Evolution of Hopfield Model of Associative Memory by the Breeder Genetic Algorithm

Akira Imada
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

Keijiro Araki
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 Hopfield's neural network model of associative memory. Although there have been a lot of researches which apply evolutionary techniques to layered neural networks, their applications to Hopfield neural networks remain few so far. Previously we reported that a genetic algorithm using *discrete chromosomes* evolves the Hebb-rule associative memory to enhance its storage capacity. We also reported that the genetic algorithm evolves a network with random synaptic weights to store eventually some number of patterns as fixed points. In this paper we present an evolution of the Hopfield model of associative memory as a parameter optimization, using the Breeder Genetic Algorithm with *real-valued chromosomes*. The goal is not to explore a method to obtain associative memories of higher performances, but rather to shed new light on the analysis of Hopfield model from different aspects, which, we believe, also give a more challenging test suite for evolutionary computations.

1 Introduction

Associative memory is a dynamical system which has a number of stable states with a domain of attraction around them [?]. If the system starts at any state in the domain, it will converge to the stable state. In 1982, Hopfield [?] proposed a fully connected neural network model of associative memory in which we can store information by distributing it among neurons, and we can recall it from its noisy and/or partial input.

The dynamical behavior of its neuron states strongly depends on synaptic strength between neurons. The synaptic strength from neuron j to neuron i is called *weight* and denoted as w_{ij}. Hopfield used the Hebbian rule [?] to determine them. So far, a lot of methods to prescribe w_{ij}'s has been proposed.

Since Hopfield's proposal, many discussions have been made extensively, nevertheless some issues still remain open. How many solutions are in the space of w_{ij} which give networks a function of associative memory or how they are distributed, were the examples. We explore those issues by applying evolutionary computations to the model.

In this paper we present an application of the Breeder Genetic Algorithm (BGA) proposed by Mühlenbin et al. [?] to the Hopfield model of associative memory. The BGA is based on "*artificial selection similar to that used by human breeders*" expecting "*artificial selection to be more efficient for optimization than natural selection*" [?]. Mühlenbein et al. applied the BGA to multimodal test functions and concluded that "*the BGA mutation scheme is able to optimize many multi-modal functions.*" However, they also say "*But are such problems typical applications? We have not encountered such a problem.*"

Returning to the Hopfield model, we take it as a test function of parameter optimization problem. In fact, for a given set of random patterns to be stored, searching an appropriate combination of $\{w_{ij}\}$ is a real parameter optimization problem. Mühlenbein wrote in the paper that "*a good theory has to be proven with challenging applications*", and we take the Hopfield model as a more challenging test suite for evolutionary computations.

At the same time, this study shed new light on the analyses of the Hopfield model. If, for example, the BGA locate all the optimum solution for a set of given patterns, we will be able to have a new information regarding the distribution of the solution in the synaptic weight space.

The remainder of this paper is organized as follows. Since we only modify synaptic weights while other scheme being maintained in the framework of Hopfield model, the model is described more fully in the following section. The other evolutionary approaches to the Hopfield model will be described in the third section. The fourth section presents our BGA implementation. After presenting experimental results and discussion, the final section describe the concluding remarks.

2 Hoppfield Model

The model consists of N neurons and N^2 synapses. Each neuron can be in one of two states ± 1, and p bipolar patterns $\xi^\nu = (\xi_1^\nu, \cdots, \xi_N^\nu)$ $(\nu = 1, \cdots, p)$ are memorized as equilibrium states. Hopfield employed a discrete-time, asynchronous update scheme. Namely, at most one neuron updates its state at a time, according to

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

where $s_i(t)$ is a state of i-th neuron at time t, w_{ij} is a synaptic weight from neuron j to neuron i.

The behaviors of the collective states of individual neurons are characterized by the synaptic weights. When these synaptic weights are determined appropriately, networks store some number of patterns as fixed points. Hopfield specified these w_{ij}'s by the Hebbian rule [?], i.e.,

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

Then giving one of the memorized patterns, possibly including a few errors, to the network as an initial state will result in the stable state after certain time steps. Hopfield suggested by using computer simulation that the maximum number p of the patterns to be stored in a network with N neurons is $0.15N$, on the condition that a small error in recalling is allowed. Later, this was theoretically calculated by Amit et al. [?] by using *replica* method. They showed that the storage capacity is $p = 0.138N$. In 1987, McEliece et al. [?] proved that when $p < N/4\ln N$ holds, the Hopfield's model is able to recall the memorized patterns without error.

The storage capacity depends on how synaptic weights are specified. The specification of the synaptic weights is conventionally referred to as *learning*. Other learning schemes instead of Hebbian rule have been proposed for their increased storage capacity. The *pseud-inverse matrix method* proposed by Kohonen et al. [?] as an extensions of the Hebbian learning is an example, and the capacity is enlarged to $p = N$.

Then, what is the ultimate capacity when only the learning scheme is explored within the Hopfield's framework? Generally the storage capacity is traded off against basin size. Gardner [?] studied the optimal storage capacity as a function of size of basin of attraction.

She showed that the ultimate capacity will be $p = 2N$ as basin size tends to zero, and also proposed an algorithm to realize it.

3 Related Works

In the classical analyses of the Hopfield model, two versions of modification of the Hebbian synaptic weights J_{ij}'s are worth to note here, among others. One is Derrida et al.'s [?]

$$J_{ij} \times c_{ij},$$

where c_{ij}'s take the value of either 0 or 1. The other is Somploinsky's [?]

$$J_{ij} + \eta_{ij},$$

where η_{ij} is a Gaussian variable of mean 0 and small standard deviation. The objective of both modifications is to show a robustness of the network for synaptic noises.

Imada et al. [?] applied the Derrida et al.'s type of modification to their GA implementation for the Hopfield model. They reported that the GA enhances the storage capacity of the Hebb rule associative memory. They also successfully evolves the random synaptic weights to store some number of random patterns with the GA [?].

On the other hand, we apply the Somploinsky's modification to real encoded genetic algorithms. We also succeeded in evolving both Hebbian and random synaptic weight matrices [?].

One major difference of the modifications used in those GA implementations from the above classical studies is that the synapses to be modified are selected *adaptively* and *reproductively* rather than *randomly*.

In our previous implementation, an allele in the chromosome represents the perturbation η_{ij}, and this is added to a component of the original matrix which remains fixed during evolution. The genetic operations worked on these perturbations η_{ij}'s. In the experiments in this paper, on the other hand, it is the components of weight matrices w_{ij}'s that are represented by alleles, and perturbations are given by a mutation operator. In the former case, search points explores only a small domain around a specific point to be started, while in the latter case they are allowed to wander all over search space.

4 The BGA Implementation

In this simulation, a set of p random bipolar patterns ξ^ν ($\nu = 1, 2, \cdots, p$) are produced before each run. The goal of the BGA is to search weight matrices which store all of these patterns as fixed points. At the beginning of a run, a population of chromosomes are initialized. Chromosome is a real-valued N^2-dimensional vector which represents a weight matrix of individual network, i.e.,

$$(w_{11}, w_{12}, \cdots, w_{1N}, w_{21}, \cdots\cdots w_{NN}).$$

In each generation, these chromosomes are modified through crossover and mutation operation, and fitness value of individual phenotype is evaluated. According to the fitness values, individuals of the next generation are selected, using a $(\mu+\lambda)$-strategy in ES terminology. The cycle of reconstructing the new population with better individuals and restarting the search is repeated until a perfect solution is found or a maximum number of generation has been reached. Specific details of the operations are as follows.

Fitness evaluation: When one of memorized patterns ξ^ν is given to the network as an initial state, the state of neurons varies from time to time afterwards (unless ξ^ν is a fixed point). In order for the network to function as associative memory, the instantaneous network state must be similar to the input pattern. The similarity is defined by,

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

where $s_i^\nu(t)$ is the state of neuron i at time t. This $m^\nu(t)$ is referred to as *overlap*. The quality of retrieval of memorized patterns are represented by $\langle m^\mu \rangle$, a temporal average of $m^\nu(t)$ over certain time interval t_0. We evaluate the fitness value of each network by further averaging these $\langle m^\mu \rangle$'s over all memorized patterns. Namely, the fitness f is

$$f = \frac{1}{t_0 \cdot p} \sum_{t=1}^{t_0} \sum_{\nu=1}^{p} m^\nu(t).$$

In this paper, t_0 is set to $2N$, twice the number of neurons. Note that the fitness 1 implies all the p patterns are stored as fixed points, while all other cases have a fitness less than 1.

Selection: We use the truncation-selection [?], i.e., two parent chromosomes are chosen *randomly* from the best T% of the population. and mated randomly (except for itself) to produce one offspring. This is repeated until the number of offspring is equal to the size of the population.

Recombination: Recombination is made with discrete crossover [?] which is similar to *uniform crossover* [?] in binary chromosomes. It operates on alleles of the selected parents' chromosomes. Two parents, $(u_1, \cdots, u_n)$ and $(v_1, \cdots, v_n)$, produce an offspring $(w_1, \cdots, w_n)$ so that w_i is either u_i or v_i

with equal probability. Mühlenbein et al. [?] proposed other two crossovers: *extended intermediate crossover* and *extended line crossover*. We tested these two crossovers, however the results were both disappointing for our case. This is probably because the latter two schemes change the values of some synaptic weights, while the first one only shuffles them.

Mutation: An allele x_i is chosen with probability p_m to be mutated. The BGA mutation is to give a variable x_i a small perturbation $\Delta x_i \times \delta$, where Δx_i is a mutation range for the variable x_i, and δ is calculated as

$$\delta = \frac{\alpha_0}{2^0} + \frac{\alpha_1}{2^1} + \frac{\alpha_2}{2^2} + \cdots + \frac{\alpha_{15}}{2^{15}}.$$

Each α_k takes the value 1 with probability 1/16, and otherwise takes the value 0. Then mutated new allele is

$$x_i \pm \Delta_i \times \delta,$$

with the sign $+/-$ being chosen with equal probability.

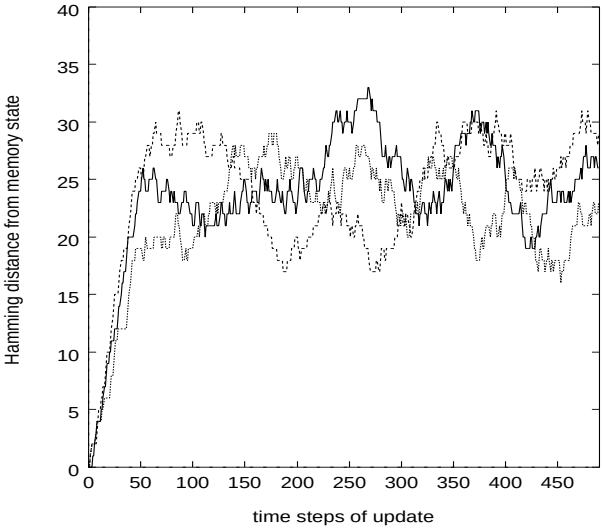

Figure 1: Three examples of chaotic trajectories

5 Experiment

All experiments in this paper are carried out on networks with 49 neurons. Update scheme are based exactly on the Hopfield model. Only synaptic weight values are explored with the BGA.

Parameters used for the BGA are as follows. The population number is 256. This is just because of our computer resources. In truncation selection, T is chosen to be 40%. The mutation range Δx_i is set to 0.1 for all x_i, since Mühlenbein et al. recommended it to be set to 0.1 times the search interval of x_i, and our x_i, synaptic weight value, takes the value out of $[-1, 1]$.

As for mutation probability p_m, Mühlenbein et al.'s suggestion is $1/n$, where n is the number of alleles in a chromosome, that is $1/49^2 = 1/2401$ in our problem. However, this is too small because of the robustness of the Hopfield network for synaptic noises. Hence we set it to $100/n$ instead.

Searching procedure is iterated until 12000 generation unless perfect solution is found.

These values were determined mainly on the basis of trial and error.

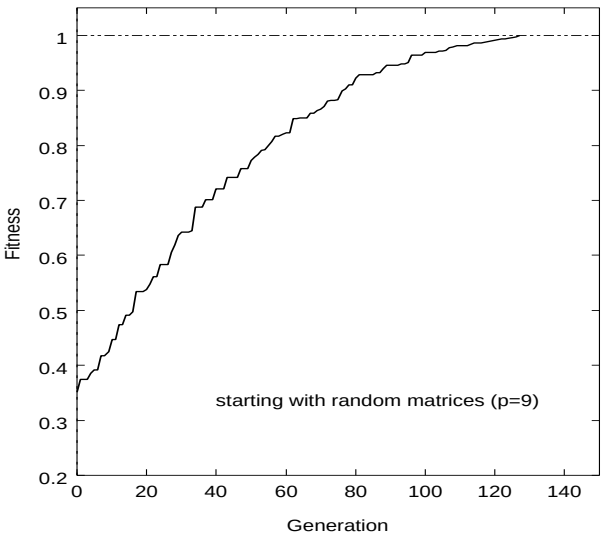

Figure 2: Best fitness vs generation

6 Results and Discussion

In the first experiment, individuals are initialized randomly. Namely, each network in the first generation has a different set of random synaptic weights. They do not

function as associative memory. Each input of a given set of random patterns to these networks results in a chaotic trajectory. In Figure ??, we show an example of these trajectories from our experiment. The Hamming distance of instantaneous network state from the input is depicted as a function of updating time step. The BGA evolves these random matrices by adapting them to a given set of random patterns. We observed that matrices emerged which eventually store a maximum of 8 patterns as fixed points. In Figure ?? we show the evolution of the best fitness value in each generation for 8 patterns.

In this experiment we have not obtained the perfect solution for a given set of more than 9 patterns. This limit is slightly higher than the capacity of the network of the same size when synaptic weights are determined by Hebbian rule.

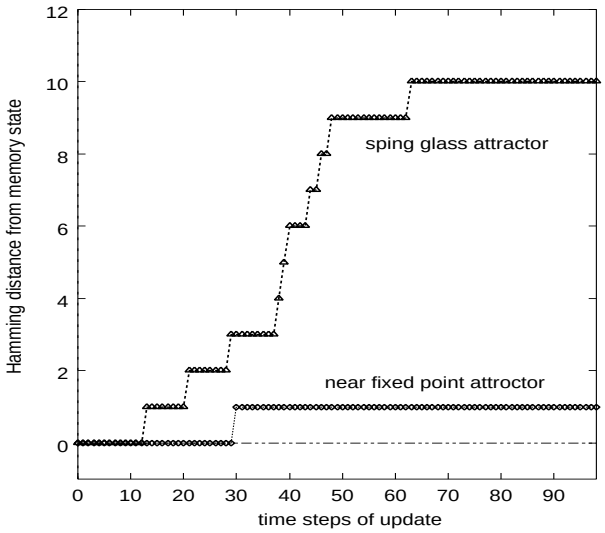

Figure 3: Attractors of an over-loaded Hebbian matrix

Then we proceed to the next experiment where we initialize all the individuals with a Hebbian matrix. Here, p, the number of given patterns, exceeds the storage capacity of the Hebbian learning.

The network of this type was analytically studied by Amit et al. [?]. As the number p increases, some memorized states are shifted slightly, and finally the system is dominated by many spin-glass attractors and can not function as associative memory. We observe these attractors again by the time evolution of Hamming distance between instantaneous neuron states and input pattern. The example is taken from our experiment in which a network has learned a set of 25 patterns by the Hebbian rule. Although each input of these 25 learned patterns converged to a stable state, none of them were not memorized states. The Hamming distances of these stable states from their corresponding memorized patterns were 1, 1, 2, 4, 5, 10, 11, 12, 12, 12, 13, 14, 14, 14, 15, 15, 16, 17, 17, 17, 19, 19, 19, 21 and 22. No fixed point attractor was found. In Figure ?? two out of 25 trajectories for the inputs of learned patterns are shown.

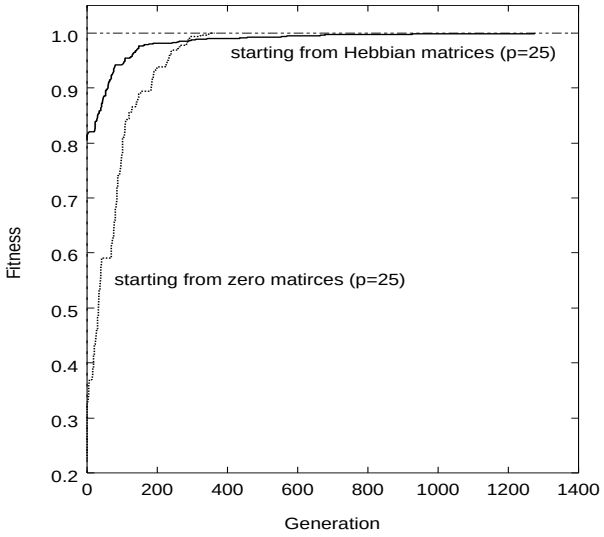

Figure 4: Best fitness vs generation

The network of the above example was evolved by the BGA. Starting with a population of these same Hebbian matrices, the BGA evolved out a matrix which create fixed point attractors at the positions of learned patterns. The maximum p we have observed succesfull evoluntions for was 25. This is surprisingly large compared to our previous implementations of other evolutionary computations. At first, we thought that this success is due to the a priori Hebbian learning. However the next experiment will reveals us this is incorrect.

In the third experiment we initialize individuals with an all-zero-component matrix (*tabula rasa*) instead of the Hebbian matrix. Starting with these individuals, perfect solution emerges which stores all the given 25 patterns as fixed points.

In Figure ??, we show the results of both the 2nd and 3rd experiment. Surprisingly, the evolution starting with zero matrix presents rather better performance than starting with Hebbian matrix. Each experiment is repeated 30 times with different random number seed. The BGA starting with zero matrices locates the optimum in 25 runs, while 7 runs locate the optimum starting with Hebbian matrices. In addition, the comparison of the best performance of both experiments show that

the zero matrix converges more rapidly to the optimum than the Hebbian one as shown in Figure ??.

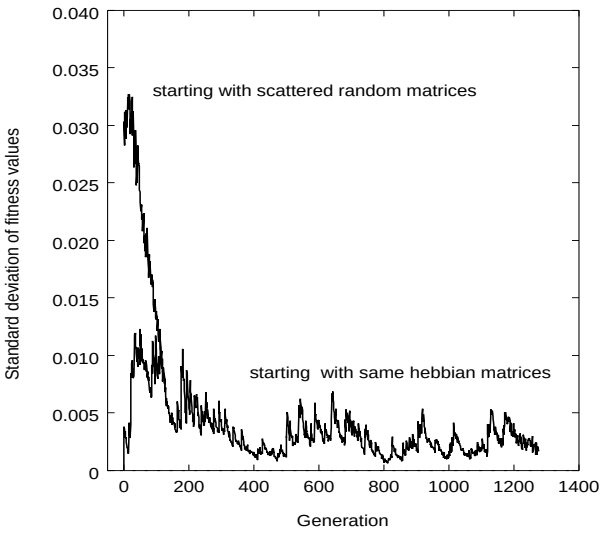

Figure 5: Standard deviation vs generation

Usually individuals are initialized randomly as our first experiment, so that starting points are distributed randomly over the search space. In fact, Mühlenbein et al. wrote "convergence means that all individuals have the same value." In the latter two experiments all individuals start the search from a specific point in the space. Unless the search stagnates by some local minima near the point, this initialization works much more effective for the current problem. In this case, the discrete crossover does not work initially, since all the individuals are similar. However the mutation increases diversity of the population quickly, which enables evolution. In Figure ??, the standard deviation of the fitness value is plotted against generation, for these both cases of initialization. We can see, in the 2nd and 3rd experiment, a rapid growth of fitness diversity in early stages. Then it converges again to the similar solution. In Figure ??, we also show the fitness distribution of both cases at some specific generations.

To see further differences between the two experiments, we investigated the time evolution of degree-of-symmetry of weight matrices. Following Krauth et al. [?], we define the degree-of-symmetry as follows:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}w_{ji} \cdot \left(\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^2\right)^{-1}.$$

We show the results of the 2nd and 3rd experiment in Figure ??. At the beginning the Hebbian matrix is totally symmetric. As evolution proceeds, the degree of symmetry decreases and asymptotically approaches to the value around 0.6. The BGA starting with the Hebbian matrix seems to hillcrimb by introducing small amount of asymmetry to the Hebbian matrix, while in the BGA starting with zero matrix, this is not possible.

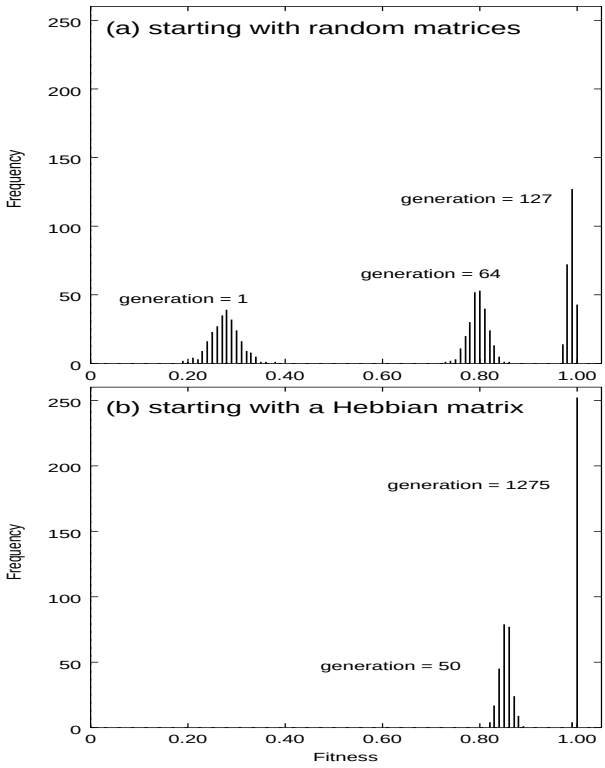

Figure 6: Fitness distribution

Hertz et al. [?] suggested that an asymmetry of synaptic weights destabilizes spin-glass attractors, only affecting mildly the memorized states. The effect of stabilizing spin-glass attractors will enlarge the basin of attraction around the memorized states. We observed the basin size of the network of the above experiments as follows. An input is produced from the memorized patterns by flipping d bits at random. Then this is given to the network, and temporal average of *overlaps* of the network state visited by the dynamics is calculated. This is repeated 800 times for each d. In Figure ??, averaged overlap over the 800 repetitions are plotted against the number of noisy-bit d, together with the results derived from the Gardner's optimized algorithm [?] for the same set of memorized patterns. The BGA starting with a Hebbian matrix can establish a fair amount of asymmetry which leads to about a same size of basin of attraction of the

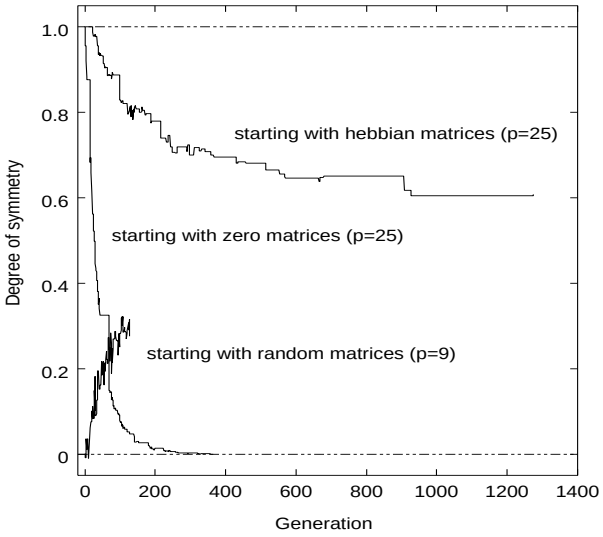

Figure 7: Degree of symmetry vs generation

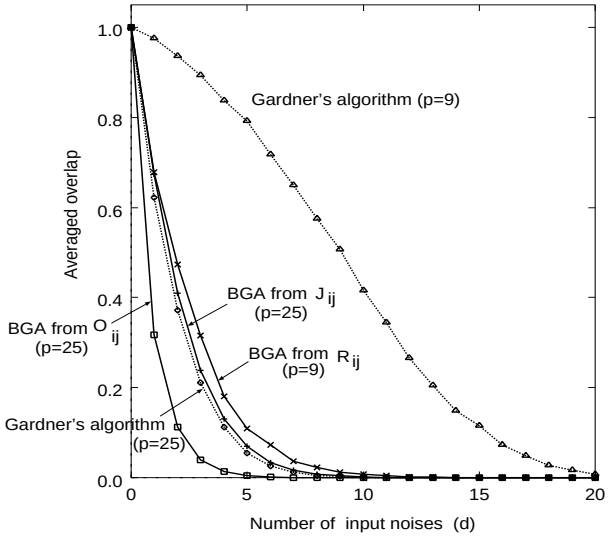

Figure 8: Similarity of updated states with noisy input

Gardner's optimized network. The suggestion by Hertz et al. seems to be supported here. In the figure, the results from the 1st experiment is also shown, together with a result of the Gardner's algorithm for convenience.

Parisi [?] suggested another effect of asymmetry, i.e., converting the spin-glass attractors into chaotic trajectories. We leave this issue to our future research.

7 Conclusions

We have described an application of the Breeder Genetic Algorithm to the Hopfield model of associative memory. Networks with 49 neurons evolved to create fixed points attractors at the exact positions of a given set of 25 random patterns. The network of this size would store only 8 patterns with Hebbian learning rule. Our previous experiment using Evolution Strategy (ES), for example, found the network which stores a maximum of 17 random patterns. The evolutionary algorithms using adaptive mutation like ES require elaborate tunings of the relevant parameters. Mülenbein et al. [?] wrote: "the BGA mutation does not need an adaptation of the mutation range." All we had to determine were population size and probability p_m. (Note that $p_m = 1/n$, where n is a number of variables, is recommended for usual applications by Mühlenbein et al.)

We expect the BGA will be able to find solutions for more number of given patterns. Furthermore, if, for example, the BGA is able to locate all the solutions in the synaptic weight space which give networks a function of associative memory, then we will have an important information about the number and the distribution of these solutions.

Although this multi-modality of the Hopfield model might present us many hurdles to clear. Especially, the problem of locating the global optimum in the sense of Gardner will be a needle in a haystack. However we believe these studies will shed new light on the analysis of the Hopfield model, as well as present a more challenging test suite for evolutionary computations.

Other problems like the fate of the spin-glass attractors or the chaotic trajectories which exist before evolution, are also waiting for our explorations.

Acknowledgments

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

References

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

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

[3] D. O. Hebb (1949) *The Organization of Behavior.* Wiley.

[4] H. Mühlenbein, M. Schomish, and J. Born (1991) *The Parallel Genetic Algorithm as Function Optimizer.* Parallel Computing **17** pp619–632.

[5] D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) *Statistical Mechanics of Neural Networks near Saturation* Annals. of Physics 173, pp30-67.

[6] R. J. McEliced, E. C. Posner, E. R. Rodemick, and S. S. Venkatesh (1987) *The Capacity of the Hopfield Associative Memory.* IEEE Trans. Information Theory **IT-33**. pp461-482.

[7] T. Kohonen, and M. Ruohonen (1973) *Representation of Associated Data by Matrix Operators.* IEEE Trans. Computers **C-22(7)**. pp701–702.

[8] E. Gardner (1988) *The Phase Space of Interactions in Neural Network Models.* J. Phys A: Math Gen. **21**. pp257–270.

[9] B. Derrida, E,Gardner, A. Zippelius (1987) *An Exactly solvable Asymmetric Neural Network Model* Europhysics Letter 4(2) pp167-173.

[10] H. Sompolinsky (1986) *Neural Network with Nonlinear Synapses and Static Noise.* Phys. Rev. **A34** pp2571-2574.

[11] A. Imada, and K. Araki (1995) *Genetic Algorithm Enlarges the Capacity of Associative Memory.* Proceedings of the 6th International Conference on Genetic Algorithms. pp413–420.

[12] A. Imada, and K. Araki (1995) *Genetic Algorithm Enlarges the Capacity of Associative Memory.* Proceedings of the 6th International Conference on Genetic Algorithms. pp413–420.

[13] Authors of this paper (1997) *Random Perturbations to Hebbian Synapses of Associative Memory using a Genetic Algorithm.* (submitted)

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

[15] D. J. Amit, H. Gutfreund, and H. Sompolinsky (1985) *Storing Infinite Number of Patterns in a Spin-glass Model of Neural Networks.* Physical Review Letters **55**. pp1530-1533.

[16] J. A. Hertz, G. Grinstein, and S. A. Solla (1987) *Irreversible Spin Glasses an Neural Networks.* in L. N. van Hemmen and I. Morgenstern eds. Heidelberg Colloquium on Glassy Dynamics. Lecture Notes in Physics No.275 Springer-Verlag, pp538–546.

[17] W. Krauth, J.-P. Nadal, and M.Mezard (1988) *The Roles of Stability and Symmetry in the Dynamics of Neural Networks.* J. Phys. A: Math. Gen. **21** pp2995–3011.

[18] G. Parisi (1986) *Asymmetric Neural Networks and the Process of Learning.* J. Phys. **A19** ppL675–L680.