X. Wang, X. Z. Gao, and S. J. Ovaska. 2007. A hybrid optimization algorithm based on ant colony and immune principles. International Journal of Computer Science & Applications, volume 4, number 3, pages 30-44.

Also appeared online with TMRF (International Journal of Computer Science and Applications, ISSN 0972-9038).

International Journal of Computer Science & Applications
Vol. 4 Issue 3, pp 30-44

A Hybrid Optimization Algorithm Based on Ant Colony and Immune Principles

X. Wang[1] **xiaolei@cc.hut.fi**

X. Z. Gao[1] **gao@cc.hut.fi**

S. J. Ovaska[2] **seppo.ovaska@tkk.fi**

[1] Institute of Intelligent Power Electronics, Helsinki University of Technology, Otakaari 5 A, Espoo, FI-02150 Finland

[2] Department of Electrical and Computer Engineering, Utah State University, 4120 Old Main Hill, Logan, UT 84322-4120 U.S.A.

Abstract

This paper proposes a hybrid optimization method based on the ant colony and clonal selection principles, in which the cloning and mutation operations are embedded in the ant colony to enhance its search capability. The novel algorithm is employed to deal with a few benchmark optimization problems under both static and dynamic environments. Simulation results demonstrate the remarkable advantages of our approach in diverse optimal solutions, closely tracking varying optimum, as well as improved convergence speed.

Keywords: clonal selection algorithm, ant colony algorithm, hybrid optimization method, benchmark problems, dynamic optimization

1 Introduction

Optimization has been a popular research topic for decades. Especially, dynamic optimization problems have objective functions changing over time, which causes changes in the position of optima as well as the characteristics of the search space. This leads to the fact that existing optima may disappear, while new optima may appear. Optimization under the dynamic environments is a challenging task that attracts great attention [8]. During recent years, biology-inspired soft computing methods have been widely used in different optimization problem solving cases. For example, the Clonal Selection Algorithm (CSA), an important branch of the Artificial Immune Systems (AIS), takes its inspiration from the clonal selection mechanism that describes the basic natural immune response to the stimulation of non-self cells, namely antigens [4][15][16]. The Ant Colony Optimization (ACO) algorithm is another emerging approach mimicking the foraging behavior of the ant species [6]. However, these powerful optimization methods have their inherent shortcomings and limitations. As we know, fusion of the computational intelligence methodologies can usually provide superior performances over employing them individually. Therefore, in this paper, a hybrid algorithm based on the CSA and ACO is proposed to cope with complex optimization problems under both the static and dynamic environments.

Our paper is organized as follows. The principles of the original ACO and CSA are first introduced in Section 2. Next, in Section 3, we discuss the proposed hybrid optimization algorithm in more details. In Section 4, the effectiveness of this new method is demonstrated using several static and dynamic benchmark optimization functions. A performance comparison among the ACO, CSA, and our hybrid optimization method is also made. Finally, we conclude the paper with some conclusions and remarks in Section 5.

2 Ant colony and clonal selection algorithms

2.1 Ant Colony Algorithm

It is well known that when foraging, some ant species have the behavior of depositing a kind of chemical substance called pheromone, through which they communicate with each other to choose the shortest path from their nest to the food source. This biological phenomenon lays down the basis of the ACO, a stochastic, meta-heuristic, and population-based optimization algorithm. The ACO was firstly proposed by Dorigo *et al.* for the combinatorial optimization problem solving [6], and numerous variations have been studied ever since. It is an efficient method for handling various optimization tasks, such as routing and scheduling [2][12][7][13]. The ACO can be characterized by [8]:

- Probabilistic transition rule is used to determine the moving direction of each ant,
- Pheromone update mechanism indicates the problem solution quality.

Local search is indeed important in dealing with continuous optimization problems. The continuous ACO has been extended to a hierarchical structure, in which the global search only aims at the 'bad' regions of the search space, while the goal of local search is to exploit those 'good' regions. The basic ACO algorithm for continuous optimization at each generation is described as follows [8].

1. Create n_r global ants.
2. Evaluate their fitness.
3. Update pheromone and age of weak regions.
4. Move local ants to better regions, if their fitness is improved. Otherwise, choose new random search directions.
5. Update ants' pheromone.
6. Evaporate ants' pheromone.

Obviously, the continuous ACO is based on both the global and local search towards the elitist. The local ants have the capability of moving to the latent region with the best solution, according to transition probability $P_i(t)$ of region i:

$$P_i(t) = \frac{t_i(t)}{\sum_{j=1}^{g} t_j(t)}, \tag{1}$$

where $t_i(t)$ is the total pheromone at region i at time t, and g is the number of global ants. Therefore, the better the region is, the more attraction to the successive ants it has. If their fitness is improved, the ants can deposit the pheromone increment Δt_i as in (2). Otherwise, no pheromone is left.

$$t_i(t+1) = \begin{cases} t_i(t) + \Delta t_i & \text{if fitness is improved} \\ t_i(t) & \text{otherwise} \end{cases}. \tag{2}$$

After each generation, the pheromone is updated as:

$$t_i(t+1) = (1-r) \cdot t_i(t) \ , \tag{3}$$

where r is the pheromone evaporation rate.

We can conclude that the probability for the local ants to select a region is proportional to its pheromone trail. On the other hand, the pheromone is affected by the evaporation rate, ant age, and growth of fitness. Thus, this pheromone-based selection mechanism is capable of promoting the solution candidate update, which is certainly suitable for handling the changing environments in optimization.

2.2 Clonal Selection Algorithm

The CSA is one of the most widely employed AIS approaches [15]. 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 the intruding antigens are selected to proliferate by cloning [14]. Hence, the fundamental idea of the CSA is those cells (antibodies) capable of recognizing the non-self cells (antigens) will proliferate. The flow chart of an essential CSA is shown in Fig. 1, and it involves the following nine iteration steps [5].

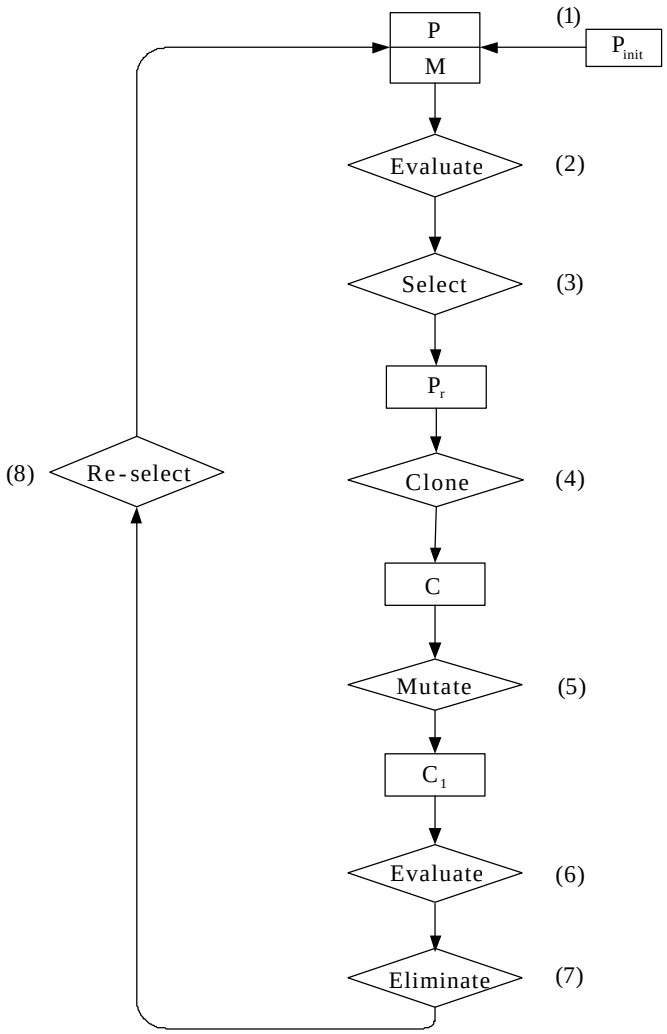


Figure 1: Flow chart of basic CSA.

1. Initialize the antibody pool including the subset of memory cells (M).
2. Evaluate the fitness of all the antibodies (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 antibody 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 antibodies with better fitness from C_1 to construct memory set M. Other improved individuals of C_1 can replace certain members with poor fitness in P to maintain the antibody diversity.
9. Return to Step 2.

Note, a unique mutation operator is used in Step 5, in which the mutated values of the antibodies 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 antibodies can also affect the overall convergence speed of the CSA. The idea of antibody suppression inspired by the immune network theory [4] is introduced to eliminate the newly generated antibodies, who are too similar to those already existing in the candidate pool (Step 7). With such a diverse antibody pool, the CSA can avoid being trapped into local minima, and provide the optimal solutions to the multi-model problems [16]. In summary, the antibody clone and fitness-related mutation are the two remarkable features of the CSA.

3 Hybrid optimization algorithm based on ACO and CSA

Based on the mechanisms of the aforementioned ACO and CSA, we here propose a hybrid optimization algorithm. In the ACO algorithm, the fitness of the ants can be improved using an effective and adequate local search rule. Unfortunately, achieving proper initial solutions for the local search is a challenging task. Compared with the worst region, the best region attracting the considerable quantities of ants may lead to over-similarity among the swarm. To overcome this drawback, the self-suppression-based affinity measurement of the CSA is utilized and combined with the ant colony in the new optimization algorithm to provide more efficient and comprehensive local exploitation. Such a fusion approach can also effectively prevent our optimization method from being trapped into local minima and high similarity among the solution candidates.

The proposed hybrid optimization algorithm is illustrated in Fig. 2, which can be described as the following steps:

1. Initialize *n* ants.
2. Sort the ants on the basis of their fitness evaluation.
3. Clone the ants with affinity-related mutation.
4. Update the pheromone and age of all the ants according to (2) and (3).
5. Exploit the selected regions by sending the ants for local search with the transition probability based on (1).
6. Retain the search direction, if the ants' fitness is improved. Otherwise, choose a new random search direction.
7. Update the pheromone and age of the local ants.
8. Evaluate the local ants. The winner in the local search is considered as the global ant in the next iteration.
9. If the preset convergence criterion is met, stop. Otherwise, continue.
10. Ants with pheromone lower than a pre-defined threshold are replaced by the newly generated ants.
11. Return back to Step 2.

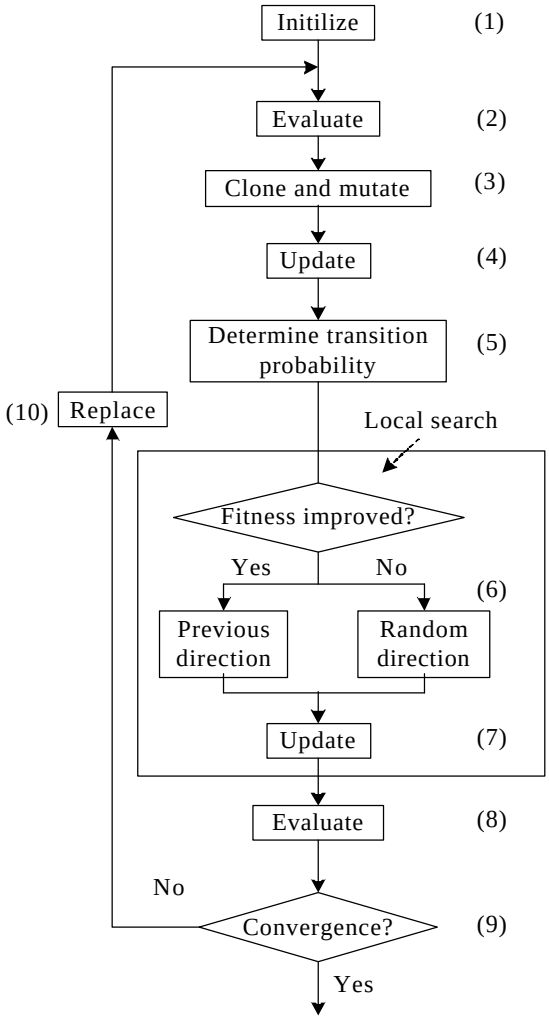


Figure 2: Hybrid optimization algorithm based on ACO and CSA.

Apparently, this hybrid optimization algorithm merges the features of both the ACO and CSA, i.e., pheromone-based elitism selection strategy, fitness- and age-depended mutation size, hieratical search structure, and selfness suppression. More precisely, at each iteration, only those ants that can increase the search performance may deposit an amount of pheromone proportional to the improved fitness. Therefore, this hybrid optimization approach is capable of weeding the outdated candidates without loosing the best ants, and is adaptable to the dynamic environments to avoid misleading the search direction as well. In addition to the fitness, the mutation size is affected by the ant age so that the exploration and exploitation in the search space can be properly balanced. We emphasize that the mutation rate is inversely proportional to the fitness associated with the ant age. Note, in the local search, the step size is regarded as the mutation, which is related to the ant age and fitness.

Lack of solution diversity can significantly slow down the convergence of the ACO. Thus, a novel affinity measurement borrowed from the CSA is embedded in our hybrid optimization algorithm. Not only the antibody-antigen affinities, but also the affinities among the antibodies are applied here in order to suppress those candidates with high similarity. Moreover, the self suppression mechanism is utilized to enhance its local search capability. Compared with the original ACO and CSA, this hybrid optimization method has a superior optimization performance of global search and convergence, which will be demonstrated using numerical simulations in Section 4.

4 Simulations

In this section, function optimization is deployed to verify our proposed method. In all the simulations, we use 10 and 30 ants for the global and local search, respectively. At each generation, the pheromone evaporation rate is 0.9. Note that the Euclidean distance is used to measure the affinities of the antibodies. The hybrid optimization algorithm is examined in both the static and dynamic cases.

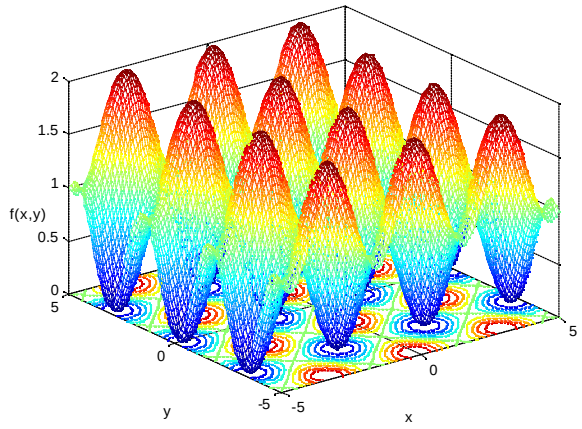
4.1 Static optimization

Example 1

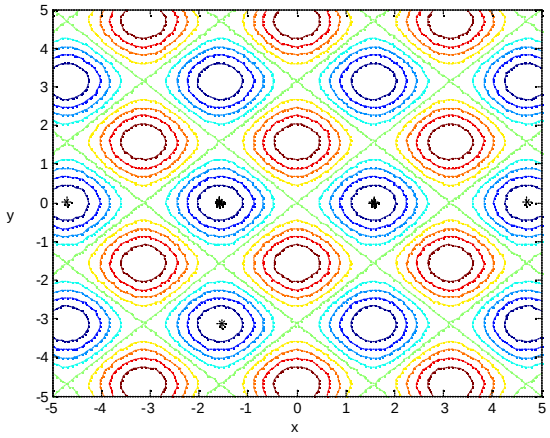
The following two-dimension function, as shown in Fig. 3 (a), is considered here [11]:

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

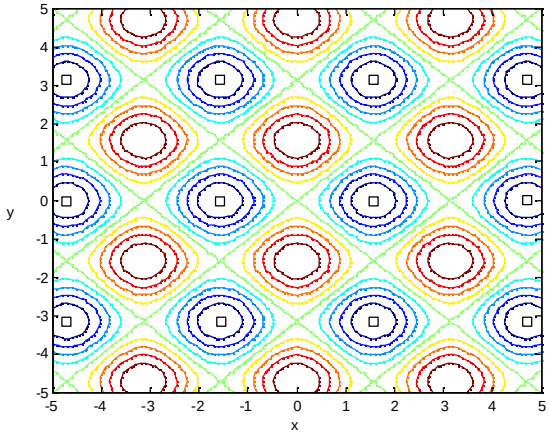
where $x \in [-5,5]$, and $y \in [-5,5]$. This nonlinear function has infinite global maxima in R^2 at points $(mp/2, np), m, n = \pm 1, \pm 2 \cdots$. Especially, there are 12 existing in $[-5,5] \times [-5,5]$. After 200 generations, the ACO can achieve only five of them, denoted by '*' in Fig. 3 (b). However, both the hybrid optimization method and CSA are capable of locating all the 12 optima, as denoted by '?' in Fig. 3 (c). The above simulation results demonstrate that the proposed hybrid optimization method has a considerably better solution diversity than the ACO.



(a)



(b)



(c)

Figure 3: Optimization results of Example 2. (a) function f(x,y) in (5). (b) optima ('*') achieved by ACO on contour map of f(x,y). (c) optima ('? ') found by CSA and hybrid optimization method on contour map of f(x,y).

Example 2

To further examine our hybrid optimization method, we employ the following four *n*-dimension functions in this example, which have been widely used as the function optimization benchmarks [8].

The Sphere function:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 . \tag{6}$$

The Bohachevsky function:

$$f_2(x) = x_1^2 + 2x_2^2 - 0.3\cos(3px_1) - 0.4\cos(4px_2) + 0.7 . \tag{7}$$

The Hyperellipsoid function:

$$f_3(x) = \sum_{i=1}^{n} x_i^2 i^2 . \tag{8}$$

The Griewank function:

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1. \tag{9}$$

The details of these unconstrained functions are given in Table 1.

Functions	**Search space**	**Global optima**
Spherical function	[-100, 100]	$f(x) = 0$
Bohachevsky function	[-50, 50]	$f(x) = 0$
Hyperellipsoid function	[-1, 1]	$f(x) = 0$
Griewank function	[-600, 600]	$f(x) = 0$

Table 1. Details of benchmark functions.

As a representative example, the minimization of a 30-dimension Spherical function is first applied for comparing the ACO, CSA, and proposed hybrid optimization method. Figure 4 illustrates their average convergence procedures over 10 runs. The CSA and ACO perform quite similarly at the beginning of their optimization. Nevertheless, the convergence speed of the ACO becomes a little higher than that of the CSA with the growth of iterations. Obviously, our hybrid optimization algorithm converges much faster than both the other two methods in this high-dimension function optimization case.

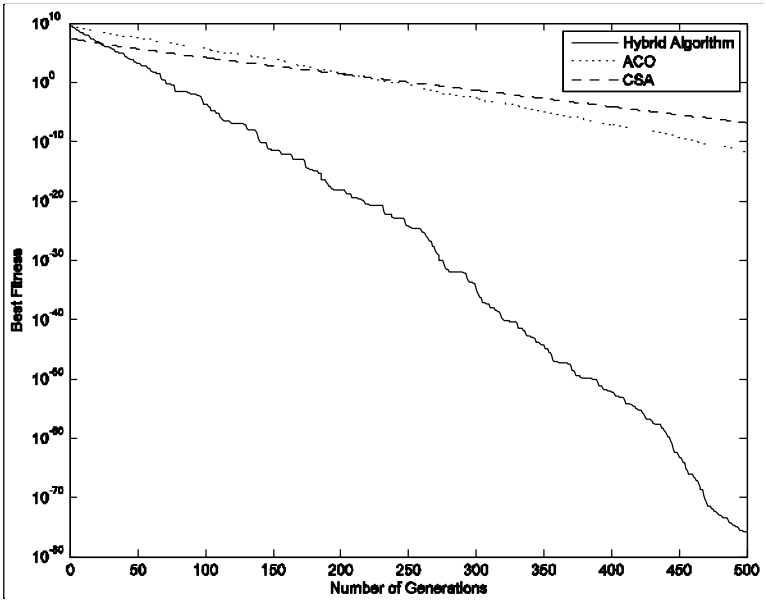


Figure 4: Convergence procedures of ACO, CSA, and hybrid algorithm in Spherical function optimization.

Moreover, we run these three optimization methods for 20 times and 1,000 iterations using the above four benchmark functions with different dimensions. The average optima and standard errors obtained by the ACO, CSA, and hybrid optimization method are summarized in Table 2. We can observe that with regard to the low-dimension and simple functions, such as the 10-dimension Sphere and Griewank functions, they all achieve the similar optimization perfomances. However, our hybrid optimization method can significantly outperform the other two in case of the high-dimensional complex functions, e.g., the 30-dimension Hyperellipsoid function.

Dimensions	**10**	**20**	**30**
Spherical function			
ACO	0	0	1.3 ± 0.1^{-16}
CSA	0	0	5.5 ± 1.2^{-20}
Hybrid Algorithm	0	0	0
Griewank function			
ACO	0	4.6 ± 0.9^{-12}	1.7 ± 0.3^{-34}
CSA	0	1.3 ± 0.4^{-8}	0.1 ± 0.03
Hybrid Algorithm	0	0	0
Hyperellipsoid function			
ACO	7.0 ± 0.9^{-25}	7.0 ± 0.2^{-16}	4.5 ± 0.5^{-15}
CSA	1.8 ± 0.2^{-19}	1.2 ± 0.2^{-12}	6.6 ± 0.7^{-12}
Hybrid Algorithm	0	0	0
Bohachevsky function ($n=2$)			
ACO	9.4 ± 0.6^{-4}		
CSA	0.88 ± 0.01		
Hybrid Algorithm	0		

Table 2. Performance comparisons of three optimization methods in benchmark functions with different dimensions (optima ± standard error).

4.2 Dynamic optimization

Since changing environments may result in the outdated memory in optimization that can misguide the search with a high probability, dynamic optimization is always a demanding topic. The goal of dynamic optimization is to track the optimum trajectory as closely as possible under the variant environments, where the changes can have various forms, such as parameters, objective functions, and constraints [1]. Morrison's two-dimension function has been intensively used to create a dynamic environment with given number of optima, which can change with the varying inputs [9]:

$$f(X,Y) = \max_{i=1,N}\left[H_i - R_i \times \sqrt{(X - X_i)^2 + (Y - Y_i)^2}\right], \tag{10}$$

where N specifies the number of cones in the search space. Each cone is specified by its location (X_i, Y_i), height H_i, and slope R_i with ranges:

$$H_i \in [H_{base}, H_{base} + H_{range}],\ R_i \in [R_{base}, R_{base} + R_{range}],\ X_i \in [-1,1], \text{ and } Y_i \in [-1,1].$$

Therefore, dynamic environments can be obtained by specifying the above parameters, and the function complexity depends on the number of peaks, as shown in Fig. 5. The typical parameters of Morrison's function are given in Table 3.

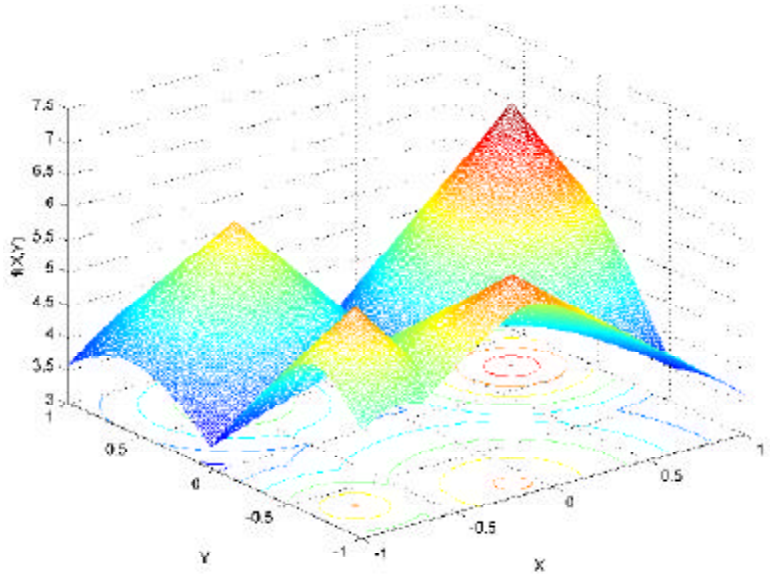


Figure 5: Morrison's function.

Parameters	Values	Parameters	Values
H_{base}	3	H_{range}	3
R_{base}	2	R_{range}	5
$X(Y)$	$[-1,1]$	N	4

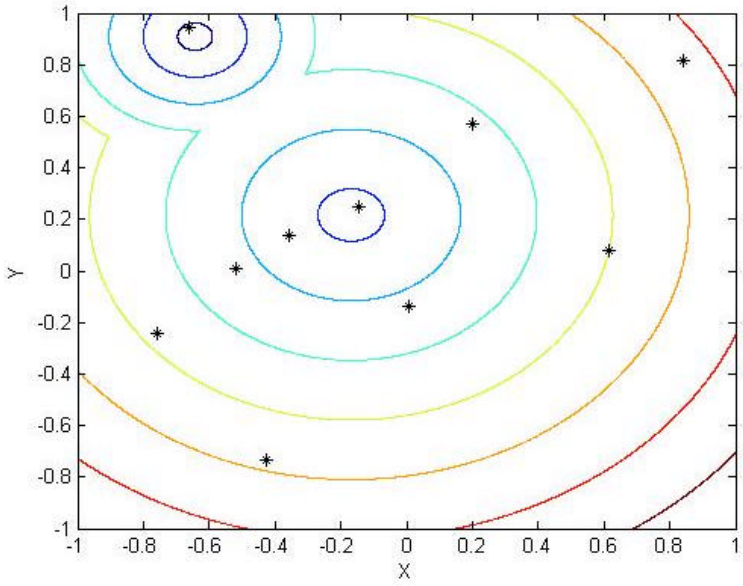
Table 3. Parameters of Morrison's function.

This dynamic function offers the height, location, and slope to generate dynamics. One of the most popular approaches is to employ a nonlinear logistics function with bifurcation transitions to progressively produce more and more complex behaviors [9], e.g.,

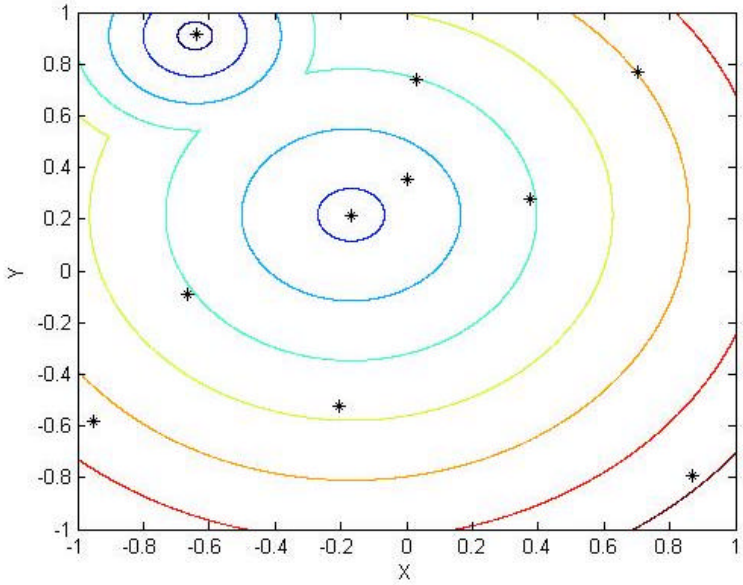
$$Y_i = A * Y_{(i-1)} * (1 - Y_{(i-1)}), \tag{11}$$

where A is a coefficient within $[1,4]$, and i is the index.

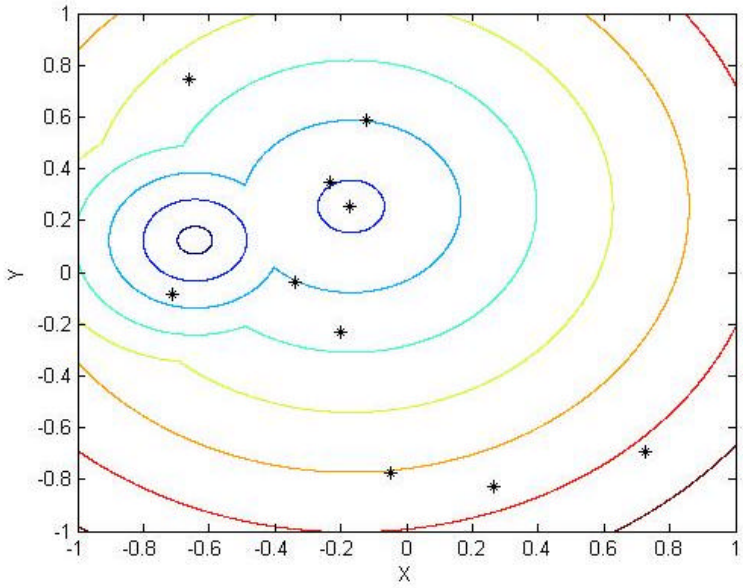
We first apply our hybrid optimization method in coping with Morrison's function. The optimal solutions acquired on the contour map for $A = 1.5$ and $f = 5$ at iteration 1, 4, 5 and 9 are shown in Figs. 6 (a), (b), (c), and (d) respectively. Obviously, the hybrid optimization algorithm can closely track the update of the dynamic environment.



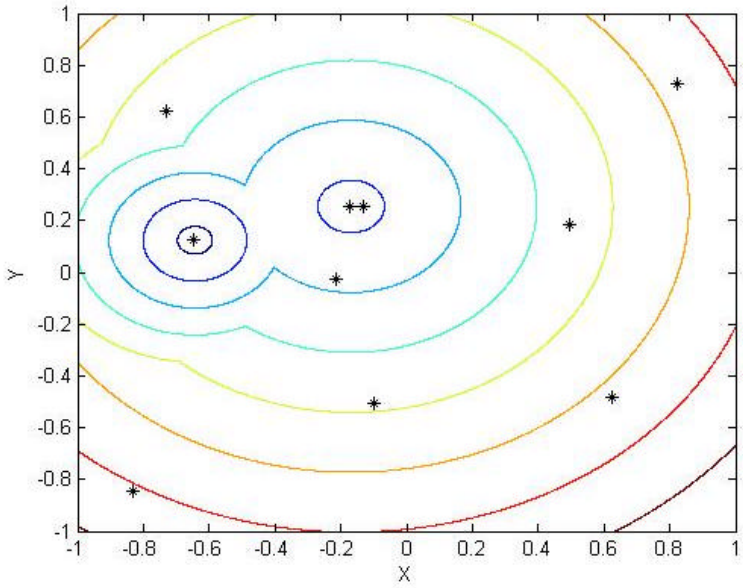
(a)



(b)



(c)



(d)

Figure 6: Optimization of Morrison's function using hybrid optimization method. (a) optima (*) achieved at the first iteration, (b) optima (*) achieved at the fourth iteration, (c) optima (*) achieved at the fifth iteration, (d) optima (*) achieved at the ninth iteration.

To investigate the effectiveness of these three optimization methods in tracking the global optima of Morrison's function under changing situations, the offline error is evaluated at each generation [3] [10]. The global error e_{gt} is the best fitness compared with the global optimum:

$$e_{gt} = 1 - \frac{f_{gt}}{g_{gt}}, \tag{12}$$

where f_{gt} is the best fitness at t, and g_{gt} is the actual global optimum. The best e'_{gt} since the previous change occurring in the environment is calculated as:

$$e'_{gt} = \min\{e_{gt}, e_{gt+1}, ..., e_{gt}\}, \tag{13}$$

where t is the last iteration, at which the environment changes. Hence, the offline error $e'_{g(offline)}$ is defined as:

$$e'_{g(offline)} = \frac{1}{T}\sum_{t=1}^{T} e'_{gt}, \tag{14}$$

where T is the number of running iterations.

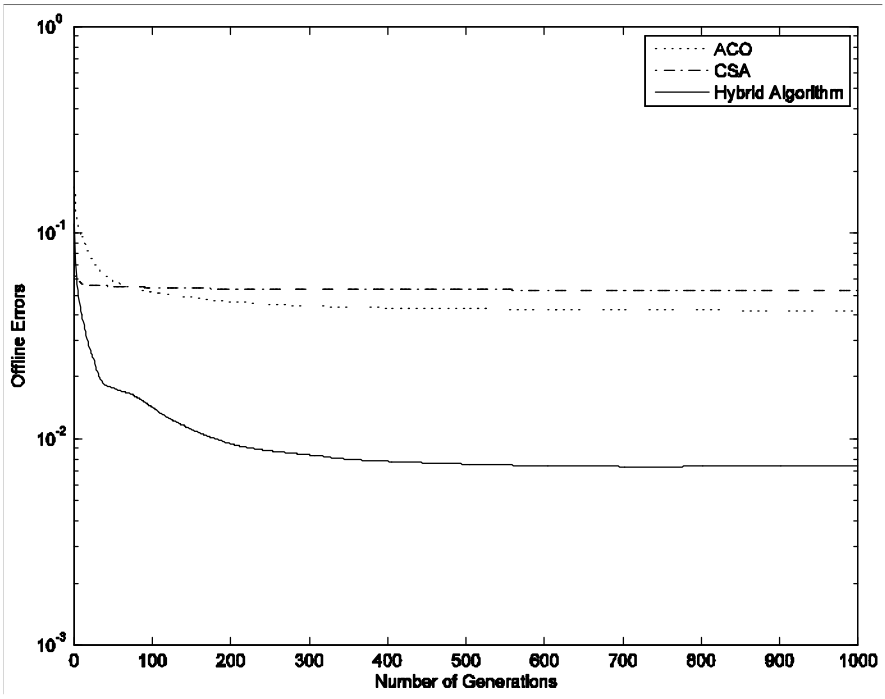
The behaviors of the offline error *versus* iterations of the three optimization methods with different iteration-dependant parameter changing frequencies f (1, 10, and 100) and values of A (1 and 1.5) in case of four cones are illustrated in Fig. 7. The performance comparison results of these optimization methods are also given in Tables 4 and 5, where N is 5 and 10, respectively. It is clearly visible that the CSA and ACO can only track the optimum with low frequencies. However, in all the cases, the proposed hybrid optimization algorithm is well capable of performing much better than the others. It can satisfactorily adapt to track the dynamic environments even when the varying frequency is high.

f	A	**ACO**	**CSA**	**Hybrid Algorithm**
1	1.5	0.018±0.001	0.077±0.004	0.080±0.006
	2.5	0.018±0.001	0.081±0.009	0.090±0.015
	3.5	0.031±0.003	0.101±0.012	0.072±0.009
10	1.5	0.015±0.002	0.021±0.004	0.014±0.002
	2.5	0.032±0.006	0.022±0.004	0.021±0.005
	3.5	0.043±0.006	0.035±0.007	0.022±0.001
100	1.5	0.009±0.001	0.011±0.001	0.006±0.002
	2.5	0.011±0.001	0.011±0.003	0.002±0.001
	3.5	0.018±0.002	0.014±0.001	0.009±0.001

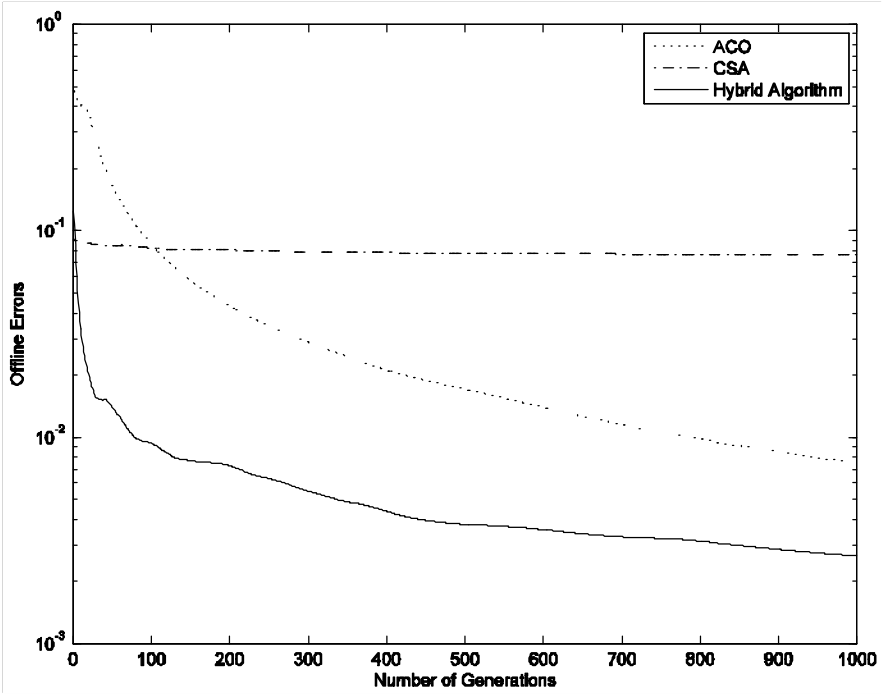
Table 4. Performance comparison of three optimization methods for Morrison's function with $N = 5$ (optima ± standard error).

f	A	**ACO**	**CSA**	**Hybrid Algorithm**
1	1.5	0.076±0.008	0.078±0.05	0.035±0.007
	2.5	0.057±0.013	0.067±0.006	0.032±0.005
	3.5	0.089±0.008	0.062±0.007	0.050±0.013
10	1.5	0.015±0.003	0.020±0.003	0.016±0.004
	2.5	0.020±0.003	0.025±0.006	0.021±0.005
	3.5	0.045±0.012	0.039±0.003	0.019±0.002
100	1.5	0.005±0.001	0.015±0.002	0.001±0.001
	2.5	0.011±0.001	0.011±0.004	0.001±0.001
	3.5	0.005±0.001	0.011±0.002	0.002±0.001

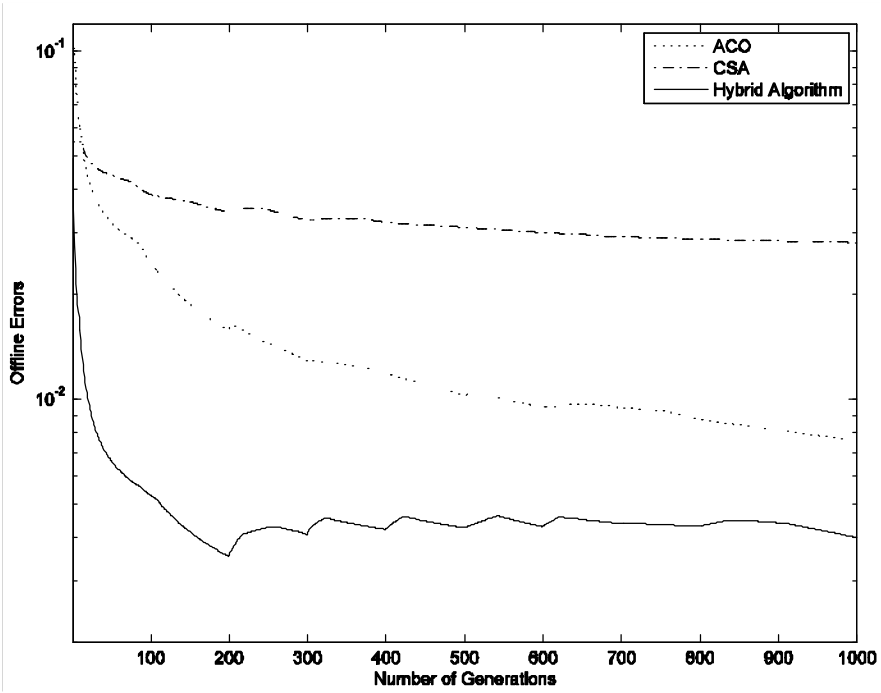
Table 5. Performance comparison of three optimization methods for Morrison's function with $N = 10$ (optima ± standard error).


ACO
CSA
Hybrid Algorithm
Offline Errors
Number of Generations
10^0
10^-1
10^-2
10^-3
0
100
200
300
400
500
600
700
800
900
1000

(a)


ACO
CSA
Hybrid Algorithm
Offline Errors
Number of Generations
10^0
10^-1
10^-2
10^-3
0
100
200
300
400
500
600
700
800
900
1000

(b)



(c)

Figure 7: Offline error *versus* iterations of three optimization methods for Morrison's function. (a) f =1, and A=1. (b) f =10, and A=1. (c) f =100, and A=3.

5 Conclusions

In this paper, we propose a hybrid optimization algorithm based on the fusion of the regular CSA and ACO, and further employ several nonlinear functions under both static and dynamic environments to evaluate its effectiveness. The pheromone-based meta-heuristic elitism and hieratical search strategy of the ACO together with the outstanding local search ability and solution diversity characteristics of the CSA are fully utilized and combined in the new optimization algorithm. Simulation results demonstrate that it can achieve an improved performance over both the CSA and ACO. Compared with these two methods, it is capable of providing more diverse and flexible solutions as well as closely tracking the optimum under dynamic environments. However, our hybrid optimization approach suffers from a moderately high computational complexity, which may lead to certain engineering implementation difficulty, and thus hinder its employment in the real-time cases. The future research work will focus on applying this novel optimization scheme for solving practical problems.

Acknowledgments

This research work was partially funded by the Academy of Finland under Grant 214144.

References

[1] Blackwell, T. and Branke, J., Multiswarms, exclusion, and anti-convergence in dynamic environments, *IEEE Transactions on Evolutionary Computation*, 10(4):459-472, 2006.

[2] Blum, C., Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling, *Computers & Operations Research*, 32(6):1565-1591, 2005.

[3] Branke, J., *Evolutionary Optimization in Dynamic Environments*, Norwell, MA: Kluwer, 2002.

[4] Dasgupta, D., Advances in artificial immune systems, *IEEE Computational Intelligence Magazine*, 1(4):40-49, 2006.

[5] De Castro, L.N. and Von Zuben, F.J., Artificial immune systems: Part I-Basic theory and applications, Technical Report RT-DCA 01/99, FEEC/UNICAMP, Brazil, 1999.

[6] Dorigo, M., Birattari, M., and Stutzle, T., Ant colony optimization, *IEEE Computational Intelligence Magazine*, 1(4):28-39, 2006.

[7] Dorigo, M., Maniezzo, V., and Colorni, A., Ant system: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29-41, 1996.

[8] Engelbrecht, A.P., *Fundamentals of Computational Swarm Intelligence*, Chichester, UK: Wiley, 2005.

[9] Morrison, R.W. and De Jong, K.A., A test problem generator for non-stationary environments, *Proc. IEEE Congress on Evolutionary Computation*, 2047-2053, 1999.

[10] Parrott, D. and Li, X., Locating and tracking multiple dynamic optima by a particle swarm model using speciation, *IEEE Transactions on Evolutionary Computation*, 10(4):440-458, 2006.

[11] Parsopoulos, K. and Vrahatis, M., Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing*, 1(2-3):235-306, 2002.

[12] Socha, K., Sampels, M., and Manfrin, M., Ant algorithms for the university course timetabling problem with regard to the state-of-the-art, *Proc. Workshop on Applications of Evolutionary Computing*, 334-345, 2003.

[13] Stutzle, T. and Hoos, H.H., The *MAX-MIN* ant system and local search for the traveling salesman problem, *Proc. IEEE International Conference on Evolutionary Computation*, 309-314, 1997.

[14] Wang, X., Clonal selection algorithm in power filter optimization, *Proc. IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*, 122-127, 2005.

[15] Wang, X., Gao, X.Z., and Ovaska, S.J., Artificial immune optimization methods and applications – A survey, *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, 3415-3420, 2004.

[16] Wang, X., Gao, X.Z., and Ovaska, S.J., A hybrid particle swarm optimization method, *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, 4151-4157, 2006.