LETTER

Adaptive Simulated Annealing in CNN Template Learning

Brett CHANDLER[†], **Csaba REKECZKY**[††], ***Nonmembers***, **Yoshifumi NISHIO**[†], ***and*** **Akio USHIDA**[†], ***Members***

SUMMARY Template learning has potential application in several areas of Cellular Neural Network research, including texture recognition, pattern detection and so on. In this letter, a recently-developed algorithm called Adaptive Simulated Annealing is investigated for learning CNN templates, as a superior alternative to the Genetic Algorithm.

key words: *adaptive simulated annealing, cellular neural network, template learning*

1. Introduction

Opportunities for the application of template optimization (or "learning") for a Cellular Neural Network (CNN) [1] are prevalent in such areas as pattern recognition and texture classification. It is highly desirable to employ an algorithm which not only can produce optimal solutions, but which can also find the solution efficiently, in as short a time as possible.

Various template-learning methods have been proposed to date [2]. In [3] and [4], a hybrid Direct-Search method and Simulated Annealing (SA) were investigated for Discrete-Time CNN template optimization. Another algorithm, which has been widely used for CNN template learning tasks, is the Genetic Algorithm (GA) [5], [6]. A variant of GA, from a class called Evolutionary Strategies, was applied in [7] to obtain feature-extraction templates.

In this letter, we compare the performance of a recently-developed optimization algorithm called Adaptive Simulated Annealing (ASA) against GA. In a published comparison study, ASA strongly outperformed GA on a set of standard benchmark optimization testing functions [8], and these excellent results provided the motivation for this letter.

2. Optimization Algorithms

2.1 Genetic Algorithm

GA is an optimization algorithm based on the concepts of natural selection and evolution. For a detailed description of GA, please refer to [5]. This describes a basic GA, which served as the foundation of our algorithm. Several extensions to this basic algorithm were reported in [7] as significantly accelerating the learning of CNN templates, and these were also implemented in our algorithm. The specific settings for the genetic algorithm are as follows. The chromosomes were encoded using the *enhanced coding* method, and 8 bits were used to encode each value in a range of $[-5, 5]$, resulting in a resolution of 0.04 (the same resolution was used for ASA). The reproduction strategy employed was a nonoverlapping population, combined with an elitist strategy. Mutation probability was chosen as 5%.

Some disadvantages of GA are that two important parameters (the population size, and the type of crossover method to employ) must be carried out for every new task, and there are no rules to determine the optimum values. Thus the algorithm relies on the experience of the operator, and/or trial-and-error testing. In addition, while the non-methodical nature of the search can help to avoid entrapment in local minima, performance tends to be somewhat inconsistent.

2.2 Simulated Annealing

In this section, some of the theory of SA is briefly revised, to serve as a background for highlighting the differences between ASA and conventional SA. A full description is contained in [9].

SA consists of three important relationships:

1. The generation probability density function, $g_T(\Delta x)$, based on the deviation Δx from the immediately previously chosen point, where $x = \{x_i\,;\, i = [1, D]\}$ are the parameters to be optimized.
2. The acceptance probability density function, $h(\Delta E)$, based on the difference in cost of the current and immediately-previous state.
3. The annealing schedule $T(k)$ for annealing-time step k, which describes the way in which the 'temperature' is adjusted.

Manuscript received March 30, 1998.
Manuscript revised August 25, 1998.

[†]The authors are with the Department of Electrical and Electronic Engineering, Tokushima University, 770–8506 Japan.
[††]The author is with the Computer and Automation Institute, Hungarian Academy of Sciences, Hungary.

2.3 Adaptive Simulated Annealing

The ASA algorithm, developed in 1989 by L. Ingber [10]–[12], is a variant of SA, but with some substantial differences from conventional SA. It has already been used in a range of fields, including physics and finance. Ingber proposed use of an annealing schedule which decreases exponentially in annealing time k:

$$T_{cost}(k) = T_0 \exp(-c_{cost} k^{1/D}) \; , \tag{1}$$

where $T_{cost}(k)$ is referred to as the *cost temperature*, T_0 is the initial cost temperature, c_{cost} is an adjusting factor, k is the annealing time index (specifying the number of transitions at each T_{cost}) and D is the number of parameters to be fitted (which defines the dimension of the search space). T_0 can be adjusted, but the algorithm is not sensitive to this value due to its adaptive nature, and setting it to a value of 1 to start the search is sufficient.

The ASA annealing schedule is significantly faster than other types of Simulated Annealing. A comparison of the convergence rates for Boltzmann Annealing, Cauchy Annealing and ASA is shown in Fig. 1.

ASA has been designed to overcome the limitations of SA for multi-dimensional optimization problems. Whereas the search process with conventional SA is purely random, ASA adaptively guides each parameter towards the most promising solution set. This is accomplished by establishing an annealing schedule for each parameter, in a similar manner to that of the cost temperature.

As with SA, a statistical guarantee exists that an optimal solution will be found after many generations. In practice, this results in a consistently-high rate of finding optimal or near-optimal solutions. It can be shown heuristically that the chosen annealing schedule of Eq. (2) will suffice to arrive at a global minimum, in other words any point in parameter space can be sampled infinitely often in annealing time [12].

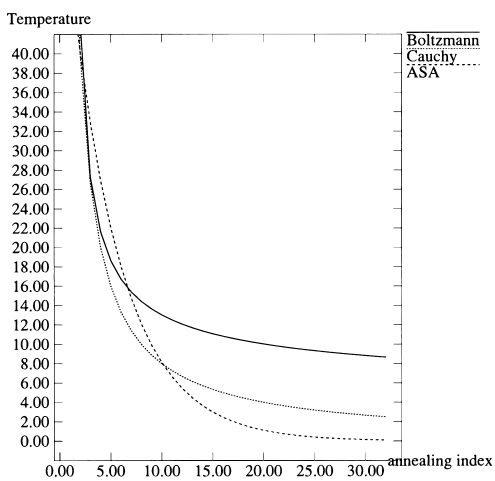


Fig. 1 Annealing schedules of BA, SA and ASA.

In general, for a multi-dimensional optimization, the sensitivities of the parameters being optimized will change as the search progresses. ASA attempts to account for the changing parameter sensitivities by adjusting the annealing schedules of the parameters periodically, resetting the annealing time index for each parameter to a new starting index every 100 (modifiable) acceptance events. The new index is calculated in a maner that is proportional to each parameter's *sensitivity* (the change in the cost with respect to that parameter.) This process is referred to as *reannealing*.

A user-friendly aspect of ASA is that tuning is often not necessary, and our experimental testing shows that it can perform well without any adjustment. We investigated the effects of a tuning parameter which has a significant effect on the annealing process, which we define as η, and is related to the adjusting factor c for each parameter in the following way:

$$c_i = m_i \exp(-n_i/D) \; ; \tag{2}$$

$$m = -\log(\eta) \; . \tag{3}$$

η can be used to scale the exponential decrease of the annealing schedule, without affecting the basic sampling proof. For example, annealing can be slowed down by using a larger value of η than the default.

3. Examples for CNN Template Learning Using ASA and GA

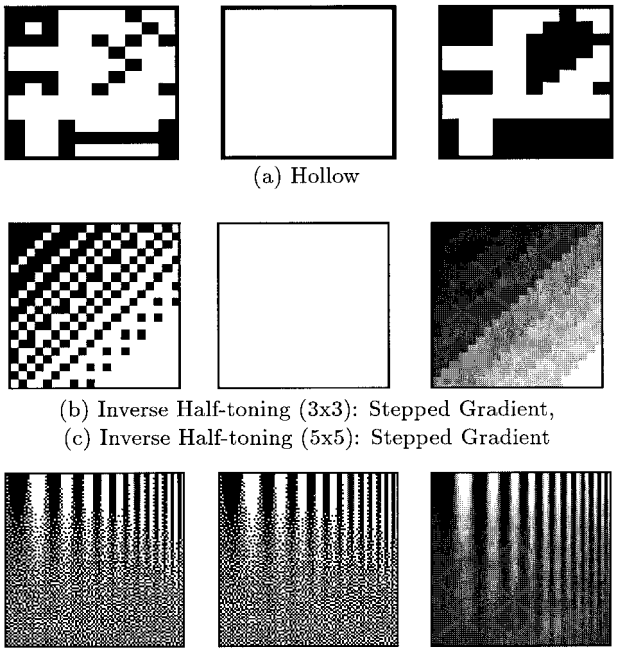
3.1 The Tasks

A number of image-processing tasks were selected as tests for finding suitable templates, focusing on coupled-system templates (i.e. those containing off-center feedback), with both gray-scale and binary output. For each task a training set was designed to provide the input image, initial state and the desired output of the network. We carried out several experiments, but present four of them here.

3.2 The Training Sets

A training set for each optimization task is shown in Fig. 2. Careful design of the training sets is vital to the chances of success of the optimization, in order to ensure the generality of the optimized templates.

For tasks such as Inverse Half-toning which use gray-scale images, the training sets cannot be designed to include a range of densities. In the case of Inverse Half-toning, tests were carried out to obtain templates for the reconstruction of both 'stepped-gradient' images, containing homogeneous density bands, and for smoothly-varying images. For the latter case, an Arden Chart was selected as a suitable training image. An Arden Chart is a chart of smoothly-changing spatial frequencies and contrast which is used to measure



(a) Hollow

(b) Inverse Half-toning (3x3): Stepped Gradient,
(c) Inverse Half-toning (5x5): Stepped Gradient

(d) Inverse Half-toning (5x5): Arden Chart

Fig. 2 Training image sets (input, initial state, desired output). Frame cells are white for (a), and for the others the frame duplicates the outer layer.

$$A = \begin{bmatrix} c & b & c \\ b & a & b \\ c & b & c \end{bmatrix} \quad B = \begin{bmatrix} f & e & f \\ e & d & e \\ f & e & f \end{bmatrix} \quad I = g$$

(a) 3x3 template (Hollow, Inverse Half-toning: Stepped Gradient)

$$A = \begin{bmatrix} f & e & d & e & f \\ e & c & b & c & e \\ d & b & a & b & d \\ e & c & b & c & e \\ f & e & d & e & f \end{bmatrix} \quad B = \begin{bmatrix} l & k & j & k & l \\ k & i & h & i & k \\ j & h & g & h & j \\ k & i & h & i & k \\ l & k & j & k & l \end{bmatrix}$$

$$I = m$$

(b) 5x5 template (Inverse Half-toning: Arden Chart, Stepped Gradient)

Fig. 3 Generalized Template Formats.

the sensitivity of the human eye. This chart was selected as a training image because it serves as a useful demonstration of how well objects of various widths and background contrasts can be reconstructed by the template [13].

The generalized template formats associated with the tasks are shown in Fig. 3. (Matrices A and B are the feedback and control matrices, respectively, and I is the bias). Some of these elements may be preset to a constant, depending on the nature of the task, to reduce the number of elements to be optimized.

3.3 Results

The results are in the form of graphs of percentage error versus number of iterations (generated states), and are plotted logarithmically to help show the convergence

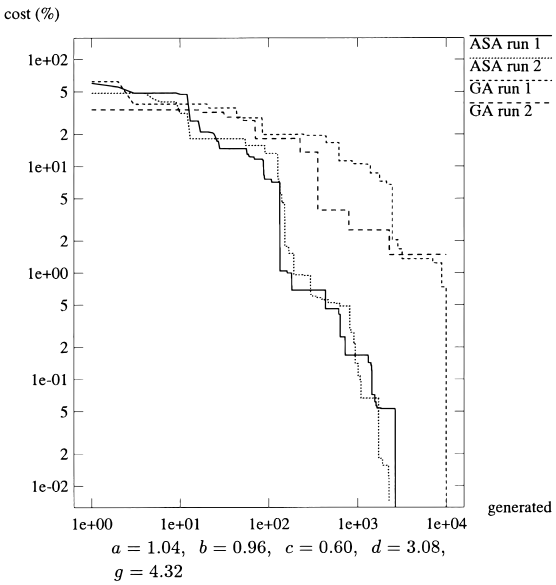


Fig. 4 Hollow (GA: population = 300, random crossover; ASA: $\eta = 10^{-3}$).

rates. Since the algorithms have an inherent random nature, each run differs from the last. Thus, two runs have been plotted for each algorithm to provide an indication of performance variation. Underneath each result is an example template found by ASA optimization (those elements which were preset to zero are omitted).

It is noted that the settings for the Genetic Algorithm were adjusted by trial and error, and the best settings found were used for these experiments.

Test 1: Hollow
Description: Concave regions (hollows) in a black object are turned to black (Fig. 4).
Elements to optimize: 5. It can be recognized that propagation of information across the image will be required to identify the hollow regions, indicating a coupled template. Propagation should be isotropic, and making this assumption reduces the number of independent A-matrix elements. In this case, the propagation should preserve the object outlines of the original image, so we make use of a control mask at the Input – the central element of the B-matrix is thus non-zero. Black pixels (the objects) are treated differently from white pixels (the background), so the current I is non-zero.
Comment: Optimization with standard ASA settings produced quite good results, although the iterations taken were a little inconsistent. Here, average iterations and their variation were reduced with η set to 10^{-3}. GA had difficulties converging past an error of approximately 1%.

Test 2: Inverse Half-toning: Stepped-Gradient
Description: A binary half-toned image is converted to a gray-scale image which contains discrete gradient

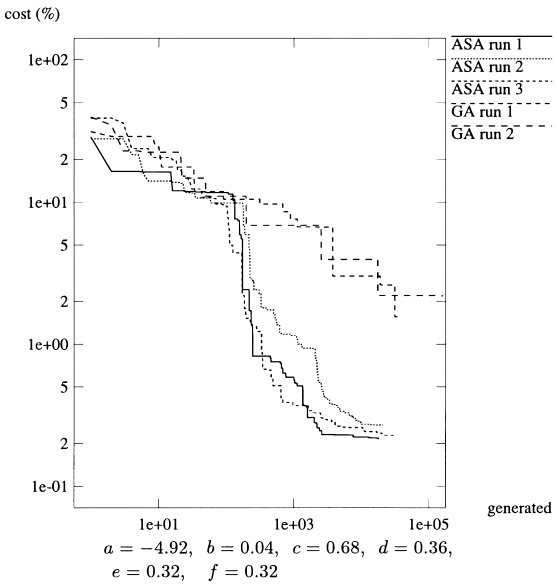


Fig. 5 Inverse Half-toning (3×3): Stepped Gradient (GA: population = 500, random crossover; ASA: default setting).

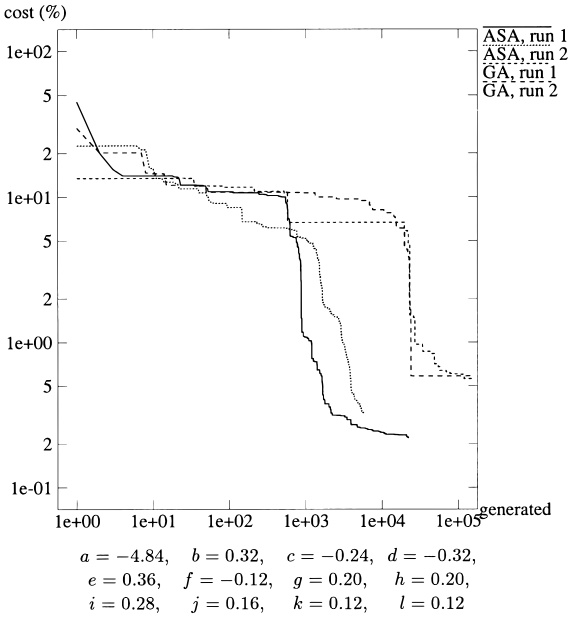


Fig. 6 Inverse Half-toning (5×5): Stepped Gradient (GA: population = 500, random crossover; ASA: $\eta = 10^{-3}$).

steps (Fig. 5).

Elements to optimize: 6 (neighborhood 1 template). In the case of Inverse Half-toning, a smoothing filtering action is required, which should be symmetrical. Black pixels and white pixels are not to be differentiated in their treatment, so the bias term may be set to zero.

Comment: Default ASA settings were suitable here. The difference in performance between the two algorithms is clear, with GA having a much slower convergence rate. GA was unable to find a solution below 1% cost, whereas ASA converged to a cost of below 0.3%.

Test 3: Inverse Half-toning: Stepped-Gradient

Elements to optimize: 12. The template has a neighborhood of 2.

Description: A binary half-toned image is converted to a gray-scale image which contains discrete gradient steps (Fig. 6).

Comment: In this case, the ASA annealing was slowed down ($\eta = 10^{-3}$), observing the complex nature of the task and large number of parameters to be optimized. This was effective in shortening the number of generations taken. Again, ASA showed superior performance to GA, with lower-cost templates and much faster convergence.

Test 4: Inverse Half-toning, Arden Chart

Elements to optimize: 12.

Description: A binary half-toned image is converted to a gray-scale image of an Arden chart (Fig. 7).

Comment: The nature of this task is similar to the previous task. In this case also, a slower annealing schedule was beneficial, with η set to 10^{-3}. GA had difficulty converging in this task.

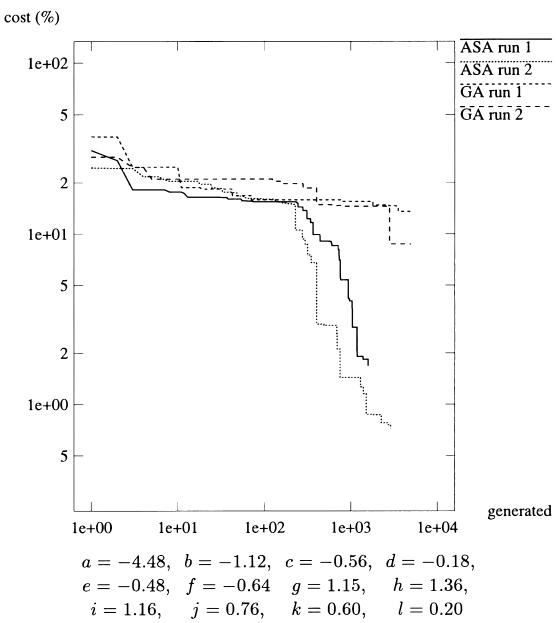


Fig. 7 Inverse Half-toning 5 × 5: Arden chart (GA: population = 500, random crossover; ASA: $\eta = 10^{-3}$).

4. Discussion

The most striking observation is that the convergence speed of ASA was generally significantly superior to GA. For template optimizations involving large training images, complex templates or for other time-intensive situations, the use of ASA has the potential to save a significant amount of time over the GA algorithm.

In addition, in terms of the quality of the solution, ASA was able to find an optimal or near-optimal template for all the tested cases. Although in many cases, GA was also successful, it took longer to reach the solution. Furthermore, in the case of the Inverse Half-toning examples, ASA also outperformed GA by providing a lower-cost solution. (A solution of zero cost is not possible, since a half-toned image lacks the information to reconstruct the original picture perfectly.) For 5×5 Inverse Half-toning, ASA found templates which produced an error of less than 0.3%, whereas GA only managed to find solutions of approximately twice this cost, and at a much slower convergence rate. In the case of the Hollow template, GA often became trapped in local minima and failed to find a zero-cost solution, whereas ASA coped with this problem successfully in all attempts.

We also carried out ASA optimization for a single constant task, using a variable number of parameters ranging from three to seven. Performance was even over the entire range, with no trend towards longer optimization runs. This suggests that subtler factors, such as the nature of the task itself and the training set used, have a strong influence on the difficulty of optimization. It also demonstrates the ability of the ASA algorithm to efficiently deal with parameters which have little significance on the output – this can be attributed to the sensitivity-based reannealing scheme.

In this letter, some tuning options with ASA were also investigated. Improved performance was obtained, in particular for the tasks representing the more difficult optimization problems, which would otherwise have taken much longer to converge, namely the three Inverse Half-toning tasks. This was achieved by decreasing the annealing rate by raising η from 10^{-5} to 10^{-3}, obtaining faster convergence.

Important to the overall process, are the pre-optimization steps, which include preparing the training set and deciding the template format. For example, if the training set holds insufficient or misleading information, then it is likely that the template found may fail to carry out the desired task, or else convergence may not occur at all. Also, where possible the dimension of the search space should be reduced by setting elements in the template format according to a-priori information.

5. Conclusions

Element optimization was carried out for a number of coupled templates, with both binary and gray-scale output. ASA was found to give consistently good results, and overall had significantly better performance than GA. With respect to convergence speed, ASA arrived at a solution in general faster than GA for all tested examples, in the best cases approximately ten times faster. For the more complex Inverse Half-toning examples in particular, ASA produced lower-cost templates than those found by GA, in addition to having faster convergence.

Overall, ASA is a powerful tool which is well-suited to situations where hand design of templates is difficult or not possible, or to complement initial hand designs. The experiments examined provide compelling evidence that ASA provides an efficient optimization method for application to CNN template learning.

Acknowledgement

This work was partly supported by the Research Grant from the Okawa Foundation for Information and Telecommunications.

References

[1] L. Chua and L. Yang, "Cellular neural networks: Theory," IEEE Trans. Circuits Syst., vol.35, no.10, pp.1257–1272, 1988.
[2] J. Nossek, "Design and learning with cellular neural networks," Proc. of CNNA-94, pp.137–146, Dec. 1994.
[3] H. Magnussen and J. Nossek, "Global learning algorithms for discrete-time cellular neural networks," Proc. of CNNA-94, pp.165–170, Dec. 1994.
[4] A. Kellner, H. Magnussen, and J. Nossek, "Texture classification, texture segmentation and text segmentation with discrete-time cellular neural networks," Proc. of CNNA-94, pp.243–248, Dec. 1994.
[5] T. Kozek, T. Roska, and L.O. Chua, "Genetic algorithm for CNN template learning," IEEE Trans. Circuits. Syst., vol.40, no.6, pp.392–402, June 1993.
[6] T. Sziranyi and M. Csapodi, "Texture classification by cellular neural networks using genetic learning," Proc. of 12th ICPR, Parallel Computing, Oct. 1994.
[7] S. Taraglio and A. Zanela, "Cellular neural networks; A genetic algorithm for parameters optimization in artificial vision applications," Proc. of CNNA-96, pp.315–320, June 1996.
[8] L. Ingber and B. Rosen, "Genetic algorithms and very fast simulated reannealing: A comparison," Math. and Comp. Modeling, vol.16, no.11, pp.87–100, Nov. 1992.
[9] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines," John Wiley & Sons, Chichester, 1990.
[10] L. Ingber, "Very fast simulated re-annealing," Math. and Comp. Modeling, vol.12, no.8, pp.967–973, 1989.
[11] L. Ingber and contributors, "Adaptive simulated annealing," (C source code), Internet address: "http://www.ingber.com/#ASA-CODE".
[12] L. Ingber, "Adaptive simulated annealing: Practice versus theory," Math. and Comp. Modeling, vol.l8, no.12, pp.29–57, 1993.
[13] K.R. Crounse, T. Roska, and L.O. Chua, "Image halftoning with cellular neural networks," IEEE Trans. Circuits Syst. II, vol.40, no.4, pp.267–283, April 1993.