i) < f(P_g))$ $P_g = P_i$
      End if
  End if
  End for
Until stopping criteria is reached

FIGURE 2. Pseudo code of MPSO algorithm

The four parameters α, α_1, α_2, α_3 help in controlling the velocity of the swarm particles. Unlike the usual velocity equation of the basic PSO (given by Equation (1)) the proposed velocity vector do not make use of inertia weight and acceleration constants and is more or less adaptive in nature. From the velocity Equation (3), we can easily see that in the beginning the velocity is large therefore the particles move rapidly but during the subsequent generations the velocity decreases and the particles slow down as they reach towards the optimum solution. The presence of the parameter P_v, which helps in stochastic application of the basic velocity vector and the proposed velocity vector, helps in preventing the algorithm in becoming greedy in nature, thereby helping in preventing the premature convergence of the swarm.

4. **Experimental Settings and Benchmark Problems.** In order to make a fair comparison of BPSO and MPSO algorithms, we fixed the same seed for random number generation so that the initial population is same for both the algorithms. The population size is taken as 50 for all the test problems. A linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters $c_1 = 2.0$ and $c_2 = 2.0$. For each algorithm, the stopping criteria is to terminate the search process when one of the following conditions is satisfied: (1) the maximum number of generations is reached (assumed 1000 generations), (2) $|f_{\max} - f_{\min}| < 10^{-4}$, where f is the value of objective function. A total of 30 runs for each experimental setting were conducted. If the run satisfies the second stopping condition then that run is called successful run.

We varied the additional probability parameter P_v for various values and observed that the best results are obtained for 0.6. The adjustable coefficient α is kept at 0.5 for MPSO. To check the efficiency of the proposed MPSO algorithm we tested it on thirty six benchmark problems having box constraints. The mathematical models of the problems are given in Appendix A.

5. **Results Discussion of Benchmark Problems.** The results of the benchmark problems $f_1 - f_{36}$ are shown in Table 1 in terms of mean best fitness, standard deviation and SR (success rate). Table 2 gives the results of all benchmark problems NFE (number of function evaluations) and time. Comparison results of MPSO algorithm with BPSO and θ-PSO algorithms are given in Table 4 and Table 5 respectively. In order to make a fair comparison of MPSO and θ-PSO, we fixed the same error goal as stated in [26] as: for f_2, f_3 and f_4 are 0.01, 0.1 and 100 respectively.

The MPSO algorithm is compared with the classical PSO in terms of average fitness function value, number of function evaluations (NFE), success rate in % (SR) and run time. As expected the proposed MPSO algorithm performed much better than the classical PSO algorithm. From Table 1, we can see that when MPSO is used to solve the given benchmark problems the improvement in terms of average fitness function value is more than 99% in comparison to the PSO for about 9 out of 36 test cases. Also, MPSO gave more than 75%, 50% and 30% improvement in 3 test cases for each in comparison with PSO in terms of fitness value. For all the remaining test cases, both the algorithms gave the same performance in terms of fitness value. In comparison of PSO and MPSO in terms of success rate, MPSO gave better performance than PSO in most of the test cases. Some of the test cases (f_2, f_3, f_5, f_{11}, f_{12}, f_{16}, f_{17}, f_{19}, f_{25} and f_{35}) PSO gave 0% SR whereas MPSO gave more than 30% SR (including 100% SR). In terms of number of function evaluation also MPSO gave much better performance than PSO; Figure 3 shows the comparison of BPSO and MPSO based on number of function evaluations. However, if we compare the convergence time of PSO and MPSO, then MPSO has taken more time for convergence than PSO in 22 test cases, that is because of the inclusion of added

velocity part in the algorithm. However, MPSO has taken less time than PSO in 14 out of 36 test cases. Figure 4 shows the comparison of BPSO and MPSO based on convergence time.

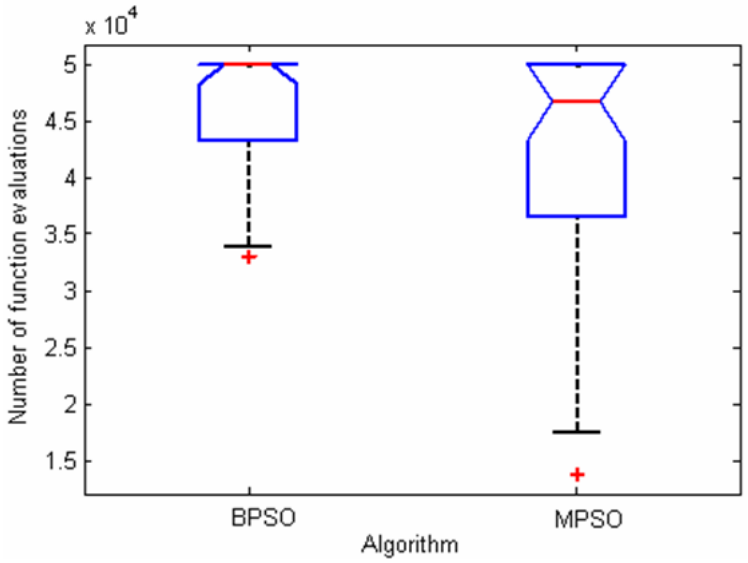


FIGURE 3. Comparison of BPSO and MPSO based on number of function evaluations of all test problems

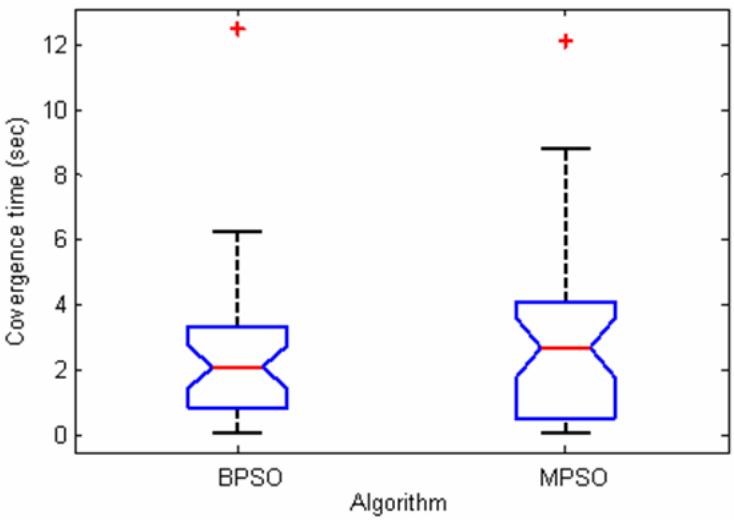
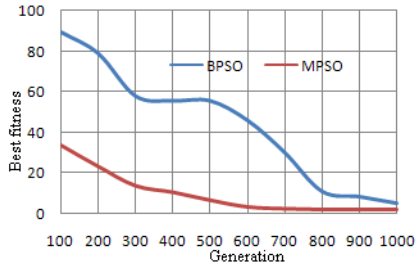
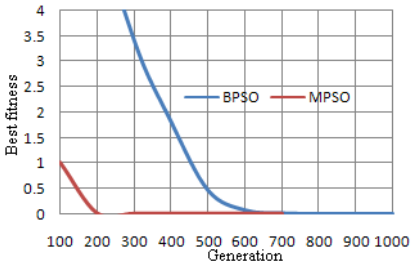


FIGURE 4. Comparison of BPSO and MPSO based on convergence time of all test problems

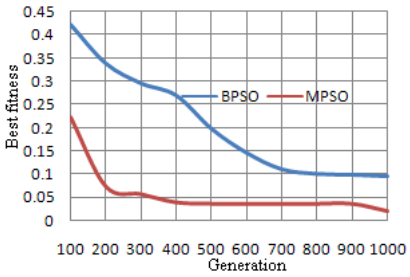
Thus, from the numerical results, it is concluded that the incorporation of the proposed velocity vector helps in improving the performance of basic PSO in terms of final objective function value, NFE and convergence rate. Also, the performance of proposed MPSO algorithm is compared with θ-PSO, a variant of BPSO. From the numerical results of Table 4 and Table 5, it is clear that the performance of proposed MPSO is better than the θ-PSO algorithm also. Performance curves of selected benchmark problems are given in Figures 5(a)-5(r). From these figures also we can see the superior performance of MPSO in comparison with basic PSO.



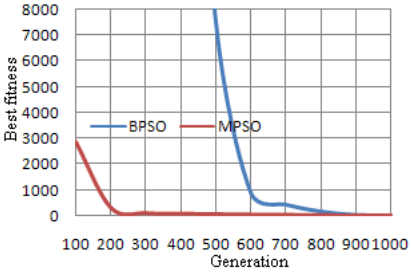
(a) Function f_1



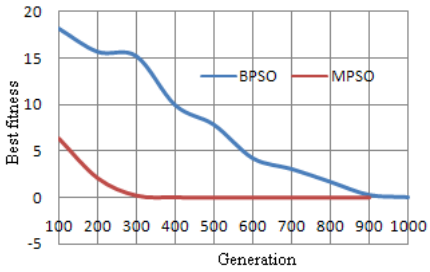
(b) Function f_2



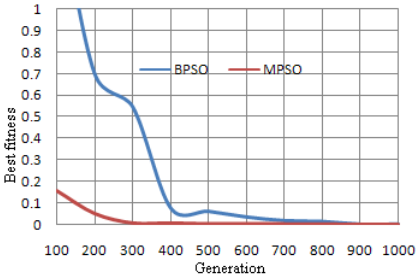
(c) Function f_3



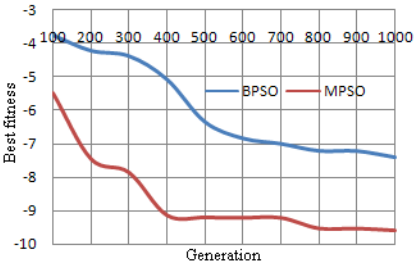
(d) Function f_4



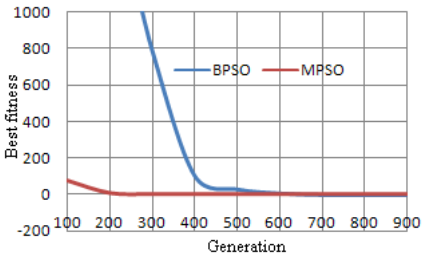
(e) Function f_5



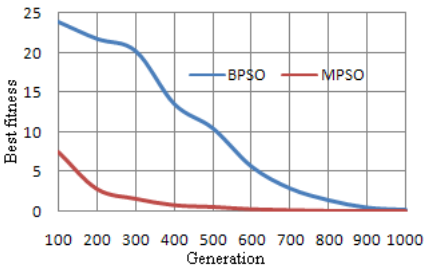
(f) Function f_6



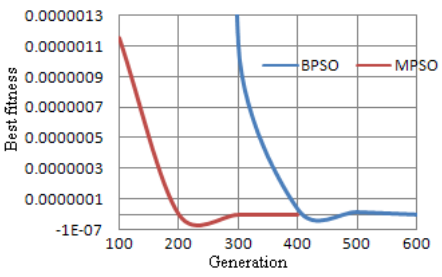
(g) Function f_9



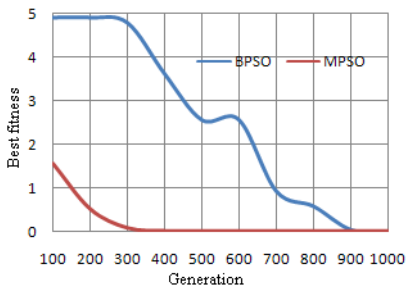
(h) Function f_{10}



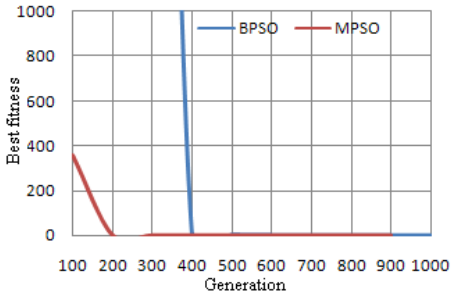
(i) Function f_{13}



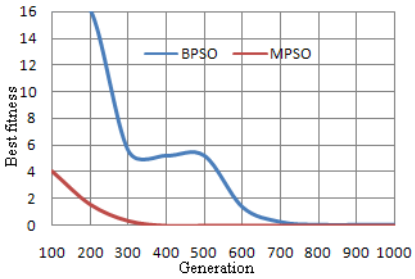
(j) Function f_{14}



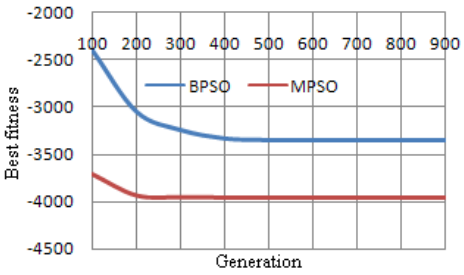
(k) Function f_{15}



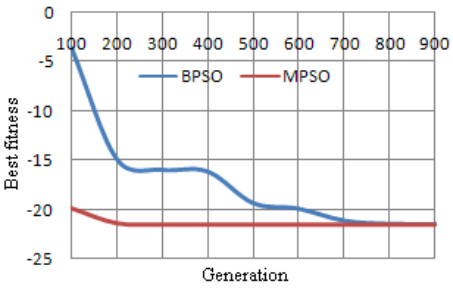
(l) Function f_{16}



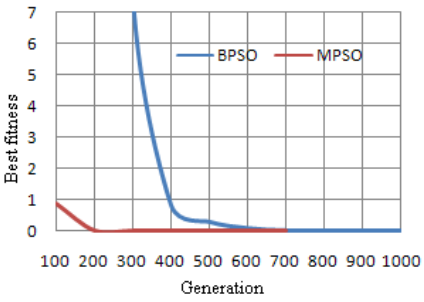
(m) Function f_{17}



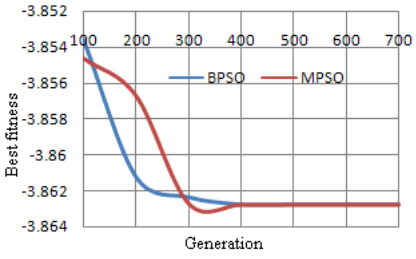
(n) Function f_{18}



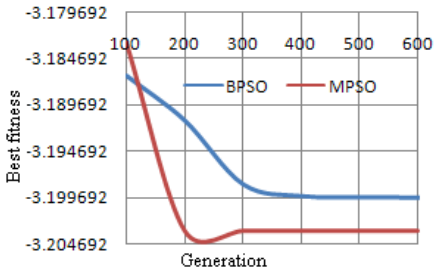
(o) Function f_{19}



(p) Function f_{20}



(q) Function f_{21}



(r) Function f_{22}

FIGURE 5. Performance curves of BPSO and MPSO for selected benchmark problems

6. **Real Life Problems and Results.** The credibility of an optimization algorithm also depends on its ability to solve real life problems. In this paper, we took three real life engineering design problems to validate the efficiency of the proposed MPSO algorithm. Mathematical models of the real life problems are given below.

(i) Gas transmission compressor design [27]:
Minimize

TABLE 1. Result comparison of PSO and MPSO (mean fitness/standard deviation/SR (%)

F	Dim	PSO			MPSO		
		Fitness	Std	SR	Fitness	Std	SR
f_1	10	8.44052	4.03724	–	**3.52897**	1.34857	–
f_2	10	2.49e-08	2.86e-08	–	**1.27e-11**	2.11e-11	100
f_3	10	0.09489	0.03934	–	**0.03903**	0.01274	30
f_4	10	23.7877	34.4229	–	**5.09976**	6.65036	–
f_5	10	9.9569	9.95228	–	**2.19e-08**	3.42e-08	50
f_6	10	0.00480	0.00191	–	**0.00296**	0.00167	–
f_7	2	–1.8013	9.93e-03	100	–1.8013	1.57e-06	100
f_8	5	–4.32364	0.44702	–	**–4.66848**	0.04535	–
f_9	10	–6.62579	0.67245	–	**–9.3255**	0.22641	–
f_{10}	10	0.00000	0.00000	60	0.00000	0.00000	70
f_{11}	10	0.83576	3.76452	–	**1.04e-12**	2.69e-12	40
f_{12}	10	0.00015	0.00018	–	**9.98e-10**	1.26e-09	60
f_{13}	10	0.09027	0.03896	–	**0.01979**	0.02145	–
f_{14}	10	1.53e-11	4.01e-11	100	**1.05e-13**	2.12e-13	100
f_{15}	10	0.00667	0.01921	–	**0.00021**	0.00065	–
f_{16}	10	–1.1504	4.23e-05	–	**–1.15044**	2.46e-11	60
f_{17}	10	1.44e-06	1.90e-06	–	**1.72e-12**	1.53e-13	90
f_{18}	10	–3201.6	369.23	20	**–3751.61**	130.28	70
f_{19}	10	–19.4018	4.2009	–	**–21.5023**	1.10e-12	30
f_{20}	2	3.87e-11	7.29e-11	30	**9.53e-16**	2.83e-15	100
f_{21}	3	–3.86278	3.97e-16	100	–3.86278	3.71e-16	100
f_{22}	6	–3.18244	0.14725	90	**–3.25608**	0.06475	100
f_{23}	2	0.00000	0.00000	100	0.00000	0.00000	100
f_{24}	2	1.81e-15	5.23e-15	100	**0.00000**	0.00000	100
f_{25}	2	–1.03163	2.22e-22	–	–1.03163	2.22e-22	80
f_{26}	10	5.58e-13	1.13e-12	80	**1.53e-15**	4.01e-15	100
f_{27}	4	0.05054	0.05190	–	**0.03758**	0.03171	–
f_{28}	2	3.00000	1.60e-15	100	3.00000	2.90e-16	100
f_{29}	2	–1.91322	0.00000	100	**–1.91322**	0.00000	100
f_{30}	2	–186.731	4.21e-14	–	**–186.731**	2.00e-14	–
f_{31}	10	–98.4351	10.4653	–	**–117.776**	1.60425	–
f_{32}	2	1.00000	1.85e-16	80	1.00000	9.93e-17	100
f_{33}	2	0.397886	0.00000	50	0.397886	0.00000	100
f_{34}	10	0.67169	0.20006	–	**0.29862**	0.08075	–
f_{35}	10	–78.3323	1.92e-06	–	–78.3323	3.55e-13	90
f_{36}	2	–3.58972	0.388473	30	**–3.78396**	0.00000	50

$$f(x) = 8.61 \times 10^5 \times x_1^{1/2} x_2 x_3^{-2/3} (x_2^2 - 1)^{-1/2} + 3.69 \times 10^4 \times x_3 + 7.72 \times 10^8 \times x_1^{-1} x_2^{0.219} - 765.43 \times 10^6 \times x_1^{-1}$$

Subject to: $10 \le x_1 \le 55$, $1.1 \le x_2 \le 2$, $10 \le x_3 \le 40$.

(ii) Optimal thermohydralic performance of an artificially roughened air heater [28]:
Maximize $L = 2.51 * \ln e^+ + 5.5 - 0.1R_M - G_H$
where $R_M = 0.95x_2^{0.53}$, $GH = 4.5(e^+)^{0.28}(0.7)^{0.57}$, $e^+ = x_1 x_3 (\bar{f}/2)^{1/2}$, $\bar{f} = (f_s + f_r)/2$, $f_s = 0.079x_3^{-0.25}$, $f_r = 2(0.95x_3^{0.53} + 2.5 * \ln(1/2x_1)^2 - 3.75)^{-2}$.
Subject to: $0.02 \le x_1 \le 0.8$, $10 \le x_2 \le 40$, $3000 \le x_3 \le 20000$.

TABLE 2. Result comparison of PSO and MPSO (diversity/NFE/time (sec))

F	Dim	PSO			MPSO		
		Diversity	NFE	Time	Diversity	NFE	Time
f_1	10	3.05529	50050+	2.7	1.02237	50050+	3.2
f_2	10	0.0805151	50050+	2.7	0.0405957	43570	2.9
f_3	10	0.713923	50050+	2.4	2.6207	47540	3.1
f_4	10	4.58719	50050+	6.3	0.678667	50050+	8.8
f_5	10	4.1804	50050+	1.7	0.00014	46800	3.7
f_6	10	0.647542	50050+	2.1	0.39727	50050+	4.6
f_7	2	0.00365	38495	0.7	0.0433284	18375	**0.5**
f_8	5	0.0524459	50050+	2.6	0.0291604	50050+	2.8
f_9	10	2.14912	50050+	5.8	1.35232	50050+	6.4
f_{10}	10	0.793882	49215	0.5	0.14526	46860	**0.3**
f_{11}	10	0.59298	50050+	2.1	0.02419	49715	2.6
f_{12}	10	0.06367	50050+	0.1	2.07e-05	48615	0.24
f_{13}	10	3.21891	50050+	0.1	0.38093	50050+	0.25
f_{14}	10	0.16440	34135	0.6	0.32737	22315	**0.21**
f_{15}	10	3.73003	50050+	0.2	0.91417	50050+	0.25
f_{16}	10	1.80214	50050+	5.4	0.38154	49550	7.4
f_{17}	10	0.443238	50050+	5	0.00436	45745	6.1
f_{18}	10	0.45694	49630	2.2	0.00704	48510	2.6
f_{19}	10	0.10776	50050+	4.8	0.46810	48680	6.9
f_{20}	2	0.14613	49970	2	0.13494	40755	4.1
f_{21}	3	0.01927	37945	2.2	0.01299	20465	**1.4**
f_{22}	6	0.01558	43920	5.1	0.00763	37295	**4.3**
f_{23}	2	0.00693	37020	1.0	0.00053	22525	**0.7**
f_{24}	2	0.18571	33945	0.2	0.16375	17655	**0.1**
f_{25}	2	0.27273	50050+	1.1	0.0069	38575	**0.9**
f_{26}	10	0.16440	45700	5	0.00153	34135	4.1
f_{27}	4	0.27199	50050+	2.2	0.16179	50050+	3.0
f_{28}	2	0.00021	40895	1.0	0.00184	19885	**0.6**
f_{29}	2	0.11743	32970	0.4	0.01701	13855	**0.2**
f_{30}	2	1.57827	50050+	0.2	0.91083	50050+	0.4
f_{31}	10	2.9924	50050+	2.0	3.38776	50050+	2.2
f_{32}	2	0.07937	39390	12.5	0.38103	37840	**12.1**
f_{33}	2	0.01523	42785	1.0	0.01535	41820	**0.6**
f_{34}	10	7.09519	50050+	2.8	1.60477	50050+	3.4
f_{35}	10	0.05004	50050+	3.9	0.00107	45840	**3.6**
f_{36}	2	1.08611	47120	1.0	0.15180	35930	**0.8**

TABLE 3. Comparison of proposed MPSO with BPSO in terms of improvement (%) and t-test values in fitness function values

Function	Improvement (%)	t-value	Function	Improvement (%)	t-value
f_1	58.190135	6.320111	f_{19}	10.82632	2.738678
f_2	99.948996	4.7662	f_{20}	99.99754	2.907591
f_3	58.868163	7.398961	f_{21}	0.00000	0.00000
f_4	78.561357	2.919559	f_{22}	2.313948	2.507455
f_5	100	5.479768	f_{23}	0.00000	–
f_6	38.333333	3.972251	f_{24}	100	1.89556
f_7	0.00000	0.00000	f_{25}	0.00000	0.00000
f_8	7.9756872	4.203663	f_{26}	99.72581	2.69725
f_9	40.745481	20.84008	f_{27}	25.64306	1.16712
f_{10}	0.00000	–	f_{28}	0.00000	0.00000
f_{11}	100	1.215997	f_{29}	0.00000	–
f_{12}	99.999335	4.564324	f_{30}	0.00000	0.00000
f_{13}	78.07688	8.679908	f_{31}	19.64838	10.00557
f_{14}	99.313725	2.075443	f_{32}	0.00000	0.00000
f_{15}	96.851574	1.840845	f_{33}	0.00000	–
f_{16}	0.0034771	5.17941	f_{34}	55.54199	9.47145
f_{17}	99.999881	4.151155	f_{35}	0.00000	0.00000
f_{18}	17.179223	7.694049	f_{36}	5.411007	2.738662

TABLE 4. Comparison of MPSO with BPSO and θ-PSO ($\boldsymbol{w = 0.6}$, $\boldsymbol{c_1 = c_2 = 1.7}$)

F	Algorithm	Swarm size	Dimension	Number of iterations to achieve the goal		
				$\boldsymbol{w = 0.6}$, $\boldsymbol{c_1 = c_2 = 1.7}$		
				Minimum	Average	SR
f_2	BPSO	20	30	722	778	100
	θ-PSO	20	30	523	598	100
	MPSO	20	30	**236**	**324**	100
	BPSO	40	30	783	847	100
	θ-PSO	40	30	352	406	100
	MPSO	40	30	**270**	**321**	100
f_3	BPSO	20	30	368	455	100
	θ-PSO	20	30	343	512	100
	MPSO	20	30	**211**	**390**	100
	BPSO	40	30	684	836	100
	θ-PSO	40	30	231	334	100
	MPSO	40	30	**219**	**293**	100
f_4	BPSO	20	30	426	533	100
	θ-PSO	20	30	223	376	100
	MPSO	20	30	**208**	**267**	100
	BPSO	40	30	544	597	100
	θ-PSO	40	30	194	283	100
	MPSO	40	30	**184**	**214**	100

(iii) Optimal capacity of gas production facilities [27]:
Minimize

$$f(x) = 61.8 + 5.72x_1 + 0.2623\left[(40 - x_1)\ln\left(\frac{x_2}{200}\right)\right]^{-0.85} + 0.087(40 - x_1)\ln\left(\frac{x_2}{200}\right) + 700.23x_2^{-0.75} + 0.087(40 - x_1)\ln\left(\frac{x_2}{200}\right) + 700.23x_2^{-0.75}$$

TABLE 5. Comparison of MPSO with BPSO and θ-PSO ($\boldsymbol{w = 0.729}$, $\boldsymbol{c_1 = c_2 = 1.494}$)

F	Algorithm	Swarm size	Dimension	Number of iterations to achieve the goal $\boldsymbol{w = 0.729}$, $\boldsymbol{c_1 = c_2 = 1.494}$		
				Minimum	Average	SR
f_2	BPSO	20	30	598	682	100
	θ-PSO	20	30	362	734	100
	MPSO	20	30	**207**	**239**	100
	BPSO	40	30	620	707	100
	$\theta-$PSO	40	30	266	683	100
	MPSO	40	30	**203**	**233**	100
f_3	BPSO	20	30	420	686	100
	θ-PSO	20	30	385	564	95
	MPSO	20	30	**178**	**198**	100
	BPSO	40	30	883	965	100
	θ-PSO	40	30	263	356	100
	MPSO	40	30	**192**	**294**	100
f_4	BPSO	20	30	363	443	100
	θ-PSO	20	30	328	402	100
	MPSO	20	30	**140**	**185**	100
	BPSO	40	30	459	537	100
	θ-PSO	40	30	272	325	100
	MPSO	40	30	**165**	**198**	100

TABLE 6. Comparison of proposed MPSO with PSO for real life problems

Gas Transmission Compressor Design			
Item	PSO	MPSO	Results in [27]
x_1	55	53.4471	55
x_2	1.19541	1.1901	1.195
x_3	24.7749	24.7185	25.026
f(x)	296.446e+04	**296.436e+004**	296.455e+04
NFE	23631	**18270**	NA
Optimal Thermohydralic Performance of an Artificially Roughened Air Heater			
Item	PSO	MPSO	Results in [28]
x_1	0.05809	0.04227	0.052
x_2	10	10	10
x_3	10400.2	13289.4	10258
f(x)	4.21422	4.21422	4.182
NFE	6207	**3030**	NA
Optimal Capacity of Gas Production Facilities			
Item	PSO	MPSO	Results in [27]
x_1	17.5	17.5	17.5
x_2	600	600	465
f(x)	169.844	169.844	2 173.76
NFE	342	**297**	NA

Subject to: $x_1 \geq 17.5$, $x_2 \geq 200$; $17.5 \leq x_1 \leq 40$, $300 \leq x_2 \leq 600$.

Numerical results for the real life problems are listed in Table 6. As evident from the empirical results, the average number of function evaluations (NFE) required to reach

the optimum solution, the proposed MPSO algorithm gave the best results. However, in terms of function value all the algorithms gave more or less similar results. This is probably because of the fact that all the real life problems considered here are simple in nature and do not have more than three variables.

7. **Conclusion.** In the present research paper, a Modified Particle Swarm Optimization algorithm called MPSO is presented. In MPSO algorithm, a new time varying velocity update equation is introduced which is applied stochastically along with basic PSO velocity equation. A test suit of 36 benchmark problems and three real life problems were used to analyze the characteristics of MPSO; the test suit consists of scalable and nonscalable functions. The results of MPSO are compared with basic PSO and recent variant PSO, θ-PSO. The numerical results showed that MPSO outperforms the BPSO and θ-PSO with a noticeable percentage. Thus, it can be concluded that the incorporation of the proposed velocity vector helps in improving the performance of BPSO.

Acknowledgment. The authors would like to extend their thanks to the unknown referees whose comments greatly helped in improving the quality of the paper.

REFERENCES

[1] T. Bäck, D. B. Fogel and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*, Inst. Phys. and Oxford University Press, New York, 1997.

[2] J. Zhang, J. Xu and Q. Zhou, A new differential evolution for constrained optimization problems, *Proc. of the 6th Int. Conf. on Intelligent Systems, Design and Applications*, vol.2, pp.1018-1023, 2006.

[3] C. Liu and Y. Wang, A new dynamic multi-objective optimization evolutionary algorithm, *International Journal of Innovative Computing, Information and Control*, vol.4, no.8, pp.2087-2096, 2008.

[4] X. Song, Y. Zhu, C. Yin and F. Li, Study on the combination of genetic algorithms and ant colony algorithms for solving fuzzy job shop scheduling problems, *Proc. of IMACS Multiconference on Computational Engineering in Systems Applications*, vol.2, pp.1904-1909, 2006.

[5] A. Abbasy and S. H. Hosseini, A novel multi-agent evolutionary programming algorithm for economic dispatch problems with non-smooth cost functions, *Proc. of IEEE Power Engineering Society General Meeting*, pp.1-7, 2007.

[6] T. Uno, H. Katagiri and K. Kato, An evolutionary multi-agent based search method for stackelberg solutions of bilevel facility location problems, *International Journal of Innovative Computing, Information and Control*, vol.4, no.5, pp.1033-1042, 2008.

[7] C. Guangyi, G. Wei and H. Kaisheng, On line parameter identification of an induction motor using improved particle swarm optimization, *Proc. of Chinese Control Conference*, pp.745-749, 2007.

[8] E. Cao and M. Lai, An improved differential evolution algorithm for the vehicle routing problem with simultaneous delivery and pick-up service, *Proc. of the 3rd International Conference on Natural Computation*, vol.3, pp.436-440, 2007.

[9] S. S. Kim, I.-H. Kim, V. Mani and H. J. Kim, Ant colony optimization for reporting cell planning in mobile computing using selective paging strategy, *International Journal of Innovative Computing, Information and Control*, vol.5, no.6, pp.1587-1598, 2009.

[10] J. Kennedy and R. C. Eberhart, Particle swarm optimization, *Proc. of IEEE Int. Conf. on Neural Networks*, Perth, Australia, vol.4, pp.1942-1948, 1995.

[11] J. Kennedy, The particle swarm: Social adaptation of knowledge, *Proc. of IEEE Int. Conf. on Evolutionary Computation*, pp.303-308, 1997.

[12] Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, *Proc. of IEEE Int. Conf. on Evolutionary Computation*, pp.69-73, 1998.

[13] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, 2001.

[14] L. P. Zhang, H. J. Yu and S. X. Hu, Optimal choice of parameters for particle swarm optimization, *Journal of Zhejiang University Science A*, vol.6, no.6, pp.528-534, 2005.

[15] T. Y. Lee and C. L. Chen, Iteration particle swarm optimization for contract capacities selection of time-of-use rates industrial customers, *Energy Conv. Management*, vol.48, no.4, pp.1120-1131, 2007.

[16] S. S. Fan and E. Zahara, A hybrid simplex search and particle swarm optimization for unconstrained optimization, *Eur. J. Operations Research*, vol.181, no.2, pp.527-548, 2007.
[17] S. L. Ho, S. Y. Yang, G. Z. Ni and K. F. Wong, An improved PSO method with application to multimodal functions of inverse problems, *IEEE Trans. on Magnetics*, vol.43, no.4, pp.1597-1600, 2007.
[18] J. Kennedy and R. Mendes, Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms, *Proc. of the IEEE Int. Workshop on Soft Computing in Industrial Applications*, pp.45-50, 2003.
[19] B. B. Thompson, R. J. Marks, M. A. El-Sharkawi, W. J. Fox and R. T. Miyamoto, Inversion of neural network underwater acoustic model for estimation of bottom parameters using modified particle swarm optimizer, *Proc. of the Int. Joint Conference on Neural Networks*, vol.2, pp.1301-1306, 2003.
[20] T. M. Blackwell and P. J. Bentley, Dynamic search with charged swarms, *Proc. of the Genetic and Evolutionary Computation Conference*, pp.19-26, 2002.
[21] T. Peram, K. Veeramachaneni and C. K. Mohan, Fitness-distance-ratio based particle swarm optimization, *Proc. of the IEEE Swarm Intelligence Symposium*, pp.174-181, 2003.
[22] W. Jian, Y. Xue and J. Qian, An improved particle swarm optimization algorithm with disturbance, *IEEE International Conference on Systems, Man and Cybernetics*, pp.5900-5904, 2004.
[23] R. A. Krohling, Gaussian particle swarm with jumps, *Proc. of IEEE Congress on Evolutionary Computation*, Edinburgh, UK, pp.1226-1231, 2005.
[24] C. Yang and D. Simon, A new particle swarm optimization technique, *Proc. of the 18th Int. Conference on Systems Engineering*, pp.164-169, 2005.
[25] R. A. Krohling and L. S. Coelho, POS-E: Particle swarm optimization with exponential distribution, *Proc. of IEEE Congress on Evolutionary Computation*, pp.1428-1433, 2006.
[26] W.-M. Zhong, S.-J. Li and F. Qian, θ-PSO: A new strategy of particle swarm optimization, *Journal of Zhejiang University Science A*, vol.9, no.6, pp.786-790, 2008.
[27] C. S. Beightler and D. T. Phillips, *Applied Geometric Programming*, Jhon Wiley and Sons, New York, 1976.
[28] B. N. Prasad and J. S. Saini, Optimal thermo hydraulic performance of artificially roughened solar air heaters, *J. Solar Energy*, vol.47, no.2, pp.91-96, 1991.

Appendix.

1. Ratringin function

$$\min_x f_1(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right), \quad -5.12 \le x_i \le 5.12,$$
$$x^* = (0, 0, \ldots, 0), \quad f_1(x^*) = 0.$$

2. Sphere function

$$\min_x f_2(x) = \sum_{i=1}^{n} x_i^2, \quad -5.12 \le x_i \le 5.12,$$
$$x^* = (0, 0, \ldots, 0), \quad f_2(x^*) = 0.$$

3. Griewank function

$$\min_x f_3(x) = \frac{1}{4000}\sum_{i=0}^{n-1} x_i^2 - \sum_{i=0}^{n-1} \cos\left(\frac{x_i}{\sqrt{i+1}}\right) + 1, \quad -600 \le x_i \le 600,$$
$$x^* = (0, 0, \ldots, 0), \quad f_3(x^*) = 0.$$

4. Rosenbrock function

$$\min_x f_4(x) = \sum_{i=0}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \quad -30 \le x_i \le 30,$$
$$x^* = (1, 1, \ldots, 1), \quad f_4(x^*) = 0.$$

5. Ackley's path function

$$\min_x f_5(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right),$$
$$-32 \le x_i \le 32, \quad x^* = (0, 0, \ldots, 0), \quad f_5(x^*) = 0.$$

6. Dejong's function with noise

$$\min_x f_6(x) = \left(\sum_{i=0}^{n-1}(i+1)x_i^4\right) + rand[0,1], \quad -1.28 \le x_i \le 1.28,$$
$$x^* = (0, 0, \ldots, 0), \quad f_6(x^*) = 0.$$

7. Michalewicz function
Functions f_7, f_8 and f_9 are Michalewicz function with dimension 2, 5 and 10 respectively.

$$f(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(i\frac{x_i^2}{\pi}\right)\right)^{2m}, \quad m = 10, \quad -\pi \le x_i \le \pi,$$
$$f_7(x^*) = -1.8013, \quad f_8(x^*) = -4.6876, \quad f_9(x^*) = -9.66015.$$

8. Step function

$$\min_x f_{10}(x) = \sum_{i=0}^{n-1}\lfloor x_i + 1/2\rfloor^2, \quad -100 \le x_i \le 100, x^* = (0, 0, \ldots, 0),$$
$$f_{10}(x^*) = 0.$$

9. Schwefel's function 1.2

$$\min_x f_{11}(x) = \sum_{i=0}^{n-1}\left(\sum_{j=0}^{i} x_i\right)^2, \quad -100 \le x_i \le 100,$$
$$x^* = (0, 0, \ldots, 0), \quad f_{11}(x^*) = 0.$$

10. Schwefel's function 2.21

$$\min_x f_{12}(x) = \max|x_i|, \quad 0 \le i < n, \quad -100 \le x_i \le 100,$$
$$x^* = (0, 0, 0, \ldots, 0), \quad f_{12}(x^*) = 0$$

11. Schwefel's function 2.22

$$\min_x f_{13}(x) = \sum_{i=0}^{n-1}|x_i| + \prod_{i=0}^{n-1}|x_i|, \quad -10 \le x_i \le 10,$$
$$x^* = (0, 0, \ldots, 0), \quad f_{13}(x^*) = 0$$

12. Sum of different power

$$\min_x f_{14}(x) = \sum_{i=1}^{n}|x_i|^{(i+1)}, \quad -1 \le x_i \le 1,$$
$$x^* = (0, 0, 0, \ldots, 0), \quad f_{14}(x^*) = 0$$

13. Alphine function

$$\min_x f_{15}(x) = \sum_{i=1}^{n}|x_i\sin(x_i) + 0.1x_i|, \quad -10 \le x_i \le 10,$$
$$x^* = (0, 0, 0, \ldots, 0), \quad f_{15}(x^*) = 0$$

14. Generalized penalized function 1

$$\min_x f_{16}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(y_{i+1}\pi)\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i,10,100,4),$$

where $y_i = 1+\frac{1}{4}(x_i+1)$, $-50 \le x_i \le 50$, $x^* = (0,0,\ldots,0)$, $f_{16}(x^*) = 0$

15. Generalized penalized function 2

$$\min_x f_{17}(x) = (0.1)\Big\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}\left((x_i-1)^2(1+\sin^2(3\pi x_{i+1}))\right) + (x_n-1)\left(1+\sin^2(2\pi x_n)\right)\Big\} + \sum_{i=0}^{n-1} u(x_i,5,100,4),$$

$$-50 \le x_i \le 50, \quad x^* = (1,1,\ldots,-4.76), \quad f_{17}(x^*) = -1.1428$$

In problem f_{16} and f_{17},

$$u(x,a,b,c) = \begin{cases} b(x-a)^c & \text{if } x > a \\ b(-x-a)^c & \text{if } x < -a \\ 0 & \text{if } -a \le x \le a \end{cases}$$

16. Schwefel function

$$\min_x f_{18}(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}), \quad -500 \le x_i \le 500,$$

$$x^* = (420.97, 420.947, \ldots, 420.947), \quad f_{18}(x^*) = -418.9829 * n$$

17. Levy and Mantalvo function

$$\min_x f_{19}(x) = \sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right) + (x_n-1)\left(1+\sin^2(2\pi x_{n1})\right),$$

$$-10 \le x_i \le 10, \quad x^* = (1,1,\ldots,1,-9.7523), \quad f_{19}(x^*) = -21.5023$$

18. Dejong's function (no noise)

$$\min_x f_{20}(x) = \sum_{i=0}^{n-1}(i+1)x_i^4, \quad -1.28 \le x_i \le 1.28,$$

$$x^* = (0,0,\ldots,0), \quad f_{20}(x^*) = 0.$$

19. Hartmann function 1

$$\min_x f_{21}(x) = -\sum_{i=1}^{4}\alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right), \quad 0 \le x_i \le 1,$$

$$x^* = (0.114614, 0.555649, 0.852547), \quad f_{21}(x^*) = -3.86278$$

where $\alpha = \begin{bmatrix} 1 & 1.2 & 3 & 3.2 \end{bmatrix}$, $A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $P = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$

20. Hartmann function 2

$$\min_x f_{22}(x) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} B_{ij}(x_j - Q_{ij})^2\right), \quad 0 \le x_i \le 1,$$

$$x^* = (0.20169, 0.50011, 0.476874, 0.275332, 0.311652, 0.6573), \quad f_{22}(x^*) = -3.32237$$

where $\alpha = \begin{bmatrix} 1 & 1.2 & 3 & 3.2 \end{bmatrix}$, $B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$,

$$Q = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

21. Shaffer's function 6

$$\min_x f_{23}(x) = 0.5 + \frac{\sin^2\sqrt{(x_1^2 + x_2^2)} - 0.5}{1 + 0.01(x_1^2 + x_2^2)^2}, \quad -10 \le x_i \le 10,$$
$$x^* = (0, 0), \quad f_{23}(x^*) = 0$$

22. Matyas function

$$\min_x f_{24}(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2, \quad -10 \le x_i \le 10,$$
$$x^* = (0, 0), \quad f_{24}(x^*) = 0$$

23. Six hump camel back function

$$\min_x f_{25}(x) = 4x_0^2 - 2.1x_0^4 + \frac{1}{3}x_0^6 + x_0x_1 - 4x_1^2 + 4x_1^4, \quad -5 \le x_i \le 5,$$
$$x^* = (0.09, -0.71), \quad f_{25}(x^*) = -1.03163$$

24. Axis parallel hyperellipsoid

$$\min_x f_{26}(x) = \sum_{i=1}^{n} ix_i^2, \quad -5.12 \le x_i \le 5.12,$$
$$x^* = (0, 0, \ldots, 0), \quad f_{26}(x^*) = 0$$

25. Colvillie function

$$\min_x f_{27}(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1),$$
$$-10 \le x_i \le 10, \quad x^* = (1, 1, 1, 1), \quad f_{27}(x^*) = 0$$

26. Goldstein and price problem

$$\min_x f_{28}(x) = \left\{1 + (x_0 + x_1 + 1)2(19 - 14x_0 + 3x_0^2 - 14x_1 + 6x_0x_1 + 3x_1^2)\right\} \left\{30 + (2x_0 - 3x_1)^2(18 - 32x_0 + 12x_0^2 + 48x_1 - 36x_0x_1 + 27x_1^2)\right\},$$
$$-2 \le x_i \le 2, \quad x^* = (0, 1), \quad f_{28}(x^*) = 3$$

27. Mccormic function

$$\min_x f_{29}(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1, \quad -2 \le x_i \le 2,$$
$$x^* = (-0.5471, -1.5473), \quad f_{29}(x^*) = -1.9132$$

28. Shubert function 1

$$\min_x f_{30}(x) = \sum_{j=1}^{5} j\cos\left((j+1)x_1 + j\right) \sum_{j=1}^{5} j\cos\left((j+1)x_2 + j\right), \quad -10 \le x_i \le 10,$$

$f_{30}(x^*) = -186.7309$

29. Shubert function 2

$$\min_x f_{31}(x) = -\sum_{i=1}^{n}\sum_{j=1}^{5} j\sin\left((j+1)x_i + j\right),$$

$$-10 \le x_i \le 10$$

30. Shekel's Foxholes function

$$\min_x f_{32}(x) = \left(\frac{1}{500} + \sum_{j=0}^{24}\left(j+1+\sum_{i=0}^{1}(x_i - a_{ij})^6\right)^{-1}\right)^{-1}, \quad -65.54 \le x_i \le 65.54,$$

$x^* = (-31.95, -31.95), \quad f_{32}(x^*) = 1$

31. Branin function

$$\min_x f_{33}(x) = \left(x_1 - \frac{5.1}{4\pi^2}x_0^2 + \frac{5}{\pi}x_0 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_0) + 10, \quad -10 \le x_i \le 10,$$

$x^* = (9.42, 2.47), \quad f_{33}(x^*) = 0.397886$

32. Shaffer's function 7

$$\min_x f_{34}(x) = \left(\sum_{i=1}^{n} x_i^2\right)^{1/4}\left[\sin^2\left(50\left(\sum_{i=1}^{n} x_i^2\right)^{1/10}\right) + 1.0\right], \quad -32.767 \le x_i \le 32.767,$$

$f_{34}(x^*) = 0$

33. Test2N function

$$\min_x f_{35}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i), \quad -5 \le x_i \le 5,$$

$$x^* = (-2.903, -2.903, \ldots, -2.903), \quad f_{35}(x^*) = -78.3323$$

34. Modified Himmelblau function

$$\min_x f_{36}(x) = (x_2 + x_1^2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + x_1, \quad -5 \le x_i \le 5,$$

$$x^* = (-3.788, -3.286), \quad f_{36}(x^*) = -3.7839.$$