Effective Search with Hopping Chaos for Hopfield Neural Networks Solving QAP

Yoshifumi Tada, Yoko Uwate and Yoshifumi Nishio
Department of Electrical and Electronic Engineering,
Tokushima University,
2-1 Minami-Josanjima, Tokushima 770-8506, JAPAN
Email:{y-tada, uwate, nishio}@ee.tokushima-u.ac.jp

Abstract— **Many people propose the method adding chaos noise to Hopfield Neural Network for solving combinatorial optimization problems. In our past study, we solved Quadratic Assignment Problem by Hopfield neural network with various chaotic noises. However, the solution of the network is sometimes trapped in a certain area and can not find a good solution, especially for relatively larger problems. In this study, we propose a method changing the amplitude of the chaos noise. We investigate the performance of Hopfield Neural Network with the hopping chaos for Quadratic Assignment Problem.**

I. Introduction

Solving combinatorial optimization problem is one of the application of the Neural Network (abbr.NN). If we choose connection weights between neurons appropriately according to given problems, we can obtain a good solution by the energy minimization principle. However, the solutions are often trapped into a local minimum and do not reach the global minimum. In order to avoid this critical problem, several people proposed the method adding some kinds of noise for solving traveling salesman problems (TSP) with the Hopfield NN [1]. Hayakawa and Sawada pointed out the chaos near the three-periodic window of the logistic map gains the best performance [2]. They concluded that the good result might be obtained by a property of the chaos noise; short time correlations of the time-sequence. Hasegawa et al. investigated solving abilities of the Hopfield NN with various surrogate noise, and they concluded that the effects of the chaotic sequence for solving optimization problems can be replaced by stochastic noise with similar autocorrelation [3]. We have also studied the reason of the good performance of the Hopfield NN with chaotic noise. We imitated the intermittency chaos noise by the burst noise generated by the Gilbert [4] model with 2 states; a laminar state and a burst state. We concluded that the irregular switching of laminar part and burst part is one of the reasons of the good performance of the chaotic noise [5]-[7]. However, we often observed that only a very small number of solutions are found. We consider that finding a lot of nearly optimal solutions is important than finding the optimal solution for difficult problems.

In this study, we propose a method changing the amplitude of the chaos noise to escape from the area where the network have almost finished searching. By using this method named as the hopping chaos, we investigate the performance of Hopfield Neural Network for Quadratic Assignment Problem.

II. Solving QAP with Hopfield NN

Various methods are proposed for solving the QAP which is one of the NP-hard combinatorial optimization problems.The QAP is expressed as follow: given two matrices, distance matrix C and flow matrix D, and find the permutation P which corresponds to the minimum value of the objective function f(p) in Eq. (1).

$$f(P) = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} D_{\mathrm{p}(i)\mathrm{p}(j)}, \tag{1}$$

where C_{ij} and D_{ij} are the (i,j)-th elements of C and D, respectively, p(i) is the ith element of vector P, and N is the size of the problem. There are many real applications which are formulated by Eq. (1). One example of QAP is find an arrangement of the factories to make a cost the minimum. The cost is given by the distance between the factories and flow of the products between the factories. Other examples are the placement of logical modules in a IC chip, the distribution of medical services in large hospital.

Because the QAP is very difficult, it is almost impossible to solve the optimum solutions in large problems. The largest problem which is solved by deterministic methods may be only 24 in recent study. Further, computation times is very long to obtain the exact optimum solution. Therefore, it is usual to develop heuristic which search near optimal solutions in reasonable time.

For solving N-element QAP by Hopfield NN, $N \times N$ neurons are required and the following energy function is defined to fire (i,j)-th neuron at the optimal position:

$$E = \sum_{i,m=1}^{N}\sum_{i,n=1}^{N} w_{im;jn} x_{im} x_{jn} + \sum_{i,m=1}^{N} \theta_{im} x_{im}, \tag{2}$$

The neurons are coupled each other with weight between (i,m)-th neuron and (j,n)-th neuron and the threshold of the (i,m)-th neuron are described by:

$$w_{im;jn} = -2\left\{A(1-\delta_{mn})\delta_{ij} + B\delta_{mn}(1-\delta_{ij}) + \frac{C_{ij}D_{mn}}{q}\right\} \tag{3}$$

$$\theta_{im} = A + B \tag{4}$$

where A and B are positive constant, and δ_{ij} is Kroneker's delta. The state of $N \times N$ neurons are asynchronously update due to following difference equation:

$$x_{im}(t+1) = f\left(\sum_{j,n=1}^{N} w_{im;jn} x_{im}(t) x_{jn}(t) - \theta_{im} + \beta z_{im}(t) \right) \tag{5}$$

where f is sigmoidal function defined as follows:

$$f(x) = \frac{1}{1 + \exp\left(-\frac{x}{\varepsilon}\right)} \tag{6}$$

z_{im} is additional noise, and β limits amplitude of the noise.

We use the method proposed by Sato et al. to decide firing of neurons [8].

III. Chaos Noise

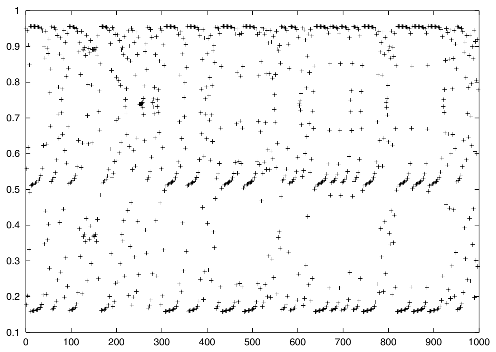
In this section, we describe chaos noise injected to the Hopfield NN. The logistic map is used to generate the chaos noise:

$$\hat{l}_{im}(t+1) = \alpha \hat{l}(t)(1 - \hat{l}_{im}(t)) \tag{7}$$

Varying parameter α_l, Eq. (7) behaves chaotically via a periodic-doubling cascade. Further, it is well known that the map produces intermittent bursts just before periodic-windows appear. Figure 1 shows an example of the intermittency chaos near the three-periodic window obtained from Eq. (7) for $\alpha_l = 3.8274$. As we can see from the figure, the chaotic time series could be divided into two phases; laminar parts of periodic behavior with period three and burst parts of spread points over the invariant interval. As increasing α_l, the ratio of the laminar parts becomes larger and finally the three-periodic window appears. We use the intermittency chaos time series after the following normalization.

$$l_{im}(t+1) = \frac{\hat{l}_{im}(t) - \bar{y}}{\sigma_l} \tag{8}$$

where $\bar{l}$ and σ_l are the average and the standard deviation of $\hat{l}(t)$, respectively



$\alpha_l = 3.8274.$

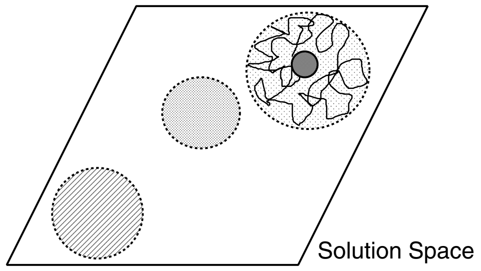
Fig. 1. Time series obtained from logistic map.

IV. Algorithm of Hopping Chaos

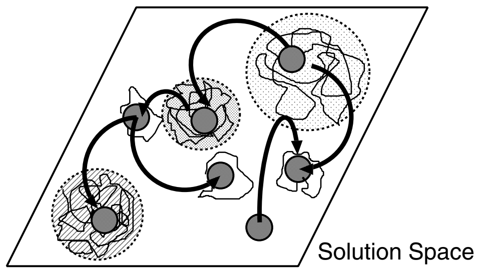
The Hopfield NN with the noises can find various solutions. However, the state of the Hopfield NN sometimes stays around one solution or a group of several solution. We consider that such a behavior is not useful to find the optimal or nearly optimal solution. So, we propose the algorithm to find many kinds of solutions in large solution space by hopping chaos. If the network finds the same solution, our proposed algorithm makes the network search another area by changing the noise amplitude β. Namely, searching area moves to a new area and the network searches many different solutions. We denote the standard value of β as β_0 and the bigger β as β_{jump}. The conceptual figure of searching in solution space is shown in Fig. 2. If the network finds the same solution as one found before, β value is changed to β_{jump}. If the network finds a new solution just after jumping, β value returns back to β_0 from β_{jump}. While if the network still find the same solution after jumping, we increase β according to the following update equation;

$$\beta = \beta_{jump} + (\beta_{jump} - \beta_0)(n - 1) \tag{9}$$

where n is the number of times that the network finds the same solution continuously after jump. Namely, if the network can not find a new solution, the amplitude β of additional noise becomes bigger and bigger. We consider that the network with constant amplitude chaos noise, the network can not search large solution area, on the other hand, the network search larger area in the solution space by using the proposed algorithm of hopping chaos.



(a) Chaos noise.



(b) Algorithm of hopping chaos.

Fig. 2. Searching in solution space.

V. Simulation Results

In this section, the simulated results of the Hopfield NN for QAP with the proposed algorithm are shown.

The first target problem is named as "Nug12" chosen from the site QAPLIB [9]. The number of the elements is 12 and the optimal solution is known as 578. The parameters of the Hopfield NN are fixed as $A = 0.9$, $B = 0.9$, $q = 140$, and $\varepsilon = 0.02$ and the amplitude of the injected noise is fixed as $\beta_0 = 0.55$. The iteration number $N_{iteration}$ of the network is fixed as 10000.

A. Frequency Distribution

We examine frequency distribution of the obtained solutions. First, we explain how to accept the solutions. The Hopfield NN with the noises can find various solutions. However, the state of the Hopfield NN sometimes stays around one solution or a group of several solutions. We consider that such a behavior is not useful to find the optimal or nearly optimal solutions. So, we take the only-different-solutions method. Namely, we take into account only the solutions whose firing patterns have not found ever in each trial. In other words, we accept only the first time if the completely same firing patterns appear more than once in each trial.

The results of the frequency distribution when the value of β_{jump} is changed are shown in Fig. 3. The frequency means the number of the accepted solutions with the corresponding costs found during 10000 iterations. We can see that a lot of solutions are found for the cases of the proposed algorithm (Fig. 3(b)-(f)). On the other hand, only a very small number of the solutions are found for the network without proposed algorithm (Fig. 3(a)). Furthermore, we confirm the performance of the random search network is shown in Fig. 4. The random search network find many solution, however almost of the solved solutions are very bad. From these results, we can say that the proposed hopping chaos method is effective for solving QAP.

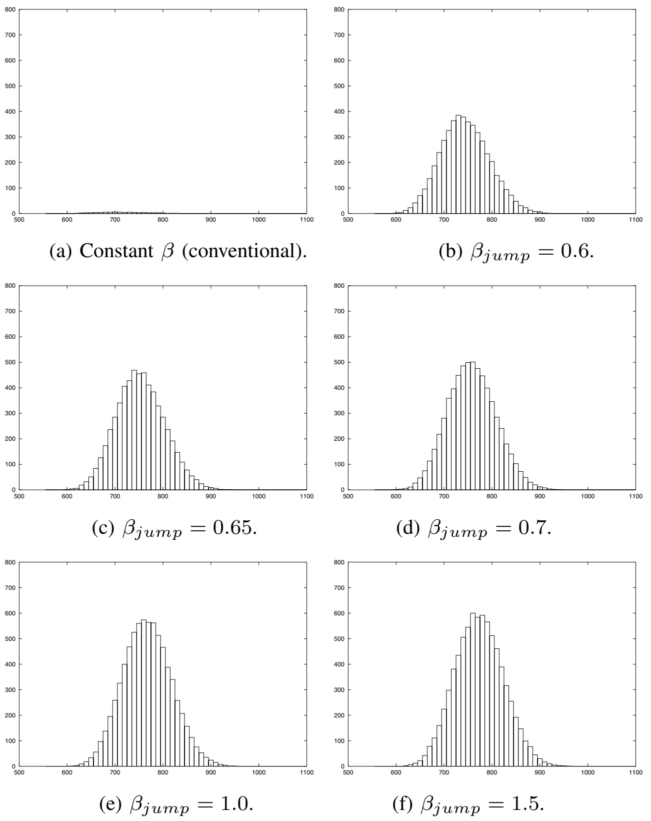

(a) Constant β (conventional). (b) $\beta_{jump} = 0.6$. (c) $\beta_{jump} = 0.65$. (d) $\beta_{jump} = 0.7$. (e) $\beta_{jump} = 1.0$. (f) $\beta_{jump} = 1.5$.

Fig. 3. Frequency distribution of solutions for Nug12.

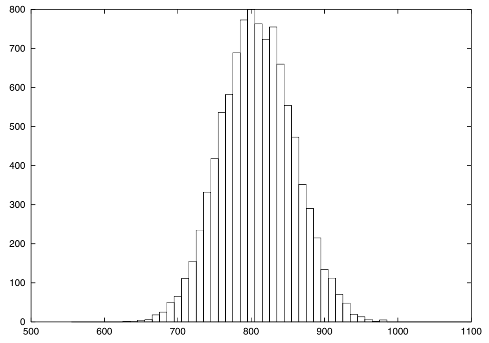


Fig. 4. Frequency distribution of solutions of random search for Nug12.

B. Depth [7]

Many people have evaluated the performance of the Hopfield NN with noises by using only the best solution found in each trial; time to find the optimal solution, frequency of finding the optimal solution during a given time, average of the minimum cost in each trial, and so on. However, such methods are not sufficient to reflect the number of the obtained solutions as shown in the previous subsection. Further, when the network can not find the optimal solution, the methods using only the optimal solution do not give a correct evaluation of the network. Namely, for very difficult problems, we consider that it is more important to find a lot of nearly optimal solutions than to happen to find the optimal solution. Therefore, we have proposed two evaluation methods to appreciate finding a lot of nearly optimal solutions.

1) Depth_1: The first evaluation method $Depth\_1$ is defined as

$$Depth\_1 = \sum_{k=0}^{n} \{f(\mathbf{p}_k) - D_\infty\}^2 \tag{10}$$

where D_∞ is a constant which is large enough to include the costs of all solutions, n is the number of the accepted solutions and the cost $f(\mathbf{p}_k)$ is calculated by Eq. (1) using the permutation $\mathbf{p}_k$ corresponding to the k-th accepted solution.

2) Depth_2: The second evaluation method $Depth\_2$ is defined as

$$Depth\_2 = \sum_{k\in\mathbf{k}_g}^{n} \{f(\mathbf{p}_k) - D_{th}\}^2 - \sum_{k\notin\mathbf{k}_g}^{n} \{f(\mathbf{p}_k) - D_{th}\}^2 \tag{11}$$

$$\text{where} \qquad \mathbf{k}_g = \{k \mid f(\mathbf{p}_k) \le D_{th}\}.$$

This evaluation has an advantage such that we can set the threshold D_{th} according to the requirement. We consider that finding a lot of bad solutions makes the performance of the network worse. However, the value of $Depth\_1$ increases even if the obtained solution is very bad. Hence, in this evaluation, we not only set up a threshold but give a penalty according to the cost. Namely, if the network finds a solution with the cost more than a given threshold value, the value of $Depth\_2$ is reduced.

The calculated results of $Depth\_1$ and $Depth\_2$ and the number of the solved solution during 10000 iterations are shown in Figs. 5, 6 and 7. The horizontal axis is the value of β_{jump}. The data at $\beta_{jump} = 0.55$ means the network with constant amplitude chaos noise. We confirm that the proposed algorithm gains better performance than the network with constant amplitude chaos noise. We also noticed that the $Depth\_2$ becomes worse as β_{jump} increases. This is because the network becomes close to the random search network finding a lot of bad solutions.

Next, we compare the hopping chaos and the hopping random noise which is the method obtained by replacing the intermittency chaos by uniform random noise. In the result of the solved solutions, the hopping random noise obtains more solutions than hopping chaos around β_{jump}=0.6. However, the hopping chaos gains slightly better performance of $Depth\_2$ than the hopping random noise. We consider that the chaotic nature is important but the hopping concept makes its effect smaller.

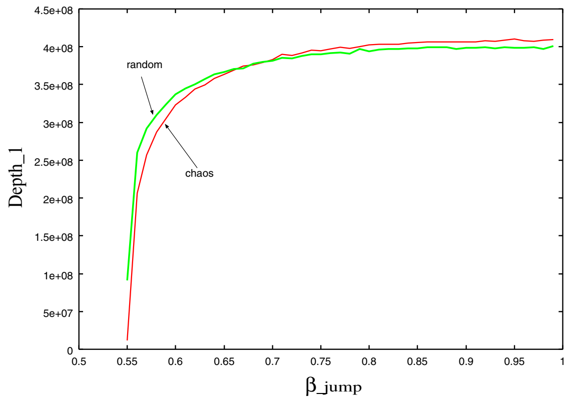


Fig. 5. Simulated result of $Depth_1$ ($D_\infty = 1000$).

VI. Conclusion

In this study, we proposed the hopping chaos method which is changing the amplitude of the chaos noise injected to Hopfield Neural Network to escape from the area where the network have almost finished searching. By computer simulation, we confirmed that the network with the proposed algorithm can find many good solutions.

References

[1] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," Proc. Natl. Acad. Sci. USA, vol.81, pp.3088–3092, 1984.

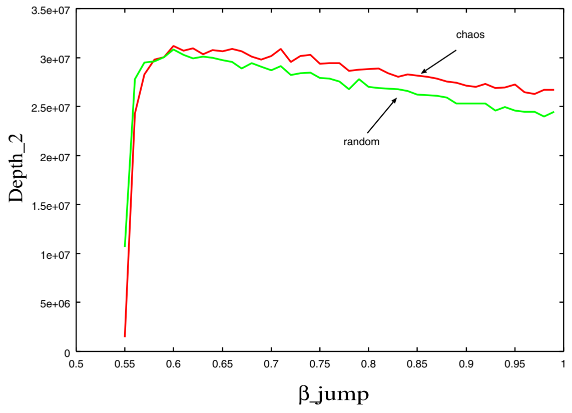


Fig. 6. Simulated result of $Depth_2$ ($D_{th} = 812$).

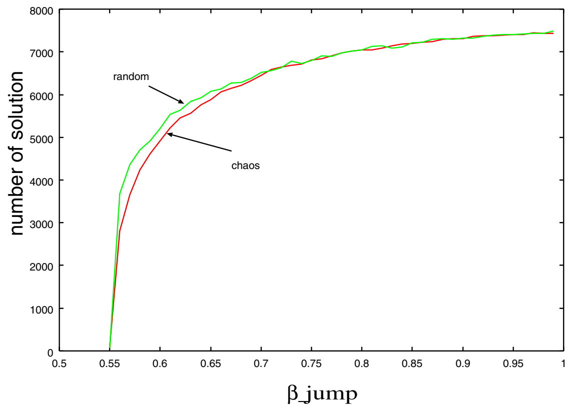


Fig. 7. Simulated result of number of solved solutions.

[2] Y. Hayakawa and Y. Sawada, "Effects of the chaotic noise on the performance of a neural network model for optimization problems," Physical Review E, vol.51, no.4, pp.2693–2696, Apr. 1995.

[3] M. Hasegawa, T. Ikeguchi, T. Matozaki and K. Aihara, "An analysis on additive effects of nonlinear dynamics for combinatorial optimization," IEICE Trans. Fundamentals, vol.E80-A, no.1, pp.206–213, Jan. 1997.

[4] M.C. Jeruchim, P. Balaban and K.S. Shanmugan, Simulation of Communication Systems, pp.245–247, Plenum Press, New York, 1992.

[5] T. Ueta, Y. Nishio and T. Kawabe, "Comparison between Chaotic Noise and Burst Noise on Solving Ability of Hopfield Neural Networks," *Proc. NOLTA'97*, vol. 1, pp. 409-412, Nov. 1997.

[6] Y. Uwate, Y. Nishio and A. Ushida, "Markov Chain Modeling of Intermittency Chaos and its Application to Hopfield NN," *IEICE Transactions on Fundamentals*, vol. E87-A, no. 4, pp. 774-779, Apr. 2004.

[7] Y. Uwate, Y. Nishio, T. Ueta, T. Kawabe and T. Ikeguchi, "Performance of Chaos and Burst Noise Injected to the Hopfield NN for Quadratic Assignment Problems," *IEICE Transactions on Fundamentals*, vol. E87-A, no. 4, pp. 937-943, Apr. 2004.

[8] K. Sato, T. Ikeguchi, M. Hasegawa and K. Aihara, "An optimization method for quadratic assignment problems by chaotic dynamics and its characterization by Lyapunov dimensions," IEICE Tech. Rep., vol.NLP64-13, pp.13–20, 2001 (in Japanese).

[9] R.E. Berkard, S.E. Karisch and F. Rendl, "QAPLIB – a quadratic assignment problem library," `http://www.opt.math.tu-graz.ac.at/qaplib`