Can a Niching Method Locate Multiple Attractors Embedded in the Hopfield Network?

Akira Imada[1] and Keijiro Araki[2]

[1] Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-01 Japan
akira-i@is.aist-nara.ac.jp

[2] Department of Computer Science and Computer Engineering
Graduate School of Information Science and Electrical Engineering
Kyusyu University
6-1 Kasuga-koen, Kasuga, Fukuoka, 816 Japan
araki@c.csce.kyusyu-u.ac.jp

Abstract. We apply evolutionary computations to the Hopfield's neural network model of associative memory. In the model, a number of patterns can be stored in the network as attractors if synaptic weights are determined appropriately. So far, we have explored *weight space* to search for the optimal weight configuration that creates attractors at the location of patterns to be stored. In this paper, on the other hand, we explore *pattern space* to search for attractors that are created by a fixed weight configuration. All the solutions in this case are *a priori* known. The purpose of this paper is to study the ability of a niching genetic algorithm to locate these multiple solutions using the Hopfield model as a test function.

1 INTRODUCTION

Associative memory is a dynamical system which has a number of stable states with a domain of attraction around them (Komlós et al. 1988). If the system starts at any state in the domain, it will converge to the stable state. Hopfield (1982) proposed a fully connected neural network model of associative memory in which information is stored by being distributed among neurons.

The Hopfield model consists of N neurons and N^2 synapses. Each neuron state is either active $(+1)$ or quiescent (-1). When an arbitrary N-bit *bipolar pattern*, a sequence of $+1$ and -1, is given to the network as an initial state, the dynamical behavior of neuron states afterwards are characterized by the strengths of the N^2 synapses. The synaptic strengths are called *weights* and the weight from neuron j to neuron i is denoted as w_{ij} in this paper. Provided the synaptic weights are determined appropriately, network can store some number of patterns as attractors. Hopfield employed the so-called Hebbian rule (Hebb, 1949) to prescribe the weights. That is, to store p bipolar patterns ξ^ν:

$$\xi^\nu = (\xi_1^\nu, \cdots, \xi_N^\nu), \quad \nu = 1, \cdots, p,$$

the weight values are determined as:

$$w_{ij} = \frac{1}{N} \sum_{\nu=1}^{p} \xi_i^\nu \xi_j^\nu \quad (i \neq j), \quad w_{ii} = 0.$$

An instantaneous state of a neuron is updated asynchronously (one neuron at a time) as:

$$s_i(t+1) = sgn\left(\sum_{j \neq i}^{N} w_{ij} s_j(t) \right),$$

where $s_i(t)$ is a state of the i-th neuron at time t. If an initial state converges to one of the stored patterns ξ^ν as an equilibrium state, then the pattern is said to be *recalled*. Furthermore, if an initial state chosen from the stored patterns remains unchanged from the beginning, then the pattern is said to be stored as a *fixed point*.

In analyzing the Hopfield model, there have been basically two different approaches: one is to explore *pattern space* searching for attractors under a specific weight configuration, and the other is to explore *weight space* searching for an appropriate weight configuration that stores a given set of patterns. To be more specific, the former is an analysis of the Hamiltonian energy as a function of all the possible configurations of bipolar pattern given to the network, where the synaptic weights are pre-specified using a learning algorithm, usually the Hebb's rule, so that the network stores a set of p given patterns. In this context, the model for $p = 1$ corresponds to the Mattis model of spin-glass (Mattis 1976), in which the Hamiltonian energy has two minima, while the model for infinitely large p corresponds to the Sherrington-Kirkpatrick model (1975), in which the synaptic weights become Gaussian random variables. Analyses of the former type have been made in between these two extreme cases (see Amit, 1989). The latter analysis was addressed by Gardner (1988). She discussed the optimal weight configurations for a *fixed* number of given patterns in terms of the volume of the solutions in weight space, suggesting that the volume shrinks to vanish when p approaches to $2N$. In short, the former approach searches for the optimal pattern configurations which minimize the Hamiltonian energy *in pattern space with the weights being fixed*, while the latter searches for the weight configurations *in weight space* that optimally store a set of given *fixed patterns*.

So far, we have studied the model with the latter approach. We have explored fitness landscapes of the model *defined on weight space*, and have found many solutions that store more patterns or store them with larger basin of attractions than, e.g., the Hebbian synaptic weights (Imada et al. 1997a, 1997b). Now, our interest is on the number and distribution of these solutions over the whole weight space, which is still an open problem. We think the niching GA is one of the appropriate tools to pursue these problems. However, since the N^2-dimensional *continuous* weight space is much more difficult to wander around than the N-dimensional *discrete* pattern space, we explore here the *pattern space* to see preliminary how our fitness function works under the niching method. In

other words, we use the model as a test function of the niching technique in the sense that all solutions are *a priori* known, like in other studies using pure mathematical test functions.

Since we explore the fitness landscape defined on the pattern space in this paper, the fitness function of an arbitrary configuration of N-bit bipolar pattern should be defined. Here we evaluate the fitness of a pattern according to how the instantaneous neurons' states $s_i(t)$ after the pattern is given to the network, are similar to either of the stored patterns. The similarity as a function of time is defined as:

$$m^{\nu}(t) = \frac{1}{N} \sum_{i=1}^{N} \xi_i^{\nu} s_i^{\nu}(t).$$

The temporal average of $m^{\mu}(t)$ is calculated for each stored pattern, and the maximum value among them is assigned to the input pattern as fitness. Thus, the fitness value f is evaluated as follows:

$$f = \max\{ \frac{1}{t_0} \sum_{t=1}^{t_0} m^{\nu}(t) \quad | \quad \mu = 1, 2, \cdots, p\}.$$

In this paper, t_0 is set to $2N$, twice the number of neurons. Note that the fitness 1 implies the pattern is a fixed point attractor, while fitness less than 1 includes many other cases.

2 Niching Methods

Niching genetic algorithm is a genetic algorithm (GA) (Holland 1975) that was devised to locate multiple solutions simultaneously. To do this, *sharing* (Goldberg & Richardson 1987), for example, derates the fitness value of each individual using *sharing function* which reflects the number of similar individuals in the population. Or *crowding* (De Jong 1975) reduces the number of similar individuals in the population by replacing some of the individuals with new individuals. Here, we employ the *deterministic crowding GA* (Mahfoud 1992) because of its niching capability (Mahfoud 1995) as well as the simplicity for implementation.

In each generation, as neatly summarized by Mahfoud (1995), the current population is reproduced as follows.

(1) Choose two parents, p_1 and p_2, at random, with no parent being chosen more than once.
(2) Produce two children, c_1 and c_2, with uniform crossover (Syswerda 1989).
(3) Mutate the children by flipping bit chosen at random with probability p_m, yielding c_1' and c_2'.
(4) Replace parent with child as follows:
- IF $d(p_1, c_1') + d(p_2, c_2') > d(p_1, c_2') + d(p_2, c_1')$
  - IF $f(c_1') > f(p_1)$ THEN replace p_1 with c_1'
  - IF $f(c_2') > f(p_2)$ THEN replace p_2 with c_2'
- ELSE

* IF $f(c_2') > f(p_1)$ THEN replace p_1 with c_2'
* IF $f(c_1') > f(p_2)$ THEN replace p_2 with c_1'

where $d(\zeta_1, \zeta_2)$ is the Hamming distance between two points (ζ_1, ζ_2) in pattern configuration space. The process of producing child is repeated until all the population have taken part in the process. Then the cycle of reconstructing a new population and restarting the search is repeated until all the global optima are found or a set maximum number of generation has been reached.

3 Experimental Results

Experiments were carried out on networks with 49 neurons. For a given set of p patterns, a configuration of synaptic weights in a network is calculated by the Hebb's rule. Thus, the network with 49 neurons now stores these p patterns as fixed point attractors, so far as p dose not exceed the storage capacity. Namely, the fitness landscape defined on pattern space has p global peaks exactly at the location of the patterns. Then the deterministic crowding GA searches for these attractors. The parameters of the GA employed in this paper are as follows. The population size is 200; mutation probability p_m is set to 0.05; and the number of iterations is limited to a maximum of 12,000 generations.

To take the bird's-eye view of the landscape, we picked up 240,000 samples randomly from pattern space. The number of samples was chosen to be equal to the typical number of function evaluations of our usual GA implementations. We observed that the fitness values were distributed ranging from 0.01 to 0.95. At first glance, this broad distribution might seem that the search for the optimal solution(s) is not so difficult. In fact, Davis (1990) pointed out that some problems can be solved more easily with a simple hill-climber than genetic algorithms. So, we experimented a random hill climbing on the landscape. Starting at a randomly chosen position in the space, a point is mutated 200 times, and then moves to the highest fitness point among mutants. We repeated a run varying p, but all we observed were early stagnations to a local minimum. The solutions of our problem were not reachable by a simple hill-climber. which satisfies Whitley et al.'s (1995) demand that test functions should be resistant to simple hill climbing searches.

Before applying the deterministic crowding, we also studied the problem with a simple GA where two parents are randomly selected from the best fit 40% of the population, and the worst 60% were replaced with the children produced, with other schemes such as recombination and mutation being the same as those described above. The simple GA founds easily the embedded patterns, but only one of them at a time. The typical result of the best and average fitness versus generation obtained by the simple GA is shown in Figure 1.

Next, we apply the deterministic crowding GA to this problem. We investigated how many of the embedded patterns can be located. For a specific number of patterns p, starting with $p = 2$, we studied 30 runs with varying random number seed at the start of a run. The experiments are iterated with p being

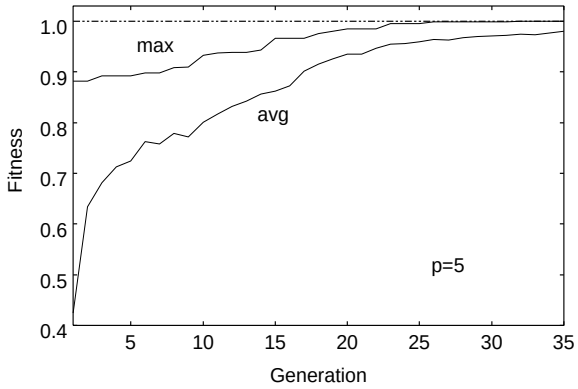

Fig. 1. Best and average fitness versus generation obtained by a simple GA.

incremented if any of the 30 runs locates *all* the embedded solutions. As results, we observed that when $p = 2$, 19 out of 30 runs locate all the embedded patterns; then as p increases, the number of successful runs decreases, i.e., when $p = 3, 4, 5, 6$ the number of successful trials out of 30 runs was 14, 8, 3, 0, respectively. Some examples of the number of individuals that converged on each of the given patterns when a GA run terminated are shown in Table 1.

Table 1. Number of individuals converged to each attractor.

	number of located attractors						
p	#1	#2	#3	#4	#5	#6	total
4	40	96	27	59	-	-	200
4	68	26	24	82	-	-	200
4	109	7	20	64	-	-	200
4	23	52	76	49	-	-	200
4	23	43	78	56	-	-	200
4	18	79	11	92	-	-	200
4	54	8	7	131	-	-	200
4	31	28	38	103	-	-	200
5	85	71	5	34	5	-	200
5	42	54	68	11	25	-	200
5	15	102	26	52	5	-	200
6	33	17	58	50	0	42	200

We can see that all the individuals eventually find some different niches for $p < 6$. Then, how the number of individuals in each of these niches changes as an evolution proceeds? An example is shown in Figure 2. Also the total number of these located solutions are shown in Figure 3, together with a result of a simple GA. The figure shows that all the individuals in the deterministic crowding GA converge to any of the solutions, while the simple GA reaches about half of the solutions.

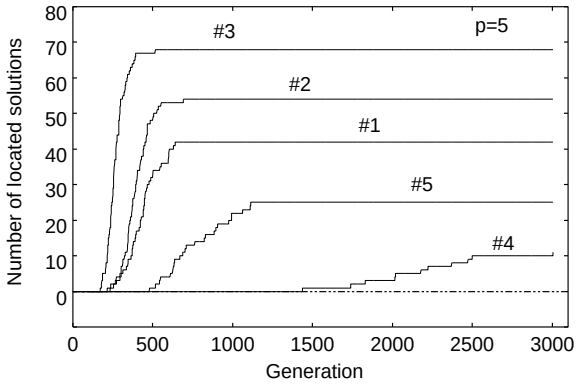

Fig. 2. Number of individuals converged on each niche.

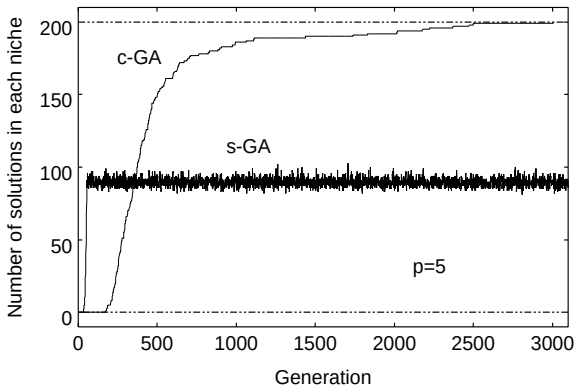

Fig. 3. Number of individuals that reach solutions.

4 Conclusion

We have described an application of the deterministic crowding GA, one of multi-modal function optimization techniques, to the Hopfield model of associative memory. What we concern is to obtain multiple solutions of weight configuration in weight space. We think the deterministic crowding GA is one of the candidates for the purpose. As a preliminary stage, the fitness landscape defined on pattern space instead of weight space was explored in this paper. As expected, we observed that convergence behaviors of the deterministic crowding GA were different from those of a simple GA. The deterministic crowding GA converges to multiple solutions, while a simple GA converges to one of these solutions. The number of solutions located by the deterministic crowding GA depends on trials. When more than 6 patterns are given, the deterministic crowding GA located only a part of them One possible reason for these results is that the number of embedded patterns exceeds the capability of the deterministic crowding GA to make niches on all of them. The other possibility might be due to our fitness evaluation. We are now experimenting with the Hamiltonian energy function for fitness evaluation.

The approach we have taken in this paper, that is, the exploration in pattern space to search for solutions, can be extended in the problem of searching for the solutions in weight space. Needless to say, much remains to be done. The simulations in this paper were made under specific limitations, such as low dimensionality and low cardinality of genes. Under experiments with the network with 49 neurons, the search space to be explored is vertices of the 49-dimensional hyper cube, while the weight space to be explored will be continuous entire region inside the 2401-dimensional hyper cube.

Although the analysis of the fully-connected network model of associative memory is rather classical, many issues are still unknown. Our goal of applying multi-modal-GA to the model is to address one of these issues, that is, the distribution of solutions in weight space. We believe these studies using evolutionary algorithms shed new light on the analysis of the model.

Acknowledgements

We thank Peter Davis at Advanced Telecommunication Research Institute (ATR) for providing us great insights into the dynamics of the Hopfield neural networks.

References

Amit, D. J. (1989) *Modeling Brain Function: The World of Attractor Neural Networks.* Cambridge University Press.

Davis, L (1991) *Bit-Climbing, Representation Bias, and Test Suite Design.* Proceedings of the 4th International Conference on Genetic Algorithms, pp.18–23.

De Jong, K. A. (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D. theses University of Michigan.

Gardner, E. (1988) *The Phase Space of Interactions in Neural Network Models.* Journal of Physics, 21A, pp257-270.

Goldberg, D. E., and J. Richardson (1987) *Genetic Algorithms with Sharing for Multimodal function Optimization.* Proceedings of 2nd International Conference on Genetic Algorithms, pp.41-49.

Hebb, D. O. (1949) *The Organization of Behavior.* Wiley.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems.* The University of Michigan Press.

Hopfield, J. J. (1982) *Neural Networks and Physical Systems with Emergent Collective Computational Abilities.* Proceedings of the National Academy of Sciences, USA, 79, pp2554-2558.

Imada, A., and K. Araki (1997a), *Random Perturbations to Hebbian Synapses of Associative Memory using a Genetic Algorithm.* Proceedings of International Work-Conference on Artificial and Natural Neural Networks. Springer Verlag, Lecture Notes in Computer Science, No.1240, pp398–407.

Imada, A., and K. Araki (1997b) *Evolution of Hopfield Model of Associative Memory by the Breeder Genetic Algorithm.* Proceedings of the 7th International Conference on Genetic Algorithms, pp784–791.

Komlós, J., and R. Paturi (1988) *Convergence Results in an Associative Memory Model.* Neural Networks 1, pp239–250.

Mahfoud, S. W. (1992) *A Comparison of Parallel and Sequential Niching Methods.* Proceedings of the 2nd Parallel Problem Solving from Nature, pp.27–36.

Mahfoud, S. W. (1995) *A Comparison of Parallel and Sequential Niching Methods.* Proceedings of the 6th International Conference on Genetic Algorithms, pp.136–143.

Mattis, D. C. (1976) *Solvable Spin Systems with Random Interactions.* Physics Letters, 56A, pp421–422.

Sherrington, D., and S.Kirkpatrick (1975) *Solvable Model of a Spin Glass.* Physical Review Letters 35, pp1792–1796.

Syswerda, G. (1989) *Uniform Crossover in Genetic Algorithms.* Proceedings of the 3rd International Conference on Genetic Algorithms, pp2–9.

Whitley, D., K. Mathias, S. Rana, and J. Dzubera (1995) *Building Better Test Functions.* Proceedings of the 6th International Conference on Genetic Algorithms, pp239–246.

This article was processed using the LaTeX macro package with LLNCS style