X. Wang, X. Z. Gao, and S. J. Ovaska. 2008. A novel particle swarm-based method for nonlinear function optimization. International Journal of Computational Intelligence Research, volume 4, number 3, pages 281-289.

International Journal of Computational Intelligence Research.
ISSN 0974-1259 Vol.4, No.3 (2008), pp. 281–289

A Novel Particle Swarm-based Method for Nonlinear Function Optimization

X. Wang[1], X.Z. Gao[1] and S.J. Ovaska[1]
[1] Department of Electrical Engineering, Helsinki University of Technology,
Otakaari 5 A, FI-02150 Espoo, Finland

xiaolei@cc.hut.fi, gao@cc.hut.fi, seppo.ovaska@tkk.fi

***Abstract*: This paper proposes a hybrid Particle Swarm Optimization (PSO) method, which is based on the fusion of the PSO, Clonal Selection Algorithm (CSA), and Mind Evolutionary Computation (MEC). The clone function borrowed from the CSA and MEC-characterized similartaxis and dissimilation operations are embedded in the original PSO algorithm. Simulations of nonlinear function optimization are made to compare this hybrid PSO with the regular PSO method. It has been demonstrated that our hybrid optimization algorithm can achieve a better convergence performance, and provide diverse solutions to the multi-model optimization problems.**

***Keywords*:** Clonal Selection Algorithm (CSA), Mind Evolutionary Computation (MEC), Particle Swarm Optimization (PSO), optimization**.**

I. Introduction

During the past decade, we have witnessed a significant growth of research interest in the nature inspired optimization methods, among which the Clonal Selection Algorithm (CSA), Mind Evolutionary Computation (MEC), and Particle Swarm Optimization (PSO) method are three representative examples. As an important constituent of the Artificial Immune System (AIS), the CSA is an emerging optimization approach based on the natural immune mechanism [1] [2]. The MEC is a novel kind of soft computing method oriented from the human mind thinking principles [3]. Inspired by the swarm intelligence from birds, fishes, and even human social behaviors, the PSO method is an evolutionary computation technique developed by Kennedy and Eberhart [4] [5]. Due to its simple principles and low computational complexity, it has been widely applied in such areas as electromagetics optimization [6], optimal circuit design [7], and data clustering [8]. However, the original PSO algorithm has certain drawbacks of slow convergence and premature in multi-model optimization [9]. As we know, fusion of the computational intelligence methods can often provide superior performances over employing them individually [10] [11]. Therefore, in this paper, we are going to study a hybrid PSO method based on merging the CSA, MEC, and PSO together.

Our paper is organized as follows. The principles of the original PSO, CSA, and MEC are first introduced in Section II. Next, we discuss the hybrid PSO algorithm in more details. In Section III, the effectiveness of this new optimization method is demonstrated using multi-model functions. Performance comparison between the original and our hybrid PSO method is also made. Finally, we conclude the paper with some remarks and conclusions in Section IV.

II. Hybrid Particle Swarm Optimization Algorithm

A. Particle Swarm Optimization

It is well known the movement of bird flocks and fish schools is an outcome of the individual efforts to maintain an optimal distance from their neighbors [12]. The implicit rule that they are able to move synchronized without colliding has been studied and simulated by scientists. The phenomena suggest information sharing in the colony can provide an evolutionary advantage. Inspired by these social behaviors of the particles, Kennedy and Eberhart proposed the PSO method to deal with multi-model optimization problems [4]. In the original PSO, the position of each particle in the swarm represents a possible solution. The position and velocity of particle i at iteration n are denoted as x_{id}^n and v_{id}^n, respectively. The new velocity at the next iteration, v_{id}^{n+1}, is calculated using its current velocity v_{id}^n, the distance between the particle's best previous position p_{id}^n and x_{id}^n, as well as the distance between the position of the best particle in the swarm p_{gd}^n and x_{id}^n:

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

where w is the inertia weight, c_1 and c_2 are two positive constants, namely cognitive and social parameters, respectively, and r_1 and r_2 are two random values in the range of $[0,1]$. The above deterministic and probabilistic parameters reflect the effects on the particle positions from both the individual memory and swarm influence. The position of particle i, x_{id}^n, is iteratively updated as follows:

$$x_{id}^{n+1} = x_{id}^n + v_{id}^{n+1}. \qquad (2)$$

The optimal solutions can, thus, be acquired by choosing the best particles in the swarm. From (1) and (2), we observe that during the evolution, each particle learns from not only its own past experiences but also the swarm social behaviors. In other words, taking advantage of collective intelligence is the distinguishing property of the PSO method, which has been proved to be efficient in handling numerous challenging optimization tasks [9]. However, empirical experiments also show that the convergence speed of the PSO often slows down with the growth of iteration steps, because the stagnant particles gradually prevail the whole swarm [6].
Therefore, some improved PSO methods have emerged to overcome such a drawback. For example, a fuzzy logic-based turbulence operator is used to control the velocity of particles [13]. In this paper, we embed the advantageous features borrowed from the CSA and MEC into the original PSO. This hybrid PSO method demonstrates an improved performance in nonlinear function optimization in our simulations.

B. Clonal Selection Algorithm

The CSA is one of the most widely employed AIS approaches [14]. It is based on the Clonal Selection Principle (CSP), which explains how an immune response is mounted, when a non-self antigenic pattern is recognized by the B cells. In the natural immune systems, only the antibodies that can recognize intruding antigens are selected to proliferate by cloning [15]. Hence, the fundamental of the CSA is those cells (antibodies) capable of recognizing the non-self cells (antigens) will proliferate. A basic CSA involves the following iteration steps, as shown in Fig. 1.

1. Initialize the antibody pool including the subset of memory cells (M).
2. Evaluate the fitness of all the individuals (affinity with the antigen) in population P.
3. Select the best candidates (P_r) from population P, according to their fitness.
4. Clone P_r into a temporary pool (C).
5. Generate a mutated antibody pool (C_1). The mutation rate of each individual is *inversely* proportional to its fitness.
6. Evaluate all the antibodies in C_1.
7. Eliminate those antibodies similar to the ones in C, and update C_1.
8. Re-select the individuals with better fitness from C_1 to construct memory set M. Other improved individuals of C_1 can replace certain antibodies with poor fitness in P to maintain the antibody diversity.
9. Return back to Step 2.

A unique mutation operator is used in Step 5, through which the mutated values of individuals are inversely proportional to their fitness by means of choosing different mutation variations. That is to say, the better fitness the antibody has, the less it may change. The similarity among candidates can also affect the convergence speed of the CSA. The idea of antibody suppression inspired by the immune network theory is introduced to eliminate the newly generated antibodies, who are similar to those already existing in the candidate pool (Step 7). With such a diverse antibody pool, the CSA can effectively avoid being trapped into local minima, and provide optimal solutions to multi-model problems. In summary, the antibody clone and fitness-based mutation are the two remarkable characteristics of the CSA. To build a hybrid PSO method with enhanced optimization capability, we incorporate these two useful features into the original PSO algorithm.

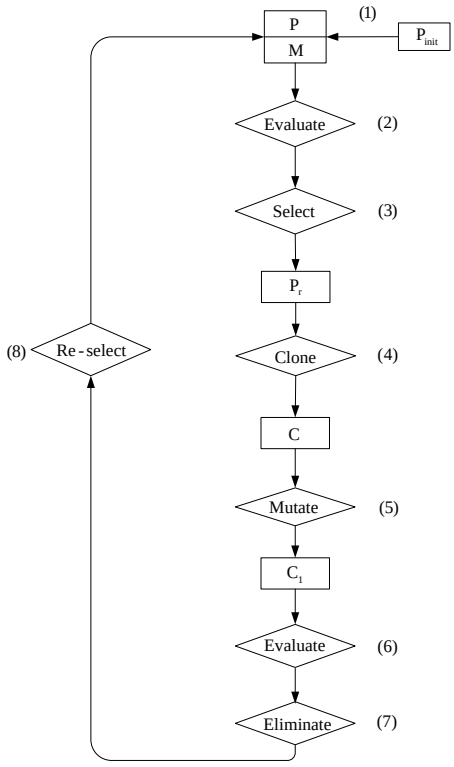


Figure. 1. Diagram of clonal selection algorithm.

C. Mind Evolutionary Computation

The MEC is an evolutionary optimization approach firstly proposed by Sun in 1998 [3]. In the MEC, the whole population of chromosomes is divided into groups. Two billboards, local and global billboards, are used to store the evolution history. The global billboard can record winners in the global competition among groups, while the local billboard is for the local winners among individuals of each group in the local competition. Especially, similartaxis and dissimilation are the two unique operations of the MEC. In the similartaxis, starting from their initial centers, individuals of every group compete against each other in the local areas to be the winners, i.e., local optima. A group is matured, if no new winners appear there. The similartaxis actually serves as ‘exploitation’ in the MEC. On the other hand, the dissimilation is an ‘exploration’ process, in which individuals or groups compete to be the global winners in the whole solution space. The function of this operation is two-fold: 1) some best individuals are chosen as the initial scattering centers of new groups; 2) the current global optima are selected from the local optima of all the groups obtained by the similartaxis. Different from the Genetic Algorithms (GA), there is no separate selection operation in the MEC. In

fact, the selection is implicitly employed in the processes of both the similartaxis and dissimilation.

The iteration procedure of the basic MEC is described as follows. Let S denote the population size, S_G the group size, and N_G the number of groups in the MEC, and there is $S = S_G N_G$.

Step 1. Generate S random chromosomes in the solution space, and select N_G individuals from them as the initial scatter centers of the N_G groups.

Step 2. Perform the similartaxis on these groups, i.e., for every group, $S_G - 1$ chromosomes are scattered based on a preset probability density function around the group center. The resulting S_G individuals are next evaluated, and compared with each other. A local winner is chosen as the new group center of the next generation. Update the local billboard by recording the local winners on it. The above process is repeated until all the groups are matured.

Step 3. Select the best solutions from all the local optima, which are obtained in Step 2. Store them on the global billboard, and expunge certain poor chromosomes from it.

Step 4. If the optimization criterion is satisfied, terminate. Otherwise, return back to Step 2.

Actually, Step 1 initializes the appropriate scattering group centers. Step 2 implements the similartaxis operation, and serves as the local competition, in which a local optimal solution can be found with chromosomes starting from the initial center of each group. Step 3 evaluates these local optima obtained by the similartaxis. Steps 1 and 3 indeed act together as the aforementioned dissimilation operation. More details and variants of the MEC can be found in [3]. Due to the contributions from the distinguishing similartaxis and dissimilation, the MEC has been shown to outperform the GA in the nonlinear multi-dimension function optimization [16]. Therefore, our hybrid PSO method can deploy these two MEC operations to accelerate its convergence.

D. Hybrid Particle Swarm Optimization Method

In this section, we develop a hybrid PSO method based on the fusion of the above PSO, CSA, and MEC. The flow chart of this new optimization algorithm is given in Fig. 2, which can be explained as follows.

1. Initialize the swarm consisting of n particle groups.
2. For each group, generate j particles around the scattering group centers.
3. Clone these particles using affinity-related mutation.
4. Update the velocities of the cloned particles according to (1).
5. Update the positions of the cloned particles according to (2).
6. Evaluate the fitness of each particle in the group, and the best one is considered as the group winner to compete with other groups.
7. If the preset convergence criterion is met, terminate.
8. Select the groups with the best fitness, and randomly generate the supplementary groups to replace the matured ones. Return back to Step 2.

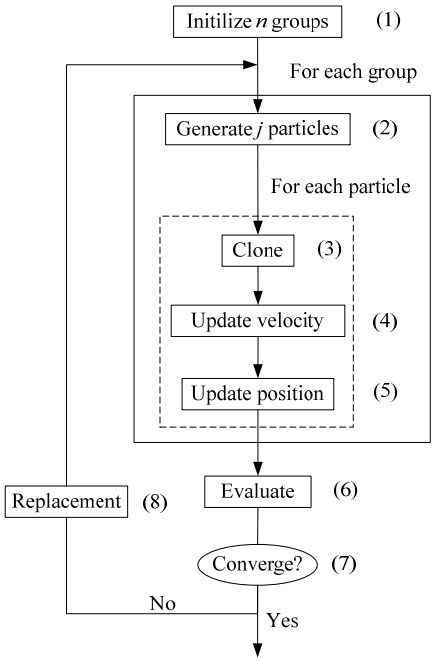


Figure. 2. Hybrid particle swarm optimization algorithm.

Obviously, based on the regular PSO method, our hybrid PSO algorithm utilizes the unique operations from both the MEC and CSA. Like in the MEC, the particles here are divided into competing groups. The similartaxis and dissimilation operations are employed for the local and global search, respectively. This approach brings the features of exploitation in the local space and exploration in the global space to our hybrid optimization method, which can efficiently deal with the well-known premature problem existing in the regular PSO method. The individual particles in this hybrid PSO algorithm are also subjected to the clone operation borrowed from the CSA. The clone size in Step 3 is defined as a monotonic function of the particle affinity [17]. Note, similarly with the CSA, the particle affinity includes the measures between the particles and objective function as well as among the particles. Those particles, whose affinities are lower than a preset threshold, are excluded, and fresh particles can be added thereafter. In other words, self-eliminating the particles with high similarity for introducing affinity-related randomness maintains the diversity of the candidate pool.

In summary, the original PSO method takes advantage of information sharing inside the swarm. The particles learn not only from their own experiences but also the social interaction. However, they often become clustered and trapped in the local search. Embedded with the MEC and CSA, our hybrid PSO algorithm can overcome this drawback, and achieve a better optimization performance. The similartaxis and dissimilation operations from the MEC provide balanced exploitation and exploration in searching for the global optimum. The CSA-based clone operation yields a dynamical particle pool to help finding the optimal as well as diverse solutions. In the next section, we are going to demonstrate the efficiency of our hybrid PSO algorithm in nonlinear function optimization.

III. Simulations

Multi-model function optimization is deployed to verify the proposed hybrid PSO method in this section. As we know, inertia weight w controls the impact of history velocities on the current ones. In the simulations, it is defined as a monotonically decreasing function with regard to the generations:

$$w_i = 1 - \frac{i}{P}, \tag{3}$$

where $i = 1,2,\cdots,P$, and P is the given number of generations. Other PSO parameters, e.g., self confidence c_1 and swarm confidence c_2, are shown in Table 1. A preset threshold discriminates those particles with high similarities, which is chosen to be 0.05. Note, the original and our hybrid PSO algorithms are initialized with the same populations. The former starts from a pool of 30 random particles, and the latter is initialized with five particle groups, in each of which five extra candidates are cloned from the initial group center.

Table 1. Parameters in original and hybrid PSO methods.

Parameters	Values
Swarm Size	30
Self Confidence c_1	0.5
Swarm Confidence c_2	0.5
Group Size	5
Affinity Threshold	0.05
Clone Size	5

Example 1

The first example is a nonlinear function with single input variable [18]:

$$\mathrm{f}(x) = e^{-2(\ln 2)(\frac{x-0.1}{0.8})^2} \left|\sin(5\pi x)\right|, \tag{4}$$

where $x \in [-1,1]$. Actually, this simple function has several local maxima. However, there is only one global maximum, as shown in Fig. 3 (a). The convergence procedures of the original and hybrid PSO methods, represented by dotted and solid lines, respectively, are illustrated in Fig. 3 (b). It is clearly visible our hybrid PSO method has a moderately faster convergence speed than that of the original PSO method. All the ten optimal solutions obtained by the hybrid PSO method are given in Fig. 3 (a) (denoted by 'o') and Table 2, which reveal that this hybrid optimization algorithm can successfully find a group of diverse solutions including the unique global and all the nine local optimal solutions.

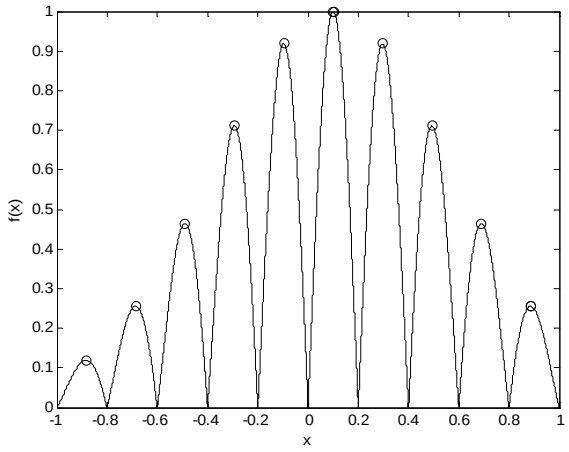
Example 2

The following two-dimension function, as shown in Fig. 4 (a), is used here [9]:

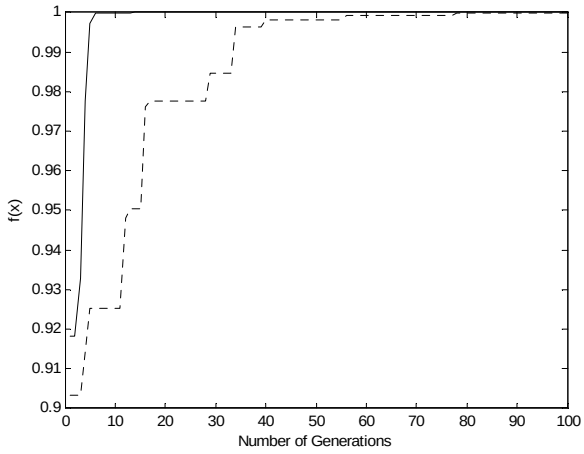
$$\mathrm{f}(x, y) = \cos(x)^2 + \sin(y)^2, \tag{5}$$

where $x \in [-5,5]$, and $y \in [-5,5]$. This function has infinite global maxima in R^2 at points $(m\pi/2, n\pi), m, n = \pm 1, \pm 2 \cdots$.

The iteration behaviors of the two optimization methods are illustrated in Fig. 4 (b). Apparently, both of them can converge in finding the global maxima. However, the hybrid PSO algorithm converges much faster than its counterpart. Furthermore, we examine their multi-model optimization performances. Figure 4 (c) shows the global optima search procedures of these two methods. After about 70 generations, the hybrid PSO method is capable of locating all the 12 global optima, as denoted by '*' in Fig. 4 (d), while the original PSO method can achieve only one, and the others found are only local optima (denoted by 'o'). This simulation example demonstrates that the hybrid PSO algorithm not only converges faster than the original PSO method, but also has a better multi-model optimization capability.

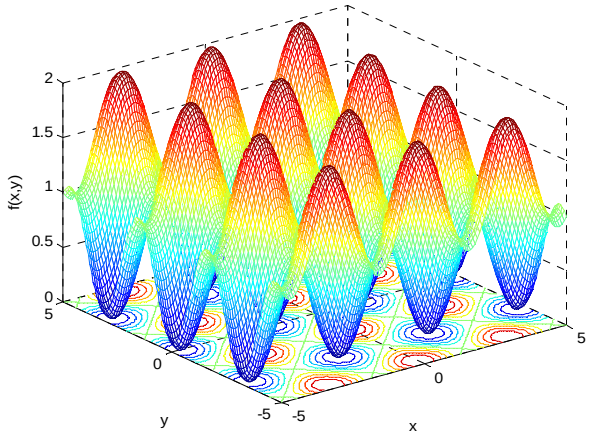


(a)

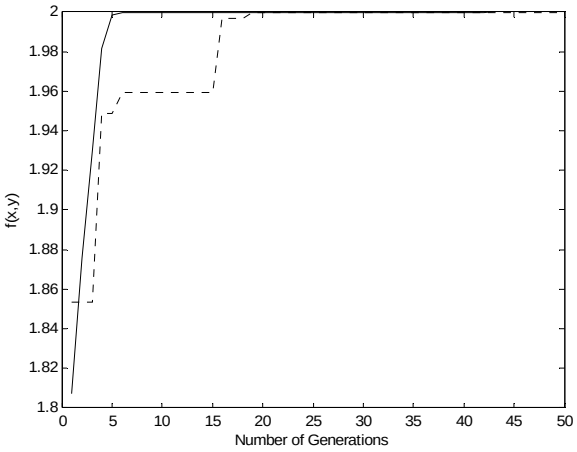


(b)

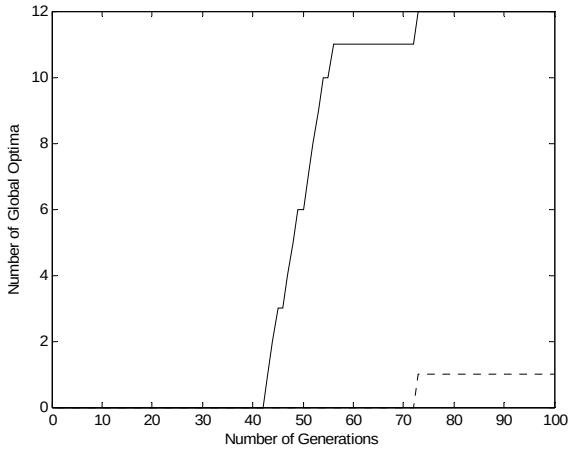
Figure. 3. Optimization of f(x) using original and hybrid PSO methods.
(a) f(x) with optimal solutions ('o') obtained by hybrid PSO method.
(b) convergence procedures of two optimization methods, dotted line: original PSO method, solid line: hybrid PSO method.



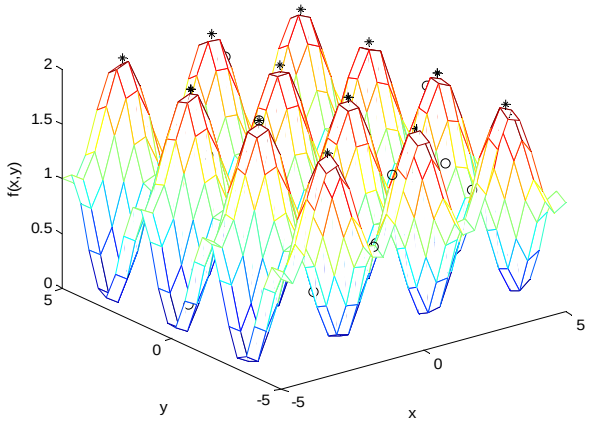
(a)



(b)



(c)



(d)

Figure. 4. Optimization of function (5) using original and hybrid PSO methods.
(a) function (5).
(b) convergence procedures of two optimization methods, dotted line: original PSO method, solid line: hybrid PSO method.
(c) global optima search procedures of optimization methods, dotted line: original PSO method, solid line: hybrid PSO method.
(d) optima obtained by optimization methods, 'o': original PSO method, '*': hybrid PSO method.

Table 2. Optimization results of $f(x)$ achieved by hybrid PSO method.

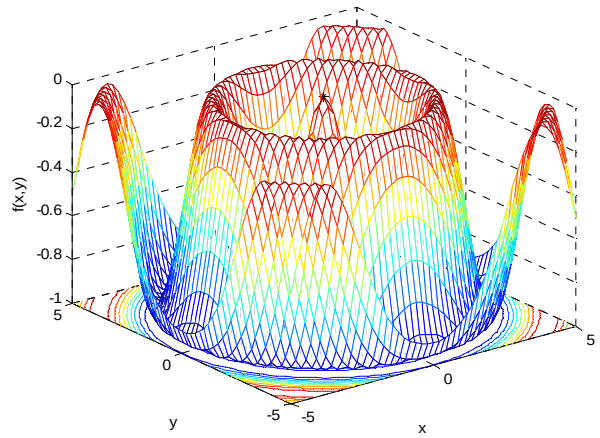
x	$f(x)$	x	$f(x)$
0.1000	1.0000	-0.4897	0.4647
-0.0966	0.9184	0.6897	0.4647
0.2966	0.9184	0.8864	0.2560
-0.2931	0.7113	-0.6864	0.2560
0.4932	0.7113	-0.8831	0.1189

Example 3

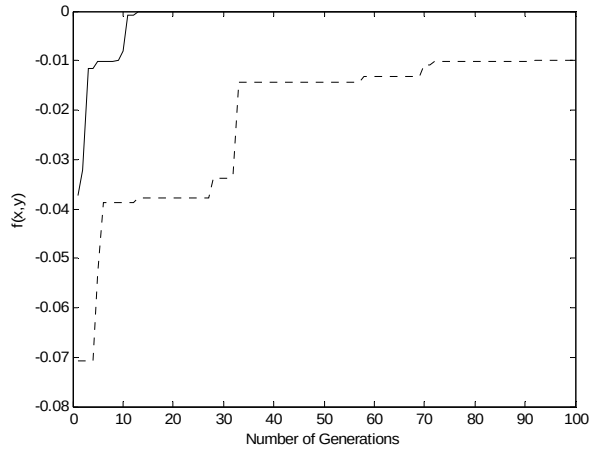
In this example, a more complex function, Scaffer's function, is employed [19]:

$$f(x,y) = -\left\{0.5 + \frac{\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{\left[1+0.001\left(x^2+y^2\right)\right]^2}\right\}, \qquad (6)$$

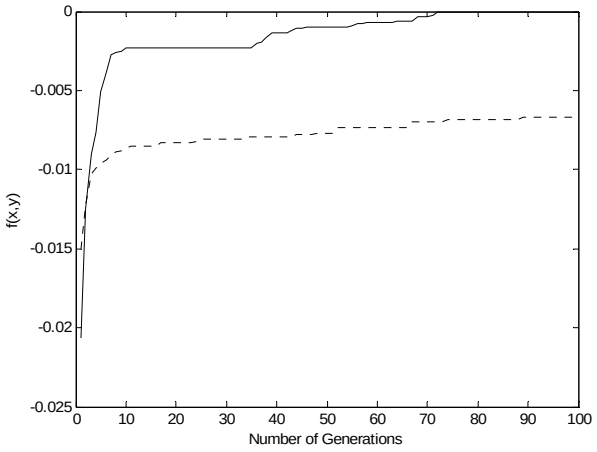
where $x \in [-5,5]$, and $y \in [-5,5]$. Finding the global optimum of Scaffer's function is a challenging task for all optimization methods, because it has only one global optimum at $f(0,0) = 0$ with numerous neighboring local optima within the distance of about 10^{-2}, as shown in Fig. 5



(a)



(b)



(c)

Figure. 5. Optimization of function (6) using original and hybrid PSO methods.
(a) function (6).
(b) convergence procedures of two optimization methods, dotted line: original PSO method, solid line: hybrid PSO method.
(c) statistics optimization results of optimization methods, dotted line: original PSO method, solid line: hybrid PSO method.

(a). Therefore, conventional optimization approaches, such as the GA and PSO, can be easily trapped into these surrounding local optima. Figure 5 (b) illustrates the convergence procedures of the original and hybrid PSO methods, which depicts that the latter can successfully locate the single global optimum, while the former indeed becomes stuck at a certain local optimum. To make statistics comparisons, we run both two optimization algorithms for 100 times, and their average optimal results *vs.* number of generations is given in Fig. 5 (c). This clearly reveals that our hybrid PSO method can always achieve the global optimum of Scaffer's function, but the original PSO algorithm usually fluctuates around one of its local optima.

Example 4

To further examine our hybrid PSO algorithm, we deploy the following ten *n*-dimension functions in this example, which have been widely used as the optimization benchmarks [19].

Sphere function:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 . \tag{7}$$

Rosenbrock function:

$$f_2(x) = \sum_{i=1}^{n} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]. \tag{8}$$

Rastrigin function:

$$f_3(x) = \sum_{i=1}^{n} x_i^2 + 10 - 10\cos(2\pi x_i) . \tag{9}$$

Griewank function:

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1 . \tag{10}$$

Dixon & Price function:

$$f_5(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(x_i^2 - x_{i-1})^2 . \tag{11}$$

Zakharov function:

$$f_6(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i \right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i \right)^4 . \tag{12}$$

Sum Squares function:

$$f_7(x) = \sum_{i=1}^{n} i x_i^2 . \tag{13}$$

Levy function:

$$\begin{aligned} f_8(x) &= \sin^2(\pi y_1) + z_1 + z_2, \\ y_i &= 1 + \frac{x_i - 1}{4}, i = 1,2,\cdots,n. \\ z_1 &= \sum_{i=1}^{n-1} \left\{ (y_i - 1)^2 \left[1 + 10\sin^2(\pi y_1 + 1) \right] \right\} \\ z_2 &= (y_n - 1)^2 \left[1 + 10\sin^2(\pi y_n) \right] \end{aligned} . \tag{14}$$

Schwefel function:

$$f_9(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| . \tag{15}$$

Quadric function:

$$f_{10}(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2 . \quad (16)$$

The details of these unconstrained functions are given in Table 3. A constriction factor χ is introduced to weight the velocities in updating the particle positions [9]:

$$x_{id}^{n+1} = x_{id}^{n} + \chi v_{id}^{n+1} . \quad (17)$$

We choose $\chi = 5$ here based on *trial and error*. As a representative example, the minimization of a 20-dimension Rosenbrock function is first employed for verifying the original and hybrid PSO methods. Figure 6 illustrates their convergence procedures in 400 generations. Apparently, our hybrid PSO can converge significantly faster than the original PSO method in this high-dimension function optimization case.

Table 3. Details of benchmark functions.

References	Search range	Global optima
Spherical	$[-100, 100]^n$	$f(x) = 0$
Rosenbrock	$[-50, 50]^n$	$f(x) = 0$
Rastrigin	$[-5.12, 5.12]^n$	$f(x) = 0$
Griewank	$[-600, 600]^n$	$f(x) = 0$
Dixon & Price	$[-10, 10]^n$	$f(x) = 0$
Zakharov	$[-10, 10]^n$	$f(x) = 0$
Sum Squares	$[-10, 10]^n$	$f(x) = 0$
Levy	$[-10, 10]^n$	$f(x) = 0$
Schwefel	$[-10, 10]^n$	$f(x) = 0$
Quadric	$[-100, 100]^n$	$f(x) = 0$

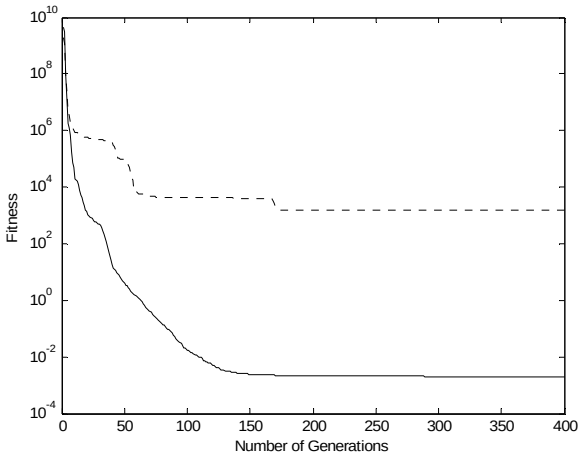


Figure. 6. Convergence procedures of 20-dimension Rosenbrock function optimization using original and hybrid PSO methods,
dotted line: original PSO method,
solid line: hybrid PSO method.

Moreover, we run the original and hybrid PSO methods for 10 times for the optimization of all the above ten benchmark functions with the dimensions of 10, 30, and 50, respectively.

Table 4. Average optima of benchmark functions with different dimensions obtained by original and hybrid PSO methods.

Dimensions	10	30	50
Sphere Function			
Original PSO	8.9×10^{-11} (6.7×10^{-11})	1.0×10^{-7} (6.7×10^{-8})	4.7×10^{-3} (9.8×10^{-4})
Hybrid PSO	4.8×10^{-24} (3.7×10^{-25})	5.7×10^{-18} (9.6×10^{-20})	9.7×10^{-11} (3.5×10^{-11})
Rosenbrock Function			
Original PSO	2.2 (0.2)	2.9×10^{2} (1.4×10^{2})	2.2×10^{5} (4.9×10^{3})
Hybrid PSO	2.9×10^{-11} (3.7×10^{-12})	1.9×10^{-2} (1.0×10^{-3})	4.1 (0.7)
Rastrigin Function			
Original PSO	1.7×10^{-1} (3.9×10^{-2})	20.9 (5.0)	45.5 (10)
Hybrid PSO	1.0×10^{-7} (9.4×10^{-8})	2.3×10^{-2} (1.0×10^{-3})	1.1 (0.4)
Griewank Function			
Original PSO	3.7×10^{-8} (2.2×10^{-8})	7.1×10^{-2} (6.8×10^{-3})	1.4×10^{-1} (4.9×10^{-2})
Hybrid PSO	2.5×10^{-24} (1.8×10^{-25})	2.5×10^{-17} (1.9×10^{-18})	2.5×10^{-6} (3.7×10^{-7})
Dixon & Price Function			
Original PSO	1.3×10^{-4} (6.6×10^{-5})	2.8×10^{-1} (1.5×10^{-1})	1.9 (0.8)
Hybrid PSO	2.8×10^{-14} (2.0×10^{-15})	6.7×10^{-8} (5.5×10^{-10})	6.7×10^{-4} (5.9×10^{-5})
Zakharov Function			
Original PSO	4.0×10^{-7} (3.7×10^{-8})	4.9×10^{-2} (1.5×10^{-2})	3.8 (0.7)
Hybrid PSO	4.8×10^{-18} (3.8×10^{-18})	1.0×10^{-6} (3.6×10^{-7})	1.7×10^{-4} (2.0×10^{-5})
Sum Squares Function			
Original PSO	1.9×10^{-9} (1.1×10^{-10})	2.0×10^{-2} (2.9×10^{-3})	0.8 (1.9×10^{-1})
Hybrid PSO	2.0×10^{-17} (1.0×10^{-18})	3.4×10^{-7} (1.9×10^{-7})	1.6×10^{-3} (4.9×10^{-4})
Levy Function			
Original PSO	2.7×10^{-1} (7.0×10^{-2})	4.7 (2.6)	4.2 (0.6)
Hybrid PSO	7.0×10^{-10} (6.2×10^{-11})	1.2×10^{-6} (4.0×10^{-7})	2.9×10^{-2} (9.0×10^{-3})
Schwefel Function			
Original PSO	1.8×10^{-3} (4.2×10^{-4})	3.2 (0.2)	9.1 (1.3)
Hybrid PSO	1.0×10^{-8} (2.7×10^{-9})	0.2×10^{-3} (1.8×10^{-5})	1.0×10^{-1} (0.3×10^{-3})
Quadric Function			
Original PSO	2.0×10^{-5} (3.7×10^{-6})	2.1×10^{-3} (1.4×10^{-4})	7.2×10^{-1} (2.3×10^{-2})
Hybrid PSO	4.3×10^{-15} (2.7×10^{-15})	1.1×10^{-7} (2.6×10^{-8})	3.0×10^{-5} (2.3×10^{-6})

The average optima and standard derivations (in brackets) obtained by the two PSO methods after 1,000 generations are given in Table 4. We can observe that with regard to low-dimension and simple functions, such as 10-dimension Sphere function and 10-dimension Griewank function, they achieve similar optimization performances. However, our hybrid PSO method can obviously outperform the original PSO algorithm in case of high-dimension functions, e.g., 50-dimension Griewank function. We also emphasize that due to its diversity of solutions, the proposed hybrid optimization method has the superior performances for the multi-optima function (such as Griewank function and Schwefel function) optimization over other modified PSO methods [13].

IV. Conclusions

In this paper, a hybrid PSO method based on the fusion of the PSO, CSA, and MEC is first proposed, and further employed for nonlinear function optimization. The information sharing capability of the PSO algorithm, solution diversity feature of the CSA, as well as anti-premature function of the MEC are fully utilized and combined in the new algorithm. Simulation results have demonstrated that our hybrid PSO method achieves an enhanced optimization performance over the original PSO algorithm. It can also provide diverse and flexible solutions to multi-model problems. How to apply this novel optimization scheme for coping with more real-world engineering tasks is currently under exploration.

Acknowledgements

This research work was funded by the Academy of Finland under Grants 214144 and 124721. The authors would like to thank the anonymous reviewers for their insightful comments and constructive suggestions that have improved the paper.

References

[1] L. N. de Castro and F. J. von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239-251, June 2002.

[2] A. Freitas and J. Timmis, "Revisiting the foundations of artificial immune systems for data mining," *IEEE Transactions on Evolutionary computation*, vol. 11, no. 4, pp. 521-540, August 2007.

[3] C. Sun, Y. Sun, and W. Wang, "A survey of MEC: 1998-2001," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia, October 2003, pp. 648-656.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, December 1995, pp. 1942-1945.

[5] S. Ho, S. Yang, G. Ni, and K. Wong, "An improved PSO method with application to multimodal function of inverse problems," *IEEE Transactions on Magnetics*, vol. 43, no. 4, pp. 1597-1600, April 2007.

[6] G. Ciuprina, D. Ioan, and I. Munteanu, "Use of intelligent-particle swarm optimization in electromagetics," *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 1037-1040, March 2002.

[7] P. Tawdross and A. Konig, "Investigation of particle swarm optimization for dynamic reconfiguration of field-programmable analog circuits," in *Proceedings of the International Conference on Hybrid Intelligent Systems*, Rio de Janeiro, Brazil, November 2005, pp. 259-264.

[8] V. D. Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Proceedings of the IEEE* Congress *on Evolutionary Computation*, Canberra, Australia, June 2003, pp. 215-220.

[9] K. Parsopoulos and M. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, no. 2-3, pp. 235-306, 2002.

[10] X. Wang, X. Z. Gao, and S. J. Ovaska, "A hybrid optimization algorithm in power filter design," in *Proceedings of the 31st Annual Conference of the IEEE Industrial Electronics Society*, Raleigh, NC, November 2005, pp. 1335-1340.

[11] X. Wang, X. Z. Gao, and S. J. Ovaska, "A hybrid optimization algorithm based on ant colony and immune principles," *International Journal of Computer Science and Applications*, vol. 4, no. 3, pp. 30-44, 2007.

[12] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Seoul, Korea, May 2001, pp. 81-86.

[13] H. Liu, A. Abraham, and W. Zhang, "A fuzzy adaptive turbulent particle swarm optimization," *International Journal of Innovative Computing and Applications*, vol. 1, no. 1, pp. 39-47, 2007.

[14] X. Wang, X. Z. Gao, and S. J. Ovaska, "Artificial immune optimization methods and applications – A survey," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, The Hague, The Netherlands, October 2004, pp. 3415-3420.

[15] X. Wang, "Clonal selection algorithm in power filter optimization," in *Proceedings of the IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*, Espoo, Finland, June 2005, pp. 122-127.

[16] C. Sun, X. Qi, and O. Li, "Pareto-MEC for multi-objective optimization," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia, October 2003, pp. 5045-5049.

[17] A. Acan, "Clonal selection algorithm with operator multiplicity," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Portland, OR, June 2004, pp. 1909-1915.

[18] J. Wang, Y. Sun, and C. Sun, "Comparison of performance of basic MEC and DC niching GAs," in *Proceedings of the 4th IEEE* Congress *on Intelligent*

Control and Automation, Shanghai, China, June 2002, pp. 2297-2303.

[19] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 35, no. 6, pp. 1272-1282, December 2005.