A PARTICLE SWARM OPTIMIZATION-THRESHOLD ACCEPTING HYBRID ALGORITHM FOR UNCONSTRAINED OPTIMIZATION

Y. Maheshkumar[*][†], *V. Ravi*[†], *Ajith Abraham*[‡]

Abstract: In this paper, we propose a novel hybrid metaheuristic algorithm, which integrates a Threshold Accepting algorithm (TA) with a traditional Particle Swarm Optimization (PSO) algorithm. We used the TA as a catalyst in speeding up convergence of PSO towards the optimal solution. In this hybrid, at the end of every iteration of PSO, the TA is invoked probabilistically to refine the worst particle that lags in the race of finding the solution for that iteration. Consequently the worst particle will be refined in the next iteration. The robustness of the proposed approach has been tested on 34 unconstrained optimization problems taken from the literature. The proposed hybrid demonstrates superior preference in terms of functional evaluations and success rate for 30 simulations conducted.

Key words: *Particle swarm optimization, threshold accepting algorithm, hybrid metaheuristics, unconstrained optimization*

Received: March 13, 2012
Revised and accepted: March 18, 2013

1. Introduction

Optimization is as ageless as time and is omnipresent, existing in every realm. Optimization in general refers to finding a best solution out of all feasible solutions. A feasible solution for which the optimization function has the best possible value is called an optimal solution. Optimization problems in several realms are formulated

[*]Y. Maheshkumar
Department of Computer and Information Sciences, University of Hyderabad, Hyderabad – 500 046 (A.P), India, E-mail: `rav_padma@yahoo.com`

[†]Y. Maheshkumar, V. Ravi
Institute for Development and Research in Banking Technology, Road #1, Castle Hills, Masab Tank, Hyderabad – 500 067 (A.P), India, E-mail: `ymaheshkumar527@gmail.com`

[‡]Ajith Abraham – Corresponding Author
Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, WA, USA; Faculty of Electrical Engineering and Computer Science, VSB – Technical University of Ostrava, Ostrava, Czech Republic, E-mail: `ajith.abraham@ieee.org`

into mathematical formulation and many approaches are proposed in literature to solve optimization problems. Metaheuristics are one of the best approaches followed to solve optimization problems which are flexible and easy to implement. These metaheuristics are developed by basing on the natural phenomenon like the way in which swarm of birds move, how human generation has evolved, how ants search for their food, etc. Some of them are Particle Swarm Optimization (PSO) [1], Genetic Algorithms (GA) [2,3], Ant Colony Optimization (ACO) [4,5,6], Simulated Annealing (SA) [7], Threshold Accepting (TA) [8], Great Deluge Algorithm (GDA) [9], Modified Great Deluge Algorithm (MGDA) [10], Differential Evolution (DE) [11], and Tabu Search (TS) [12,13].

In optimization, a local optimal solution is a point in the search space where all neighboring solutions are worse than the current solution, and a global optimum is a point in the search space where all other points are worse than (or equal to) the current one. Globally optimizing a function in a given search domain consists of finding its global minima without getting trapped into one of its local minima. To localize a “promising area”, an area where a global minimum is likely to be present, it is necessary to well “explore” the whole domain. When a promising area is detected, appropriate tools must be used to “exploit” this area and obtain the optimum as quickly as possible.

Many researchers use metaheuristics as an efficient way of solving optimization problems. Differential Evolution, a global optimization metaheuristic proposed by Storn and Price [11], is a simple and efficient algorithm. Threshold Accepting, proposed by Dueck and Scheur [8], is a local search technique that works on an individual solution to search for the optimal state. Global search techniques like PSO, DE, ACO, etc., are highly efficient in searching for “promising regions”, while exploitation could be very well done through local search techniques like Nelder-Mead Simplex Search (NMSS), Threshold Accepting (TA).

The process of hybridization of the existing metaheurstics has been followed by researchers for more than last 15 years. The hybrid optimization algorithms benefit from the advantages of the component metaheuristic algorithms. To start with, Ravi et al. [14] hybridized the Non-Equilibrium Simulated Annealing (NESA) with a simplex-like heuristic to develop a new algorithm called Improved Non-Equilibrium Simulated Annealing (INESA). This is one of the earliest hybrid algorithm proposed in the literature. In their paper, the authors improved the Non-Equilibrium Simulated Annealing (NESA) by taking the solutions at regular intervals of the progress of NESA and then combining them with the best solutions obtained before the termination of NESA part of algorithm. At this stage they applied a simplex-like heuristic to obtain the global optimum. Chauhan and Ravi [20] hybridized the Differential Evolution (DE) with Threshold Accepting (TA), which takes the advantage of efficient exploration of DE and exploitation of TA. They reported spectacular reduction function evaluations when tested on test problems. Schmidt and Thierauf [16] hybridized the TA and DE where the TA was first applied to all solutions of solution space and the resultant was passed to the DE to move towards global optimal solution. Bhat et. al. [17] hybridized the DE by employing reflection property of the simplex method for fast convergence to global optima. Trafalis et. al. [18] hybridized 3 heuristics namely scatter search, GA and TS. They introduced the notion of memory to explore the solution space more extensively. It used the scatter search by combining the concepts of trajectory and clustering methods. The later stages of the algorithm combined the characteristics of TS and GA to test the status of new solutions and to direct the search towards global optima. Srinivas and Rangaiah [19] developed a hybrid employing the DE by using tabu lists for solving global optimization problems. Chelouah et al. [16] hybridized the GA and NMSS search method. They used the GA to do detect promising regions where we can find the optimal solution and used the NMSS for intensification i.e., to search locally for global optimum in this promising region.

Many hybrid metaheuristics using the PSO have been proposed in the literature. Deep and Bansal [21] hybridized the PSO with Quadratic Approximation, where the whole swarm was split into two sub-swarms and the PSO was applied to one sub-swarm and QA to the other, ensuring that sub-swarms were updated using the global best particle of the whole swarm. Milie et al. [22] improved the PSO by including a cross-over operator to maintain diversity in the PSO. Li et al. [23] hybridized the PSO with Improved Genetic Algorithm (IGA). where the worst solutions in every iteration particles providing worst solutions were passed to the IGA and the remaining ones were modified using PSO. Shelokar et al. [24] hybridized the PSO with Ant Colony Optimization (ACO) for solving highly non-convex optimization problems. First, the PSO explores the search space thoroughly and when reaches the promising area, the solutions are passed to ACO that does the exploitation part by performing local search. The concepts of diversification and intensification used in this solution were first introduced by Glover [12]. Diversification refers to finding the promising regions that have the most probable solution by considering the entire search space. Intensification is to search locally for global optimum in the promising regions. Jiang et al. [25] proposed Improved PSO by dividing the whole swarm into several sub-swarms; the PSO was applied on each sub-swarm individually and at periodic stages of evolution the population was shuffled to ensure information sharing. Fan and Zahara [26] hybridized the Nelder-Mead Simplex Search (NMSS) with the PSO where, at every stage of evolution, the particles providing elite solutions were passed to the NMSS, and the worst ones were passed to the PSO. Kulakarni and Moorthy [27] proposed an estimation of distribution improved PSO where they restricted the particles to best solutions by probabilistic modeling of an archive of best solutions. Zhang and Xie [28] hybridized the PSO with Differential Evolution Operator (DEPSO), which provided the bell-shaped mutations with consensus on the population diversity along with the evolution, while keeping the self-organized particle swarm dynamics. Ben and Li [29] developed a hybrid metaheuristic using the PSO and DE where they divided the entire swarm into two in which the PSO and DE were applied on two sub-swarms and move towards optimal solution ensuring that they both share same global best particle.

In the proposed hybrid algorithm we tightly coupled PSO and TA so that the PSO is the main optimizer, and the TA is the fine tuner. It starts with the PSO applied on all particles which move towards the optimal solution, and every time after the end of iteration we invoke the TA probabilistically to update the particle that provides the worst solution for that iteration. The TA is a robust local search algorithm that helps the particle in moving towards the optimal solution. The modified particle will join the group for the next iteration. In this way every time by the end of iteration, this probability based calls are made to the TA, which speeds up the algorithm in approaching the solution and the algorithm terminates on convergence criterion. Comparison with other evolutionary algorithms is outside the scope of the paper.

The rest of the paper is organized as follows: The *PSOTA HYBRID* section describes the traditional PSO and TA and then our proposed algorithm is described. The *RESULTS and DISCUSSION* section discusses the results obtained on various benchmark problems, and then finally conclusion is drawn.

2. Psota Hybrid

2.1 Overview of PSO

A novel population-based metaheuristic named PSO, developed by Kennedy and Eberhart [1], is very efficient in solving optimization problems. It was developed based on the idea of how a swarm of birds or a school of fish moves in search for their food. A swarm is a set of disorganized moving individuals that tend to cluster, whereas each individual is moving in random direction. The choreography of these swarms inspired Kennedy and Eberhart to formulate it into a powerful metaheuristic.

PSO is very simple to implement and efficacious. It has very few parameters to tweak, which makes it simpler, and also, with the absence of greediness in the algorithmic design, it makes it faster. PSO progresses towards the solution by mutual sharing of knowledge of every particle collectively.

PSO consists of a swarm of particles where each particle has its own position and velocity. Each particle is initialized randomly at the beginning of the algorithm and the heuristics update the position and velocity in the latter stages of algorithm. The change in position of the particle is influenced by the particle's own experience and that of the best particle in the swarm, i.e. the particle that provides a better optimal solution than any other particle in the swarm. Each particle is evaluated using fitness function that indicates the closeness to the optimal solution. Every particle has a memory variable that stores the best position obtained by the particle so far; it is termed as $p_i best$ – the best position for i^{th} particle. The particle that provides the best fitness value in the swarm is stored as *gbest.* At each iteration, the position of the particle in each dimension is updated by adding velocity to its position making it move towards *gbest* and $p_i best.$

The pseudo code of PSO for global minimization problems is as follows:

Let $P = \{p_1, p_2, p_3, \ldots p_n\}$ be set of particles where each p_i is of d dimensions; $p_{id} = \{p_{i1}, p_{i2}, \ldots, p_{id}\}$. Each particle has its own individual velocities v_i i.e., $V = \{v_1, v_2, v_3, \ldots v_n\}$.

```
Start
Initialize positions and velocities randomly of permissible range for each particle.
While convergence criteria is not met
DO
For each particle
    Calculate the fitness value
```

```
        If the fitness value of the particle is less than $p_i best$ (old) then
    Update $p_i best$ to present value
    End if
End
Update gbest with the particle that provides best fitness value of all the particles in
the swarm
For each particle
    For each dimension
    $v_{id} = w * v_{id}(old)$ +c1*rand*($p_i best_d - p_{id}$)+c2*rand*($g_i best_d - p_{id}$) // update
        particles velocity
    $p_{id} = p_{id}$ (old)+$v_{id}$ // updating particles position
    end for
end for
END
```

Where $c1, c2$ are acceleration coefficients; *rand:* random number between 0, 1; and w is the inertia weight.

The quantities p_i $best_d$ and $g_i best_d$ share the knowledge of particles previous experience and experience of the whole swarm. The way in which birds in a flock move towards their food by considering their experience and taking the help of their neighbors to move further, each particle in the PSO, like a bird in the flock, moves towards the optimal solution.

2.2 Overview of TA

Threshold Accepting algorithm was proposed by Deuck and Sheur in 1990 [8]. It is a point-based search technique. It is a variation of Simulated Annealing (SA) algorithm; however, while in the SA new solutions are accepted probabilistically, in the TA they are accepted based on a deterministic criterion. In the TA, any new solution that is not much worse than the previous solution is accepted.

The pseudo code of TA is as follows:

```
Initialize the solution and set global iteration counter itr=0, old=99999, thresh=2
$f_i \leftarrow$ fitness value of initial solution
while itr<gitr // gitr is the number of global iterations
DO
    itr $\leftarrow$ itr+1
    ii $\leftarrow$ 0 // ii - inner iteration value
    while ii < limit or del1 > thresh
    DO
        ii $\leftarrow$ ii+ 1
        Generate a candidate solution vector using the following equation
            candidate solution = old solution+(max-min)*(2*u-1)$^{pindex}$
        $f_j \leftarrow$ fitness value for the candidate solution
        del1 $\leftarrow f_i - -f_j$
    END
If del1 < thresh, set $f_i = f_j$
If thresh < thrtol, set del2 = (new – old) / old
Report current solution as the optimal one if abs (del2) < acc and exit if itr < gitr
Else
    old ← new
    thresh = thresh * (1-eps)
END
```

TA is applied to a single solution. It is run for *gitr* number of global iterations, and at every iteration, we generate a candidate solution for every inner iteration *ii*. The fitness value is calculated for each candidate solution and the solutions that are not much worse than the previous one are selected for exploring. The algorithm terminates when the difference between the previous and present objective function values is very small. This is determined by the parameter *acc* which is set to 10^{-6} to obtain highly accurate solution. The parameter *thresh* is used to determine the acceptance of candidate solution and is generally set to 2 at the beginning; however, it is gradually decreased in a geometric progression based on an epsilon value that is generally set to 0.01. *limit* is the number of inner iterations. *max, min* are the boundaries of the decision variables, and *pindex* is generally an odd integer between 3 and 33 that is used to generate a value that is added to the old solution to generate a neighborhood solution.

2.3 Proposed PSOTA hybrid algorithm

We proposed a new hybrid algorithm by tightly coupling the PSO with the TA. In our approach we initially started the searching procedure with the PSO and then fine tuned the worst solutions using the TA probabilistically. After the swarm initialization, the particles are updated as in the traditional PSO, and at the end of every iteration we generate a random number between 0 and 1. If the random number ≥ 0.9, then we invoke the TA to update the worst solution-. The worst solution is the solution or particle that gives the worst fitness value of all the particles. Here we set the PSO as the main optimizer and use the TA to tweak the worst solution. This is made with an idea that the worst solution that fails to provide a better fitness value is helped by TA to move towards the optimal solution. The TA, which is a powerful local search technique, ushers the particle in moving towards the global optimal solution. Here we are not using the TA for the whole particles to provide the global optimal solution, but we apply it finely so that it helps in reenergizing the particle towards optimal solution; then the updated particle is made to rejoin the swarm for the next round of the PSO. In this way the process is repeated until convergence criteria are met.

The step wise description of PSOTA hybrid for global minimization problems is as follows:

1. *Initialize the particle swarm*

   Here every particle is initialized randomly within the range of the decision variables in the problem and initialize the velocities of the particles within the range [0, 1]. The acceleration constants $c1, c2$ are set to 2 and the inertia is set to 1. Coming to TA parameters epsilon '*eps*' is set to 0.01, *acc* set to 0.0000012, *thrtol* to 0.000001, *pindex* is set to 29.

```
2. While the convergence criteria is not met
3. DO
4. For each particle
   4.1 Calculate the fitness value by computing objective function for the particle.
   4.2 If the fitness value of the particle is less than $p_i best$ (old) then
       3.2.1 Update $p_i best$ to present value
       End if
   End for
5. Update gbest with the particle that provides best fitness value of all the particles in the swarm
6. For each particle
   5.1 For each dimension
       $v_{id}$= w*$v_{id}$(old) + c1*rand*($p_i best_d$ − $p_{id}$)+c2*rand*($g_i best_d$ − $p_{id}$) // update particle's velocity
       $p_{id}$ = $p_{id}$ (old)+$v_{id}$ // updating particle's position
       end for
   end for
7. if (rand() $\geq$ 0.9) then
       Calculate Threshold Accepting algorithm (TA) for the worst particle
   End if
8. Replace the worst particle in the swarm by the solution given by the TA
9. If a convergence criterion was satisfied the PSO is called and algorithm steps from 3 are repeated
10. Else algorithm exits providing the obtained optimal solution
11. END
```

We invoke the TA whenever the random number generated ≥ 0.9 i.e., with 10% probability. This sort of biased sampling helps in reducing the overhead of TA on the PSO. Here every particle is of '*d*' dimensions; the values of *limit* and *gitr* are set differently for lower and higher dimensional problems which are specified in the experimental settings. The sequence of event flow is presented in the form of flow chart denoted as Fig. 1.

By incorporating the TA, the performance of PSO is considerably enhanced. The main intention of invoking the TA probabilistically is to enhance the traditional PSO without having any overload of TA. This approach pushes up the particle that

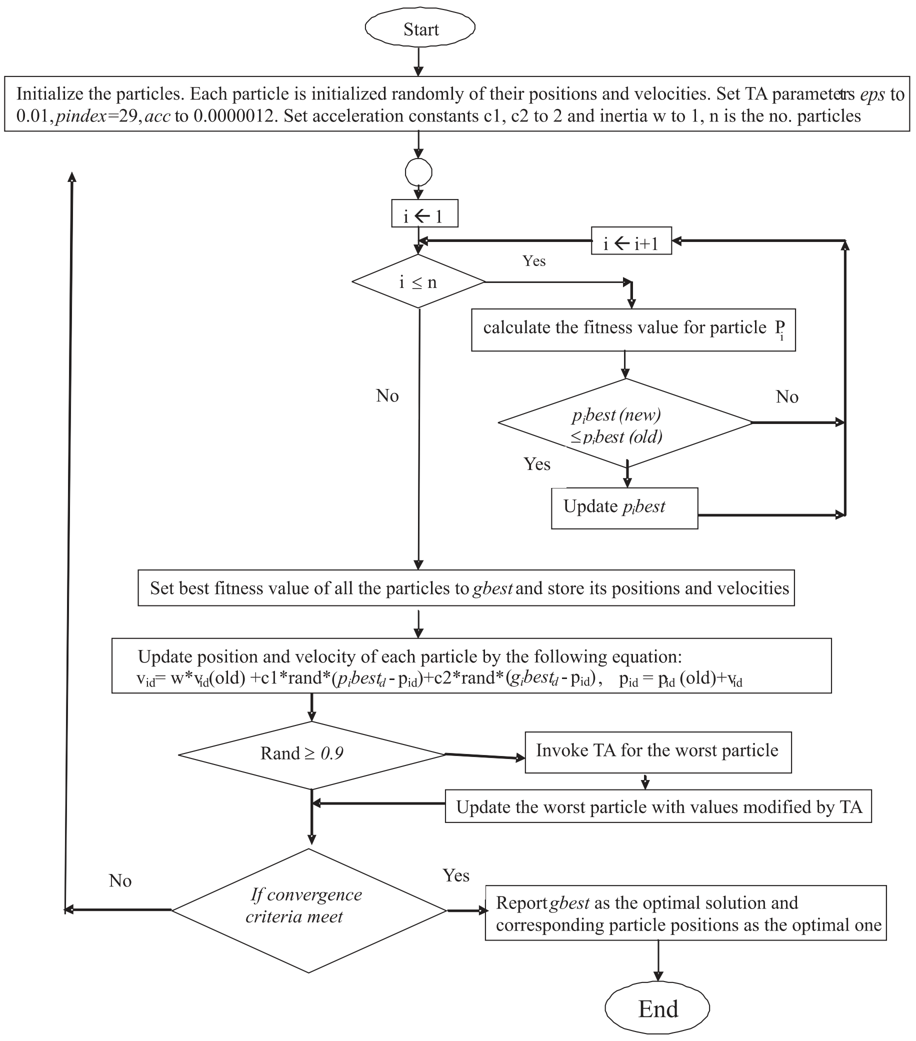

Start
Initialize the particles. Each particle is initialized randomly of their positions and velocities. Set TA parameters eps to 0.01, pindex=29, acc to 0.0000012. Set acceleration constants c1, c2 to 2 and inertia w to 1, n is the no. particles
i ← 1
i ← i+1
i ≤ n
Yes
No
calculate the fitness value for particle P_i
p_ibest (new) ≤ p_ibest (old)
No
Yes
Update p_ibest
Set best fitness value of all the particles to gbest and store its positions and velocities
Update position and velocity of each particle by the following equation:
v_id = w*v_id(old) + c1*rand*(p_ibest_d - p_id) + c2*rand*(g_ibest_d - p_id), p_id = p_id (old) + v_id
Rand ≥ 0.9
Invoke TA for the worst particle
Update the worst particle with values modified by TA
No
If convergence criteria meet
Yes
Report gbest as the optimal solution and corresponding particle positions as the optimal one
End

Fig. 1 *Schematic representation of our proposed hybrid: W represents the worst particle, N is the number of particles and Rand represents random number generator.*

lags behind other particles in that iteration, rejoins the swarm and continues to search the global optimal solution. We use the TA to update only the worst particle because we find that at every iteration, the worst particle is the particle that gains least knowledge from the other particles in the swarm; thus, invoking the TA will help to reenergize the particle so that it will certainly move towards the near optimal solution in the next iteration. The use of TA probabilistically causes that we have less functional evaluations, but on the other hand, it ensures that the robustness of PSO is preserved. This approach really helped reduce functional evaluations without any compromise in the accuracy, which is described below.

The success rate of our hybrid is higher when compared to the normal PSO. Our proposed approach utilizes the best features of both the PSO and TA in a very balanced manner. The schematic representation of our proposed approach is presented in Fig. 2. A fixed number of functional evaluations was set for both the PSO and PSOTA. Both algorithms were set to terminate upon completion of complete number of iterations or if the *present best – previous best < epsilon.*

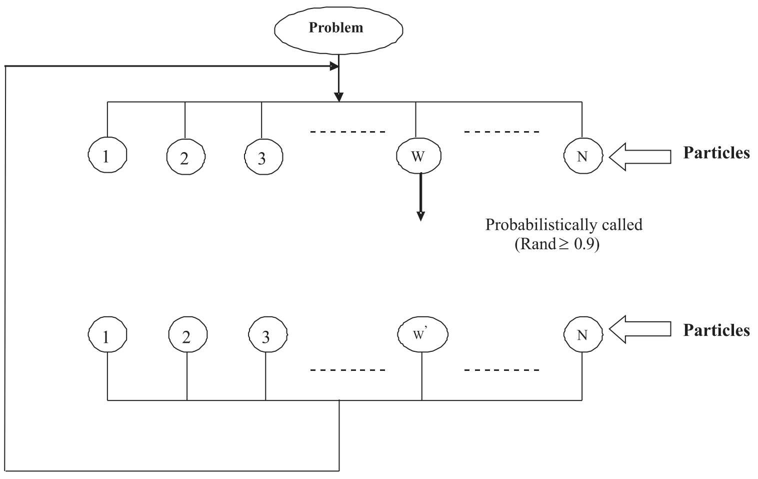


Fig. 2 *Graph comparing accuracy of fitness values of PSO and PSOTA for lower dimensions.*

3. Results and Discussion

The proposed algorithm is tested on 34 unconstrained optimization problems. The most of the problems are taken from Ali et al. [30], others from the Web. The algorithm is run for 30 simulations for different random seeds. The average, standard deviations of the objective functions over 30 simulations are computed. The experimental settings for the proposed hybrid are presented in Tab. II.

The algorithms PSO, PSOTA are implemented in ANSI C using C-Free 4.0. The tests are made on a system with Intel Pentium 4 processor with 2.39 GHz clock speed and 1.99 GB of RAM. Both PSO and PSOTA hybrid are run on 34 unconstrained optimization problems (26 lower dimensional and 8 higher dimensional). The performance of PSO and PSOTA is presented in Tabs. III, IV, V and VI. The results indicate the performance of PSOTA in terms of accuracy and consistency. Our proposed algorithm provides high accuracy in some cases and at least the same accuracy as that of PSO with less functional evaluations. The use of TA to refine the particle helped us in reducing the functional evaluations and also improved accuracy.

Tab. III presents comparativeness of PSO and PSOTA in terms of accuracy and functional evaluations for lower dimensional problems. The result shows that for most of the specified optimization problems, the PSOTA provides efficient re-

SNo	**Parameter name**	**Value**
For PSO part		
1	C1	2
2	C2	2
3	W	1
4	no. of particles	25 (for lower dimensions)
5	no. of particles	50 (higher dimensions)
For TA part		
1	eps	0.01
2	acc	0.0000012
3	thrtol	0.000001
3	pindex	29
4	limit	50
5	gitr	25

Tab. I *Parameters fixed for the PSOTA hybrid.*

SNo.	**Name of the function**	**Dimension**	**Reported objective function value**
1	Sphere [31]	2,30	0
2	RosenBrock [32, 33]	2,20	0
3	Goldstein [34]	2	3
4	Schaffer1 [35]	2	0
5	Schaffer2 [35]	2	0
6	Bohachevsky1[36]	2	0
7	Bohachevsky2 [36]	2	0
8	Periodic [37]	2	0.9
9	Camelback6 [34, 35]	2	0
10	Becker and Lago [37]	2	0
11	Ackley [11]	2	0
12	Salomon [38]	5	0
13	Kowalik [39]	4	$3.0748 * 10^{-4}$
14	Levy and Montalvo1 [40]	3,30	0
15	Levy and Montalvo2 [40,41]	5,30	0
16	Meyer and Roth [42]	3	0
17	Miele and Cantrell [42]	4	0
18	Neumaier2 [43]	4	0
19	Powells [42]	4	0
20	Woods [35,42]	4	0
21	Zakharov [44]	2,20,30,50	0
22	Axis Parallel Hyper Ellipsoidal [45]	2,30	0
23	Rastrigin [11,46]	2	0
24	Shekel5 [34]	4	-10.1499
25	Shekel7 [34]	4	-10.3999
26	Shekel10 [34]	4	-10.5319

Tab. II *Details of test functions.*

Function number	PSO	FE's of PSO	PSOTA	FE's of PSOTA
1	0.000000	1275	0.000000	830
2	0.000000	10025	0.000003	7401
3	3.000000	1275	3.000000	1299
4	0.003715	14040	0.004210	13780
5	0.000000	6275	0.000000	5024
6	0.000000	14040	0.000000	11844
7	0.000000	14040	0.000000	11735
8	0.900000	7525	0.900000	7316
9	0.000000	2275	0.000000	957
10	0.000000	2025	0.000000	1171
11	0.009334	3775	0.000004	1957
12	0.099833	17525	0.006845	14029
13	0.000307	12525	0.000308	4407
14	0.000000	3775	0.000000	1218
15	0.000000	3775	0.000000	3039
16	0.000083	12525	0.000069	10386
17	0.000005	5025	0.000003	2591
18	0.010448	15025	0.007210	13005
19	0.000001	5025	0.000005	3509
20	0.020200	25025	0.029904	18204
21	0.000000	1275	0.000000	911
22	0.000000	650	0.000003	388
23	0.000000	3775	0.000000	2624
24	-9.309608	162525	-10.153199	75109
25	-10.035613	162525	-10.402944	77128
26	-9.997189	162535	-10.536411	77681

Tab. III *Objective function values and functional evaluations for lower dimension problems.*

Function number	PSO	FE's of PSO	PSOTA	FE's of PSOTA
1 @ 30d	0.000005	25030	0.000000	12292
14@ 30d	0.017285	35050	0.000000	24125
15@ 30d	0.000000	75050	0.000000	33178
21@ 20d	0.000000	20040	0.000003	15374
21@ 30d	0.000000	35050	0.000000	30050
21@ 50d	0.000724	75050	0.000054	58956
22@ 30d	0.000000	45050	0.000001	26766
2@ 20d	14.854886	400050	0.069339	400100

Tab. IV *Objective function values and functional values for higher dimensional problems.*

Function number	SR % of PSO	S.D. of PSO	SR% of PSOTA	S.D. of PSOTA
1	100	0	100	9.68468E-07
2	100	0	93.33	9.63805E-06
3	100	0	100	8.89918E-07
4	57	0.004706	57	0.004897
5	96.7	1.82574E-07	96.7	7.48147E-06
6	100	0	100	0
7	100	0	100	0
8	100	4.51681E-16	100	4.51681E-16
9	100	1.82574E-07	100	2.53708E-07
10	100	1.82574E-07	100	1.11211E-06
11	96.66	0.051124	93.33	1.29E-05
12	0	2.82E-17	83.33	0.025298
13	100	1.82574E-07	90	1.17248E-06
14	100	4.61133E-07	100	4.61133E-07
15	100	0	100	0
16	23.33	0.000114927	23.33	2.90144E-05
17	96	3.32169E-05	100	1.19998E-05
18	0	0.013178	3.33	0.009086
19	80	1.02889E-06	80	5.46893E-06
20	0	0.043618	0	0.023845
21	100	1.82574E-07	100	1.82574E-07
22	96.66	0	96.7	5.16E-06
23	100	0	100	0
24	83	1.918587	100	1.81E-15
25	90	1.336582506	100	4.49776E-07
26	90	1.645332	100	1.81E-15

Tab. V *Success Rate (SR) and Standard Deviation (S.D.) for lower dimensional problems.*

Function number	SR% of PSO	S.D. of PSO	SR% of PSOTA	S.D. PSOTA
1 @ 30d	100	1.64701E-06	100	0
14@ 30d	83.33	0.039312	97	1.83077E-06
15@ 30d	96.6	0.004159	100	1.82574E-07
21@ 20d	100	0	96.67	8.13966E-06
21@ 30d	96.6	3.83705E-06	100	6.68675E-07
21@ 50d	23.33	0.001633	33.33	8.79E-05
22@ 30d	100	0	100	2.34128E-06

Tab. VI *Success Rate (SR) and Standard Deviation (S.D.) for higher dimensional problems.*

sults with less functional evaluations. For some problems like Camelback6, Becker and Lago, Kowalik, Levy and Montalvo1, Miele, Axis Parallel Hyper Ellipsoidal, Shekel 5, 7, 10 problems, our hybrid outperformed the PSO by providing better results with very less functional evaluations. Tab. IV presents the performance of PSO and PSOTA in terms of accuracy and functional evaluations in case of higher dimensional problems. In the first column, *number@dimesion* notation is used; for example, 1@30d means the function of serial number 1 in Tab. I of 30 dimensions; dimension indicates the number of decision variables. The results show the robustness of our proposed hybrid, the proposed hybrid provides better solutions with less functional evaluations. Regarding higher dimensional problems, in RosenBrock (20 dimensions), Sphere (30 dimensions), Levy and Montalvo2 (30 dimensions) and Zakharov (50 dimensions), our hybrid outperformed the PSO in terms of functional evaluations. Tabs. V and VI compare the PSO and PSOTA in terms of success rate and standard deviation. The success rate is considered, based on the results obtained in the 30 simulations. Here we considered the resultant value as successful if the obtained objective function value fell within 10^{-6} difference of the reported value. So the success rate is defined as *number of successful results / total number of simulations *100.* Tab. V indicates that the success rate of PSOTA is better and more consistent than that of PSO. The consistency of PSOTA is its main asset. The standard deviations of PSOTA and PSO are also presented, and as can be seen, our hybrid outperformed the PSO in many functions. Tab. VI presents the success rate and standard deviation values for higher dimensional problems by the PSO and PSOTA. The success rate of our hybrid is better than that of PSO, and when it comes to standard deviation values, our hybrid outperformed the PSO in all functions.

T-test has been applied to find out whether the obtained results are statistically significant or not. It was found that for functions: Sphere, Goldstein, Salomon-5d, Kowalik, Shekel5, Powells, Sphere-30d, Levy and Montalvo1-30d, Levy and Montalvo2-30d, Axis Parallel Hyper Ellipsoidal-30d, Rosenbrock-20d, the results of PSOTA hybrid are statistically significant with 95 percent level of confidence.

The proposed experimental setup is made for Un-Constrained optimization problems only, as this experimental work is meant to be. PSOTA can also be applied to solve Constrained optimization problems.

3.1 Visual presentation of the results

The performance of PSO and PSOTA is depicted in Figs. 3–10, where the dark line denotes our proposed PSOTA hybrid and the dotted line denotes the PSO. The optimal objective function values obtained by both algorithms for lower and higher dimensional problems are depicted in Figs. 3 and 4, respectively. The average functional evaluations consumed by the PSO and PSOTA for lower and higher dimensions are depicted in Figs. 5 and 6, respectively. For the sake of convenience, we depicted the function evaluations consumed in case of Shekel 5, 7 and 10 functions in Fig. 6 because both algorithms consumed a very high number of function evaluations there. The success rate obtained by the PSO and PSOTA for both lower and higher dimensional problems is depicted in Figs. 7 and 8, respectively, whereas the standard deviation values obtained by the PSO and PSOTA for lower and higher dimensional problems are depicted in Figs. 9 and 10, respectively. Here X-axis represents optimization functions and Y-axis represents measuring parameter. The performance of PSO and PSOTA on individual functions is depicted in the rest of the figures. Out of 30 simulations run for each function, the results of the best simulation are plotted in these figures. Figs. 11–36 depict the performance of PSO and PSOTA on lower dimensional functions, and Figs. 37–44 depicts the performance of PSO and PSOTA on higher dimensional problems. Evidently, all the above mentioned plots demonstrate the superiority of PSOTA over PSO not only in the success rate but also in function evaluations. Fig. 11 and the following ones represent how the algorithms PSO and PSOTA performs on every individual function for the first 100 iterations: X-axis represents functional evaluations and Y-axis represents fitness values.

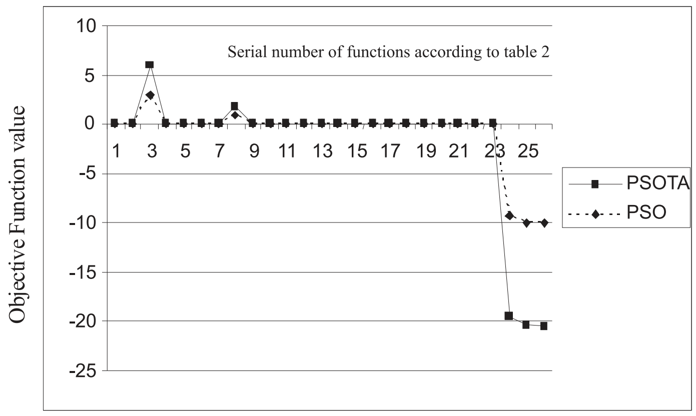


Fig. 3 *Graph comparing accuracy of fitness values of PSO and PSOTA for lower dimensions.*

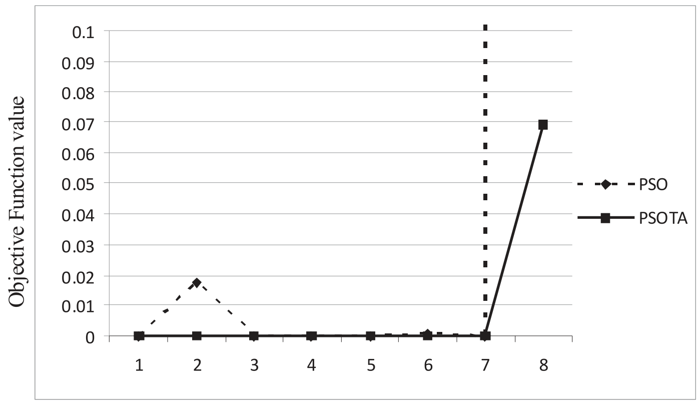


Fig. 4 *Graph comparing accuracy of fitness values of PSO and PSOTA for lower dimensions.*

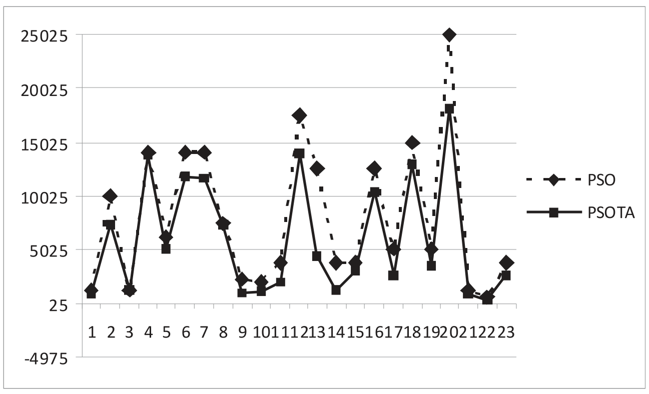


Fig. 5 *Graph comparing accuracy of fitness values of PSO and PSOTA for higher dimensions.*

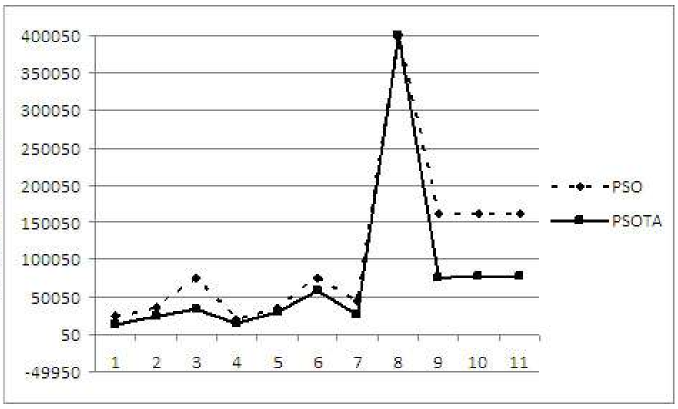


Fig. 6 *Graph comparing accuracy of fitness values of PSO and PSOTA for lower dimensions.*

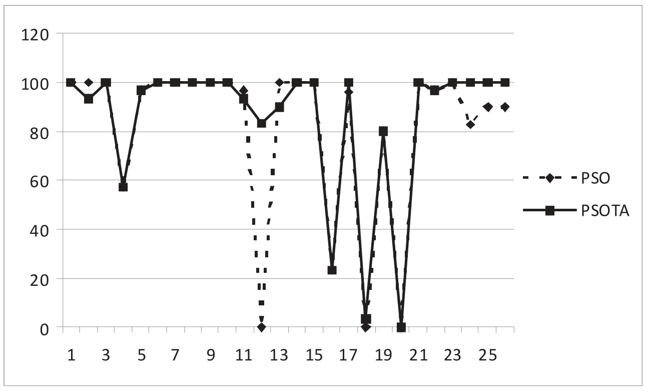


Fig. 7 *Graph comparing accuracy of fitness values of PSO and PSOTA for higher dimensions.*

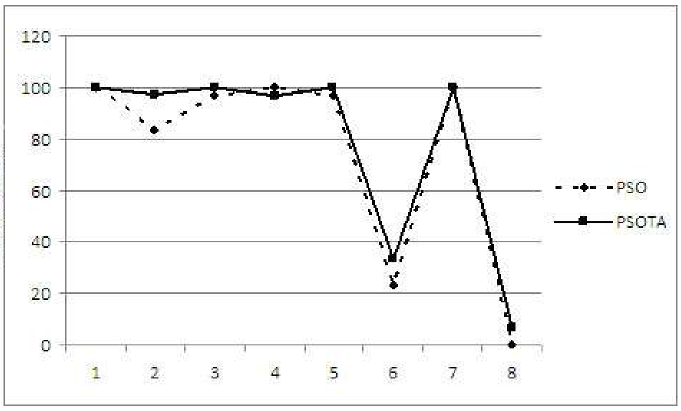


Fig. 8 *Graph comparing accuracy of fitness values of PSO and PSOTA for higher dimensions.*

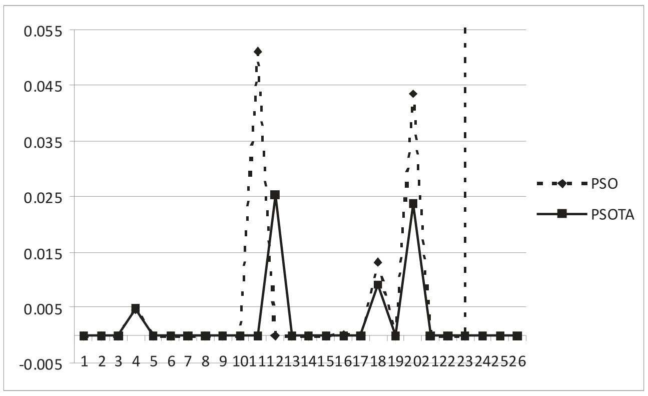


Fig. 9 *Graph comparing accuracy of fitness values of PSO and PSOTA for lower dimensions.*

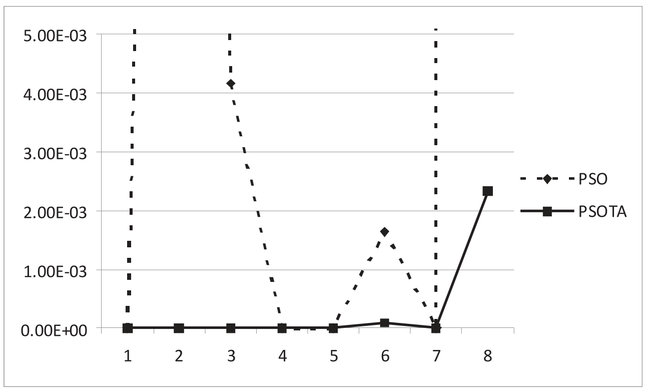


Fig. 10 *Graph comparing accuracy of fitness values of PSO and PSOTA for higher dimensions.*

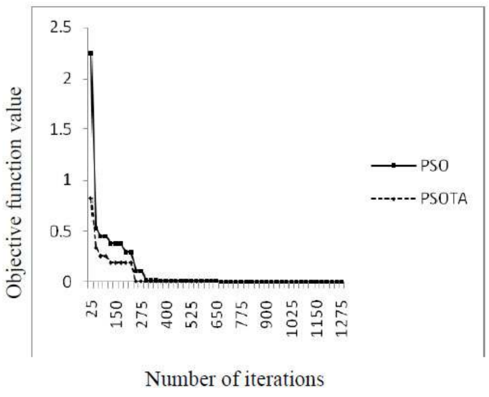


Fig. 11 *Sphere function.*

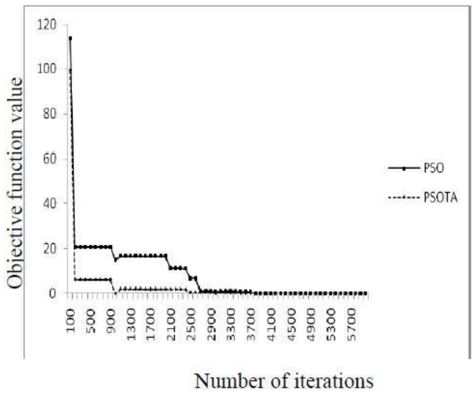


Fig. 12 *Rosenbrock function.*

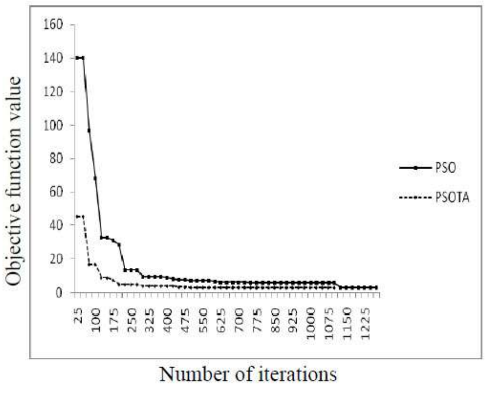


Fig. 13 *Goldstein function.*

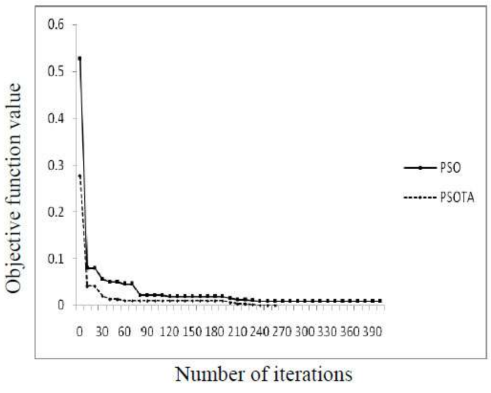


Fig. 14 *Schaffer1 function.*

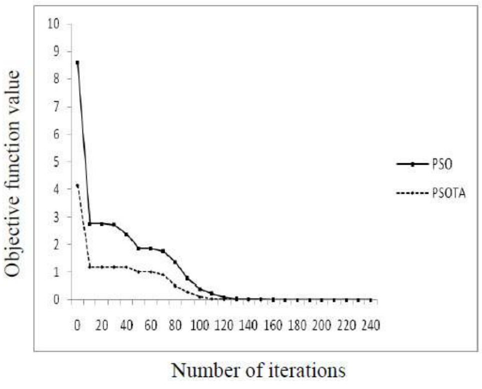


Fig. 15 *Schaffer2 function.*

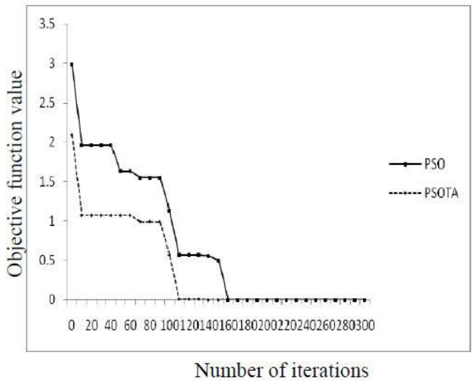
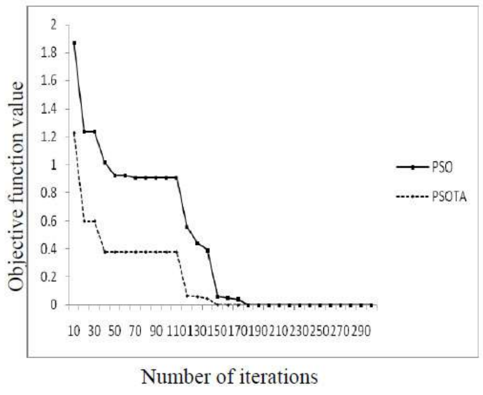


Fig. 16 *Bohachevsky1 function.*



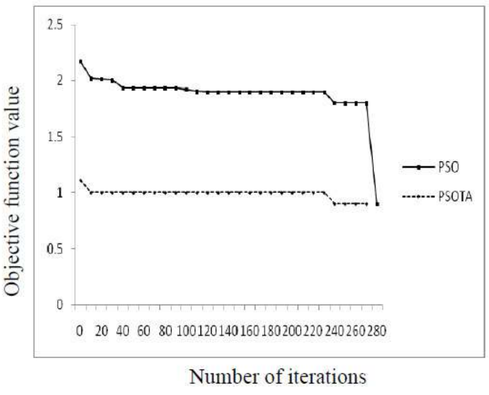
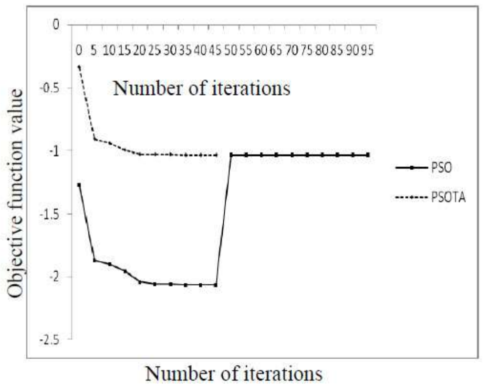


Fig. 17 *Bohachevsky2 function.* **Fig. 18** *Periodic function.*



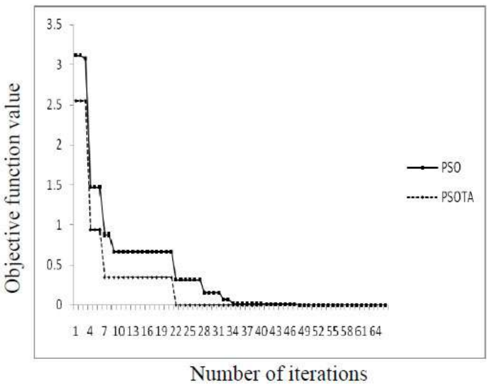
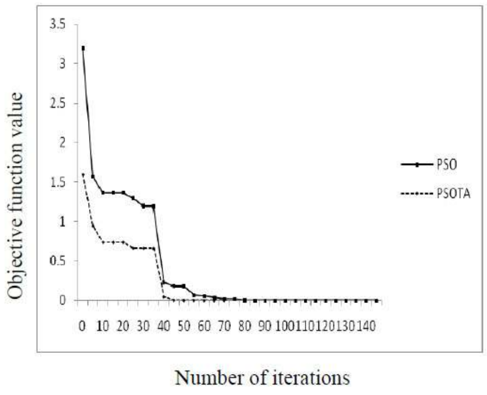


Fig. 19 *CamelBack6 function.* **Fig. 20** *Becker and Lago function.*



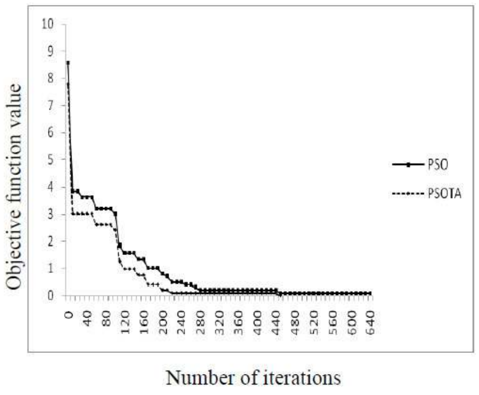


Fig. 21 *Ackley function.* **Fig. 22** *Salomon function.*

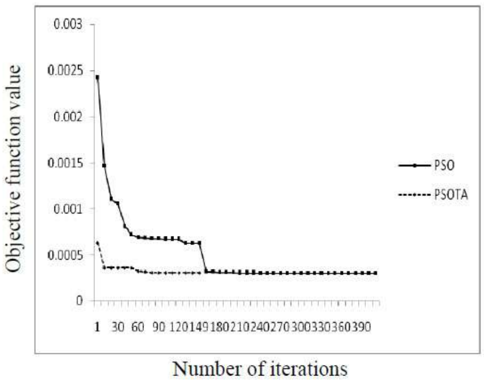


Fig. 23 *Kowalik function.*

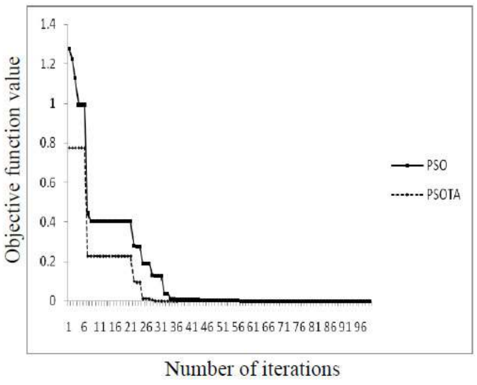


Fig. 24 *Levy and Montalvo1 function.*

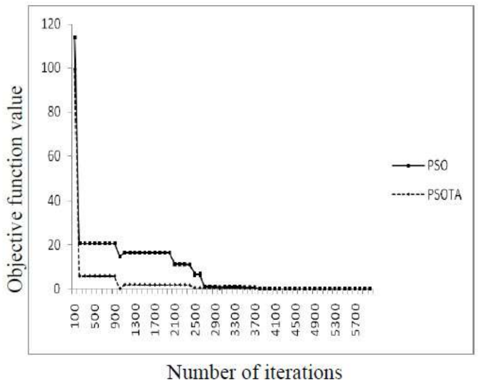


Fig. 25 *Levy and Montalvo2 function.*

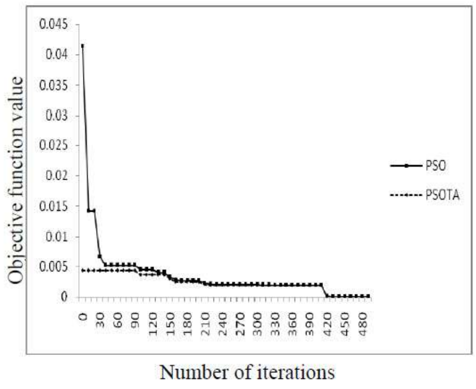


Fig. 26 *Meyer and Roth function.*

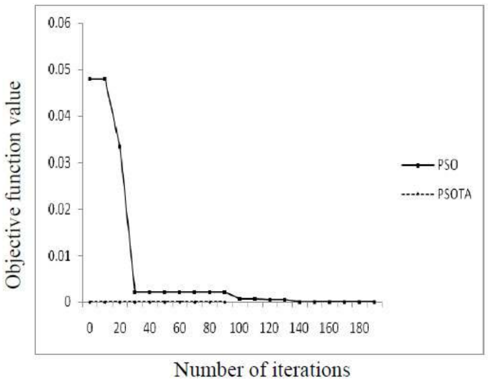


Fig. 27 *Miele and Cantrell function.*

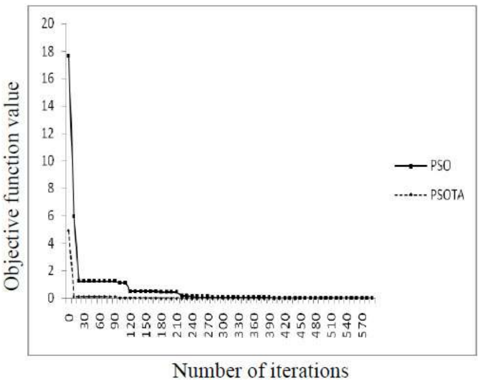


Fig. 28 *Neumaier function.*

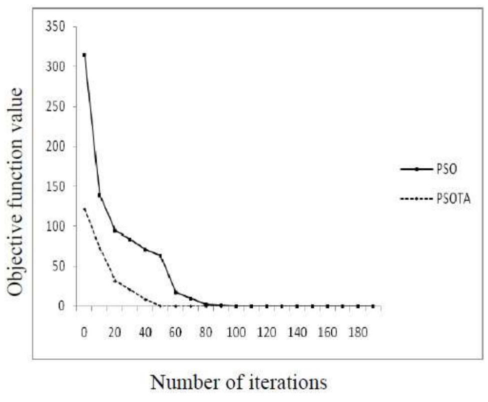


Fig. 29 *Powell's function*

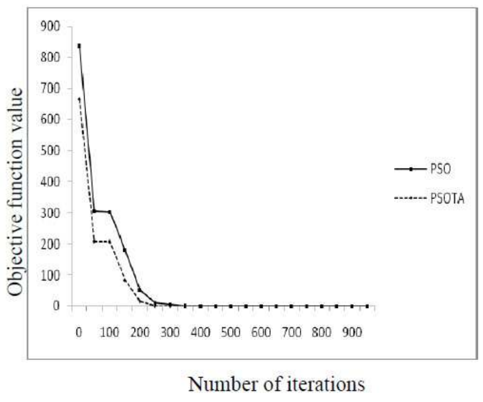


Fig. 30 *Woods function.*

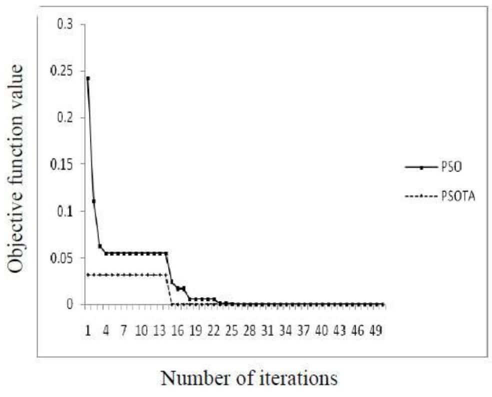


Fig. 31 *Zakharov function.*

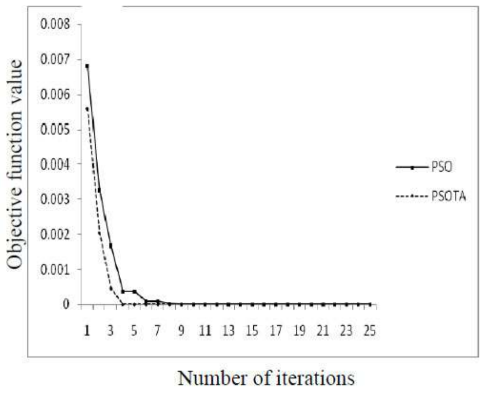


Fig. 32 *Axis parallel hyper ellipsoidal.*

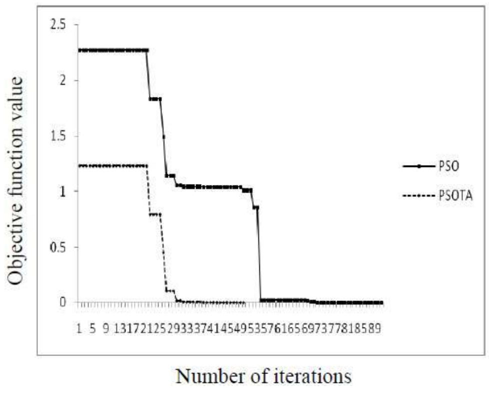


Fig. 33 *Rastrigin function.*

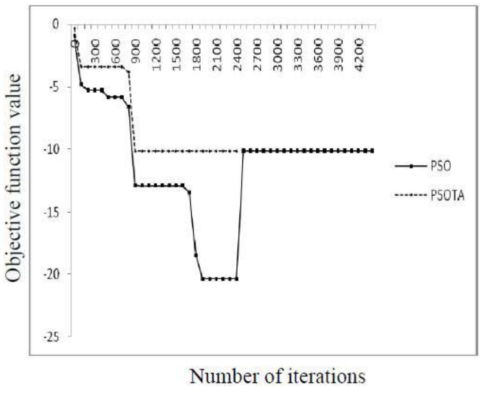


Fig. 34 *Shekel5 function.*

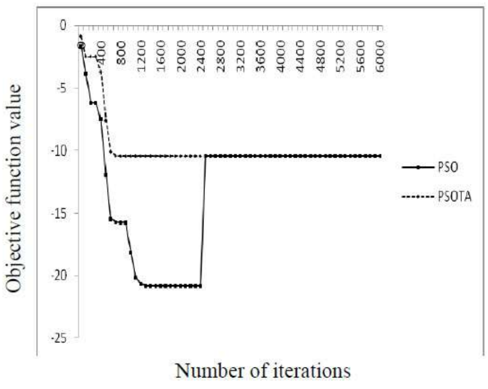


Fig. 35 *Shekel7 function.*

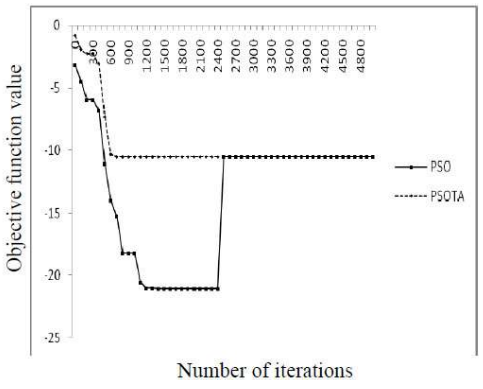


Fig. 36 *Shekel10 function.*

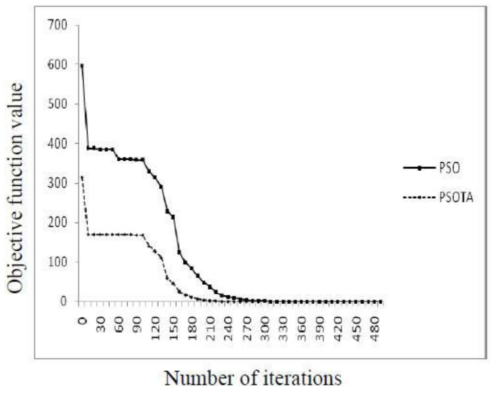


Fig. 37 *Sphere Function: 30d.*

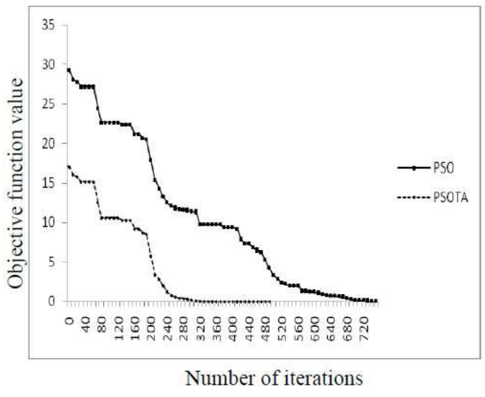


Fig. 38 *Levy and Montalvo1 Function: 30d.*

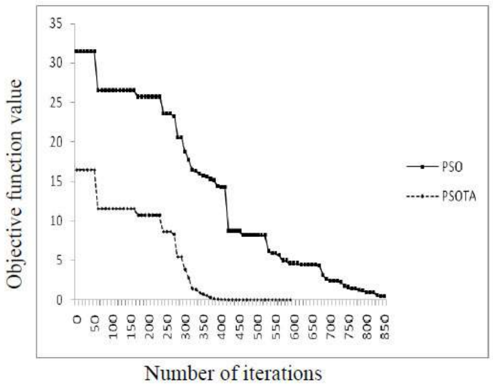


Fig. 39 *Levy and Montalvo2 function: 30d.*

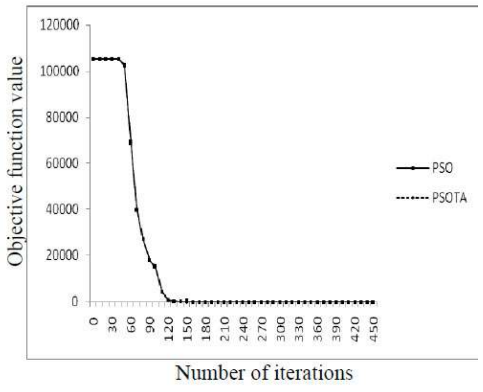


Fig. 40 *Zakharov function: 20d.*

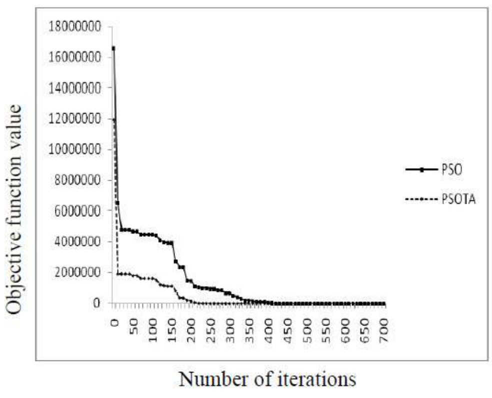


Fig. 41 *Zakharov function: 30d.*

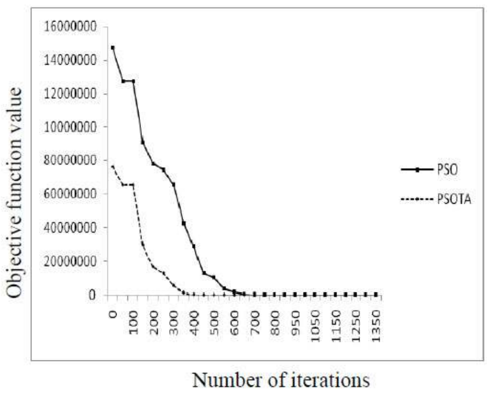


Fig. 42 *Zakharov function: 50d.*

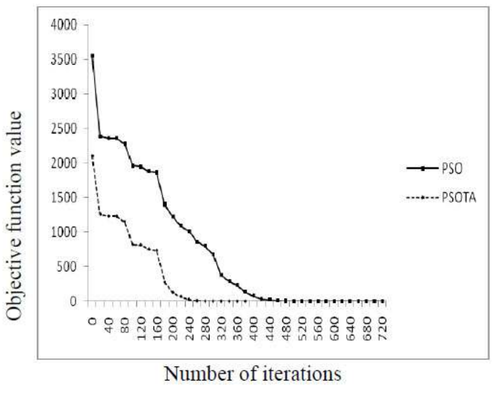


Fig. 43 *Axis parallel hyper ellipsoidal function: 30d.*

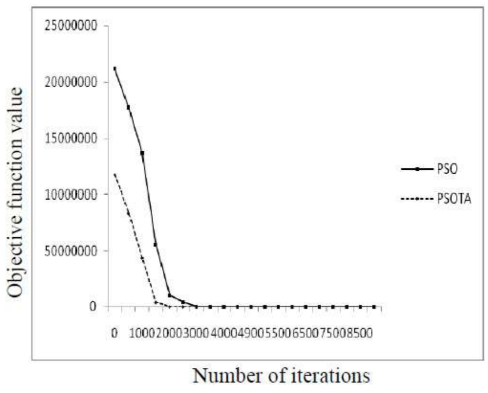


Fig. 44 *RosenBrock function: 20d.*

4. Conclusions

A novel PSOTA hybrid is presented in this paper. The proposed hybrid metaheuristic is compared with the traditional PSO by testing it on a set of 34 unconstrained problems. The results show that our hybrid provides consistent and efficient fitness values with less functional evaluations. The process of invoking TA at the end of each iteration probabilistically helped the PSO approach the optimal solution faster. By using the TA we are enabling the PSO to do local search that it never does as per original algorithm. This process really helps improvise the PSO; and also, with the biased sampling aspect of invocation of TA, we do not increase the complexity of the algorithm, but reduce functional evaluations taken to solve the objective function, as presented in the *results* section. Here the TA acts like a catalyst that speeds up the PSO in moving towards the global optimal solution by enhancing the performance of the worst particle and also increasing its experience by making it do local search. The success rate and standard deviation of PSO and our hybrid are compared in several graphs which demonstrate the robustness of our algorithm. Based on numerical results it is concluded that our proposed hybrid is more promising in providing global optimal solutions.

References

[1] Kennedy J., Eberhart R.: Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[2] Holland J. H.: Outline for logical theory of adaptive systems. Journal of ACM, 1962, **3**, pp. 297–314.

[3] Holland J. H.: Adaptation in natural and artificial systems, University of Michigan press, Ann Harbor, MI, Internal Report, 1975.

[4] Dorigo M.: Optimization, Learning and Natural algorithms, PhD Thesis, 1992, Politecnico di Milano, Italy.

[5] Dorigo M.: Maniezzo V., Colorni A.: Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B, 1996, **26**, 1, pp. 29–41.

[6] Dorigo M., Stützle T.: Ant Colony Optimization. MIT Press, 2004, Cambridge, MA.

[7] Kirkpatrick S., Gelatt C. D. Jr., Vecchi M. P.: Optimization by Simulated Annealing. Science, 1983, No. 4598.

[8] Dueck G., Scheur T.: Threshold Accepting: A General Purpose Optimization Algorithm appearing Superior to Simulated Annealing. Journal of Computational Physics, 1990, **90**, pp. 161–175.

[9] Holland J. H.: Adaptation in natural and artificial systems. University of Michigan press, Ann Harbor, MI, Internal Report, 1975.

[10] Ravi V.: Optimization of Complex System Reliability by a Modified Great Deluge Algorithm. Asia-Pacific Journal of Operational Research, 2004, **21**, 4, pp. 487–497.

[11] Storn R., Price K.: Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 1997, **11**, pp. 341–359.

[12] Glover F.: Tabu Search, Part I. ORSA Journal on computing, 1989, **1**, 1, pp. 190–206.

[13] Glover F.: Tabu search, Part II. ORSA Journal on Computing, 1990, **2**, 1, pp. 4–32.

[14] Ravi V., Murty B. S. N., Reddy P. J.: Non-equilibrium simulated annealing-algorithm applied to reliability optimization of complex systems. IEEE Transactions on Reliability, 1997, **46**, pp. 233–239.

[15] Trafalis T. B., Kasap S.: A novel metaheuristics approach for continuous global optimization, Journal of global optimization, 2002, **23**, pp. 171–190.

[16] Chelouah R., Siarry P.: Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multi-minima functions, European Journal of Operational Research, 2003, **148**, pp. 335–348.

[17] Schmidt H., Thierauf G.: A Combined Heuristic Optimization Technique. Advance in Engineering Software, 2005, **36**, 1, p. 11.

[18] Bhat T. R., Venkataramani D., Ravi V., Murty C. V. S.: Improved differential evolution method for efficient parameter estimation in biofilter modeling, Biochemical Engineering Journal, 2006, **28**, pp. 167–176.

[19] Srinivas M., Rangaiah G. P.: Differential Evolution with Tabu list for Global Optimization and its Application to Phase Equilibrium and Parameter Estimation. Problems Ind Engg Chem. Res., 2007, **46**, pp. 3410–3421.

[20] Chauhan N., Ravi V.: Differential Evolution and Threshold Accepting Hybrid Algorithm for Unconstrained Optimization. International Journal of Bio-Inspired Computation, 2010, **2**, pp. 169–182.

[21] Zhang W., Xie X. F.: DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. IEEE International Conference on Systems, Man & Cybeetics (SMCC), Washington DC, USA, 2003.

[22] Shelokar P. S., Siarry P., Jayaraman V. K., Kulkarni B. D.: Particle Swarm and Ant Colony Algorithms Hybridized for Improved Continuous Optimization. Applied Mathematics and Computation Journal, 2006, **188**, pp. 129–142.

[23] Jiang Y., Tiesong H., Huang C. C., Xianing W.: An Improved Particle Swarm Optimization Algorithm. Applied Mathematics and Computation, 2007, **193**, pp. 231–239.

[24] Fan S. K. S., Zahara E.: A Hybrid Simplex Search and Particle Swarm Optimization for Unconstrained Optimization European Journal of Operational Research, 2007, **181**, pp. 527–548.

[25] Kulkarni R., Venayagamoorthy G. K.: An Estimation of Distribution Improved Particle Swarm Optimization Algorithm. Intelligent Sensors, Sensor Networks and Information Processing Conference, Melbourne, December 2007.

[26] Li W. T., Shi L., Xu X. W., Hei Y. Q.: Improved GA and PSO Culled Hybrid Algorithm for Antenna Array Pattern Synthesis. Progress in Electromagnetic Research, PIER, 2008, **80**, pp. 461–476.

[27] Deep K., Bansal J. C.: Hybridization of Particle Swarm Optimization with Quadratic Approximation. Opsearch, Springer, 2009, **46**, 1, pp. 3–24.

[28] Pant M., Radha T.: Particle Swarm Optimization with Crossover Operator and its Engineering Applications. IAENG International Journal of Computer Science, 2009, **36**, 2.

[29] Ben N., Li L.: A Novel PSO – DE based Hybrid Algorithm for Global Optimization. Advanced intelligent computing Theories and Applications with Aspects of Artificial Intelligence, 2009, **5227**, pp. 156–163.

[30] Ali M. M., Charoenchai K., Zelda B. Z.: A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. Journal of Global optimization, 2005, **31**, pp. 635–672.

[31] Sphere problem: global and local optima [cited on 2010 Nov 20]available at: www.optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page113.htm

[32] Schwefel H. P.: Evolution and Optimum Seeking, John Wiley and Sons, New York, 1995.

[33] More J., Garbow B., Hillstrom K.: Testing Unconstrained Optimization Software. ACM Transaction on Mathematical Software, 1981, **7**, pp. 17–4.

[34] Dixon L., Szego G.: Towards Global Optimization, North Holland, New York, 1978, **2**.

[35] Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin/Heidelberg/New York, 1996.

[36] Bohachevsky M. E., Johnson M. E., Stein M. L.: Generalized simulated annealing for function optimization. Techno metrics, 1986, **28**, pp. 209–217.

[37] Price W. L.: Global Optimization by Controlled Random Search. Computer Journal, 1977, **20**, pp. 367–370.

[38] Salomon R.: Reevaluating Genetic Algorithms Performance under Co-ordinate Rotation of Benchmark Functions. Bio Systems, 1995, **39**, 3, pp. 263–278.

[39] Jansson C., Knüppel O.: A Branch and Bound Algorithm for Bound Constrained Optimization Problems without Derivatives. Journal of Global Optimization, 1995, **7**, pp. 297–331.

[40] Levy A. V., Montalvo A.: The Tunneling Algorithm for the Global Minimization of Functions. Society for Industrial and Applied Mathematics, 1985, **6**, pp. 15–29.

[41] Dekkers L., Szego G.: Global Optimization and Simulated Annealing. Mathematical Programming, 1991, **50**, pp. 367–393.

[42] Wolfe MA. Numerical Methods for Unconstrained Optimization, Van Nostrand Reinhold Company, New York, 1978.

[43] Neumaier A.: Global and local optimization [cited 2010 Nov 20]. www-document available at http://solon.cma.univie.ac.at/∼neum/glopt.html.

[44] Zakharov Problem Global and Local Optima [cited on 2010 Nov 20] available at: www.optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page3088.htm

[45] Axis Parallel Hyper-Ellipsoidal Problem: Global optimum www document [cited 2010 Nov 19] available from: http://www.geatbx.com/docu/fcnindex-01.html.

[46] Torn A., Zilinskas A.: Global Optimization. Lecture Notes in Computer Science, Springer-Verlag, Berlin/Heidelberg/New York, 1989, **350**.

APPENDIX

1. Sphere Problem

Sphere function is defined as

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2$$

where n is the number of variables and its search domain ranges from $-5.12 \le x_i \le 5.12, i = 1, 2, \ldots, n$. There are no local minima and has a global minima $x^* = (0, 0, \ldots .0), f(x^*) = 0$.

2. Rosenbrock's Function (Schwefel, 1995)

$$\text{Minimize } f(x) = \sum_{i=1}^{n} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 - 2.048 \le x_i \le 2.048$$

It is a classic optimization problem, also known as Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult. The global minimum occurs at $(1, 1, \ldots, 1)$ with objective function value of 0. We tested the two dimensional version.

3. Goldstein-Price (Dixon and Szego, 1978)

Minimize

$$\begin{aligned} f(x) &= [1 + (x_1 + x_2 + 1)^2.(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ &\times [30 + (2x_1 - 3x_2)^2.(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\ & -2 \le x_1, x_2 \le 2 \end{aligned}$$

There are four local minima and the global minima occurs at $(0, -1)$ with $f(0, -1) = 3$.

4. Schaffer1 problem (SF1) (Michalewicz, 1996)

$$\min_x f(x) = 0.5 + \frac{(\sin \sqrt{(x_1^2 + x_2^2)})^2 - 0.5}{1 + 0.001(x_1^2 + x_2^2)^2}$$

Subject to $-100 \le x_1, x_2 \le 100$, the number of local minima is not known but global minima is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

5. Schaffer2 Problem (SF2) (Michalewicz, 1996)

$$\min_x f(x) = (x_1^2 + x_2^2)^{0.25}(\sin^2(50((x_1^2 + x_2^2)^{0.1}) + 1)$$

Subject to $-100 \le$ x$_1$,x$_2$ ≤ 100, the number of local minima is not known but global minima is located at x* = (0,0) with f(x*) = 0.

6. Bohachevsky1 Problem (BF1) (Bohachevsky et al., 1986)

$$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.7$$

Subject to $-50 \leq x_1, x_2 \leq 50$. The number of local minima is unknown but the global minimizer is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

7. Bohachevsky2 Problem (BF1) (Bohachevsky et al., 1986)

$$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$$

Subject to $-50 \leq x_1, x_2 \leq 50$. The number of local minima is unknown but the global minimizer is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

8. Periodic Problem (PRD) (Price, 1977)

$$\min_x f(x) = 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1\exp(-x_1^2 - x_2^2)$$

Subject to $-10 \leq x_1, x_2 \leq 10$. There are 49 local minima all with minimum values 1 and global minimum located at $x^* = (0, 0)$ with $f(x^*) = 0.9$.

9. Camel Back – 6 Six Hump Problem (CB3) (Dixon and Szego, 1978; Michalewicz, 1996)

$$\min_x f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

Subject to $-5 \leq x_1, x_2 \leq 5$. This function is symmetric about the origin and has three conjugate pairs of local minima with values $f \approx -1.0316, -0.2154, 2.1042$. The function ≈ -1.0316.

10. Becker and Lago Problem (BL) (Price, 1977)

$$\min_x f(x) = (|x_1| - 5)^2 + (|x_2| - 5)^2$$

Subject to $-10 \leq x_1, x_2 \leq 10$. The function has four minima located at $(\pm 5, \pm 5)$, all with $f(x^*) = 0$.

11. Ackley's Problem (ACK) (Storn and Price, 1997)

$$\min_x f(x) = -2\exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n}\cos(2\pi x_i)}\right) + 20 + e$$

Subject to $-30 \leq x_i \leq 30, i \in \{1, 2, \ldots, n\}$. The number of local minima is not known. The global minimum is located at the origin with $f(x^*) = 0$. Tests were performed for $n = 10$.

12. Salomon Problem (SAL) (Salomon, 1995)

$$\min_x f(x) = 1 - \cos(2\pi||x||) + 0.1||x||$$

where $||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$ and $-100 \leq x_i \leq 100$. The number of local minima is not known but global optimum lies at $x^* = (0, 0, \ldots 0)$ with $f(x^*) = 0$ and n is the number of variables.

13. Kowalik Problem (KL) (Jansson and Knuppel, 1995)

$$\min_x f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(1 + x_2 b_i}{1 + x_3 b_i + x_4 b_i^2}\right)^2$$

subject to $0 \leq x_i \leq 0.42$. The global optimum lies at $x^* \approx (0.192, 0.190, 0.123, 0.135)$ and $f(x^*) \approx 3.0748 * 10^{-4}$.

The values for a_i, b_i constants are:
$a[1{-}11] = \{0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, 0.0323, 0.0235, 0.0246\}$
$b[1 - 11] = \{0.25, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0\}$

14. Levy and Montalvo1 Problem (LM1) (Levy and Montalvo, 1985)

$$\begin{aligned}\min_x f(x) &= (\pi/n)\left(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right)\\ &\quad \text{where } y_i = 1 + 1/4(x_i + 1)\end{aligned}$$

where $y_i = 1/4(x_i + 1)$ for $-10 \leq x_i \leq 10$. $i \in \{1, 2, \ldots n\}$. There are 5^n local minima and global minima is $f(x^*) = 0$ with $x^* = (-1, -1, \cdots -1)$. Here '$n$' is the number of variables.

15. Levy and Montalvo2 Problem (LM1) (Levy and Montalvo, 1985; Dekker and Aarts, 1991)

$$\begin{aligned}\min_x f(x) &= 0.1\Bigg(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + \\ &\qquad +(x_n-1)^2[1+\sin^2(2\pi x_n)]\Bigg)\end{aligned}$$

Subject to -5 $\leq$ x$_i$ $\leq$ 5, i$\in \{1,2,\dots\ n\}$. There are approximately 15n local minima and global optimum is f(x*)=0 at x* $= (1,1,\dots 1)$. Here 'n' is the number of variables.

16. Meyer and Roth Problem (MR) (Wolfe, 1978)

$$\min_x f(x) = \sum_{i=1}^{5}\left(\frac{x_1x_3t_i}{(1+x_1t_1+x_2v_i)} - y_i\right)^2$$

Subject to $-10 \leq x_i \leq 10$, $i \in \{1,2,\dots,n\}$. This is a least squares problem with minimum value $f(x^*) = 0.4 * 10^{-4}$ located at $x^* = (3.13, 15.16, 0.78)$.

data:
$t[1-5] = \{1.0, 2.0, 1.0, 2.0, 0.1\}$
$v[1-5] = \{1.0, 1.0, 2.0, 2.0, 0.0\}$
$y[1-5] = \{0.126, 0.219, 0.076, 0.126, 0.186\}$

17. Miele and Cantrell Problem (MCP) (Wolfe, 1978)

$$\min_x f(x) = (\exp(x_1)-x_2)^4 + 100(x_2-x_3)^6 + (\tan(x_3-x_4))^4 + x_1^8$$

Subject to $-1 \leq x_i \leq 1, i \in \{1,2,3,4\}$. The number of local minima is unknown but the global optimizer is located at $x^* = \{0,1,1,1\}$ with $f(x^*) = 0$.

18. Neumaier2 Problem (NF2) (Neumaier, 2003b)

$$\min_x f(x) = \sum_{k=1}^{n}\left(b_k - \sum_{i=1}^{n} x_i^k\right)^2$$

Subject to $0 \leq x_i \leq n, i \in \{1,2,\dots,n\}$. The case considered here is $n = 4$ and $b = \{8, 18, 44, 114\}$. The global minimum is $f(1,2,2,3) = 0$.

19. Powell's Quadratic Problem (PWQ) (Wolfe, 1978)

$$\min_x f(x) = (x1 + 10x1)2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

Subject to $-10 \leq x_i \leq 10, i \in \{1, 2, 3, 4\}$. This is a unimodal function with $f(x^*) = 0, x^* = (0, 0, 0, 0)$. This minimiser is difficult to obtain with accuracy as the hessian matrix at the optimum is singular.

20. Wood's function (WF) (Michalewicz, 1996; Wolfe, 1978)

$$\begin{aligned}\min_x f(x) &= 100(x2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + \\ &+ 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)\end{aligned}$$

Subject to $-10 \leq x_i \leq 10, i \in \{1, 2, 3, 4\}$. The function has a saddle near $(1, 1, 1, 1)$. The only minimum is located at $x^* = (1, 1, 1, 1)$ with $f(x^*) = 0$.

21. Zakharov (Z_n) (n variables) (Chelouah and Siarry, 2003)

$$Z_n(x) = \left(\sum_{j=1}^n x_j^2\right) + \left(\sum_{j=1}^n 0.5jx_j\right)^2 + \left(\sum_{j=1}^n 0.5jx_j\right)^4;$$

5 functions were considered: Z_2, Z_5, Z_{10}, Z_{50} and Z_{100}; search domain: $-5 \leq x_j \leq 10, j = 1, \ldots, n; n$ several local minima (exact number unspecified in usual literature); 1 global minimum: $x* = (0, \ldots, 0); Z_n(x*) = 0$.

22. Axis Parallel Hyper-ellipsoidal Function

$$\min_x f(x) = \sum_{i=1}^n ix_i^2$$

Subject to $-5.12 \leq x_i \leq 5.12$. It is also known as weighted sphere model. It is continuous, convex and unimodal. It has a global minimum at $f(x^*) = 0$ at $x^* = \{0, 0, \ldots, 0\}$.

23. Rastrigin Problem (RG) (Storn and Price, 1997; Torn and Zilinskas, 1989)

$$\min_x f(x) = 10n + \sum_{i=1}^n \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Subject to $-5.12 \leq x_i \leq 5.12$, $i\{1, 2, \ldots, n\}$. The total number of minima for this function is not exactly known but global optimizer is located at $x^* = \{0, 0, \ldots 0\}$ with $f(x^*) = 0$.

24. Shekel5 Problem (S5) (Dixon and Szegö, 1978)

$$\min f(x) = -\sum_{i=1}^{5} \frac{1}{\sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i}$$

Subject to $0 \leq x_j \leq 10$, $j \in \{1,2,3,4\}$ with constants a_{ij} and c_j given in Tab. VII. There are five local minima and the global minimiser is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.1499$.

	i	a_{ij}				c_i
		j = 1	2	3	4	
S5	1	4	4	4	4	0.1
	2	1	1	1	1	0.2
	3	8	8	8	8	0.2
	4	6	6	6	6	0.4
	5	3	7	3	7	0.4
S7	6	2	9	2	9	0.6
	7	5	5	3	3	0.3
S10	8	8	1	8	1	0.7
	9	6	2	6	2	0.5
	10	7	3.6	7	3.6	0.5

Tab. VII *Data for Shekel5, Shekel7, Shekel10 problem.*

25. Shekel7 Problem (S7) (Dixon and Szegö, 1978)

$$\min f(x) = -\sum_{j=1}^{7} \frac{1}{\sum_{i=1}^{4} (x_j - a_{ij})^2 + c_i}$$

Subject to $0 \leq x_j \leq 10$, $j \in \{1,2,3,4\}$ with constants a_{ij} and c_j given in Tab. VII. There are seven local minima and the global minimiser is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.3999$.

26. Shekel10 Problem (S10) (Dixon and Szegö, 1978)

$$\min f(x) = -\sum_{j=1}^{10} \frac{1}{\sum_{i=1}^{4} (x_j - a_{ij})^2 + c_i}$$

Subject to $0 \leq x_j \leq 10$, $j \in \{1,2,3,4\}$ with constants a_{ij} and c_j given in Tab. VII. There are 10 local minima and the global minimiser is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.5319$.