A Dynamical N-Queen Problem Solver using Hysteresis Neural Networks

Takao Yamamoto†, Kenya Jin'no† and Haruo Hirose†
†Department of Electrical and Electronics Engineering, Faculty of Engineering
Nippon Institute of Technology, Saitama, Japan
Tel. +81-480-33-7672, Fax.: +81-480-33-7680

Abstract: In previous study about combinatorial optimization problem solver by using neural network, since Hopfield method, to converge into the optimum solution sooner and certainer is regarded as important. Namely, only static states are considered as the information. However, from a biological point of view, the dynamical system has lately attracted attention. Then we propose the "dynamical" combinatorial optimization problem solver using hysteresis neural network. In this article, the proposal system is evaluated by the N-Queen problem.

1. Introduction

The problem to find the optimum combination on some constraint conditions is called combinatorial optimization problem. This problem is discussed at the various cases[1]. It is difficult to solve such problem in the case where the problem has huge number of combinations. N-Queens problem which this article deals with is one of the combinatorial optimization problems.There are precedents to apply neural networks to combinatorial optimization problems, for famous example, mutually connected neural network called Hopfield model is proposed by D.W.Tank and J.J.Hopfield[2], [3]. In Hopfield model, a monotonously decreasing function called energy function is defined. The cost function defined to evaluate the solution in combinatorial optimization problem corresponds to the energy function. Then, when the energy function converges into its global minimum, the cost function indicates the optimum solution. However, if many local minima exist in the energy function, the system hardly converges into the global minimum. Therefore, this system can find the optimum solution in few cases. Then, the method which can escape from such local minima by using the chaos neuron is proposed[4], [5].

In this article, the network which can find the optimum solution is constructed by using the hysteresis neuron. In previous combinatorial optimization problem solver, the equilibrium point of the network corresponds to the optimum solution to obtain the solution[6], [7]. Namely, static states are considered as the information. Then we have made a study to remove all the oscillating state and have published this[7]. However, from a biological point of view, the dynamical system has lately attracted attention. Then, in this article, we propose the dynamical combinatorial optimization problem solver using hysteresis neural network. Hysteresis neuron behaves as a relaxation oscillator and cannot generate chaos. However, it is proved that the system, which has large connected hysteresis neurons, generates chaos by proper parameters[8], [9]. In this article, the system behaves as chaos and searches the optimum solution dynamically. Namely, the system does not converge into the optimum solution, but continues to search the solution. At the present time, it has been researched that the real neuron behaves as chaos[10]. It is considered that the proposal system behaves as such real neuron.

2. System

2.1 Hysteresis Neural Network

The objective system is described by the following piecewise linear differential equation.

$$\begin{cases} \dfrac{dx_i(t)}{dt} = -x_i(t) + g_i(y(t)) + \Delta(y_i(t)) \\ y_i(t) = h(x_i(t)) = \begin{cases} +1, \text{for } x_i(t) > 1 \\ -1, \text{for } x_i(t) < -1 \end{cases} \\ \Delta(y_i(t)) = \begin{cases} \Delta_+, \text{for } y_i(t) = 1 \\ \Delta_-, \text{for } y_i(t) = -1 \end{cases} \end{cases} \tag{1}$$

, where $x_i(t)$ is the inner state of i-th neuron, $y_i(t)$ is the output of i-th neuron, $g_i(y(t))$ describes the connection of each neuron. Δ_+, Δ_- is constant. $h(x_i(t))$ is the piecewise linear bipolar hysteresis function as shown in Fig.1.

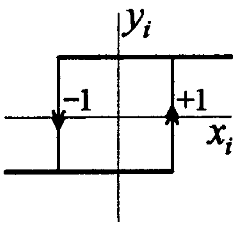


Figure 1. piecewise linear bipolar hysteresis function

The equation 1 is simplified as the following.

$$\begin{cases} \dfrac{dx_i(t)}{dt} = -x_i(t) + p_i(y(t)) \\ y_i = h(x_i(t)) \end{cases} \tag{2}$$

$p_i(y(t))$ describes the equilibrium point of the inner state of each neuron. The output does not change if the equilibrium point is on the hysteresis branch as shown in Fig.2(a). On the contrary, the output will change in the case of Fig.2(b) and continues to oscillate. For dynamical search, the equilibrium points are set as shown in Fig.2(b).

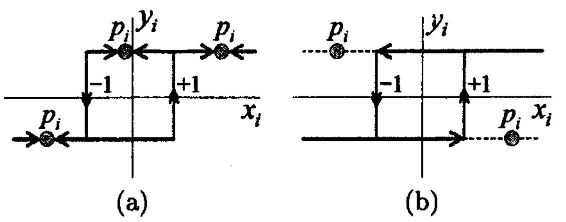


Figure 2. behavior of hysteresis neuron

2.2 Condition of Dynamical Search

From Fig.2(b), if the following condition is satisfied, the inner state does not converge into the equilibrium point, and the output will change.

$$y_i(t)p_i(y(t)) < -1 \tag{3}$$

The equation following is given by equation 1 and 2.

$$p_i(y(t)) = g_i(y(t)) + \Delta(y_i(t)) \tag{4}$$

Then the condition of dynamical search is described as the following equation.

$$\begin{cases} \Delta_+ < -2 \\ \Delta_- > 2 \end{cases} \tag{5}$$

, where $g_i(y(t))$ has the value as the following.

$$-1 < g_i(y(t)) < 1 \tag{6}$$

If this condition is satisfied, the condition 3 is satisfied too and the output continues to oscillate.

2.3 Relation between the Position of Equilibrium Point and Arrival Time at Threshold Point

From equation 2, the further the equilibrium point locates, the larger difference of inner state becomes. Then, in the case of Fig.3(b), inner state will reach threshold point earlier than of (a); shown in Fig.4.

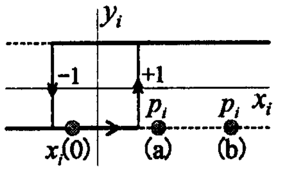


Figure 3. position of equilibrium point

2.4 Apply to Combinatorial Optimization Problem Solver

In the combinatorial optimization problem solver, cost function is defined to evaluate each solution. Cost function becomes minimum(or maximum) value when the solution is optimum. Then if the connection function$g_i(y(t))$ is defined as cost function simultaneously, each neuron's output is controlled by the cost function. Therefore, the network can find the optimum solution.

From equation 1, the equilibrium point is as shown in Fig.5.

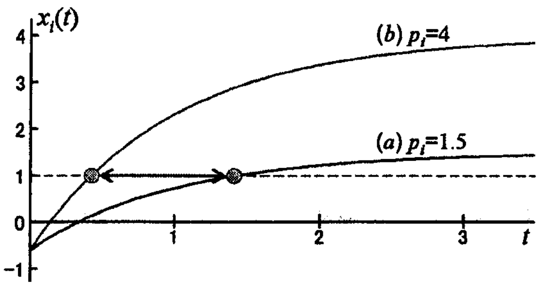
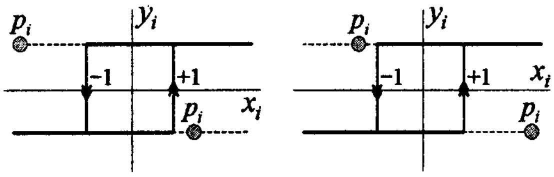


Figure 4. arrival time at threshold point



(a)$g_i(y(t)) < 0$ (b)$g_i(y(t)) > 0$

Figure 5. equilibrium point

From Fig.5(a), in the case of $g_i(y(t)) < 0$, equilibrium point becomes nearer when $y_i(t) = -1$ and becomes further when $y_i(t) = +1$. Then in this case, the inner state should reach threshold point when $y_i(t) = +1$ and should not reach threshold point when $y_i(t) = -1$. Contrary, from Fig.5(b), in the case of $g_i(y(t)) > 0$, the inner state should not reach threshold point when $y_i(t) = +1$ and should reach threshold point when $y_i(t) = -1$. Therefore, the output should become $y_i(t) = -1$ when $g_i(y(t)) < 0$ and should become $y_i(t) = +1$ when $g_i(y(t)) > 0$.

If the connection function $g_i(y(t))$ is set as the following, the output vector keeps the state near the optimum solution, and continues to transit. Namely, the network continues to search the optimum solution.

$$\begin{cases} g_i(y(t))|_{y_i(t)=+1} < g_i(y(t))|_{y_i(t)=-1} < 0 \\ g_i(y(t))|_{y_i(t)=+1} > g_i(y(t))|_{y_i(t)=-1} > 0 \end{cases} \tag{7}$$

3. N-Queens Problem

3.1 N-Queens Problem

Queen is the piece used in chess. Queen moves for vertical, horizontal and diagonal freely. N-queens problem is the problem to assign N queens with no collision in $N \times N$ chess board. One of the optimum solutions of 4 queens problem is as shown in Fig.6.

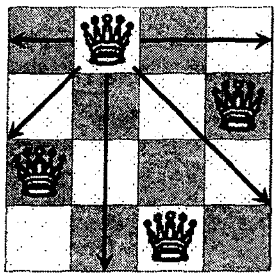


Figure 6. N-queens problem($N = 4$)

Neurons are assigned to 2-dimensions for describing the solution of N-queens problem. Then equation 1 is reformed as the following.

$$\begin{cases} \dfrac{dx_{ij}(t)}{dt} = -x_{ij}(t) + g_{ij}(y(t)) + \Delta(y_{ij}(t)) \\ y_{ij}(t) = h(x_{ij}(t)) = \begin{cases} +1, \text{ for } x_{ij}(t) > 1 \\ -1, \text{ for } x_{ij}(t) < -1 \end{cases} \end{cases} \tag{8}$$

Each neuron corresponds to each square on the chess board. Queen is assigned onto the square which the ignited neuron corresponds to.

3.2 Constraint Condition of N-Queens Problem

The constraint condition of N-queens problem is described as the following.

1. Each row and column has only one queen.
2. Each diagonal line has only one queen or no queen.

The cost functions following denotes the above condition.

$$g_{1ij}(y(t)) = 2 - \sum_{k=1}^{N} \left(\frac{y_{ik}(t)+1}{2} + \frac{y_{kj}(t)+1}{2} \right) \tag{9}$$

$$\begin{cases} g_{2ij}(y(t)) = f\left(0.5 - \displaystyle\sum_{k=1}^{N} \frac{y_{i-j+k,k}(t)+1}{2}\right) \\ \qquad\qquad +f\left(0.5 - \displaystyle\sum_{k=1}^{N} \frac{y_{i+j-k,j}(t)+1}{2}\right) \\ f(x) = \begin{cases} 0, & \text{for } -1 < x < 1 \\ x, & \text{otherwise} \end{cases} \end{cases} \tag{10}$$

The connection function $g_{ij}(y(t))$ is obtained by adding these cost functions.

$$g_{ij}(y(t)) = g_{1ij}(y(t)) + g_{2ij}(y(t)) \tag{11}$$

The connection function $g_{ij}(y(t))$ is satisfied the condition 7. Then the system can search the optimum solution by using this function.

4. Simulation

The simulation to find the optimum solution of N-queens problem for the above system is carried out. The system is evaluated by the following way.

1. How long time does the output vector keep the optimum solution?
2. Does the output vector show all the optimum solutions?

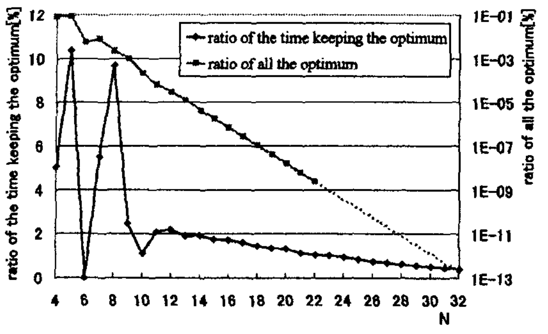


Figure 7. ratio of the time keeping the optimum solution
($\Delta_+ = 0, \Delta_- = 1.000000001$)

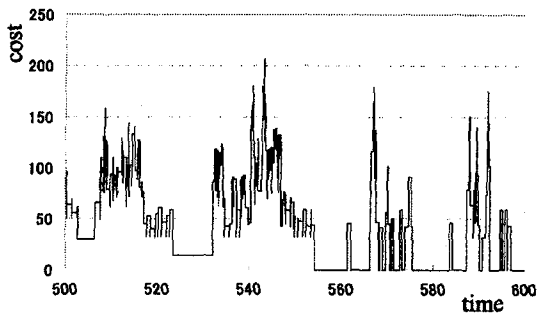


Figure 8. transition of cost function($N = 8$)
($\Delta_+ = 0, \Delta_- = 1.000000001$)

4.1 Ratio of the Time Keeping the Optimum Solution

The time how long does the output vector keep the optimum solution is shown in table 1 and Fig.7. From Fig.7, though ratio of the optimum solution decrease exponentially, the time how long the system keeps the optimum solution hardly decrease. Therefore, it is considered that the system has the ability to search the optimum solution. The transition of cost function is shown in Fig.8. From Fig.8, it is considered that the output changes busily when the cost function has high value, but keep the value comparatively in the case of the low cost.

4.2 Can the System Have All the Optimum Solutions?

It is considered that the proposed system can find all the optimum solutions because the system does not converge into an optimum solution and continue to search other optimum solution. Then it is measured how many kind of optimum solution is obtained by the system. The result is shown in table 2. From table 2, it is confirmed that the

Table 1. ratio of the time keeping the optimum solution
($\Delta_+ = 0, \Delta_- = 1.000000001$)

N	♯of all optima*	♯of solutions	time[%]
4	2	24	5.05
5	10	120	10.38
6	4	720	0.00626
7	40	5.04×10^{03}	5.51
8	92	4.03×10^{04}	9.68
9	352	3.63×10^{05}	2.46
10	724	3.63×10^{06}	1.10
11	2.68×10^{03}	3.99×10^{07}	2.09
12	1.42×10^{04}	4.79×10^{08}	2.19
13	7.37×10^{04}	6.23×10^{09}	1.92
14	3.66×10^{05}	8.72×10^{10}	1.92
15	2.28×10^{06}	1.31×10^{12}	1.77
16	1.48×10^{07}	2.09×10^{13}	1.73
17	9.58×10^{07}	3.56×10^{14}	1.60
18	6.66×10^{08}	6.40×10^{15}	1.44
19	4.97×10^{09}	1.22×10^{17}	1.34
20	3.90×10^{10}	2.43×10^{18}	1.32
21	3.15×10^{11}	5.11×10^{19}	1.13
22	2.69×10^{12}	1.12×10^{21}	1.04

* from Queens Intro(http://queens.x2o.net/)

Table 2. the number of the optimum solutions which the system found
($\Delta_+ = 0, \Delta_- = 1.000000001$)

N	♯of all optima	♯of found	found time
4	2	2	1.28×10^2
5	10	10	1.47×10^3
6	4	4	1.55×10^3
7	40	40	4.04×10^3
8	92	92	1.55×10^4
9	352	352	4.83×10^4
10	724	724	1.88×10^5
11	2680	2680	8.03×10^5

system find all the optimum solutions to $N = 11$. It is considered that the system can find all the optimum solution over $N = 12$.

5. Conclusions

In this article, the dynamical N-queen problem solver is considered. The proposed system is hardly influenced by the exponential increase of the number of combinations. It is considered that the proposed system does not fall into limit cycle and can find all the optimum solutions because the system shows like the chaotic behavior. It is been examined whether the system generates chaos.

References

[1] Modern Heuristics Techniques for Combinatorial Problems: edited by Colin Reeves, Nikkan Kogyo Shimbun., 1997.(Japanese edition)

[2] J.J.Hopfield and D.W.Tank: ""Neural" Computation of Decisions in Optimization Problems," Biol. Cybern., vol.52, pp.141–152, 1985.

[3] D.W.Tank and J.J.Hopfield: "Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," IEEE Trans. CAS, vol.33, pp.533–541, 1986.

[4] L.Chen and K.Aihara: "Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos," Neural Networks, vol.8, no.6, pp.915–930, 1995.

[5] M.Hasegawa, T.Ikeguchi and K.Aihara: "Exponential and Chaotic Neurodynamical Tabu Searches for Quadratic Assignment Problems," Control and Cybernetics, vol29, no.3, pp.773–788, 2000.

[6] S.Bharitkar and J.M.Mendel: "The Hysteretic Hopfield Neural Network," IEEE Transactions on Neural Networks, vol.11, no.4, pp.879–888, 2000.

[7] T.Nakaguchi, K.Jin'no and M.Tanaka: "Hysteresis Neural Networks for N-Queens Problems," IEICE Trans. Fundamentals, vol.E82-A, no.9, pp.1851–1859, 1999.

[8] K.Jin'no: "Chaos and Related Bifurcation from a Simple Hysteresis Network," IEICE Trans. on Fundamentals., vol.E79-A, no.3, pp.402–414, 1996.

[9] K.Jin'no, T.Nakamura and T.Saito: "Chaos and Related Bifurcation from a 3 Cells Hysteresis Neural Network," IEEE Trans., Circuits and Systems, Part I, vol.46, no.7, pp.851–857, 1999.

[10] A.Celletti and A.E.P.Villa: "Low-Dimensional Chaotic Attractors in the Rat Brain," Biol. Cybern., vol.74, pp.387–393, 1996.