Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Fusion of clonal selection algorithm and differential evolution method in training cascade–correlation neural network

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

Department of Electrical Engineering, Helsinki University of Technology, Otakaari 5 A, FI-02150 Espoo, Finland

ARTICLE INFO

Article history:
Received 5 April 2008
Received in revised form
2 November 2008
Accepted 2 November 2008
Communicated by X. Li
Available online 14 November 2008

Keywords:
Artificial immune systems (AIS)
Clonal selection algorithm (CSA)
Differential evolution (DE)
Optimization
Cascade–correlation (C–C) neural network

ABSTRACT

In this paper, based on the fusion of the clonal selection algorithm (CSA) and differential evolution (DE) method, we propose a novel optimization scheme: CSA–DE. The DE is employed here to improve the affinities of the clones of the antibodies (Abs) in the CSA. Several nonlinear functions are used to verify and demonstrate the effectiveness of our hybrid optimization approach. It is further applied for the construction of the cascade–correlation (C–C) neural network, in which the optimal hidden nodes can be obtained.

I. Introduction

Natural immune systems are complex and enormous self-defense systems with the remarkable capabilities of learning, memory, and adaptation [1]. Artificial immune systems (AIS), inspired by the natural immune systems, are an emerging kind of soft computing methods [2]. With the features of optimization, pattern recognition, anomaly detection, data analysis, and machine learning, the AIS have recently gained considerable research interest from different communities [3]. As an important constituent of the AIS, artificial immune optimization (AIO) algorithms have been successfully applied to attack numerous challenging optimization problems with superior performances over the classical techniques [4]. Clonal selection algorithm (CSA) is one of the most widely employed AIO approaches [5]. 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. Unfortunately, empirical study has shown that the CSA usually suffers from a slow search speed. The differential evolution (DE) method is a simple but universal numerical optimizer [6]. All the individuals in the DE are updated by an amount of the difference between two randomly chosen ones. The DE has the distinguishing advantages of computation simplicity as well as convergence efficiency. In this paper, inspired by the fusion of the CSA and DE, we propose a novel optimization algorithm: CSA–DE. The CSA–DE is also used in the nonlinear function optimization and optimal training of the cascade–correlation (C–C) neural network. Simulations have demonstrated that our CSA–DE can outperform the original CSA with regard to the convergence speed.

The rest of this paper is organized as follows. We briefly discuss the essential principles of both the CSA and DE in Sections 2 and 3, respectively. In Section 4, by merging the CSA and DE together, we present a hybrid optimization method: CSA–DE, in which the affinities of the Ab clones of the CSA are improved by the DE. Moreover, our CSA–DE is deployed to acquire the optimal hidden nodes for constructing the C–C neural network in Section 5. Simulation examples are demonstrated in Section 6. Finally, in Section 7, we conclude the paper with some remarks and conclusions.

2. Clonal selection algorithm (CSA)

As we know, in the natural immune systems, only the antibodies (Abs) that can recognize the intruding antigens are selected to proliferate by cloning [7]. Therefore, the fundamental of the clonal optimization method is that those Abs capable of recognizing the non-self cells (antigens) will proliferate. More precisely, the underlying principles of the CSA borrowed from the CSP are

- maintenance of memory cells functionally disconnected from repertoire,
- selection and cloning of most stimulated Abs,

* Corresponding author. Tel.: +358 9 451 2434; fax: +358 9 451 2432.
E-mail addresses: gao@cc.hut.fi (X.Z. Gao), xiaolei@cc.hut.fi (X. Wang), seppo.ovaska@tkk.fi (S.J. Ovaska).
URL: http://powerelectronics.tkk.fi/ (X.Z. Gao).

- suppression of non-simulated cells,
- affinity maturation and reselection of clones with higher affinities, and
- mutation rate proportional to Ab affinities.

The diagram of the basic CSA is shown in Fig. 1, in which the corresponding iteration steps are explained as follows:

1. Initialize the Ab pool (P_{init}) including the subset of memory cells (M).
2. Evaluate the fitness of all the individuals in population P. The fitness here refers to the affinity measure.
3. Select the best candidates (P_r) from P, according to their fitness (affinities with the antigens).
4. Clone these Abs into a temporary pool (C).
5. Generate a mutated Ab pool (C_1). The mutation rate of each individual is *inversely* proportional to its fitness.
6. Evaluate all the Abs in C_1.
7. Eliminate those Abs similar to the ones in C, and update C_1.
8. Reselect the individuals with better fitness from C_1 to build M. Other improved individuals of C_1 can replace certain members with poor fitness in P for maintaining the overall Ab diversity.

The CSA can be terminated, when a preset performance criterion is met.

We should point out that the clone size in Step 4 is generally defined as a monotonic function of the affinity measure. 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. In other words, the better fitness an Ab has, the less it may change. The similarity among the Abs can also affect the convergence speed of the CSA. The idea of Ab suppression based on the immune network theory is introduced to eliminate the newly generated Abs, which are too similar to those already in the candidate pool (Step 7). With such a diverse Ab pool, the CSA can avoid being trapped into local optima. In contrast to the popular genetic algorithms (GA), which usually tend to bias the whole population of chromosomes towards only the best candidate solution [8], it can effectively handle the challenging multimodal optimization tasks [9–12].

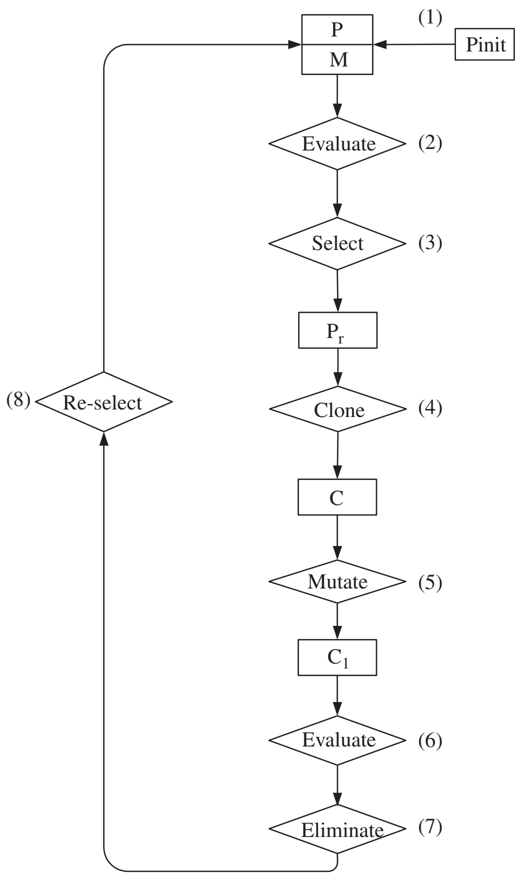

Fig. 1. Diagram of basic clonal selection algorithm (CSA).

3. DE method

The DE method is a robust population-based optimization technique firstly proposed by Storn and Price [6]. It has been applied to cope with a large variety of engineering problems in parameter identification [13], power system planning [14], data clustering [15], etc. The principle of the DE is similar to that of other evolutionary computation methods, e.g., the GA. However, the uniqueness of the DE is that it generates new chromosomes by adding the weighted difference between two chromosomes to the third. If the fitness of the resulting chromosome is improved, this newly generated chromosome replaces the original one. Suppose there are three chromosomes, $r_1(k)$, $r_2(k)$, and $r_3(k)$, under consideration in the current population. Note, $r_1(k)$ and $r_2(k)$ are randomly selected and mutually different. Fig. 2 depicts that a trial update of $r_3(k)$, $r'_3(k+1)$, is

$$r'_3(k+1) = r_3(k) + \lambda[r_1(k) - r_2(k)], \tag{1}$$

where λ is a predetermined weight. In order to further increase the diversity of these chromosomes, a crossover operator is employed to generate $r''_3(k+1)$ by randomly combining the parameters of $r_3(k)$ and $r'_3(k)$ together. If $r''_3(k+1)$ yields a higher fitness than $r_3(k)$, we get

$$r_3(k+1) = r''_3(k+1). \tag{2}$$

Otherwise, $r''_3(k+1)$ is eliminated, and the above procedure restarts until all the chromosomes have been successfully updated. In fact, the difference between two chromosomes is an estimation of the gradient information in the zone, where both chromosomes belong to. Therefore, the DE method can be regarded as a simple gradient descent-based stochastic search scheme.

During the recent years, hybridization of evolutionary algorithms has gained growing popularity, which can overcome their individual drawbacks while benefit from each other's strengths [16–19]. As aforementioned, the affinity-related Ab cloning and somatic mutation are two interesting features of the CSA, and the DE method has the remarkable advantages of effective search and computation simplicity. Thus, in this paper, we propose a new fusion of the CSA and DE: CSA–DE. The CSA–DE has been

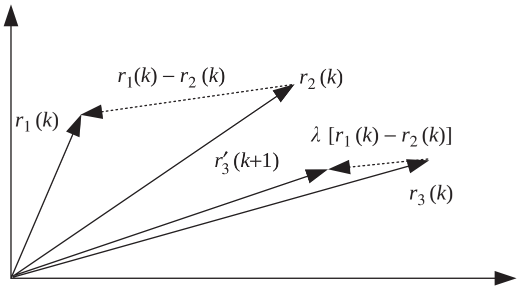

Fig. 2. Principle of differential evolution (DE) method.

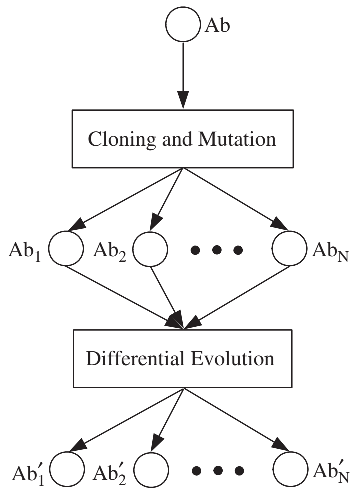

Fig. 3. DE-based optimization of Ab clones in CSA–DE.

demonstrated to significantly outperform the original CSA in optimization.

4. Fusion of CSA and DE method: CSA–DE

In the CSA, the cloning and mutation of the Abs have the important role of 'blind' search because of their random characteristics. However, such a blind search is not always sufficiently efficient for dealing with the demanding complexity in real-world optimization problems. Combining these two operations and gradient-based local search strategies can indeed improve the convergence of the regular CSA. Unfortunately, the gradient information is often difficult if not impossible to obtain in practice. The CSA–DE, a fusion of the CSA and DE, is proposed here on the basis of the estimated gradient directions from the DE method. The principle of our CSA–DE is that all the clones of the CSA Abs are updated and optimized using the DE method, as illustrated in Fig. 3. Suppose an Ab in the CSA has been cloned and mutated into N clones, $\mathrm{Ab}_1, \mathrm{Ab}_2, \ldots, \mathrm{Ab}_N$. These Ab clones can be alternatively considered as the individuals in a population of the DE. For example, Ab_1 is updated to Ab'_1 after a given number of the DE iterations. Hence, the resulting $\mathrm{Ab}'_1, \mathrm{Ab}'_2, \ldots, \mathrm{Ab}'_N$ are expected to occupy their higher affinities than that of $\mathrm{Ab}_1, \mathrm{Ab}_2, \ldots, \mathrm{Ab}_N$, and they will continue the normal CSA evolution.

The proposed CSA–DE has two remarkable features. Firstly, in the original CSA, the Ab clones are generally independent, and have no interaction with each other. In our CSA–DE, aided by the DE method, they make full use of the information sharing and exchange among themselves, and work collectively to improve their affinities. This approach can efficiently prevent the harmful prematurity of the CSA. Secondly, to build the CSA–DE, we do not need to make any major modification on the regular CSA. The DE method is embedded into the CSA as an 'internal' fine-tuning procedure. Thus, the implementation of our CSA–DE is fairly easy and straightforward. We will next discuss its application in the C–C neural network training in Section 5.

5. CSA–DE in C–C neural network training

5.1. C–C neural network

During the past decade, neural networks have been widely employed to deal with difficult real-world problems, e.g., human face detection [20] and exchange rate forecasting [21]. The C–C is an adaptive learning algorithm for the self-growing feedforward neural network [22,23]. The C–C neural network has found intensive applications in such areas as noise cancellation [24] and harmonic source detection [25]. Compared with the conventional back-propagation (BP) neural network, the C–C neural network does not have a fixed size. Instead, it grows from the smallest structure with no hidden nodes, and the hidden nodes are actually added one by one until a given performance criterion, e.g., the maximal number of training epochs or minimal approximation error, is satisfied. Fig. 4(a) shows an example of the initial C–C neural network (without any hidden nodes) with three inputs and one output. This C–C neural network is first trained with the available learning data. When there is no significant reduction in the approximation error or a certain number of training epochs have been reached, the training procedure is terminated, and all the weights obtained are frozen. The residual approximation error will next be eliminated by adding extra hidden nodes, as illustrated in Fig. 4(b). The strategy of incrementally cascading the hidden nodes to the C–C neural network is explained as follows. The new hidden nodes are cascaded with the network input nodes as well as existing hidden nodes. Therefore, the weights of the hidden nodes to be trained consist of two parts: input weights connecting with both the input nodes and preexisting hidden nodes and output weights connecting with the output nodes. The input weights of the hidden nodes are trained with the learning data to maximize S, which is the sum over all the output nodes of the magnitude of the correlation between V_p, the hidden nodes' outputs, and $E_{p,o}$, the residual output error at output node o. S can be defined as

$$S = \sum_o \left| \sum_p (V_p - \bar{V})(E_{p,o} - \bar{E}_o) \right|, \tag{3}$$

where p is the index of the training patterns, and $\bar{V}$ and $\bar{E}_o$ are the averaged values of V and E_o over all the training patterns, respectively. After the above training phase is finished, the input weights of the hidden nodes are also frozen, and they are cascaded

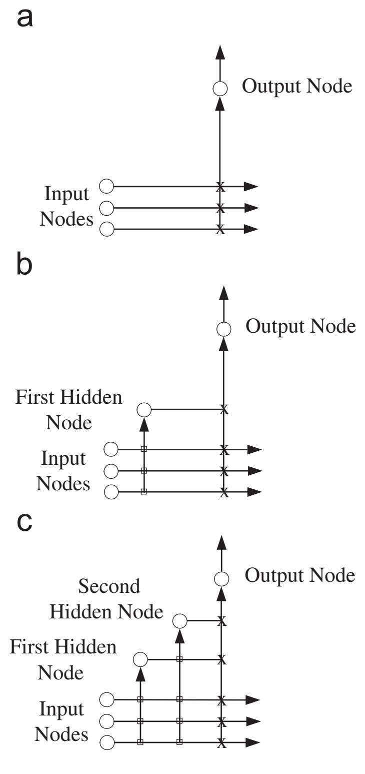

Fig. 4. Cascade–correlation (C–C) neural network training.

to the output nodes in the C–C neural network. The output weights of these new hidden nodes are further updated using the regular BP learning algorithm to minimize the network output error. Note that the update of all the input and output weights of the hidden nodes is based on only a single-layer structure of the feedforward neural network. This iterative procedure of involving more and more hidden nodes is repeated, as shown in Fig. 4(c), so as to achieve a satisfactory approximation performance.

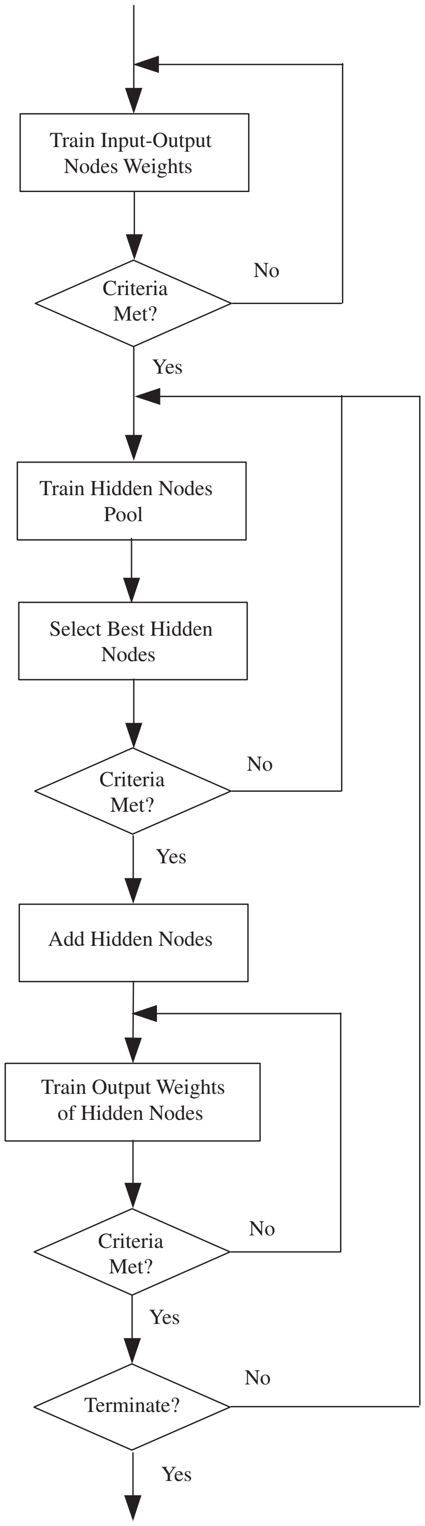

Fig. 5. Construction of C–C neural network.

We emphasize that a pool of hidden nodes containing a certain number of candidates with different initial input weights is usually constructed and trained in order to choose the optimal hidden nodes that can maximize (3) [22]. After the most suitable hidden nodes are selected from the pool, they are connected to the existing network. The flow chart of the C–C neural network construction is given in Fig. 5. However, since the BP method is used to acquire the input weights of all the hidden node candidates in the pool, the shortcoming of being trapped into the local optima during training is not avoidable. In other words, it cannot be guaranteed that the best hidden nodes are always obtained for constructing the C–C neural network. To handle this difficult nonlinear optimization problem, we apply our CSA–DE to optimize the input weights of the hidden nodes in the regular C–C training.

5.2. *CSA–DE in optimal C–C neural network construction*

Instead of utilizing a large candidate pool to select the appropriate hidden nodes, we can employ the CSA–DE to search for the hidden nodes with the best input weights. To put it into more details, each candidate set of the hidden node is encoded as an Ab in our approach. With the growth of the C–C neural network, the number of the input weights of the hidden nodes increases, and the size of the Abs is also variant. All the Abs with the affinities defined in (3) evolve in the CSA–DE. The ultimate optimization goal is to find those hidden nodes, which can maximize (3), so that the size of the C–C neural network (number of hidden nodes) is minimized. After the optimal hidden nodes are obtained, they are first connected to the present C–C neural network, and their output weights are next trained with the BP learning method. The flowchart of our CSA–DE-based optimal C–C neural network construction is shown in Fig. 6.

Due to the global optimization capability of the CSA–DE, the size of the resulting neural network can be smaller than that from using the regular C–C training method. In Section 6, we deploy the popular two-spirals problem as a challenging testbed to examine this novel scheme.

6. Simulations

In this section, we investigate the effectiveness of the proposed CSA–DE method with two simulation examples: nonlinear function optimization and C–C neural network training.

6.1. *Nonlinear function optimization*

The following five n-dimension nonlinear functions ($n = 50$), which have been extensively used as the optimization benchmarks [26], are employed here to compare the optimization (minimization) capabilities between the CSA and our CSA–DE:

Ackley function:

$$f(x) = -20e^{-0.2\sqrt{1/n\sum_{i=1}^{n}x_i^2}} - e^{1/n\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 - e, \quad x \in [-32, 32]. \tag{4}$$

Griewank function:

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}} + 1, \quad x \in [-100, 100]. \tag{5}$$

Rastrigin function:

$$f(x) = \sum_{i=1}^{n}x_i^2 + 10 - 10\cos(2\pi x_i), \quad x \in [-5.12, 5.12]. \tag{6}$$

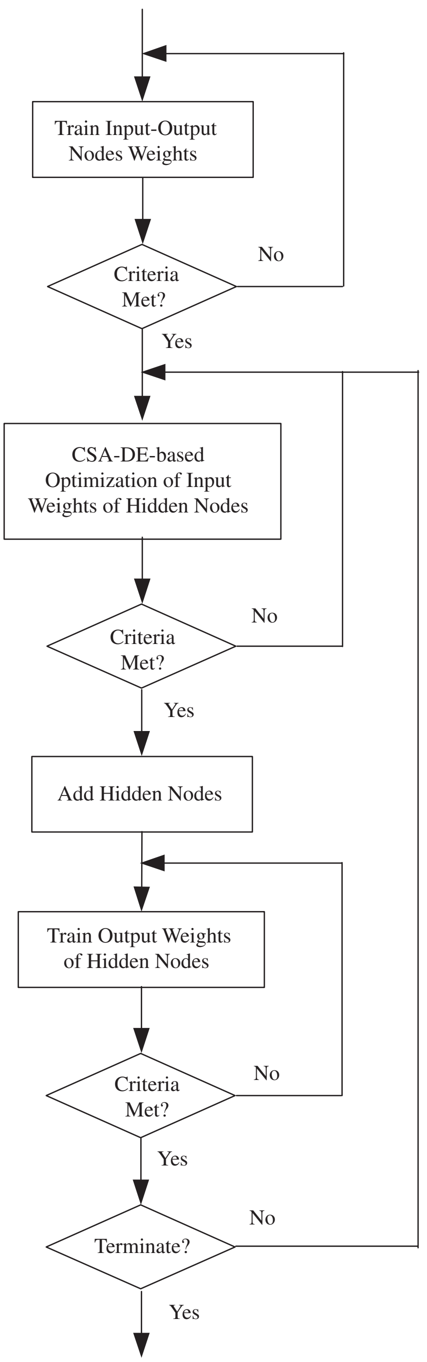
Train Input-Output Nodes Weights
Criteria Met?
No
Yes
CSA-DE-based Optimization of Input Weights of Hidden Nodes
Criteria Met?
No
Yes
Add Hidden Nodes
Train Output Weights of Hidden Nodes
Criteria Met?
No
Yes
Terminate?
No
Yes

Fig. 6. CSA–DE in optimal C–C neural network construction.

Rosenbrock function:

$$f(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \quad x \in [-10, 10]. \tag{7}$$

Sphere function:

$$f(x) = \sum_{i=1}^{n} x_i^2, \quad x \in [-100, 100]. \tag{8}$$

It is well known that the global minima of the above functions are all at $f(x) = 0$. There are totally 10 Abs involved in both the CSA and CSA–DE. Their evolution procedures are terminated after 10,000 iterations. The optimal solutions acquired by the two methods are given in Table 1. We stress that the results are based on the average of 100 independent trials. As an illustrative example, the optimal solutions to the Rosenbrock function from the CSA and CSA–DE during these trials are shown in Figs. 7(a) and (b), respectively. Apparently, compared with the original CSA, our CSA–DE can acquire much better optimization results within the same numbers of iterations, because of the DE-based effective refinement of the Abs. That is, the CSA–DE has a superior nonlinear function optimization performance over the CSA.

Furthermore, we compare the convergence speeds of the CSA and CSA–DE using the same nonlinear functions. A minimization goal is set beforehand for the optimization of each function. The optimization procedure is stopped after the goal has been

Table 1
Optimal solutions acquired by CSA and CSA–DE in function optimization.

Function type	CSA	CSA–DE
Ackley	19.7493	1.0419×10^{-4}
Griewank	135.7764	115.0552
Rastrigin	455.9070	9.4486×10^{-6}
Rosenbrock	1.5870×10^{6}	48.8484
Sphere	7.6837×10^{4}	4.5101×10^{-4}

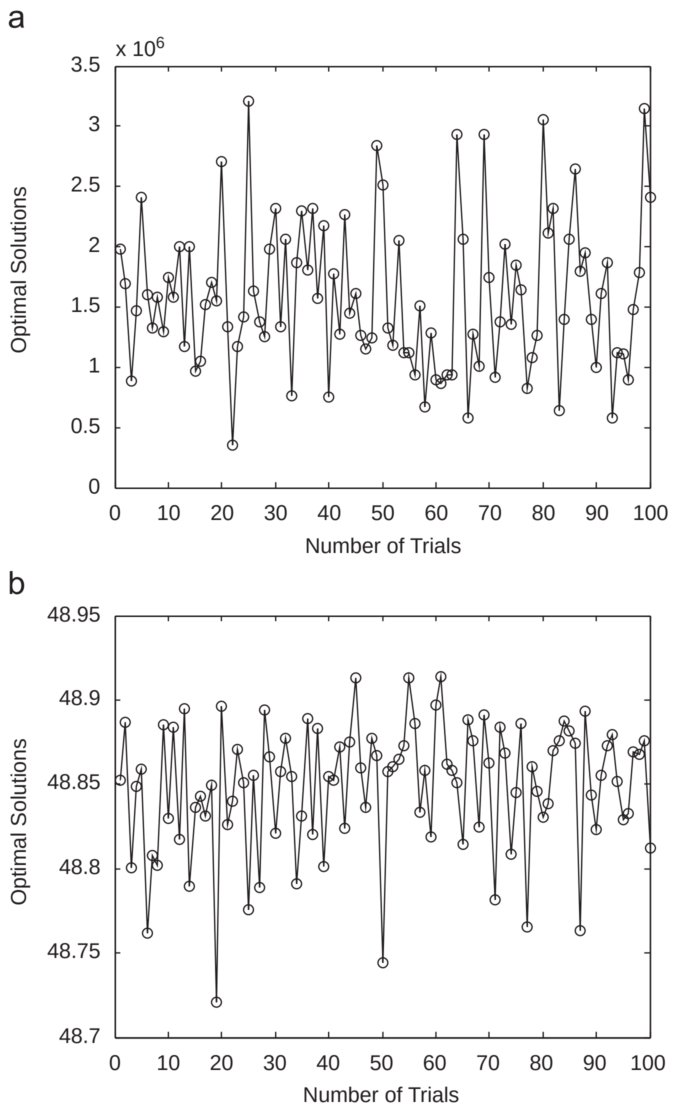

Fig. 7. Optimal solutions in Rosenbrock function optimization: (a) CSA and (b) CSA–DE.

achieved, and the corresponding calculation time elapsed is recorded. The simulations are made under the MATLAB 7.0 environment on a Dell Optiplex GX 745 computer with a 2.4 GHz Core 2 Duo E6600 CPU and 2 G system memory. Again, 100 separate trials have been run. Table 2 shows the average calculation time of the CSA and CSA–DE in this convergence performance comparison. To simplify our presentation, we only illustrate the calculation time of the CSA and CSA–DE in the Rosenbrock function optimization in Figs. 8(a) and (b), respectively. From Table 2 and Fig. 8, we can observe that although the DE method moderately increases the computational complexity of the CSA–DE, it can still converge much faster than the CSA in optimizing those nonlinear functions. As a matter of fact, the incorporation of the DE into the CSA significantly improves the optimization effectiveness of our new hybrid technique.

Table 2
Calculation time (in seconds) of CSA and CSA–DE in function optimization.

Function type	Minimization goal	CSA	CSA–DE
Ackley	20	26.6773	0.1159
Griewank	150	45.6679	0.1129
Rastrigin	500	47.6159	0.0924
Rosenbrock	1×10^6	358.8081	0.0643
Sphere	1×10^5	44.4684	0.0657

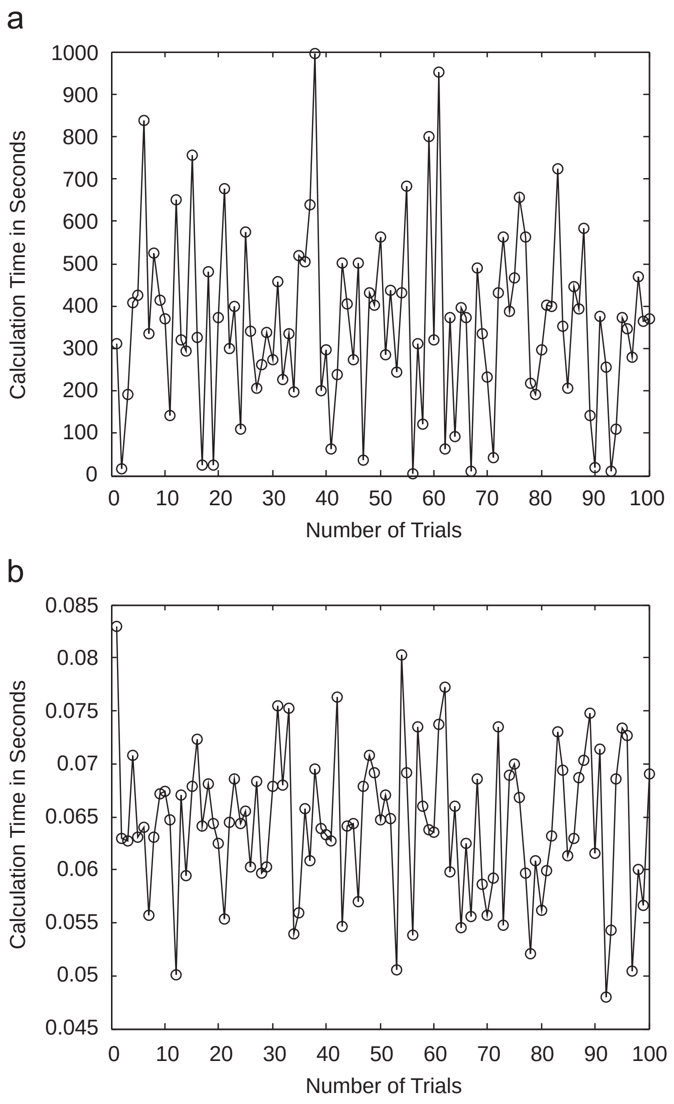

Fig. 8. Calculation time in Rosenbrock function optimization: (a) CSA and (b) CSA–DE.

The above optimization results of typical nonlinear functions clearly demonstrate our CSA–DE has a much better convergence property than that of the original CSA. It is well capable of not only converging faster but also achieving improved optimal solutions. The considerable optimization performance enhancement of the CSA–DE is due to the employment of the DE to increase the affinities of all the Abs in the CSA. In other words, the DE-based approach can yield those Abs with higher fitness, which lay an improved basis for the consequent evolution of the CSA. The CSA–DE will be next examined in the optimal training of the C–C neural network to attack the difficult two-spirals problem.

6.2. C–C neural network optimization in two-spirals problem

The two-spirals problem is a popular benchmark for the data classification methods [27], which consists of 194 pairs of X–Y samples. One half of these samples belong to Class 1, and the other half Class 2, as shown in Fig. 9. In this simulation, we target at constructing an optimal C–C neural network with the smallest size (number of hidden nodes) that can correctly classify the two-spirals samples. In [22], Fahlman and Lebiere apply the C–C method to solve the two-spirals problem. A pool of eight hidden node candidates is deployed in their experiments. For each trial, the best-trained hidden nodes are selected from the pool and incrementally added to the C–C neural network until the two-spirals samples are properly classified. All the 100 trial runs are successful, in which the number of the resulting hidden nodes varies from 12 to 19 with an average of 15.2. In our CSA–DE-based optimization approach, we also use 10 Abs, as in the function optimization example, to optimize the hidden nodes, and the iterations are limited to 5000 steps. The range of the numbers of the optimal hidden nodes acquired is from 9 to 18 with an average of 12.9. Fig. 10 illustrates the trials-related distributions of the hidden nodes from the C–C and CSA–DE-based methods. Table 3 gives the relationship between the number of hidden nodes and number of trials in both two schemes. Obviously, for this two-spirals problem, our CSA–DE can outperform the C–C method. The average size of the C–C neural network obtained by the former is much smaller than that by the later, i.e., 12.9 vs. 15.2, because the pure gradient descent-based C–C method is usually trapped into

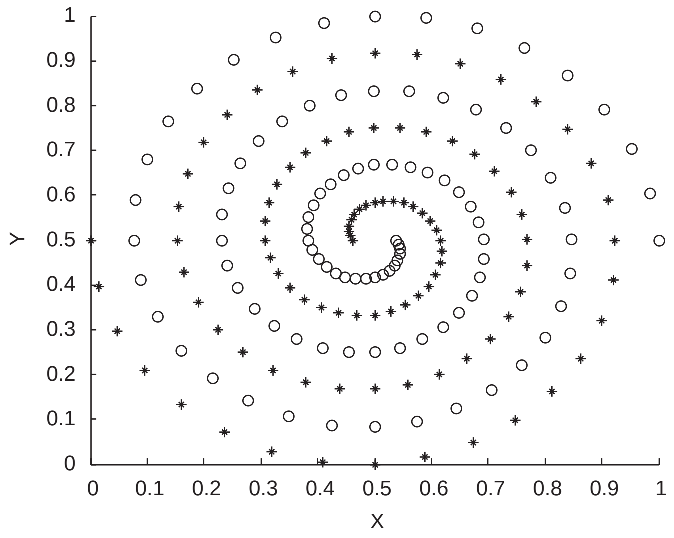

Fig. 9. Samples in two-spirals problem: '○': Class 1 and '*': Class 2.

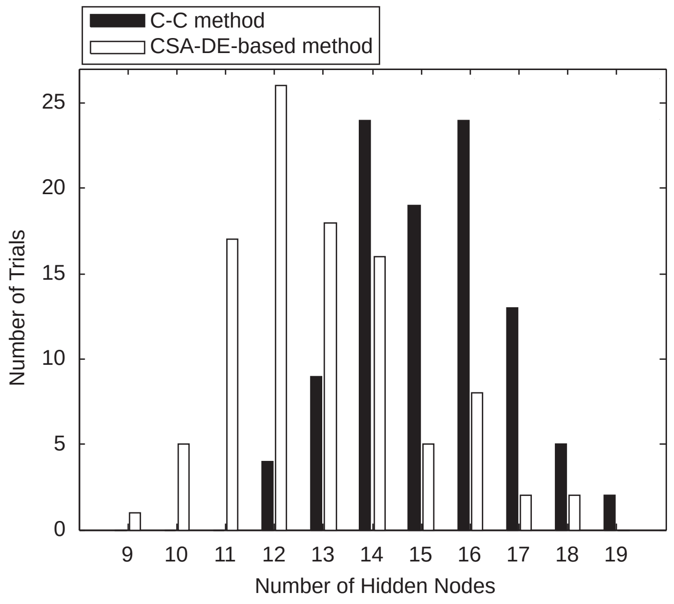

Fig. 10. Distributions of sizes of C–C neural network in two-spirals problem.

Table 3
Training of C–C neural network in two-spirals problem.

Number of hidden nodes	Number of trials (C–C approach)	Number of trials (CSA–DE-based approach)
9	0	1
10	0	5
11	0	17
12	4	26
13	9	18
14	24	16
15	19	5
16	24	8
17	13	2
18	5	2
19	2	0

the local optima. However, as aforementioned, only a single-layer feedforward neural network needs to be trained in the C–C method, while the evaluation, cloning, and mutation of the Abs in the CSA–DE are rather time-consuming. Therefore, compared with the C–C method, the major drawback of our CSA–DE-based approach is its relatively high computational complexity, which is mainly caused by the calculation of (3) for the Abs evolution.

6.3. *Discussions of simulation results*

The proposed CSA–DE has been validated using examples of nonlinear function optimization and C–C neural network training in this section. As we can observe, the CSA–DE has the remarkable advantage of fast convergence speed. Unfortunately, there are two shortcomings with regard to its computational complexity and convergence analysis. The computational complexity of our CSA–DE is moderately higher than that of the regular CSA, because the DE embedded is indeed a simple optimization method. Nevertheless, the computational complexity of the CSA–DE can be reduced by using less DE iterations, which may, on the other hand, lead to the overall optimization performance deterioration. Moreover, the convergence of the CSA–DE still needs a comprehensive theoretical analysis, and should be examined with more engineering problems.

7. Conclusions

In this paper, based on the fusion of the CSA and DE, we propose a hybrid optimization method: CSA–DE. The Abs in the CSA are fine-tuned by the DE to improve their affinities so that enhanced optimization performances can be achieved. We also discuss the application of the new CSA–DE in the optimal construction of the C–C neural network. Two simulation examples, nonlinear function optimization and C–C neural network training, have been employed to verify the effectiveness of the proposed hybrid technique. Compared with the original CSA, better optimization results are obtained with our CSA–DE. However, a few important issues of the CSA–DE, such as analysis of convergence and computational complexity, need to be further explored. In addition, we are going to investigate how to apply it for manipulating the real-world problems.

Acknowledgments

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

References

[1] G.A. Goldsby, T.J. Kindt, J. Kuby, B.A. Osborne, Immunology, fifth ed., Freeman, New York, NY, 2003.
[2] L.N. de Castro, J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer, London, UK, 2002.
[3] D. Dasgupta, Advances in artificial immune systems, IEEE Computational Intelligence Magazine 1 (4) (2006) 40–49.
[4] X. Wang, X.Z. Gao, 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.
[5] L.N. de Castro, F.J. von Zuben, Learning and optimization using the clonal selection principle, IEEE Transactions on Evolutionary Computation 6 (3) (2002) 239–251.
[6] R. Storn, K. Price, Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[7] G.L. Ada, G.J.V. Nossal, The clonal selection theory, Scientific American 257 (2) (1987) 50–57.
[8] R. Poli, W.B. Langdon, Foundations of Genetic Programming, Springer, Berlin, Germany, 2002.
[9] J. Yoo, P. Hajela, Immune network simulations in multicriterion design, Structural Optimization 18 (2–3) (1999) 85–94.
[10] X. Wang, X.Z. Gao, 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. Xu, J. Zhang, An improved immune evolutionary algorithm for multimodal function optimization, in: Proceedings of the Third International Conference on Natural Computation, Haikou, China, August 2007, pp. 641–646.
[12] T. Tang, J. Qiu, An improved multimodal artificial immune algorithm and its convergence analysis, in: Proceedings of the Sixth World Congress on Intelligent Control and Automation, Dalian, China, June 2006, pp. 3335–3339.
[13] M. Toman, G. Stumberger, D. Dolinar, Parameter identification of the Jiles–Atherton hysteresis model using differential evolution, IEEE Transactions on Magnetics 44 (6) (2008) 1098–1101.
[14] G.Y. Yang, Z.Y. Dong, K.P. Wong, A modified differential evolution algorithm with fitness sharing for power system planning, IEEE Transactions on Power Systems 23 (2) (2008) 514–522.
[15] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Transactions on Systems, Man, and Cybernetics—Part A 38 (1) (2008) 218–237.
[16] C. Grosan, A. Abraham, H. Ishibuchi (Eds.), Hybrid Evolutionary Algorithms, Springer, Berlin, Germany, 2007.
[17] X. Wang, X.Z. Gao, S.J. Ovaska, A hybrid optimization algorithm based on ant colony and immune principles, International Journal of Computer Science and Applications 4 (3) (2007) 30–44.
[18] X. Wang, X.Z. Gao, S.J. Ovaska, A novel particle swarm-based method for nonlinear function optimization, International Journal of Computational Intelligence Research 4(3) (2008), in press.
[19] X. Wang, X.Z. Gao, S.J. Ovaska, A simulated annealing-based immune optimization method, in: Proceedings of the International and Interdisciplinary

Conference on Adaptive Knowledge Representation and Reasoning, Porvoo, Finland, September 2008, pp. 41–47.
[20] K. Curran, X. Li, N. McCaughley, Neural network face detection, The Imaging Science Journal 53 (2) (2005) 105–115.
[21] L. Yu, K.K. Lai, S. Wang, Multistage RBF neural network ensemble learning for exchange rates forecasting, Neurocomputing 71 (16–18) (2008) 3295–3302.
[22] S.E. Fahlman, C. Lebiere, The cascade–correlation learning architecture, Technical Report, CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, 1991.
[23] J.J.T. Lahnajärvi, M.I. Lehtokangas, J.P.P. Saarinen, Evaluation of constructive neural networks with cascaded architectures, Neurocomputing 48 (1–4) (2002) 573–607.
[24] J. Dheeba, A. Padma, Intelligent adaptive noise cancellation using cascaded correlation neural networks, in: Proceedings of the International Conference on Signal Processing, Communications and Networking, Chennai, India, February 2007, pp. 178–182.
[25] W.-M. Lin, C.-H. Lin, K.-P. Tu, C.-H. Wu, Multiple harmonic source detection and equipment identification with cascade correlation network, IEEE Transactions on Power Delivery 20 (3) (2005) 2166–2173.
[26] S. Janson, M. Middendorf, A hierarchical particle swarm optimizer and its adaptive variant, IEEE Transactions on Systems, Man, and Cybernetics—Part B 35 (6) (2005) 1272–1282.
[27] K.J. Lang, M.J. Witbrock, Learning to tell two spirals apart, in: Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA, June 1988, pp. 52–59.

Xiao-Zhi Gao received his B.Sc. and M.Sc. degrees from the Harbin Institute of Technology, China in 1993 and 1996, respectively. He earned a D.Sc. (Tech.) degree from the Helsinki University of Technology, Finland in 1999. Since January 2004, he has been working as a Docent at the same university. Dr. Gao has published more than 150 technical papers on refereed journals and international conferences. He is an Associate Editor of the *Journal of Intelligent Automation and Soft Computing* and an editorial board member of the *Journal of Applied Soft Computing, International Journal of Bio-Inspired Computation* and *Journal of Hybrid Computing Research*. Dr. Gao was the General Chair of the *2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*. His current research interests are neural networks, fuzzy logic, evolutionary computing, swarm intelligence and artificial immune systems with their applications in industrial electronics.

Xiaolei Wang received her B.Sc. degree in Technical Physics from the Heilongjiang University, China in August 2001. She earned the M.Sc. degree (Hons.) from the Helsinki University of Technology, Finland in February 2005. She is now working towards her D.Sc. (Tech.) degree at the same university.

Seppo J. Ovaska received his D.Sc. (Tech.) in electrical engineering from Tampere University of Technology, Finland, in 1989. He is currently a Professor in the Faculty of Electronics, Communications, and Automation at Helsinki University of Technology, Finland. Before joining Helsinki University of Technology in 1996, he was a Professor in the Department of Information Technology at Lappeenranta University of Technology, Finland (1992–1996). From 1980 to 1992, he held engineering, research, and R&D management positions with Kone Elevators and Nokia Research Center, both in Finland and in the United States. In the academic year of 2006–2007, he was a Visiting Professor of electrical and computer engineering at Utah State University. His research interests are in computationally intelligent hybrid systems, evolutionary computation, artificial life, and artificial immune systems. During his career, he has published more than 80 journal articles and more than 140 conference publications. He edited the book ''Computationally Intelligent Hybrid Systems: The Fusion of Soft Computing and Hard Computing'' (Wiley-Interscience, 2004). Dr. Ovaska served as an associate/guest editor for several IEEE journals, including a guest-editorship for *Proceedings of the IEEE*. In addition, he was the coordinating chair of technical committees in the area of systems, and an elected member of the board of governors, IEEE Systems, Man, and Cybernetics Society. He is a recipient of two Outstanding Contribution Awards, and the Most Active SMC Technical Committee Award of the IEEE Systems, Man, and Cybernetics Society.