A Self-Learning Simulated Annealing Algorithm for Global Optimizations of Electromagnetic Devices

Shiyou Yang, Jose Marcio Machado, Guangzheng Ni, S. L. Ho, and Ping Zhou

***Abstract*—A self-learning simulated annealing algorithm is developed by combining the characteristics of simulated annealing and domain elimination methods. The algorithm is validated by using a standard mathematical function and by optimizing the end region of a practical power transformer. The numerical results show that the CPU time required by the proposed method is about one third of that using conventional simulated annealing algorithm.**

***Index Terms*—Domain elimination method, global optimization, self-learning ability, simulated annealing algorithm.**

I. Introduction

GENERALLY the optimization of electromagnetic problems can be classified commonly as constrained, multimodal, and nonlinear programming ones. To determine a local optimum point, one only needs to test if it satisfies the Karush–Kuhn–Tucker conditions which arise from the first-order necessary conditions. But for a global optimum point, there is no mathematical condition to characterize the differences between global and local ones. Therefore, the global optimization problems are much harder to solve. Many authors have even concluded that this kind of problems is unsolvable from a mathematical point of view. Thanks to developments both in computer science and numerical techniques, one could however use numerical approaches to find an approximate solution in practice.

Robust algorithms for local minimization are well developed and are readily available. For global optimization problems, the competitive performance of "general purpose" optimization algorithms concerning both the robustness and the speed is still a challenging topic, although several stochastic algorithms such as simulated annealing, genetic, and tabu search methods, all have been investigated and applied in the literatures [1]–[4]. Nevertheless, the convergence speeds of all these algorithms are still very low. Therefore it is essential to search for improved global optimization methods. One feasible approach for developing new algorithms is to combine the conventional global optimization algorithms in such a way that their offsprings are superior to their ancestors in terms of both robustness and convergence speeds.

This paper focuses on the development of a self-learning simulated annealing (SA) algorithm. For this purpose, the concept of the memorization of the previously searched history are incorporated into conventional simulated annealing algorithms. Numerical applications of the proposed algorithm demonstrate its superiority over traditional SA methods.

Manuscript received October 25, 1999. The work of S. Yang was supported by FAPESP, Brazil. This work was supported in part by the National Natural Science Foundation of China under the Project 59877022.

S. Yang and S. L. Ho are with the EE Department, Hong Kong Polytechnic University (e-mail:yangwj@looksmart.com; eeslho@polyu.edu.hk).

J. M. Machado is with the Computer Science and Statistics Department, Sao Paulo State University, Sao Jose do Rio Preto - SP, Brazil.

G. Ni is with the EE Department, Zhejiang University, China.

P. Zhou is with the Ansoft Corporation, USA.

Publisher Item Identifier S 0018-9464(00)06391-3.

II. Self-Learning Simulated Annealing Algorithm

As it is well known, SA algorithm is a Markov chain Monte Carlo method, and is therefore, memoryless. By definition, the distribution of the random variable $x(t+1)$ is mediated entirely by the value of $x(t)$: the past history does not influence the current move, i.e. the information accumulated in the searched parameter space is not used to guide the generation of new states. This will, however, be very computationally inefficient, especially in the case where new states are very near or equal to previously searched ones. Thus different studies on the memorization of the searched spaces are reported [7].

As with other stochastic methods, the domain elimination method has also two phases, the global phase where the cost function is evaluated at a number of randomly sampled points in the feasible domain, and a local phase where the sample points are manipulated (by means of local search) to yield a candidate global points. This algorithm can be viewed as a modification of a multi-start procedure but, unlike simulated annealing, it has the ability to learn as the search progresses by means of comparing the new states with those memorized in the sets such as the starting point set, the local minimum point set, the rejected point set, etc. Hence the proposed algorithm is expected to be more efficient. The details about this method are already reported in [5] and in a complementary paper written by the authors [6].

Motivated by the novel characteristics of domain elimination method, the inclusion of the memorization of the searched histories into conventional SA algorithms is reported in this paper. Hence the proposed algorithm would then have self-learning abilities during the searching process. In order to improve the search efficiency, a local phase is also introduced into the self-learning SA algorithm.

A. History Memorization

Just as mentioned before, one of the main disadvantages of the conventional simulated annealing algorithm is the "blindness" in generating new random moves. In other words, the algorithm does not "know" whether the new random moves had been explored previously, or whether they can lead to some new

local optima. In order to improve the searching efficiency, one should try to develop an enhanced algorithm which does possess a self-learning or memorization ability. The memorization of the searched subspaces in the parameter spaces is realized by memorizing the following sets in this paper:

Starting point set: $X_0 = \{x | x = x^0;\ x^0$ is a starting point when executing a local optimization procedure$\}$.

Local optimum set: $X_l = \{x | x = x^l;\ x^l$is a local optimum point searched by the local optimization procedure$\}$

Trajectory set: $X_t = \{Y^i | Y^i = (x^{i1}, x^{i2}, \cdots, x^{im});\ x^{ik}(k = 1, 2, \cdots, m)$ is the kth sample point in the ith trajectory from the start point x^{i0} to the ith local optimum point x^{il}.

Here the trajectory means the searching history of the local optimization procedure from a starting point to the corresponding local optimum point. For practical engineering problems, there could be many different trajectories meeting at one local optimum point.

B. Rejection of Random Moves

In order to determine if a new random move (point) has already been searched or if it can lead to a new local optimum, the random point, denoted using x^r, is compared with the points stored in the three sets X_0, X_l, and X_t. If it is equal to or near a point or a line segment of some trajectories comprising any of the three mentioned sets, i.e., the distance between them is within a certain critical value, the proposed point is discarded and a new random point is generated again. Only if the distances from the proposed points to all the points or trajectory segments of the aforementioned sets satisfy the distance condition, the self-learning SA algorithm can start from this point to begin a new cycle of iterations. Such checking avoids unnecessary minimization steps that would otherwise lead to an already known local optima, and tends to enforce the algorithm to explore the un-searched subspaces. Hence unnecessary computations are avoided with no negative effects. Since the generation of a random point and the evaluation of it are much cheaper than beginning a new iterative cycle with this kind of points, the self-learning SA algorithm will increase the computation efficiency and enhance significantly the chance of finding a new local optimum in the unexplored parameter spaces.

C. Trajectory Representation and Distance Computation

A trajectory is represented by passing linear straight-line segments through the sampled points along the trajectory in this paper. In order to reduce the storage requirements, the procedure will store the points once after every M points. When checking the proximity of two points, a hyper-prism around the specified point x^s (a point belonging to one of the three sets mentioned above) is constructed. The proposed point x^r is rejected if it lies inside the hyper-prism. In checking the distance between a proposed point and a trajectory, the distance from the proposed point to the nearest line section of the trajectory is used. The proposed point is rejected if this distance is smaller than a critical distance D_{cr}.

D. Speed up Comparison Efficiency

Although the CPU time required for comparing the distances between the random moves and points or trajectory segments stored in the three aforementioned sets are very small compared with those needed for starting new iterative cycles with this kind of points, the accumulation of the overall time is still significant, especially for the case where the quantities in the three sets are very large. In order to reduce the computing time, the following approaches are proposed:

1) the quantities of set X_0 and set X_l are stored in a sequence from the maximum to the minimum values of the cost functions;
2) the trajectories of set X_t are stored in such a way that the cost function values of the starting points are reduced sequentially;
3) Besides storing the variable x and the corresponding cost function for those points of X_0 and X_l, an additional variable, which is the sum of every point in all directions, denoted as $x_{\sum}$, is also memorized.

Due to the nature of the data structures as defined by (1) and (2), one can use the well known fast comparison algorithms such as bisect search algorithm to reduce the comparison space of sets X_0, X_l and X_t significantly. For the subspaces reduced by the fast search algorithms, one can further reduce the subspace by discarding those points whose $x_{\sum}$ do not agree with that of the random move. Hence the CPU time needed to carry out the comparisons will be reduced further.

Moreover, there could be substantial differences in the order of magnitudes for the various directional variables. For example, the range for a variable in the ith direction may be in the order of 10^{-2}, whereas its value in the $(i+1)$th direction is of the order of 10^2, then even a very significant variation in the ith direction variable may not lead to any significant changes in the computed distances mentioned in Section II-B or in the variable $x_{\sum}$. Hence the search will be much more efficient when one uses the per unit values of the variables.

E. Description of the Self-Learning SA Algorithm

The self-learning SA algorithm as proposed can thus be summarized in the following steps:

STEP 0	
(initialization)	Select a value for the parameter IM that specifies the number of local search iterations to be performed to determine an intermediate point x^{IM}; Empty the sets X_0, X_l, and X_t; Set the initial values for other iteration parameters; Generate an initial feasible point x^0 and compute the cost function $f(x^0)$; Let $x^i = x^0$, $f(x^i) = f(x^0)$;
STEP 1	Starting from point x^i, generate a random point x^{i0};
STEP 2	If point x^{i0} is in or near X_0 or X_l, or if it is inside or near a trajectory in X_t, then go to Step 1; Else add x^{i0} to X_0 and go to the next Step;
STEP 3	Execute a local minimization procedure up to IM iterations; If the local minimization

procedure yields a new local optimum point x^{i*}, then store the trajectory from x^{i0} to x^{i*} after every M intermediate points, then go to Step 4; Else go to Step 4 directly;

STEP 4 If x^{i*} or x^{IM} is within or near X_0 or X_l, or if it is within or near a trajectory in X_t, go to Step 7 directly; Otherwise, accept x^{i*} or x^{IM} with the probability min $\{\exp(-\Delta f/T)\}$ (here $\Delta f = f(x^{i*}) - f(x^i)$ or $\Delta f = f(x^{IM}) - f(x^i)$, and T is the control parameter of the simulated annealing algorithm);

STEP 5 If x^{i*} or x^{IM} is accepted, then set $x^i = x^{i*}$ or $x^i = x^{IM}$ and go to Step 6; Else go to Step 6 directly;

STEP 6 If $f(x^{i*})$ or $f(x^{IM}) < f_{opt}$ then set $f_{opt} = f(x^{i*})$ or $f_{opt} < f(x^{IM})$, and let $x_{opt} = x^{i*}$ or $x_{opt} = x^{IM}$;

STEP 7 Test the equilibrium of the Metropolis process; If the test is passed, go to Step 8; Else go to Step 1;

STEP 8 Test the terminating criterion. If test is passed, go to Step 9; Else reduce the value of the control parameter, let $x^i = x_{opt}$, and go to Step 1;

STEP 9 Output x_{opt}, f_{opt}, and terminates the program.

III. Numerical Results

A mathematical test function and a practical engineering problem are used to evaluate the performance of the proposed Self-Learning SA (SL–SA) algorithm.

A. Mathematical Function

The mathematical test function selected in this paper is

$$\text{minimize} \quad f(x) = \frac{\pi}{n}\left\{10\sin^2(\pi x_1) + \sum_{i=1}^{n-1}[(x_i-1)^2 \cdot(1+10\sin^2(\pi x_{i+1}))] + (x_n-1)^2\right\}$$

$$\text{subject to} \quad -10 \le x_i \le 10 \quad (i = 1, 2, \cdots, n; n = 5)$$

The function has roughly 10^5 local optima and the global one is at $x_i = 1 (i = 1, 2, \cdots, 5)$ with $f_{opt} = 0.0$.

Table I gives the optimization results obtained by using SL–SA and the traditional SA from different initial starting points. From these results it can be seen that:

1) Both the present SL–SA and the conventional SA algorithms converge to the global optimum;
2) The function calls used by the proposed method is about one third of that used by conventional SA algorithm, but the CPU time used by the proposed method is more than one half of that used by the conventional SA algorithm. This is because additional CPU time is needed to compare the random moves with the memorized search histories.

TABLE I
Optimization Result Comparison of Different Methods

Algorithm	Optimal Solution[1]	Function value[1]	No. of iterations[2]	CPU time[2] (s)
SA	(0.9999429, 0.9995698, 1.0004129, 0.9996216, 1.0016973)	2.32762×10^{-6}	410610	100
SL-SA	(0.9999986, 0.9999859, 1.0000012, 1.0000531, 0.9999995)	2.02881×10^{-9}	120814	60

Note:
[1]The optimizations begin from 4 different starting points at x_a=(1,2,3,4,5), x_b=(-1,-2,-3,-4,-5), x_c=(1,3,0,-4,-5), and x_d=(0,0,0,0,0). The optimal solution and function values are corresponding to point x_a.
[2]The number of iterations and CPU time are the mean values of 4 different starting points.

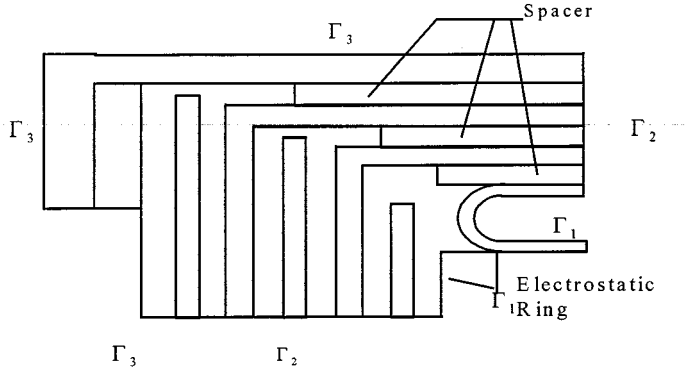


Fig. 1. Schematic diagram of the end region of the transformer being studied.

TABLE II
Optimization Results and Comparisons

	x_1 (mm)	X_2 (mm)	x_3 (mm)	x_4 (mm)	E_{max} (pu)	CPU* (hour)
Initial value	42.0	62.0	88.0	112.0	1.00	/
Optimized SL-SA #*	40.5	39.8	85.0	112.2	0.91	3.25
Optimized SA*	40.6	40.0	84.7	113.1	0.91	10.03
Optimized Simplex	40.0	50.2	81.2	114.0	0.95	0.188

Here SL- SA algorithm is the proposed self-learning SA algorithm
* The CPU time here for SA and SL-SA is the mean value of 10 runs.

B. Application

The realization of minimizing the maximum electric fields in the end region of power transformers under normal and faulty operating conditions is of paramount importance for the designers, since a reduction in the maximum electric fields would prevent the transformer from unnecessary flashovers. The optimization problem is thus to determine the optimal geometrical parameters of the end region so as to have an ideal electric

field distribution. Although there are many optimization algorithms reported in the literatures, the number of suitable algorithms that can be used to solve such problem is very few due to the multi-modal nature of the objective function. Hence this problem is selected as the benchmark problem for the assessment of the proposed method. Here the geometrical optimization problem is formulated as

$$\begin{aligned} &\min \quad E_{\max}(x) \\ &\text{subject to} \quad x_{ib} \le x_i \le x_{ia} (i = 1, 2, \cdots, 4) \end{aligned} \tag{1}$$

where

$E_{\max}$ is the maximum value of the electric field in the end region,

x_1 is the width of the electrostatic rings of the winding and

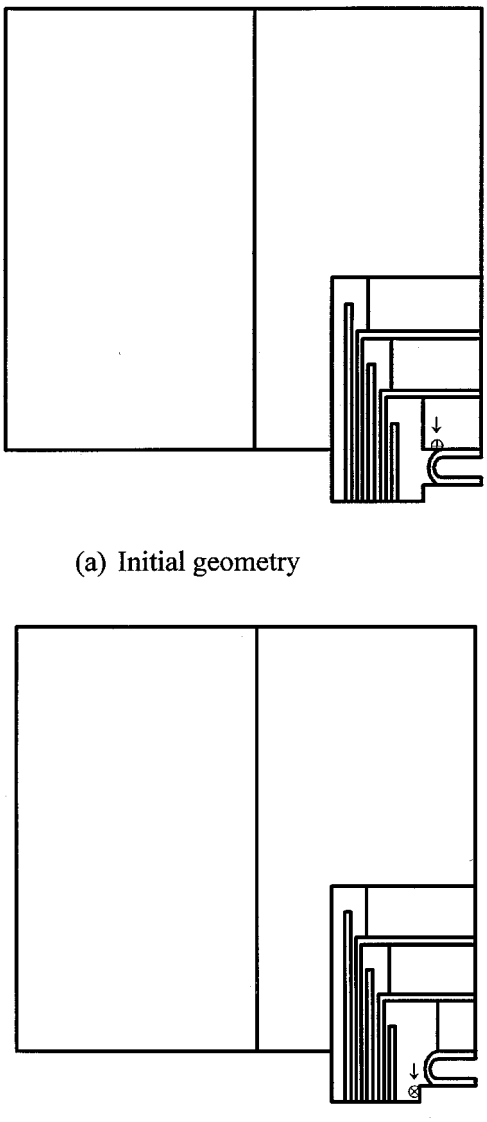
$x_i (i = 2, 3, 4)$ is the width of the spacer i, as shown in Fig. 1.

The end fields are computed by using finite element method. The boundary value problem is

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0 \quad \varphi|_{\Gamma_1} = 1, \varphi|_{\Gamma_3} = 0 \quad \frac{\partial \varphi}{\partial n}|_{\Gamma_2} = 0 \tag{2}$$

The geometry optimization results of the end region of a 63 MVA, 110 kV transformer using the proposed SL–SA, the traditional SA, and a simplex algorithms are given in Table II. The CPU times for the SL–SA and SA algorithms are the mean values of their 10 respective runs. The positions in which the maximum electric density occurs with the initial geometry and the optimized geometry are depicted, respectively, in Fig. 2(a) and (b). From these results it can be seen that:

1) The proposed and the conventional SA algorithms converge to almost the same global optimum point and solution;
2) The percentage value of the CPU time used by the proposed method to that used by the traditional SA algorithm is significantly reduced when compared with that for the mathematical test function, because the additional CPU time used by the proposed method for the history memorization and comparison is relatively small compared with those required for finite element method calculations.
3) The simplex method cannot find the global solution.
4) The maximum value of the electric field with the optimized geometry decreases from 1 pu to 0.91 pu and the position in which the maximum electric field occurs is moved from point $\oplus$ to point $\otimes$.



(a) Initial geometry

(b) Optimized geometry

Fig. 2. The position of the maximum electric density.

IV. Conclusion

By incorporating a feature that includes the memorization of the searched parameter spaces into the proposed algorithm and by introducing a local phase to the conventional simulated annealing algorithm, a new self-learning simulated annealing algorithm is reported in this paper.

The numerical results on both the mathematical test and engineering problems show:

1) The incorporation of memorization of the searched histories into the conventional SA algorithm do enable the proposed algorithm to have self-learning ability in the search process.
2) The reduction in the value of the mean CPU time required for the proposed method is very significant for inverse problems encountered in the study of electromagnetics, since the additional CPU time needed for the history memorization and comparison is very small compared to those needed for the numerical calculations of electromagnetic fields.
3) The inclusion of memorization features in the searched histories of the parameter spaces could be incorporated into other stochastic algorithms for improving the searching efficiency of these methods.

REFERENCES

[1] T. Renyuan, Y. Shiyou, L. Yan, and W. Geng, "Combined strategy of improved simulated annealing and genetic algorithm for inverse problem," *IEEE Trans. Magn.*, vol. 32, no. 3, pp. 1326–1329, 1996.

[2] Y. Shiyou, N. Guangzheng, L. Yan, and T. Renyuan, "Shape optimization of pole shoes in harmonic exciting synchronous generators using a stochastic algorithm," *IEEE Trans. Magn.*, vol. 33, no. 2, pp. 1920–1923, 1997.

[3] G. Drago and A. Manella, "A combined strategy for optimization in nonlinear magnetic problems using simulated annealing and search techniques," *IEEE Trans. Magn.*, vol. 28, no. 2, pp. 1541–1544, 1992.

[4] R. R. Saldanha, J. A. Vasconcelos, A. N. Moreira, and G. B. Alvarenga, "Optimization of the cross-section shape of a ridge waveguide using the ellipsoid and the tabu search algorithm," *IEEE Trans. Magn.*, vol. 32, no. 3, pp. 1254–1257, 1996.

[5] O. A. Elwakeil and J. S. Arora, "Two algorithms for global optimization of general NLP problems," *Int. Journal Numer. Method Engineering*, vol. 39, pp. 3305–3325, 1996.

[6] S. L. Ho, Y. Shiyou, Z. Ping, H. C. Wong, and M. A. Rahman, "A combined finite element-domain elimination method for minimizing torque ripples in inverter-fed motor drive systems," in *COMPUMAG'99*, Sapporo, Japan, Oct. 25–28, 1999.

[7] M. Laguna, "Intensification and diversification with elite tabu search solutions for the linear ordering problem," *Computers Ops Res.*, vol. 26, pp. 1217–1230, 1999.