A NOVEL HYBRID EVOLUTIONARY ALGORITHM FOR MULTI-MODAL FUNCTION OPTIMIZATION AND ENGINEERING APPLICATIONS

L. IDOUMGHAR
LMIA-MAGE and LORIA
Université de Haute Alsace
68093 Mulhouse, France
lhassane.idoumghar@uha.fr

M. MELKEMI
LMIA-MAGE
Université de Haute Alsace
68093 Mulhouse, France
mahmoud.melkemi@uha.fr

R. SCHOTT
IECN and LORIA, Nancy-Université
Université Henri Poincaré, BP 239,
54506 Vandoeuvre-lès- Nancy, France
rene.schott@loria.fr

ABSTRACT
This paper presents a novel hybrid evolutionary algorithm that combines Particle Swarm Optimization (PSO) and Simulated Annealing (SA) algorithms. When a local optimal solution is reached with PSO, all particles gather around it, and escaping from this local optima becomes difficult . To avoid premature convergence of PSO, we present a new hybrid evolutionary algorithm, called PSOSA, based on the idea that PSO ensures fast convergence, while SA brings the search out of local optima because of its strong local-search ability. The proposed PSOSA algorithm is validated on ten standard benchmark functions and two engineering design problems. The numerical results show that our approach outperforms algorithms described in [1, 2].

KEY WORDS
Evolutionary Algorithm, Particle Swarm Optimization, Simulated Annealing, Hybrid algorithms.

1 Introduction

Several optimization algorithms have been developed over the last decades for solving real-world optimization problems. Among the many heuristics let us mention Simulated Annealing (SA) [3, 4] and optimization algorithms that make use of social or evolutionary behaviors like Particle Swarm Optimization (PSO) [1, 5]. SA and PSO are quite popular heuristics for solving complex optimization problems but each method has strengths and limitations.

Particle Swarm Optimization (PSO) is based on the social behavior of individuals living together in groups. Each individual tries to improve itself by observing other group members and imitating the better ones. That way, the group members are performing an optimization procedure which is described in [5]. The performance of the algorithm depends on the way the particles (i.e., potential solutions to an optimization problem) move in the search space with a velocity that is updated iteratively. Large body of research in the field has been devoted to the analysis and proposal of different motion rules (see [6, 7, 8] for recent accounts of PSO research).

To avoid premature convergence of PSO, we combine it with SA: PSO contributes to the hybrid approach in a way to ensure that the search converges faster, while SA makes the search jump out of local optima due to its strong local-search ability. We present a new hybrid optimization algorithm, called PSOSA, which exploits cleverly the features of PSO and SA. PSOSA has been validated on ten benchmark functions [9] and two engineering problems [10, 11] and compared with classical PSO, ATREPSO, QIPSO and GMPSO algorithms described in [1] and TL-PSO presented in [2]. The simulation results show that PSOSA outperforms the above mentioned algorithms. Hence PSOSA is a good alternative for dealing with complex numerical function optimization problems.

This paper is organized as follows. Section 2 introduces briefly PSO and SA algorithms. Section 3 is devoted to a precise detailled description of PSOSA . In Section 4, PSOSA is applied to ten benchmark functions and used for solving two engineering problems. In addition, the simulation results are compared with those of [1, 2]. Conclusions and further research aspects are given in Section 5.

2 Particle Swarm Optimization and Simulated Annealing

2.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [12], inspired by social behavior patterns of organisms that live and interact within large groups. In particular, it incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior.

The idea of PSO algorithm is that particles move through the search space with velocities which are dynamically adjusted according to their historical behaviors. Therefore, the particles have the tendency to move towards the better and better search area over the course of search process. PSO algorithm starts with a group of random (or not) particles (solutions) and then searches for optima by updating each generation. Each particle is treated as a volume-less particle (a point) in the n- dimensional search space. The i^{th} particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{in})$. At each generation, each particle is updated by following two $best$ values:

- The first one is the best solution (fitness) it has achieved so far (the fitness value is also stored). This value is called $cbest$.
- Another best value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called $gbest$. When a particle takes part of the population as its topological neighbours, the best value is a local best and is called $lbest$.

At each iteration, theses two best values are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new move for the particle. The portion of the adjustment to the velocity influenced by the individual's previous best position ($cbest$) is considered the *cognition* component, and the portion influenced by the best in the neighborhood ($lbest$ or $gbest$) is the social component.

With the addition of the inertia factor, ω, by Shi and Eberhart [13] (for balancing the global and the local search), these equations are:

$$v_{i+1} = \omega v_i + c_1 * random(0,1) * (cbest_i - x_i) + c_2 * random(0,1) * (gbest_i - x_i) \quad (1)$$

$$x_{i+1} = x_i + v_i \quad (2)$$

where $random(0,1)$ is a random number independently generated within the range of [0,1] and c_1 and c_2 are two learning factors which control the influence of the social and cognitive components (Usually, c_1 = c_2 = 2).

In equation 1 if the sum on the right side exceeds a constant value, then the velocity on that dimension is assigned to be $\pm V_i max$. Thus, particles' velocities are clamped to the range of $[-V_i max; V_i max]$ which serves as a constraint to control the global exploration ability of PSO algorithm. This also reduces the likelihood of particles for leaving the search space. Note that this does not restrict the values of x_i to the range $[-V_i max; V_i max]$; it only limits the maximum distance that a particle will move during one iteration.

2.2 Simulated Annealing

Simulated annealing (SA) was proposed by S. Kirkpatrik, C. D. Gelatt and M. P. Vecchi in 1983 [14]. SA is a probabilistic variant of the local search method, but it can, in contrast, escape local optima [15].

SA is based on an analogy taken from thermodynamics: to grow a crystal, we start by heating a row of materials to a molten state. Then, we reduce the temperature of this crystal melt gradually, until the crystal structure is frozen in.

A standard SA procedure begins by generating an initial solution at random. At initial stages, a small random change is made in the current solution s_c. Then the objective function value of the new solution s_n is calculated and compared with that of the current solution. A move is made to the new solution if it has better value or if the probability function implemented in SA has a higher value than a randomly generated number.

Otherwise a new solution is generated and evaluated. The probability of accepting a new solution is given as follows:

$$p = \begin{cases} 1 & \text{if} \quad f(s_n) - f(s_c) < 0 \\ exp(\frac{-|f(s_n)-f(s_c)|}{T}) & \text{otherwise} \end{cases} \quad (3)$$

The calculation of this probability relies on a temperature parameter T, which is referred to as temperature, since it plays a similar role as the temperature in the physical annealing process. To avoid getting trapped at a local minimum point, the rate of reduction should be slow. In our problem the following method to reduce the temperature has been used:

$$T_{i+1} = \gamma \ T_i \quad (4)$$

where $i = 0, 1, ...$ and $\gamma = 0.99$.

Thus, at the start of SA most worsening moves may be accepted, but at the end only improving ones are likely to be allowed. This can help the procedure jump out of a local minimum. The algorithm may be terminated after a certain volume fraction of the structure has been reached or after a pre-specified run time.

Additional concepts about SA and some of its applications can easily be found in literature [3, 4].

3 Our PSOSA Hybrid Algorithm

This section presents a new hybrid PSOSA algorithm which combines the advantages of both PSO (that has a strong global-search ability) and SA (that has a strong local-search ability).

This hybrid approach makes full use of the exploration ability of PSO and the exploitation ability of SA and offsets the weaknesses of each other. Consequently, through introducing SA to PSO, the proposed algorithm is capable of escaping from a local optimum. However, if SA is introduced to PSO at each iteration, the computational cost will increase sharply and at the same time the fast convergence ability of PSO may be weakened. In order to perfectly integrate PSO with SA, SA is introduced to PSO every K iterations if no improvement of the global best solution does occur. Therefore, the hybrid PSOSA approach is able to keep fast convergence (most of the time) thanks to PSO, and to escape from a local optimum with the aid of SA. To help PSO to jump out of a local optimum, SA is applied to the best solution in the swarm found so far, each K iterations that is predefined to $270 \sim 500$ according to our experimentations.

The hybrid PSOSA algorithm works as illustrated in Algorithm 1, where:

Description of a particle: Each particle (solution) $X \in S$ is represented by its $n > 0$ components, i.e.,

$X = (x_1, x_2, ..., x_n)$, where $i = 1, 2, ..., n$ and n represents the dimension of the optimization problem to solve.

Initial Swarm: Initial Swarm corresponds to population of particles that will evolve. Each particle x_i is initialized with uniform random value between the lower and upper boundaries of the interval defining the optimization problem.

Evaluate function: Evaluate (or fitness) function in PSOSA algorithm is typically the objective function that we want to minimize in the problem. It serves for each solution to be tested for suitability to the environment under consideration.

Generate function: To generate a $Neighbour$ solution, a small random change is made to the $current\_solution$.

Accept function: Function $Accept(current\_cost, Neighbour\_cost, T)$ is decided by the acceptance probability given by equation 3, which is the probability of accepting configuration $Neighbour$.

4 Experiments results

4.1 Benchmark functions

In order to compare the performance of our hybrid PSOSA algorithm with those described in [1, 2], we use ten benchmark functions [9] described in the next subsections. These functions provide a good launch pad for testing the credibility of an optimization algorithm. For these functions, there are many local optima in their solution spaces. The amount of local optima increases with increasing complexity of the functions, i.e. with increasing dimension. In our experiments, we used 20-dimensional functions except in the case of Himmelblau and Shubert functions that are two-dimensional by definition. In the following, we describe in some detail each of the benchmark functions used in our study.

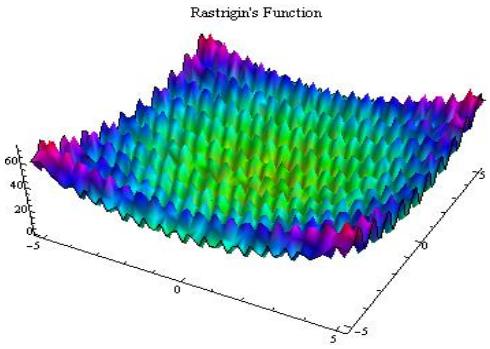

Figure 1. Two-dimensional Rastrigin's function.

```
1  iter ←0, cpt ← 0,
2  Initialize swarm_size particles
3  stop_criterion←maximum iterations or
   Optimal_solution is not attained
4  while Not stop_criterion do
5      for each particle i ← 1 to swarm_size do
6          Evaluate(particle(i))
7          if the fitness value is better than the best fitness
           value (cbest) in history then
8              Update current value as the new cbest.
9          end
10     end
11     Choose the particle with the best fitness value in the
       neighborhood (gbest)
12     for each particle i ← 1 to swarm_size do
13         Update particle velocity according equation (1)
14         Enforce velocity bounds
15         Update particle position according equation (2)
16         Enforce particle bounds
17     end
18     if there is no improvement of global best solution
       then
19         cpt ← cpt + 1
20     else
21         Update global best solution
22         cpt ← 0
23     end
24     if cpt = Kmax then
25         cpt ← 0
26         //Apply SA to global best solution
27         iterSA ← 0
28         initialize(T)
29         stop_criterion ←maximum iterations or
           Optimal_solution is not attained
30         current_solution ← global_best_solution
31         current_cost ←
           Evaluate(current_solution)
32         while Not stop_criterion do
33             while inner-loop stop criterion do
34                 Neighbour ←
                   Generate(current_solution)
35                 Neighbour_cost ←
                   Evaluate(Neighbour)
36                 if Accept(current_cost,
                   Neighbour_cost, T) then
37                     current_solution ←
                       Neighbour
38                     current_cost ←
                       Neighbour_cost
39                 end
40                 iterSA ← iterSA + 1
41                 Update (global_best_solution)
42             end
43             Update(T) according equation 4
44             Update (stop_criterion)
45         end
46     end
47 end
48 iter ← iter + 1
49 Update (stop_criterion)
```

Algorithm 1: PSOSA Hybrid Algorithm.

4.1.1 Rastrigin's Function

Rastrigin's function is mathematically defined as:

$$F_1(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

where $\vec{x}$ is an n-dimensional vector located within the range $[-5.12; 5.12]^n$. The location of local minima is regularly distributed. The global minimum is located at the origin and its function value is zero.

Figure 1 shows a two-dimensional Rastrigin function.

4.1.2 Sphere's Function

The Sphere function is a highly convex, unimodal test function, and is defined as:

$$F_2(x) = \sum_{i=1}^{n} x_i^2$$

where $\vec{x}$ is an n-dimensional vector located within the range $[-5.12; 5.12]^n$. The global minimum is located at the origin with a function value of zero.

Figure 2 shows a two-dimensional Sphere function.

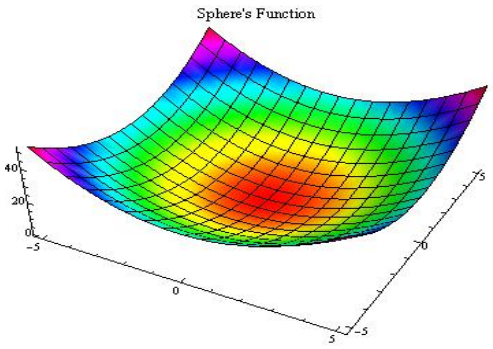

Figure 2. Two-dimensional Sphere's function.

4.1.3 Griewank's Function

The mathematical formula that defines Griewank's function is:

$$F_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$$

where $\vec{x}$ is an n-dimensional vector located within the range $[-600.0; 600.0]^n$. The location of local minima is regularly distributed. The global minimum is located at the origin and its value is zero.

Figure 3 shows a two-dimensional Griewank's function.

4.1.4 Rosenbrock's Function

Rosenbrock's function is mathematically defined as:

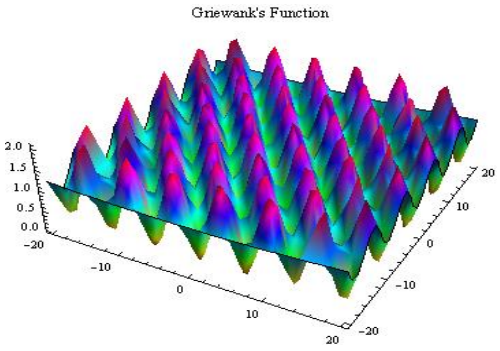

Figure 3. Two-dimensional Griewank's function.

$$F_4(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$

where $\vec{x}$ is an n-dimensional vector located within the range $[-30.0; 30.0]^n$. The global minimum is located at $(1, ..., 1)$ with a function value of zero.

This function exhibits a parabolic-shaped deep valley. To find the valley is trivial, but to achieve convergence to the global minimum is a difficult task.

In the optimization literature it is considered a difficult problem due to the nonlinear interaction between variables [16]. Figure 4 shows a two-dimensional Rosenbrock's function.

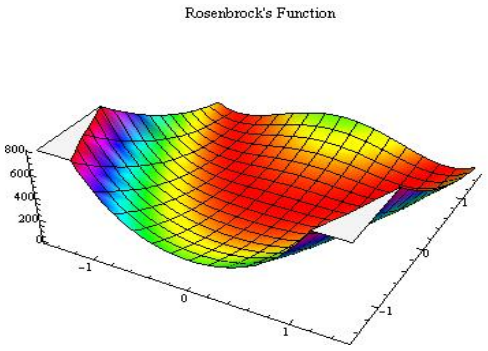

Figure 4. Two-dimensional Rosenbrock's function.

4.1.5 Noisy Function

Noisy function is mathematically defined as:

$$F_5(x) = (\sum_{i=1}^{n}(i+1)x_i^4) + rand[0, 1]$$

where $\vec{x}$ is an n-dimensional vector located within the range $[-1.28; 1.28]^n$. Function value of the global minimum is zero.

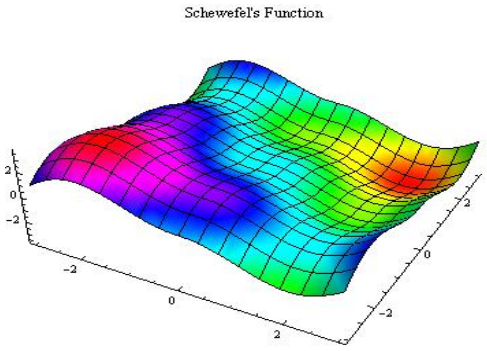

Figure 5. Two-dimensional Schwefel's function.

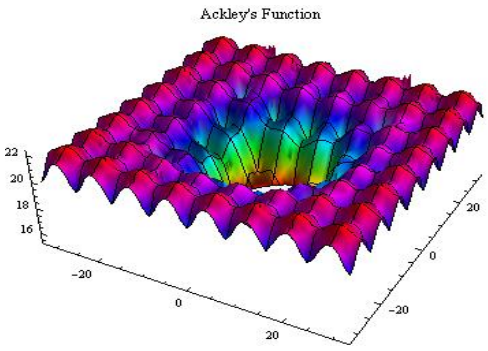

Figure 6. Two-dimensional Ackley's function.

4.1.6 Schwefel's Function

This function is also known as Schwefel's sine root function. It is defined as:

$$F_6(x) = 418.9829\ n - \sum_{i=1}^{n}(x_i \sin(\sqrt{|x_i|})$$

or

$$f_6(x) = \sum_{i=1}^{n}(x_i \sin(\sqrt{|x_i|}) = -418.9829\ n$$

where $\vec{x}$ is an n-dimensional vector located within the range $[-500.0; 500.0]^n$.

The surface of this function is composed of a great number of peaks and valleys. The main difficulty of this function is that the second best minimum is very far from the global optimum where many search algorithms are trapped. Moreover, the global minimum is near the bounds of the domain. The search algorithms are potentially prone to convergence in the wrong direction in the optimization of this function. Function value of the global minimum of Schwefel's function is zero (or, for $f6$ the global minimum is $-418.9829 \times 20 = -8379.658$).

Figure 5 shows a two-dimensional Schwefel's function.

4.1.7 Ackley's Function

Ackley's function, generalised to n dimensions by Bäck [17], is defined as:

$$F_7(x) = 20 + e - 20\ e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$$

where $\vec{x}$ is an n-dimensional vector that is normally located within the range $[-32.0; 32.0]^n$. Ackley's function is a highly multimodal function with regularly distributed local optima. The global minimum is located at the origin and its value is zero.

Figure 6 shows a two-dimensional Ackley's function.

4.1.8 Michalewicz's Function

Michalewicz's function is a parameterized, multi-modal function with many local optima ($n!$) located between plateaus. The higher the parameter m, the more difficult it is to find the global minimum. In our experiments, m is set to 10. The mathematical formula that defines the Michalewicz's function is:

$$F_8(x) = -\sum_{i=1}^{n}\sin(x_i)\sin^{2m}(\frac{i - x_i^2}{\pi})$$

where $\vec{x}$ is an n-dimensional vector that is normally located within the range $[-\pi; \pi]^n$.

Figure 7 shows a two-dimensional Michalewicz's function.

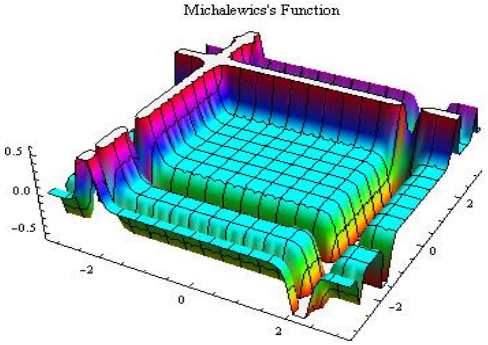

Figure 7. Two-dimensional Michalewicz's function.

4.1.9 Himmelblau's Function

In mathematical optimization, Himmelblau's function is a multi-modal function, used to test the performance of optimization algorithms. The function is defined by:

$$F_9(x) = (x_2 + x_1^2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + x_1$$

where $-5 \leq x_i \leq 5$ for i = 1, 2. Function of the global minimum of Himmelblau's function is -3.78396.

Figure 8 shows a two-dimensional Himmelblau's function.

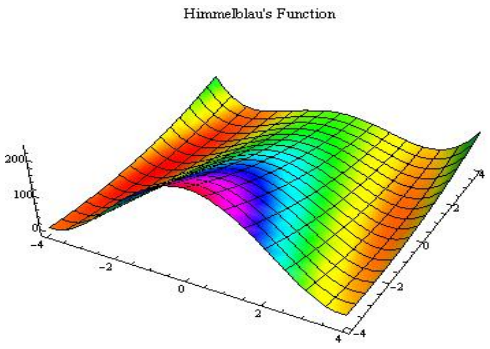

Figure 8. Two-dimensional Himmelblau's function.

4.1.10 Shubert's Function

This is a very interesting function which can be defined as:

$$F_{10}(x) = \sum_{i=1}^{5} i\cos((i+1)x_1 + i) \sum_{i=1}^{5} i\cos((i+1)x_2 + i)$$

where $-10 \leq x_i \leq 10$ for i = 1, 2. It has 760 local minima, 18 of which are global minima with function value of -186.7309. Moreover, the global optima are unevenly distributed.

Figure 9 shows a two-dimensional Shubert's function.

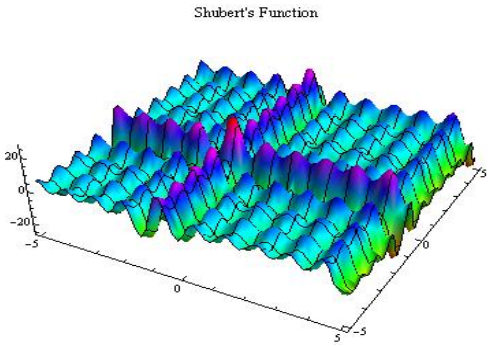

Figure 9. Two-dimensional Shubert's function.

4.2 Simulation Results and Discussions

To testify the efficiency and effectiveness of PSOSA hybrid algorithm, the experimental results of our approach are compared with those obtained by [2, 1]. Note that our PSOSA hybrid algorithm is written in C++. The C++ compiler used is gcc version 2.95.2 (Dev-cpp). The laptop (Intel Core 2 Quad Q9000, 2 GHz CPU, 4 Gb memory) used to run our experimentation is running under Windows Vista x 64 Premium Home Edition.

4.2.1 Comparison with results obtained by using TL-PSO algorithm [2]

In this section we compare our approach with TL-PSO method [2] that is based on combining the excellence of both PSO and Tabu Search.

As described in [2], we apply our algorithm to the four following benchmark problems: Rastrigin's, Schwefel's, Griewank's and Rosenbrock's function.

Here, number of dimension of searching $n = 20$ and iteration number is 2000.

In numerical simulations, the obtained results are shown in Table 1. These results indicate the Mean, Best, and Worst values obtained under the same condition of 50 times trials.

By analysing Table 1 we can see that the results obtained by our algorithm are excellent in comparison with those obtained by TL-PSO algorithm.

4.2.2 Comparison with other PSO algorithms described in [1]

Performance of four Particle Swarm Optimization algorithms, namely classical PSO, Attraction-Repulsion based PSO (ATREPSO), Quadratic Interpolation based PSO (QIPSO) and Gaussian Mutation based PSO (GMPSO) are evaluated in [1]. Whereas all the algorithms presented in this paper are guided by the diversity of the population to search the global optimal solution of a given optimization problem, GMPSO uses the concept of mutation and QIPSO uses the reproduction operator to generate a new member of the swarm.

In order to make a fair comparison of classical PSO, ATREPSO, QIPSO, GMPSO and our PSOSA approach we fixed, as indicated in [1], the same seed for random number generation so that the initial swarm population is same for all the five algorithms. The number of particles in the swarm is 30. The four first algorithms use a linearly decreasing inertia weight ω which starts at 0.9 and ends at 0.4, with the user defined parameters $c_1 = c_2 = 2.0$. PSOSA uses inertia weight $\omega = 0.6$ and $c_1 = c_2 = 2.0$.

For each algorithm, the maximum number of iterations allowed was set to 10000. A total of 30 runs for each experimental setting were conducted and the average fitness of the best solutions throughout the run was recorded.

The mean solution and the standard deviation [1] found by the five algorithms are listed in Table 2.

The numerical results given in Table 2 show that:

- all the algorithms outperform the classical Particle Swarm Optimization.
- PSOSA algorithm gives much better performances in comparison to PSO, QIPSO, ATREPSO and GMPSO, out of the example of Sphere's and Ackley's functions.

[1] Note that the standard deviation indicates the stability of the algorithms

- On Sphere's function, QIPSO obtains better results than those obtained by PSOSA approach. But when the maximum number of iterations is fixed to 1.5×10^6, PSOSA obtains the optimal value.

- The analysis of the results obtained for Ackley's function shows that QIPSO obtains better mean results than PSOSA algorithm. But, PSOSA has a much smaller standard deviation.

4.2.3 Engineering design problems

This section presents the performances of all algorithms obtained on two engineering design problems common in the field of mechanical and electrical engineering. A brief description of these problems is given below.

Problem 1: Design of a Gear Train [1, 19]
The compound Gear Train problem shown in Figure 10 was introduced by Sandgren [10]. It is to be designed such that the Gear ratio is as close as possible to $1/6.931$. For each Gear i the number of teeth x_i must be between 12 and 60. Since the number of teeth is to be an integer, the variables must be integers. The mathematical model of gear train design is given by,

$$Minimize\ \ G(x) = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right)$$

where x_1, x_2, x_3 and x_4 are the integer number, $\in\ [12, 60]$, of teeth on gears A, B, D and F respectively.

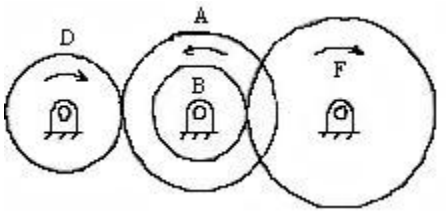

Figure 10. Compound Gear Train.

Problem 2: Transistor Modelling [1, 11]
The mathematical model of the Transistor design is given by,

$$Minimize\ T(x) = \gamma^2 + \sum_{i=1}^{4}(\alpha_i^2 + \beta_i^2)$$

where:

- $\alpha_i = (1 - x_1 x_2)x_3(exp[x_5(g_{1i} - g_{3i}x_7 10^{-3} - g_{5i}x_8 10^{-3})] - 1) - g_{5i} + g_{4i}x_2$
- $\beta_i = (1 - x_1 x_2)x_4(exp[x_6(g_{1i} - g_{2i} - g_{3i}x_7 10^{-3} + g_{4i}x_9 10^{-3})] - 1) - g_{5i}x_1 + g_{4i}$
- $x_i \geq 0$
- g_{ki} are the numerical constants given by the matrix:

$$\begin{pmatrix} 0.485 & 0.752 & 0.869 & 0.982 \\ 0.369 & 1.254 & 0.703 & 1.455 \\ 5.2095 & 10.0677 & 22.9274 & 20.2153 \\ 23.3037 & 101.779 & 111.461 & 191.267 \\ 28.5132 & 111.8467 & 134.3884 & 211.4823 \end{pmatrix}$$

The objective function $T(x)$ provides a least-sum-of-squares approach to the solution of a set of nine simultaneous nonlinear equations, which arise in the context of Transistor Modelling.

To solve these two problems, the number of particles in the swarm is taken as 20. The maximum number of generations allowed for Gear Train Design and Transistor Modelling is set to 2000 and 5000 respectively (as in [1]).

Table 3 shows the results for Gear Train Design. The examination of this table shows that PSOSA, GMPSO and ATREPSO algorithms obtain the best results in comparison with those obtained by QIPSO and classical PSO.

Table 4 shows the experimental results of Transistor Modelling problem. As can be observed from this table, PSOSA algorithm outperforms the other four algorithms.

5 Conclusion

In this paper, we have designed a hybrid algorithm (PSOSA) that combines the exploration ability of PSO with the exploitation ability of SA, and is capable of preventing premature convergence. Compared with QIPSO, ATREPSO and GMPSO [1] and Tl-PSO [2] on ten well-known benchmark functions, and two engineering problems, it has been shown that our approach has better performances in terms of accuracy, convergence rate, stability and robustness. In future work, we will compare PSOSA algorithm with other hybrid algorithms (PSO-GA, PSO-MDP, PSO-TS) whose design is in progress by the authors. Comparison will also be done on additional benchmark functions and more complex problems including functions with dimensionality larger than 20.

	Mean		Best		Worst	
Function	TL-PSO	**PSOSA**	TL-PSO	**PSOSA**	TL-PSO	**PSOSA**
Rastrigin	8.7161	**5.23034e-05**	4.0589	**4.35994e-09**	17.053	**8.15835e-04**
Schwefel(F6)	445.8830	**206.693255**	0.6748	**0**	829.4431	**355.318**
Griewank	0.0228	**1.18294e-03**	$1.0507e-5$	**1.2299e-15**	0.0739	**0.0172263**
Rosenbrock	19.5580	**0.58359742**	0.1331	**0.0442625**	71.5439	**1.36171**

Table 1. Comparison of our hybrid PSOSA algorithm with TL-PSO approach [2] (Max iteration = 2000 and $n = 20$).

Function	*PSO*	*QIPSO*	*ATREPSO*	*GMPSO*	**PSOSA**
Rastrigin	22.339158	11.946888	19.425979	20.079185	**0**
	15.932042	9.161526	14.349046	13.700202	**0**
Sphere	$1.167749e-45$	0.000000	$4,000289e-17$	$7,263579e-17$	**5.3656e-32**
	5.222331e-46	0.000000	0.000246	6.188854e-17	**2.98492e-31**
Griewank	0.031646	0.011580	0.025158	0.024462	**3,32255e-20**
	0.025322	0.012850	0.028140	0.039304	**2.68415e-20**
Rosenbrock	22.191725	8.939011	19.490820	14.159547	**0,227048188**
	1.615544e+04	3.106359	3.964335e+04	4.335439e+04	**0.243978057**
Noisy	8.681602	0.451109	8.046617	7.160675	**0,002019998**
	9.001534	0.328623	8.862385	7.665802	**0.000650347**
Schwefel (f6)	-6178.559896	-6355.586640	-6183.677600	-6047.670898	**-8379.658**
	4.893329e+02	477.532584	469.611104	482.926738	**2.20425e-19**
Ackley	$3.483903e-18$	$2.461811e-24$	0.018493	$1.474933e-18$	**7.43546e-16**
	8.359535e-19	0.014425	0.014747	1.153709e-08	**1.09382e-15**
Michalewizc	-18.159400	-18.469600	-18.982900	-18.399800	**-19,62555806**
	1.051050	0.092966	0.272579	0.403722	**0.00926944**
Himmelblau	-3.331488	-3.783961	-3.751458	-3.460233	**-3,78396**
	1.243290	0.190394	0.174460	0.457820	**3.16001e-15**
Shubert	-186.730941	-186.730942	-186.730941	-186.730942	**-186,730942**
	1.424154e-05	0.000000	1.424154e-05	1.525879e-05	**8.66746e-14**

Table 2. Comparison of mean/standard deviation of solutions obtained by using our hybrid PSOSA algorithm and approaches described in [1] (Max iteration = 10000 and $n = 20$).

Item	*BPSO*	*QIPSO*	*ATREPSO*	*GMPSO*	**PSOSA**
x_1	13	15	19	19	19
x_2	31	26	46	46	46
x_3	57	51	43	43	43
x_4	49	53	49	49	49
$G(x)$	$9,98e-11$	$2,33e-11$	$2,70086e-12$	$2,70086e-12$	$2,70086e-12$
GearRatio	0,14429	0,14428	0,14428	0,14428	0,14428

Table 3. Best results obtained in Gear Train Design. Comparison of our hybrid PSOSA algorithm with approaches described in [1].

Item	*BPSO*	*QIPSO*	*ATREPSO*	*GMPSO*	**PSOSA**
x_1	0.901019	0.90104	0.900984	0.90167	0.900038
x_2	0.88419	0.884447	0.886509	0.877089	0.459385
x_3	4.038604	4.004119	4.09284	3.532352	1.01304
x_4	4.148831	4.123703	4.201832	3.672409	2.00485
x_5	5.243638	5.257661	5.214615	5.512315	7.97399
x_6	9.932639	9.997876	9.981726	10.80285	8.063
x_7	0.100944	0.096078	$5,69e-06$	0.56264	4.97205
x_8	1.05991	1.062317	1.061709	1.074696	1.00004
x_9	0.80668	0.796956	0.772014	0.796591	1.98737
$T(x)$	0.069569	0.066326	0.066282	0.06576	**0.000235327**

Table 4. Best results obtained in Transistor Modelling. Comparison of our hybrid PSOSA algorithm with approaches described in [1].

References

[1] Pant M., Thangaraj R. and Abraham A., *Particle Swarm Based Meta-Heuristics for Function Optimization and Engineering Applications*. 7^{th} Computer Information Systems and Industrial Management Applications 2008. Vol. 7, pp. 84 - 90, Issue , 26-28 June 2008.

[2] S. Nakano, A. Ishigame and K. Yasuda, *Consideration of Particle Swarm Optimization Combined with Tabu Search*, Special Issue on "The Electronics, Information and Systems Conference Electronics, Information and Systems Society, I.E.E. of Japan", Special Issue Paper, Vol. 128-C N 7, pp. 1162-1167, July 2008.

[3] A. Al-khedhairi, *Simulated Annealing Metaheuristic for Solving P-Median Problem*. Int. J. Contemp. Math. Sciences, Vol. 3, no. 28, pp. 1357-1365, 2008.

[4] Hui Li; Landa-Silva, D., *Evolutionary Multi-objective Simulated Annealing with adaptive and competitive search direction*. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), IEEE Press, pp. 3310-3317, 2008.

[5] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Academic Press, 2001.

[6] Marco A. Montes de Oca and Thomas Sttzle. *Convergence behavior of the fully informed particle swarm optimization algorithm*. Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 71-78, 2008.

[7] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, Chichester, West Sussex, England, 2005.

[8] R. Poli, J. Kennedy, and T. Blackwell. *Particle swarm optimization. An overview*. Swarm Intelligence, vol. 1, N 1, pp. 3357, 2007.

[9] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. Technical Report 2005005, Nanyang Technological University, Singapore and IIT Kanpur, India, May 2005.

[10] Sandgren E, *Nonlinear Integer and Discrete Programming in Mechanical Design*, In Proc. of the ASME Design Technology Conference, Kissimme,Fl, pp. 95-105, 1998.

[11] Price W. L., *A Controlled Random Search Procedure for Global Optimization*, In L. C. W. Dixon and G. P. Szego, eds., Towards Global Optimization 2, Amsterdam, Holland: North Holland Publishing Company, pp. 71-84, 1978.

[12] Kennedy, J. and Eberhart, R. C., *Particle swarm optimization*. In Proceedings of the IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.

[13] Shi, Y. and Eberhart, R. C. *A modified particle swarm optimizer*. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ, pp. 69-73, 1998

[14] S. Kirkpatrick and C. D. Gelatt and Jr. and M. P. Vecchi, *Optimization by Simulated Annealing*. Journal of Science, vol. 220, pp. 671-680, 1983.

[15] Z. Michalewicz, and D. B. Fogel, *How to Solve It: Modern Heuristics*. Springer-Verlag Berlin Heidelberg New, 2000.

[16] D. Ortiz-Boyer, C. Herbaś-Martínez, and N. Garciá-Pedrajas. *CIXL2 - A crossover operator for evolutionary algorithms based on population features*. Journal of Artifical Intelligence Research, vol. 24, pp. 1-48, 2005.

[17] T. Bäck. *Evolutionary algorithms in theory and practice*. Oxford University, Press, 1996.

[18] Mudassar Iqbal and Marco A. Montes de Oca, *An estimation of distribution particle swarm optimization algorithm*, 5^{th} International Workshop, ANTS 2006, Brussels, Vol. 4150/2006 of LNCS, Springer Verlag, pp. 72-83, August 2006.

[19] Kannan B. K. and Kramer S. N, *An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design*, Journal of Mechanical Design. Vol. 116, pp. 405-411, 1994.