Constraint Satisfaction Using Neural Networks with a Local and Autonomous Annealing Technique

Yasusi Kanada*
Tsukuba Research Center, Real World Computing Partnership
Takezono 1-6-1, Tsukuba, Ibaraki 305, Japan
E-mail: *kanada@trc.rwcp.or.jp*, WWW: *http://www.rwcp.or.jp/people/yk/*

ABSTRACT

A method for solving large-scale constraint satisfaction problems using an annealed symmetrically-connected neural network, which is called DSN-FAM, is proposed in the present paper. Some conventional methods, such as Hopfield networks, often fail to find a solution. Some others, such as Boltzmann Machines, take too much time. These difficulties are solved by a type of annealing technique, which we call the frustration accumulation method (FAM). DSN-FAM works only with local information, and no global functions or global parameters such as a temperature are used. DSN-FAM thus works autonomously. That is, no external control is necessary while operating. Experiments show that this method does not fail to find a solution and the execution time is less than one tenth of Boltzmann Machines. The performance can be easily and almost linearly improved by parallel processing using tens of processors.

1. Introduction

Constraint satisfaction problems (CSP) have often been solved experimentally using symmetrically-connected neural networks. The problems are translated to maximum descent or hill-climbing of a function defined using the number of satisfied constraints [Min 92]. Ackley, Hinton, and Sejnowski [Ack 85] solved CSP using Boltzmann Machines. Takefuji [Tak 92] solved many CSP using improved Hopfield networks and other types of networks. Nakamura, Wakutsu, and Aiyoshi [Nak 94] solved map coloring problems and traveling salesperson problems using improved symmetrically-connected networks with discrete output values.

The difficulties of the above methods are as follows. Firstly, local maxima cannot be avoided and, thus, it fails to find a solution in some of the above methods, such as Hopfield networks. This difficulty does not exist in Boltzmann Machine and other annealed methods, of course. Secondly, the above methods take too much computation time on conventional computers. For example, Hopfield networks take much time because they require many iterations and costly floating-point function calculation. Boltzmann Machines require very many iterations because it requires slow temperature decrease. These difficulties make solving large-scale CSP almost impossible. There are several other difficulties in the methods using conventional annealing techniques. These difficulties are explained in detail in the next section.

To overcome these difficulties, a symmetrically-connected network with discrete output and with an annealing technique called the frustration accumulation method (FAM) [Kan 94b] is proposed in the present paper. This method is called DSN-FAM (discrete symmetric network with FAM). DSN-FAM is a method for CSP. FAM has been developed as an annealing technique for solving CSP using the chemical casting model (CCM) [Kan 94a]. CCM is a symbolic model for emergent computation and has been applied to several constraint satisfaction and optimization problems. This method is called CCM-FAM. However, FAM can also be applicable to neural networks, and it is also advantageous. Local maxima can be avoided using FAM without loosing the efficiency of DSN, the original method.

Several annealing techniques including simulated annealing are surveyed in Section 2. The method of solving CSP using DSN-FAM is explained in Section 3. This method is applied to map or graph coloring problems in Section 4. The results of performance evaluation of both sequential and parallel processing are shown in Section 5. Finally, the conclusion is given in Section 6.

2. Annealing Techniques

In the present paper, "Annealing techniques" means techniques for escaping from local optima using randomization. There are several types of annealing techniques for optimization problems. Simulated annealing (SA) and Boltzmann Machines, which are among them, are mentioned here.

* The author's current address is Intelligent Systems Research Dept., Hitachi Central Research Laboratory, Kokubunji, Tokyo 185, Japan. The email address is kanada@crl.hitachi.co.jp.

SA [Kir 83] is a method for function optimizations. The function to be optimized, $f(x)$, is usually a global function that is defined to whole system, such as global energy. The value of function f is a scalar, and the value of parameter x is usually numerical (i.e., a scalar or vector). Global parameter T, which is called temperature, is used in SA. Random noise, which is a monotonically increasing function of T, is added to x and T is gradually decreased to 0 while searching a solution, so that falling into a local optimum is avoided. Because SA is a method for *global* function optimization, SA, in its original form, does not fit for local-information-based method, such as neural networks or CCM [Kan 94a], which is briefly mentioned later. Temperature in SA is time-dependent and externally controlled. Thus, the problem-solving system does not autonomously work again in an expected way after it converges to a solution, even if the input to the system is changed so that the optimality gets worse.

The annealing technique in Boltzmann Machines is also called SA. However, random noise is added to each neuron independently in Boltzmann Machines. This means that SA in Boltzmann Machines works locally to each neuron. Thus, the nature of this method is different from normal SA, which optimizes a global function directly. However, temperature in Boltzmann Machines is still a global parameter and externally controlled.

Kanada [Kan 94b] developed a new annealing technique called FAM. This technique has been developed for avoiding local optima in CCM-based problem solving. There are two reasons why a new annealing technique was developed.

Firstly, the annealing technique used with CCM should be based only on local information, but the above techniques are not. In this context, that the computation is based on *local*-information means that the number of data used for a unit of computation is small and that the units of computation are autonomous or basically independent each other. CCM is a symbolic model for emergent computation [For 91]. Emergent computation is local-information-based computation. The value of global objective function is not computed even when CCM is applied to global optimization. The global function is expressed as a summation of functions, which are computed only using local information, and the global function is eventually optimized through the optimization of the local functions. Thus, global function based method cannot be applied to CCM.

Secondly, SA and similar techniques are inefficient. A local SA technique, which is similar to that used in Boltzmann Machines, has been developed for CCM-based problem solving [Kan 96], but it is also inefficient. More efficient technique is required for solving large-scale problems in a reasonable time.

3. Symmetrically-connected Networks with FAM

A method of solving CSP using a neural network with FAM is proposed in the present section. This method is called DSN-FAM, where DSN means discrete symmetric network, in the present paper.

The neurons used in this method are McCulloch-Pitts type, i.e., the output is discrete and time is also discrete, and they are mutually connected and symmetric. The input-output function of neuron i is as follows:

$$x_i' = threshold\left(\sum_{j=1}^{n} w_{ji}x_j + \alpha\right),$$

where x_j ($j = 1, 2, \ldots, n$) is the current output of neuron j, x_j' is its output of the next time step, and w_{ji} is the connection weight between neurons j and i ($w_{ji} = w_{ij}$). The connection weights are invariant. α is a constant if FAM is not used, but it is varied by FAM. Function threshold is defined as follows:

$$threshold(x) = \begin{cases} 0 & \text{if } x < c, \\ 1 & \text{if } x \geq c, \end{cases}$$

where c is a constant.

The problem to be solved must be described as a collection of local constraints. If there are global constraints, this method cannot be applied. The local constraints must be expressed as constraints among the input values of a neuron. For example, that the sum of outputs of several specific neurons that are connected to a certain neuron is 1 may be the constraint. This constraint means that exactly one of these neurons is active and others are inactive.

FAM is a technique for escaping from "local maxima." In this technique, each neuron has a type of energy called a *frustration*, whose value is positive. If frustration is larger, α is varied so that the state of the neuron can more easily be changed. The frustration of neuron i is noted by f_i. If the current state is 0, then

$$\alpha = a + bf_i,$$

where a and b are constants and $b > 0$. If the current state is 1, then

$$\alpha = a - bf_i.$$

Each neuron initially has a certain level of frustration, f_0, which is, for example, 10^{-10}. The frustration is increased, when the application of a reaction rule fails but there are unsatisfied constraints. This means that the frustration of a neuron is increased when the following three conditions hold.

1. The neuron is tested for a possible state transition, i.e., a possible state change from inactive to active state, or vice versa.
2. The state transition does not occur. (The output of the neuron is not changed.)
3. There are constraints, relating to the neuron that are not or will not be fully satisfied.

Because of the third condition, FAM is a method applicable only to CSP without global constraints.

A value of frustration, f_i, is replaced by cf_i, where c (> 1) is a constant. This means that

$$f_i' = cf_i,$$

where f_i' is the frustration at the next time step. The value of c is 2, for example. If a state transition occurs, the frustration of the neuron is reset to f_0 even when the constraints are not satisfied.

The order of state transition tests can be sequential (asynchronous) or parallel, and can be deterministic or randomized. Sequential and randomized order is used throughout the present paper. An algorithm of network state transition can be described as follows.

```
repeat
    Select a neuron randomly;
    Test the neuron, and change the state if it is necessary;
    if a state transition occurred then[1]
        f_i := f_0;  /* reset to the initial value */
    else if not all the constraints are satisfied then
        f_i := c f_i;  /* increased */
    end if
until no more state transition can occur;
```

The network works qualitatively as follows. The average frustration increases rapidly when there are constraints that are difficult to be satisfied. This high frustration state corresponds to high temperature state in SA. If the values of parameters, f_0 and c, has been selected appropriately, the number of violations of constraints decreases, and the average frustration thus decreases. This frustration decrease corresponds to temperature decrease in SA, but the decrease occurs autonomously. No external control is required. If all the constraints are satisfied and all the frustrations become low enough, no state transition occurs anymore. Then the execution stops.[2]

Two sample time sequences of average frustration is shown in **Figure 1**. The problem used is the USA mainland map coloring, which is explained in the next section. The horizontal axis shows the number of state transitions since the network begins to operate, and the vertical axis shows the average frustration. The stars in the upper part of the graph show the state transitions caused by frustration accumulation. (Their vertical coordinate has no meaning.) A solution was found by 118 state transitions in a trial, and by 274 state transitions in the other trial. Average frustration sometimes increases rapidly and this causes a state transition. Then, the frustration decreased rapidly.[3] Average frustration sometimes decreases without a state transition caused by a frustration accumulation.

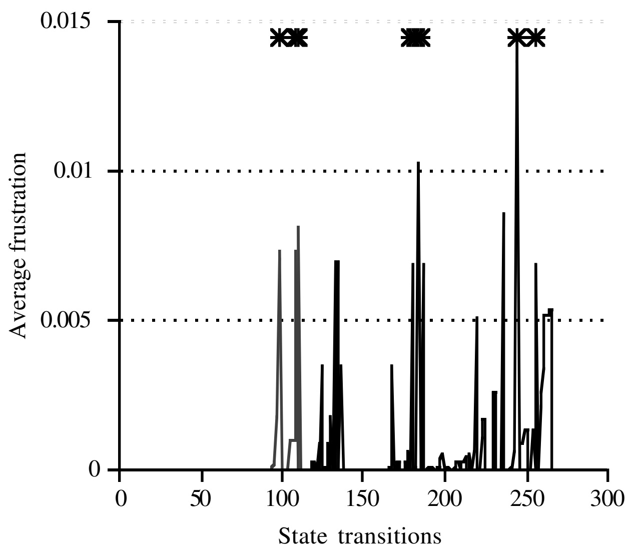

Figure 1. Two sample sequences of average frustration

Parameters, f_0 and c, are constant during the execution, and they are considered to be properties of each neuron or each type of neurons. Thus, this method works only with local information if each neuron is con-

[1] In this vertion of the algorithm, the frustration of a neuron is not reset to f_0 when the constraints are satisfied by an activation of another neuron even if the neuron is tested after the satisfaction. This statement can be modified as follows:

```
if a state transition occurred or
   all the constraints are satisfied then
        f_i := f_0;
else    f_i := c f_i;
end if
```

Then, the frustration is reset when the neuron is tested after the constraints are satisfied. This version also works. However, the parameter tuning seems to be more difficult.

[2] The termination condition in the algorithm is replaced by "false." This means that the network operates forever. Then, the network once stops changing its state but begins changing its state again when the state of a neuron is changed or new constraints are added externally. Thus, a dynamic problem or open-ended problem can be solved without an additional mechanism in this method.

[3] The frustration of the neurons, in which all the constraints are satisfied, are regarded to be the initial value, although the actual values stored are larger in this measurement.

nected only to near neurons, and there are no global parameters, such as temperature, used in simulated annealing. Both f_0 and c are constant for all the vertices in our implementation.

Frustrations may be increased additively instead of multiplicatively as above. That means, the frustration, f_i, may be replaced by $f_i + k$ when f_i is to be increased, *i.e.*, $f_i' = f_i + k$, where k (> 0) is a constant.[1] However, if the frustration increase is additive, the appropriate range of parameters, f_0 and c, will become narrower. The additive frustration is explained more in the case of CCM-FAM by Kanada [Kan 96].

4. Application to Coloring Problems

DSN-FAM is applied to map/graph coloring problems in the present section.

The problem is how to color the vertices of a graph using a specified number of colors, for example, four. Each pair of neighboring vertices must be given different colors. A map coloring problem can be converted to a graph coloring problem, if areas of the map are converted to vertices and the area borders are converted to edges. Thus, the map coloring problem can be solved by the same network as the graph coloring problems. For example, the problem of coloring the graph with five vertices, shown in **Figure 2**, is equivalent to the problem of coloring the map with five areas, which is also drawn in Figure 2.

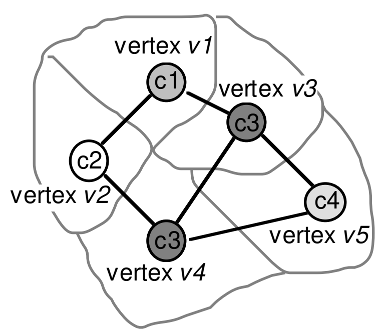

Figure 2. An example of a graph coloring problem and its representation

The network structure used for solving the problem is shown in **Figure 3**. Only part of the network connection is drawn in Figure 3, because otherwise the figure will be too complicated. The neurons, N_{11}, N_{12}, …, $N_{n_c n_v}$ (n_c is the number of colors and n_v is the number of vertices) are two-dimensionally arrayed. The neurons in a column represent the color of a vertex. The neurons in a row represent the same color. There are inhibitory connections between the neurons of the same vertex (i.e., in the same column), because a vertex has only one color at a time. There are inhibitory connections between the neurons of the neighboring (i.e., connected) vertices in the same row, because the neighboring vertices must not have the same color. The weights of the connections between neurons are defined so that the energy function takes the minimum value at solutions. This means that the constraints are embedded in this network. When FAM is used, the constraints are also embedded in the mechanism of FAM. Thus, the constraints are expressed twice in DSN-FAM.

The output values of neurons are 0 or 1.

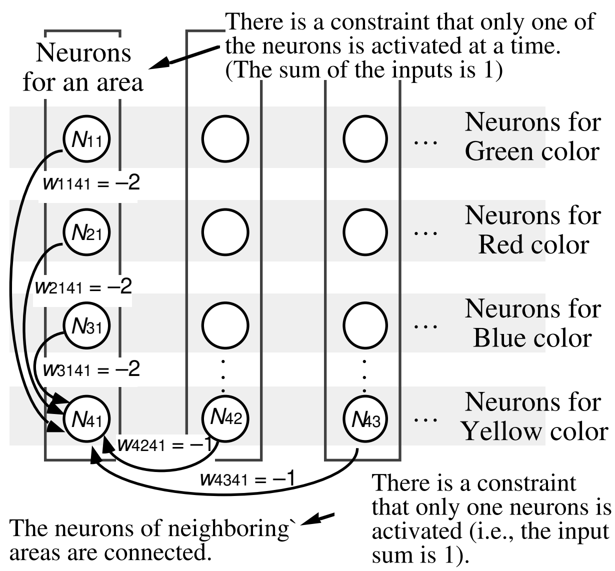

Figure 3. The network structure for solving coloring problems

Two coloring problems are used for performance evaluation. One is a problem to color the USA mainland map shown in **Figure 4** using four colors. This problem was also used by Takefuji [Tak 92]. The number of areas is 48. The other is a graph coloring problem, which is called DSJC125.1, in DIMACS benchmarks [Tri][DIM]. The number of vertices is 125 and the number of colors is five. This problem was used by Johnson, et al. [Joh 91] and by Selman and Kautz [Sel 93].

[1] Several variations of the method of increasing/decreasing frustrations are available. Additive frustration is one of them. However, the method described above is the best tested so far.

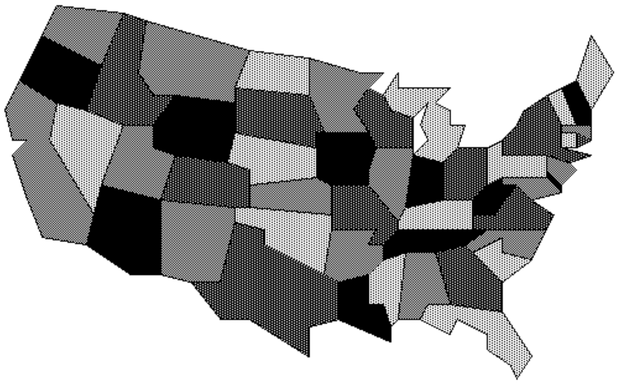

Figure 4. The USA mainland map coloring

5. Results of Performance Evaluation

Performance of DSN-FAM is shown and it is compared with other methods here. Random asynchronous execution order is used in all the experiments.

Firstly, the performance of the USA map coloring problem is described. This problem is a rather easy problem in coloring problems. DSN without FAM was tried 100 times. A solution could be found only once. The network stopped at non-solution state 99 times. Thus, DSN without FAM is unacceptable for solving coloring problems. The performance results of sequential execution using DSN-FAM, CCM-FAM and Bolzmann Machine are shown in **Table 1**. The performance results in this table are average of 20 runs. Although the temperature scheduling of Boltzmann Machine is not highly optimized, the execution time is 50 or 500 times larger than the methods using FAM.

Table 1. Performance of the USA mainland map coloring

Method	CPU time (seconds)
DSN-FAM	0.2
CCM-FAM	0.02
Boltzmann Machine	10

* $f_0 = 10^{-16}$, $c = 2$.

Secondly, the sequential and parallel performance of DSN-FAM was evaluated using the Cray Superserver 6400 with 12 CPUs and shared memory. The results are shown in **Table 2**. The performance results are also average of 20 runs. The performance of CCM-FAM was measured using the same machine and the performance of SA using the Sequent Balance 21000 by Johnson et al. are also shown. A solution was always found by both DSN-FAM and CCM-FAM. A Boltzmann Machine was also tested, but no solution can be found within several hours. Thus, the performance of DSN-FAM is also much better than the Boltzmann Machine. Because the Balance 21000 is much slower than the Cray, we cannot conclude that DSN-FAM performs better than Johnsons' SA. CCM-FAM performs better than DSN-FAM because the number of neurons used in DSN-FAM is n_c times larger than the number of data used in CCM-FAM, where n_c is the number of colors. However, if the execution time of DSN-FAM is divided by n_c, the execution time is comparable with CCM-FAM.

Table 2. Performance of DSJC125.1 (a graph coloring problem)

Method	CPU time (seconds)
DSN-FAM (sequential)*	160
DSN-FAM (12 parallel)*	16.6
CCM-FAM (sequential)**	13.2
CCM-FAM (12 parallel)**	1.5
Johnson et al. (sequential)***	720

* $f_0 = 10^{-16}$, $c = 2$.
** $f_0 = 10^{-5}$, $c = 2$.
*** Sequent Balance 21000 was used.

The performance is improved nearly linearly by the parallel processing using 12 processors as same as CCM-FAM [Kan 96]. This fact is noteworthy because annealing techniques such as SA is difficult to parallelize and the parallel performance is not good, because of their global and sequential nature. For example, see Wong [Won 95].

6. Conclusion

A method of solving CSP using a symmetrically-connected neural network with discrete-output neurons (DSN) and using a new annealing technique, which is called FAM, is proposed in the present paper.

The advantages of DSN-FAM are as follows. Firstly, this method performs much better than DSN without FAM or Boltzmann Machines. Secondly, FAM requires no global parameters nor global control. FAM is based only on local information and works autonomously.

The disadvantages of DSN-FAM are as follows. This method can be used only for CSP. The constraints are embedded twice in the connection and in the mechanism of FAM, and the expression is thus redundant. This method currently has no theoretical basis.

Possible future work is to study the following extensions of DSN-FAM. The mechanism of DSN-FAM may be simplified without performance degradation. The simplification may cause formalization easier. A method that is an extension of FAM may be applied to continuous-state and/or continuous-time neural networks.

References

[Ack 85] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J.: A Learning Algorithm for Boltzmann Machines, *Cognitive Science*, 9, 147–169, 1985.

[DIM] Center for Discrete Mathematics and Theoretical Computer Science, *http://dimacs.rutgers.-edu/*.

[For 91] Forrest, S., ed.: *Emergent Computation*, MIT Press, 1991.

[Gu 93] Gu, J.: Local Search for Satisfiability (SAT) Problem, *IEEE Trans. on Systems, Man, and Cybernetics*, 23:4, 1108–1129, 1993.

[Joh 91] Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C.: Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, *Operations Research*, 39:3, 378–406, 1991.

[Kan 94a] Kanada, Y., and Hirokawa, M.: Stochastic Problem Solving by Local Computation based on Self-organization Paradigm, *27th Hawaii Int'l Conf. on System Sciences (HICSS-27)*, 82–91, 1994.

[Kan 94b] Kanada, Y.: Methods of Controlling Locality in Problem Solving using CCM: A Model for Emergent Computation, *Summer United Workshops on Parallel, Distributed, and Cooperative Processing '94* (*Swoop '94*), 94-AI-95-4, 29-38, 1994 (in Japanese).

[Kan 96] Kanada, Y.: CCM: A Model for Emergent Computation and Its Application to Combinatorial Problems, *IEEE Trans. on Systems, Man, and Cybernetics*, Submitted.

[Kir 83] Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P.: Optimization by Simulated Annealing, *Science*, Vol. 230, No. 4598, 671–680, 1983.

[Min 92] Minton, S., Johnston, M. D., Philips, A. B., and Laird, P.: Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems, *Artificial Intelligence*, 58, 1992, 161–205, 1992.

[Mor 93] Morris, P.: The Breakout Method For Escaping From Local Minima, *AAAI 93*, 40–45, 1993.

[Nak 94] Nakamura, T., Wakutsu, T., Aiyoshi, E.: Global Optimization Using Neural Networks with Linked State Transition, *Transactions of SICE*, 30:8, 966–975, 1994 (in Japanese).

[Sel 93] Selman, B., and Kautz, H.: Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems, *Int'l Joint Conf. on Artificial Intelligence '93 (IJCAI '93)*, 290–295, 1993.

[Tak 92] Takefuji, Y.: *Neural Network Parallel Processing*, Kluwer Academic Publishers, 1992.

[Tri] Tricks, M.: The Second DIMACS Challenge, *http://mat.gsia.cmu.edu/challenge.html*.

[Won 95] Wong, K. L., and Constantinides, A. G.: Problem-Independent Parallel Realisation of Simulated Annealing on a Ring Multiprocessor Architecture Based on Speculative Computation, *2nd Int'l Conf. on Evolutionary Computation*, 220–223, 1995.