A Hybrid Artificial Immune Optimization Method

X. Wang, X. Z. Gao, and S. J. Ovaska

Department of Electrical Engineering

Helsinki University of Technology

Otakaari 5 A, FI-02150 Espoo, Finland

Tel.: +358 9 451 4965, Fax: +358 9 451 2432

http://powerelectronics.tkk.fi/

E-mail: xiaolei@cc.hut.fi, gao@cc.hut.fi, seppo.ovaska@tkk.fi

Received: 29/12/08
Accepted: 07/07/09

Abstract

This paper proposes a hybrid optimization method based on the fusion of the Simulated Annealing (SA) and Clonal Selection Algorithm (CSA), in which the SA is embedded in the CSA to enhance its search capability. The novel optimization algorithm is also employed to deal with several nonlinear benchmark functions as well as a practical engineering design problem. Simulation results demonstrate the remarkable advantages of our approach in achieving the diverse optimal solutions and improved convergence speed.

Keywords: artificial immune systems, simulated annealing, hybrid algorithm, optimization.

1. Introduction

During the past decade, biology-inspired computational intelligence techniques have been widely employed in numerous optimization areas. For example, Artificial Immune Systems (AIS), inspired by the immunology, are an emerging kind of soft computing methods. As an important branch of the AIS, the Clonal Selection Algorithm (CSA) stems from the clonal selection mechanism that describes the basic natural immune response to the stimulation of non-self cells (antigens) [1]-[3]. Another popular optimization scheme is the Simulated Annealing (SA) method, proposed by Kirkpatrick *et al.* in 1983 [4], which is based on the principle of the atoms transition in equilibrium at a given temperature. There is an analogy between the minimization of the cost function in an optimization problem and the practical procedure of gradually cooling a metal until it reaches its “freezing” point, where the energy of the system has acquired the globally minimal value [5]. However, these optimization algorithms have their inherent drawbacks and limitations, e.g., the slow convergence of the SA method. As we know, fusion of different intelligent computing methods can often provide superior performances over employing them individually [6]. Therefore, in this paper, we study a novel hybrid optimization approach based on the hybridization of the CSA and SA method.

Our paper is organized as follows. First, the principles of the original CSA and SA method are briefly 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 optimization method is demonstrated and verified using several benchmark functions and a real-world pressure vessel design problem. Performance comparisons among the CSA, SA, and our hybrid optimization method are also made. Finally, we conclude the paper with some conclusions and remarks in Section 5.

2. Principles of Clonal Selection Algorithm and Simulated Annealing Method

2.1. *Clonal Selection Algorithm (CSA)*

Inspired by the Clonal Selection Principle (CSP), the CSA has been successfully applied to handle some challenging optimization problems, due to its improved capability compared with the classical optimization techniques [2]. The CSP 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 [7]. Hence, the fundamental idea of the CSA is that 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 [8] [9].

(i) Initialize the antibody pool P_{init} including the subset of memory cells (M).
(ii) Evaluate the fitness of all the antibodies (affinity with the antigen) in population P.
(iii) Select the best candidates (P_r) from population P, according to their fitness.
(iv) Clone P_r into a temporary antibody pool (C).
(v) Generate a mutated antibody pool (C_1). The mutation rate of each antibody is inversely proportional to its fitness.
(vi) Evaluate all the antibodies in C_1.
(vii) Eliminate those antibodies similar to the ones in C, and update C_1.
(viii) 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.
(ix) Return back to Step 2, if a pre-set termination criterion is not met.

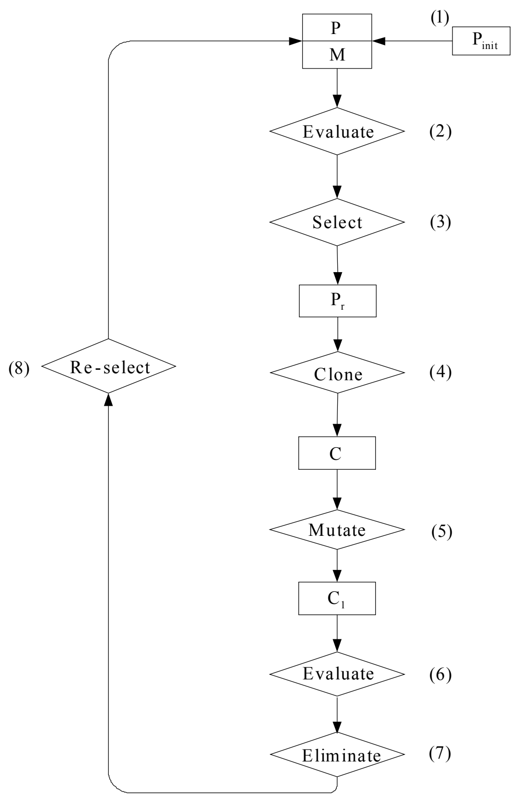


Fig. 1. Flow chart of basic CSA.

Note that 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 the antibodies can also affect the overall convergence speed of the CSA. The idea of antibody suppression inspired by the immune network theory [1] is introduced to eliminate the newly generated antibodies, which are too 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 the local minima, and provide the optimal solutions to the multi-modal problems [3]. In summary, the antibody cloning and fitness-related mutation are the two remarkable characteristics of the CSA.

2.2. *Simulated Annealing (SA) Method*

The SA is a powerful optimization method, which is based on the analogy between the statistical mechanics and optimization. The SA process consists of first "melting" the system being optimized at a high temperature, and then lowering the temperature by very slow stages until the system "freezes" and no further change occurs. At each temperature instant, the annealing must proceed long enough for the system to reach a steady state [4]. The SA method actually mimics the behavior of this dynamical system to achieve the thermal equilibrium at a given temperature. It has the distinguishing ability of escaping from the local minima by accepting or rejecting new solution candidates according to a probability function. In addition, the SA method only requires little computation resource. The flow chart of a basic SA method is illustrated in Fig. 2, and it can be described by the six steps as follows:

(i) Specify initial temperature T_0, and initialize the candidate.
(ii) Evaluate fitness E of the candidate.
(iii) Move the candidate randomly to a neighboring solution.
(iv) Evaluate the fitness of new solutions E'.
(v) Accept the new solution, if
 (a) $E' \le E$
 or
 (b) $E' > E$ with acceptance probability P.
(vi) Decrease temperature T. The SA search is terminated, if the temperature is close to zero.

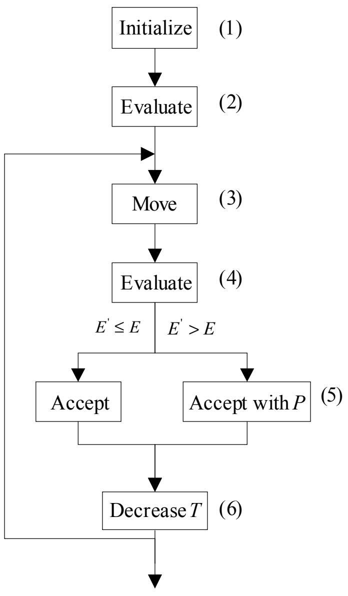


Fig. 2. Flow chart of basic SA method.

As we can see that the SA algorithm simulates the procedure of gradually cooling a metal until the energy of the system achieves the global minimum. Each configuration of the physical system and energy of the atoms correspond to the current solution to the optimization problem and fitness of the objective function, respectively, and the temperature is used to control the whole optimization procedure. At every generation, according to the Metropolis criterion [10], the candidate is updated through the random perturbation, and the improvement of its fitness is calculated. If $E' \le E$, the moving change results in a lower or equivalent energy of the system, and the new solution is accepted. Otherwise, the displacement is only accepted with a probability P:

$$P = e^{\frac{-(E'-E)}{T}}. \tag{1}$$

The temperature is updated by:

$$T(k+1) = \lambda T(k), \quad 0 < \lambda < 1, \tag{2}$$

where k is the number of generations. Obviously, the SA has the remarkable ability of escaping from the local minima by accepting or rejecting new solution candidates according to (1). As a matter of fact, the cooling schedule can be adjusted by modifying parameter λ. We set λ to be between 0.4 and 0.8 in our simulations.

The SA method is terminated when the final temperature is sufficiently low, which makes it reach the global optimal solution with a high probability. The probability-dependent acceptance policy for the new solutions helps the SA algorithm in the solution exploitation. However, slow convergence is the main disadvantage that can hinder its applications in engineering. The temperature plays an important role in the cooling procedure control. The initial temperature should be high enough to explore the whole solution space [5].

3. A Hybrid Optimization Algorithm

In this section, we develop a hybrid optimization algorithm based on the principles of both the aforementioned CSA and SA. The SA method occasionally chooses those 'uphill points' from the current place. That is, not only the improved solutions but also the relatively weak ones are accepted with a specified probability according to different temperatures. Thus, the SA method has certain advantages, e.g., robustness and flexibility, over other local search methods, and is suitable for handling nonlinear problems. Unfortunately, it always takes a considerably long time to acquire the global optimum, because the temperature indeed needs to be decreased slowly enough during the iterations. In our approach, the fitness-related mechanisms of mutation and cloning as well as the affinity-based self-suppression of the CSA are utilized and combined with the SA method so as to improve the global search and convergence speeds. The diagram of this hybrid optimization scheme is shown in Fig. 3, and the corresponding iteration steps are explained as follows.

(i) Initialize the candidate pool.
(ii) Evaluate the fitness of all the antibodies (affinities with the antigen) in the population.
(iii) Select the best candidates from the population according to their fitness.
(iv) Clone those selected antibodies.
(v) Move the candidates randomly to the neighboring states. The moving step of each candidate is inversely proportional to its fitness, which can be considered as the mutation operation in the CSA.
(vi) Evaluate the new candidates.
(vii) Accept the new solutions, if their fitness is improved $(\Delta f > 0)$. Otherwise, accept them only with probability P.
(viii) Update the temperature based on (2).
(ix) Evaluate the antibodies, and measure the affinities among these antibodies.
(x) Re-select the antibodies with better fitness, and return back to Step 4.

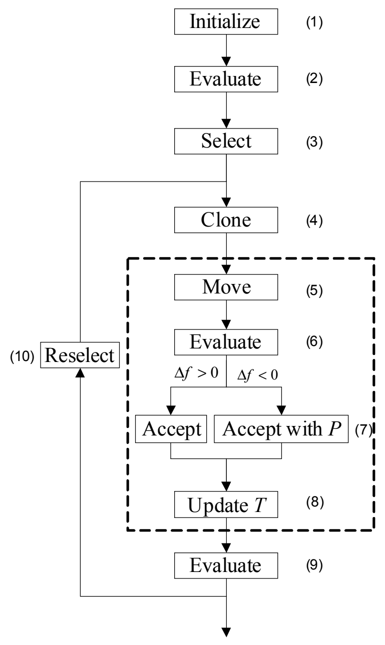


Fig. 3. Flow chart of hybrid optimization algorithm.

As can be observed from Fig. 3, the proposed hybrid optimization algorithm has the remarkable features of both the CSA and SA method, i.e., fitness-related mutation size, selfness suppression, and probability-based acceptance of worse solutions. In this hierarchical search system, the SA method is embedded in the CSA to enhance its local search ability. In Step 5, the random perturbation of the current solutions is regarded as the mutation, which is proportional to their fitness. That is to say, the better fitness the individual has, the less it changes by mutation. Furthermore, the acceptance of the low-affinity antibodies with a specified probability can efficiently protect those potential candidates that may lead to the global optimal from weeding. Therefore, the SA-aided approach provides a sufficient global search, which is well suited for challenging optimization problems. According to the evaluation criterion of the CSA, not only the antibody-antigen affinities, but also the affinities among the antibodies are employed here in order to suppress the candidates with over-similarity. The procedure from Steps 5 to 8 represents the local search that is realized by the SA. To accelerate the convergence as well as improve the search efficiency, the local search is only executed for a pre-defined number of iterations (problem-dependent) under low temperatures to maintain a high update probability. Apparently, the hybrid algorithm can avoid long perturbation state until it reaches the equilibrium. Compared with the original CSA and SA method, our hybrid optimization method has an enhanced performance of global search and convergence, which will be demonstrated using numerical simulations in Section 4.

4. Simulations

In this section, a few nonlinear functions and a practical engineering problem are employed to verify our proposed hybrid optimization method. In all the simulations, we use a total of 20 antibody candidates for evolution, and the Euclidean distance is deployed as the affinity measure.

4.1. *Nonlinear Functions Optimization*

Firstly, we examine the above three optimization methods with the following eight nonlinear functions, which have been widely used as the optimization benchmarks [11].

Function 1:

$$\mathrm{f}_1(x,y) = \cos(x)^2 + \sin(y)^2 . \tag{3}$$

Sphere function:

$$\mathrm{f}_2(x) = \sum_{i=1}^{n} x_i^2 . \tag{4}$$

Hyperellipsoid function:

$$f_3(x) = \sum_{i=1}^{n} x_i^2 i^2 . \quad (5)$$

Griewank function:

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1 . \quad (6)$$

Zakharov function:

$$f_5(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4 . \quad (7)$$

Schwefel function:

$$f_6(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| . \quad (8)$$

Bohachevsky function:

$$f_7(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 . \quad (9)$$

Sum Squares function:

$$f_8(x) = \sum_{i=1}^{n} ix_i^2 . \quad (10)$$

Note that all the functions are uni-modal functions with n dimensions except for Function 1, which is a two-dimension multi-modal function with 12 global optima. The details of these unconstrained functions are given in Table 1.

Table 1. Details of benchmark functions.

Functions	Search Range	Global Optima
Function 1	[-5, 5]	$f_1(x, y) = 0$
Sphere	[-100, 100]^n	$f_2(x) = 0$
Hyperellipsoid	[-50, 50]^n	$f_3(x) = 0$
Griewank	[-600, 600]^n	$f_4(x) = 0$
Zakharov	[-10, 10]^n	$f_5(x) = 0$
Schwefel	[-10, 10]^n	$f_6(x) = 0$
Bohachevsky	[-50, 50]	$f_7(x) = 0$
Sum Squares	[-10, 10]^n	$f_8(x) = 0$

The simulations are made under the MATLAB 7.0 environment on an AMD Athlon 64 4000+ computer with 1 G system memory. As a representative example, the minimization of the 50-dimension Zakharov function is deployed here for comparing the CSA, SA, and proposed hybrid optimization method. Figure 4 illustrates their average convergence procedures over 100 runs that are represented by the dash-dotted, dash, and solid lines, respectively. Obviously, the convergence speed of the SA method is the lowest among the three approaches. Moreover, the proposed algorithm performs moderately better than the CSA, which demonstrates that the SA method embedded can provide a more sufficient local search. Therefore, we conclude that our hybrid optimization scheme converges faster than both the other two methods in this high-dimension function optimization case. However, the computational complexity of this hybrid method is higher than that of the CSA and SA.

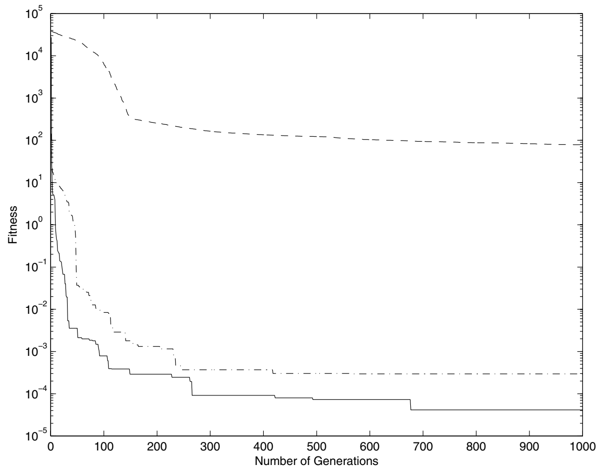


Fig. 4. Convergence procedures of CSA, SA, and hybrid algorithm in 50-dimension Zakharov function optimization. Dash-dotted line: CSA, dash line: SA method, solid line: hybrid algorithm.

We also run these three optimization methods for 100 times and 1,000 iterations using the above n-dimension benchmark functions with $n = 10$, $n = 20$, and $n = 30$. The average optima and standard errors obtained by the CSA, SA, and hybrid optimization method are given in Table 2. It is clearly visible that the hybrid method is capable of significantly outperforming the other two approaches in the high-dimension functions. Table 3 shows the performance comparison with regard to the optimization of Function 1. The optimization results of the CSA, SA method, and hybrid algorithm are illustrated in Figs. 5 (a), (b), and (c), respectively. We observe that both the CSA and hybrid algorithm successfully locate all the 12 optima (Figs. 5 (a) and (c)), while the SA method fails (Fig. 5 (b)). To summarize, for the multi-model problems, this hybrid method can take advantage of the solution diversity from the CSA to find the global optima.

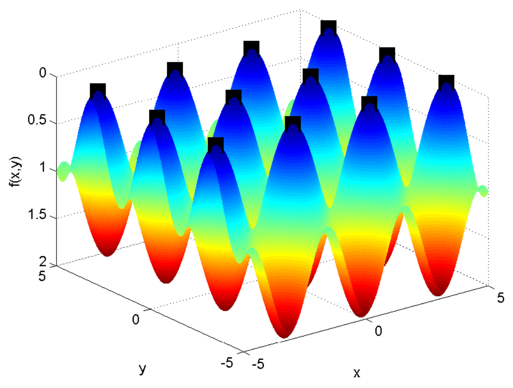
Table 2. Performance comparisons of three optimization methods in benchmark functions with different dimensions (optima ± standard error).

Dimensions	10	20	30
Sphere Function			
CSA	6.1×10^{-10} (1.7×10^{-11})	2.0×10^{-7} (2.7×10^{-8})	2.7×10^{-4} (1.8×10^{-5})
SA Method	0.1 (0.02)	11.1 (1.8)	13.4 (3.7)
Hybrid Algorithm	7.2×10^{-47} (3.1×10^{-47})	2.3×10^{-32} (9.7×10^{-31})	2.1×10^{-21} (2.3×10^{-20})
Hyperellipsoid Function			
CSA	3.4×10^{-17} (5.3×10^{-18})	1.0×10^{-14} (0.7×10^{-14})	3.7×10^{-11} (1.8×10^{-11})
SA Method	0.1 (0.02)	2.5 (2.3)	15.8 (5.9)
Hybrid Algorithm	4.8×10^{-24} (5.7×10^{-25})	5.7×10^{-18} (9.6×10^{-20})	4.0×10^{-12} (5.5×10^{-13})
Griewank Function			
CSA	8.9×10^{-11} (6.7×10^{-11})	1.0×10^{-7} (6.7×10^{-8})	4.9×10^{-6} (1.4×10^{-5})
SA Method	3.2×10^{-2} (3.4×10^{-3})	0.6 (0.1)	1.4 (0.1)
Hybrid Algorithm	2.2×10^{-15} (3.5×10^{-16})	1.7×10^{-10} (9.9×10^{-11})	6.2×10^{-8} (3.9×10^{-8})
Zakharov Function			
CSA	3.0×10^{-10} (6.7×10^{-9})	2.8×10^{-8} (1.7×10^{-8})	2.7×10^{-6} (8.6×10^{-6})
SA Method	4.8×10^{-1} (2.4×10^{-1})	1.5 (0.3)	23.5 (10.4)
Hybrid Algorithm	4.8×10^{-12} (3.7×10^{-13})	5.7×10^{-10} (9.6×10^{-10})	3.1×10^{-7} (1.0×10^{-7})
Schwefel Function			
CSA	2.5×10^{-40} (5.7×10^{-41})	2.5×10^{-37} (6.7×10^{-38})	2.7×10^{-20} (1.8×10^{-20})
SA Method	8.1×10^{-1} (7.3×10^{-1})	11.1 (3.6)	13.7 (8.0)
Hybrid Algorithm	5.6×10^{-44} (7.9×10^{-43})	2.3×10^{-40} (2.7×10^{-39})	2.1×10^{-21} (2.3×10^{-20})
Sum Squares Function			
CSA	1.5×10^{-40} (0.6×10^{-40})	5.0×10^{-36} (2.7×10^{-36})	8.2×10^{-18} (1.8×10^{-19})
SA Method	3.8×10^{-3} (6.3×10^{-4})	9.8×10^{-1} (2.3×10^{-1})	23.7 (12.0)
Hybrid Algorithm	6.7×10^{-43} (2.9×10^{-44})	2.3×10^{-39} (2.6×10^{-40})	4.2×10^{-20} (6.3×10^{-21})
Bohachevsky Function (*n*=2)			
CSA	6.2×10^{-8} (5.7×10^{-9})		
SA Method	3.0×10^{-1} (6.3×10^{-2})		
Hybrid Algorithm	2.6×10^{-14} (0.9×10^{-14})		

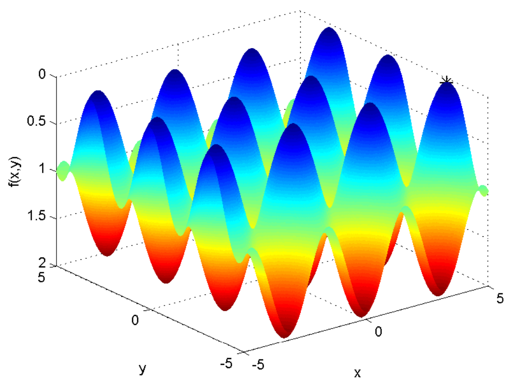
Table 3. Performance comparisons of three optimization methods in Function 1 (optima ± standard error).

Function 1

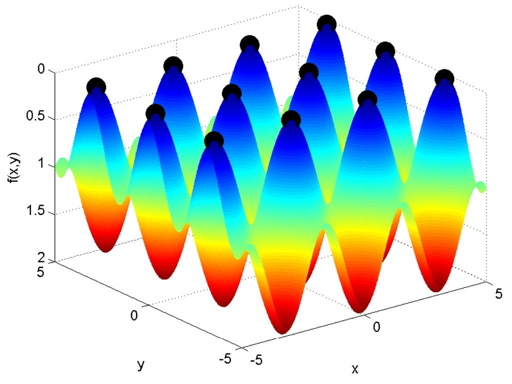
CSA	0 (12 optima)
SA Method	7.0×10^{-24} (1.9×10^{-24})
Hybrid Algorithm	0 (12 optima)



(a)



(b)



(c)

Fig. 5. Optimization results of Function 1 with CSA, SA method, and hybrid algorithm. (a) Optima obtained by CSA. (b) Optimum obtained by SA method. (c) Optima obtained by hybrid algorithm.

4.2. *Optimal Pressure Vessel Design*

The pressure vessel design is to minimize the total cost of the material, forming, and welding of a cylindrical vessel [12]. There are four design variables involved: x_1 (T_s, shell thickness), x_2 (T_h, spherical head thickness), x_3 (R, radius of cylindrical shell), and x_4 (L, shell length). The shell and spherical head thickness are the integer multipliers of 0.0625 in accordance with the available thickness of the rolled steel plates, and the radius of cylindrical and shell length have continuous values of $40 \le R \le 80$ and $20 \le L \le 60$, respectively. The mathematical formulation of this typical constrained optimization problem is as follows:

$$\mathrm{f}(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1611x_1^2x_4 + 19.84x_1^2x_3, \tag{11}$$

subject to

$$g_1(x) = 0.0193x_3 - x_1 \le 0,$$
$$g_2(x) = 0.00954x_3 - x_2 \le 0,$$
$$g_3(x) = 750\times 1728 - \pi x_3^2 x_4 - 0.75\pi x_3^3 \le 0,$$
$$g_4(x) = x_4 - 240 \le 0,$$
$$g_5(x) = 1.1 - x_1 \le 0,$$
$$g_6(x) = 0.6 - x_2 \le 0.$$

During the optimization process, the perturbed candidates resulting from mutation may potentially violate the constraints. The bounds of these candidates are checked after mutation. If they exceed the bounds, a new randomly chosen mutation parameter is used. This approach is called 'random re-initialization'. Figure 6 illustrates the average convergence procedure over 30 independent runs. The comparisons of the variable values, constraints, and objective functions with three earlier solutions from Sandgren [13], Wu & Chow [14], and Lee & Geem [12] are given in Table 4.

Table 4. Optimization comparisons of pressure vessel design.

	Sandgren	Wu and Chow	Lee and Geem	Our method
T_s	1.125	1.125	1.125	1.125
T_h	0.625	0.625	0.625	0.625
R	48.97	58.1978	58.2789	58.2891
L	106.72	44.2930	43.7549	43.6993
$g_1(x)$	-0.1799	-0.00178	-0.00022	-0.00020
$g_2(x)$	-0.1578	-0.06979	-0.06902	-0.0689
$g_3(x)$	97.760	-974.3	-3.71629	-8.9621
$g_4(x)$	-133.28	-195.707	-196.245	-196.3007
f(x)	7980.894	7207.494	7198.433	7197.831

Sandgren uses the branch and bound method, and achieve the result of 7980.894. However, the variable values do not satisfy the third constraint. Wu and Lee apply two different Genetic Algorithms (GA)-based approaches, and obtain the costs of 7207.494 and 7198.433, respectively. Apparently, the performances of the proposed hybrid optimization algorithm are better than those of the existing schemes.

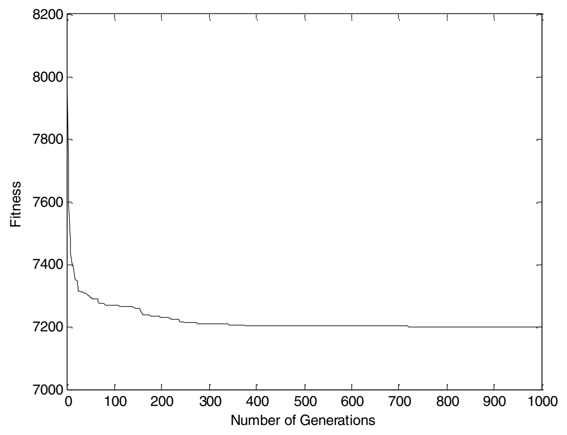


Fig. 6. Convergence procedure of hybrid optimization algorithm in pressure vessel design.

5. Conclusions

In this paper, a hybrid optimization method based on the fusion of the CSA and SA is proposed, and further examined with a few nonlinear optimization problems. The flexible global search ability of the SA and solution diversity feature of the CSA are fully utilized and combined in the new algorithm. Simulation results have demonstrated that our hybrid method achieves an enhanced optimization performance over the original CSA and SA algorithm. It can also acquire satisfactory results in providing diverse and flexible solutions to the multimodel problems. We are going to investigate its applications in a larger variety of engineering areas.

Acknowledgments

This research work was partially 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. D. Dasgupta, "Advances in artificial immune systems," *IEEE Computational Intelligence Magazine,* vol. 1, no. 4, pp. 40-49, November 2006.
2. 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.
3. X. Wang, X. Z. Gao, and S. J. Ovaska, "A novel particle swarm-based method for nonlinear function optimization," *International Journal of Computational Intelligence Research*, vol. 4, no. 3, pp. 281-289, 2008.
4. S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, May 1983.
5. D. Simopoulos, S. Kavatza, and C. Vournas, "Unit commitment by an enhanced simulated annealing algorithm," *IEEE Transactions on Power Systems*, vol. 21, no. 1, pp. 68-76, February 2006.
6. 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.
7. J. Timmis, P. Andrews, N. Owens, and E. Clark, "An interdisciplinary perspective on artificial immune systems," *Evolutionary Intelligence*, vol. 1, no. 1 pp. 2-26, 2008.
8. 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.
9. X. Z. Gao, X. Wang, and S. J. Ovaska, "Fusion of clonal selection algorithm and differential evolution method in training cascade-correlation neural network," *Neurocomputing*, vol. 72, no. 10-12, pp. 2483-2490, 2009.
10. L. N. de Castro and F. J. von Zuben, "Artificial immune systems: Part I—Basic theory and applications," Technical Report RT-DCA 01/99, FEEC/UNICAMP, Brazil, 1999.
11. A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. West Sussex, England: John Wiley & Sons Ltd, 2005.
12. K. Lee and Z. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36-38, pp. 3902-3933, 2005.
13. E. Sandgren, "Nonlinear integer and discrete programming in mechanical design optimization," *Journal of Mechanical Design*, vol. 112, pp. 223-229, 1990.
14. S. Wu and P. Chow, "Genetic algorithms for nonlinear mixed discrete-integer optimization problems via metagenetic parameter optimization," *Engineering Optimization*, vol. 24, pp. 137-159, 1995.