数理モデル化と問題解決　42－10
（２００２．　１１．　２８）

ナイトツアー問題における改良型ニューラルネット手法

青葉雅人*，武藤佳恭†

概要：ナイトツアー問題は，ハミルトン回路問題の一種として古くから数学者たちの間で議論されてきた．武藤と Lee はニューラルネットを用いた手法を提案したが，この手法は局所解に陥りやすい欠点を持っている．本論分では武藤モデルを改良した３つの手法を提案した．また，新手法と武藤モデルの比較実験を行い，収束率と計算時間の点で新手法の有効性を示した．

Improved Neural Algorithms for Knight's Tour Problems

Masato Aoba*, Yoshiyasu Takefuji†

Abstract: The knight's tour problem has been well-known to mathematicians because it is considered as a subset of the Hamiltonian circuit problem. Takefuji and Lee have used a neural network to solve knight's tour problem, however their method has a problem that the system often converges to a local minimum. In this paper, we propose new three algorithms to improve Takefuji's algorithm. We compared the proposed algorithms with Takefuji's algorithm. Our results show that the proposed algorithms improve both the convergence rate and the computation time compared with Takefuji's algorithm.

1. Introduction

The knight's tour problem has been well-known to mathmaticians as a subset of the Hamiltonian circuit problems and the procedure to find a tour is known as NP-hard [1].

The knight's tour problem can be represented as follows. A knight has to traverse all of the squares on a chessboard but each square once and only once and return to the originated square. A knight can trace only eight routes as shown in Fig.1 and the figure also shows an example of the tours for the 8x8 chessboard.

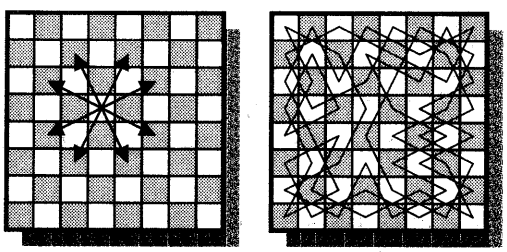

Fig.1. Legal paths of a knight and an example of knight's tour

The first serious attempt to find a knight's tour was made by Euler in 1759 [2], and many mathematicians tried to solve it as a general problem [3] [4]. Takefuji and Lee [5] have used a neural network to find a knight's tour for the general sized chess boards. In this paper, we propose new three algorithms to improve Takefuji's algorithm.

* 慶應義塾大学大学院 政策・メディア研究科
Graduate School of Media and Governance, Keio University
† 慶應義塾大学 環境情報学部
Faculty of Environmental Information, Keio University

2. Takefuji's Algorithm

2.1. Neural Representation and Motion Equations

In their algorithm, the system requires $p(p-1)/2$ hysteresis McCulloch-Pitts neurons where p is the number of squares on $m \times n$ chessboard. The hysteresis McCulloch-Pitts neuron model represents the fired neuron as $V_{ij} = 1$ and the unfired neuron as $V_{ij} = 0$, where V_{ij} is the output of the ij th neuron. The state of V_{ij} is updated by the hysteresis McCulloch-Pitts function; $V_{ij}(t) = 1$ if $U_{ij}(t) > UTP$, 0 if $U_{ij}(t) < LTP$ and unchanged otherwise, where U_{ij} is the input of the ij th neuron and UTP (Upper Trip Point) is always larger than LTP (Lower Trip Point). The ij th neuron is fired if the knight moves from the i th square to the j th square. Only the upper triangular elements in the two-dimensional array are used for the relation $V_{ij} = V_{ji}$.

The motion equation for the ij th neuron is represented as follows;

$$\frac{dU_{ij}}{dt} = \begin{cases} -\left(\sum_{k=1}^{p} V_{ik} d_{ik} - 2\right) - \left(\sum_{k=1}^{p} V_{kj} d_{kj} - 2\right) & \text{if } d_{ij} = 1 \\ 0 & \text{if } d_{ij} = 0 \end{cases} \tag{1}$$

where $d_{ij} = 1$ if the movement from i th square to the j th square is legal, and $d_{ij} = 0$ otherwise. The state of U_{ij} is updated by using the first order Euler's method at each iterative step.

2.2. Problems in Takefuji's Algorithm

In Takefuji's algorithm, the system easily converges to the local minimum whose solution has more than two subtours, especially for a large sized chessboard. In Takefuji's algorithm, once the system converges to the local minimum, the network state has to be initialized and be recalculated.

3. The Proposed Algorithms

3.1. Basic Ideas

The basic ideas are very simple. When the system converges to the local minimum with more than two subtours, the following ideas can be used to solve the problem;

1. cutting subtours,
2. connecting subtours with each other, and
3. cutting and connecting subtours at the same time.

3.2. Motion Equations

At first, the system uses Takefuji's algorithm to find a knight's tour. When the system does not converge to the local minimum with more than two subtours, previous ideas are not employed. If the system converges to the local minimum, one of these ideas is implemented by a new motion equation.

Algorithm 1. Cutting subtours for fired neurons:

$$\frac{dU_{ij}}{dt} = -F\log\left\{\left(\sum_{k=1}^{p} d_{ik} - 2\right)\left(\sum_{k=1}^{p} d_{kj} - 2\right) + 1\right\} \tag{2}$$

Algorithm 2. Connecting subtours for unfired neurons:

$$\frac{dU_{ij}}{dt} = F\log\left\{\left(\sum_{k=1}^{p} d_{ik} - 2\right)\left(\sum_{k=1}^{p} d_{kj} - 2\right) + 1\right\} \tag{3}$$

Algorithm 3. Cutting and connecting subtours:

Eq.(2) for fired neurons and Eq.(3) for unfired neurons.

where F is an arbitrary constant.

4. Experiments

The average number of iterative steps is less than 100 steps in Takefuji's algorithm [16]. Thus we define the upper limit of the number of iterative steps as 100. We tested all of the algorithms until 100 correct solutions were obtained for the following chessboards;6x6, 8x8, 10x10, 12x12, 14x14, 16x16, 18x18, 20x20 and 22x22.

The total number of iterative steps is calculated as a summation of the steps until getting required number of correct results and it includes failure in convergence. Note that the rate of the total number of iterative steps mostly means the rate of the computation time.

Fig.2 and Fig.3 show the experimental results respectively. Fig 2 shows the convergence rate. Fig.3 shows the total number of iterative steps. On the right side of the Fig.2 and Fig.3, the comparison rates with Takefuji's algorithm are plotted. The comparison rate is defined as the result of dividing the value of the proposed algorithm by that of Takefuji's algorithm.

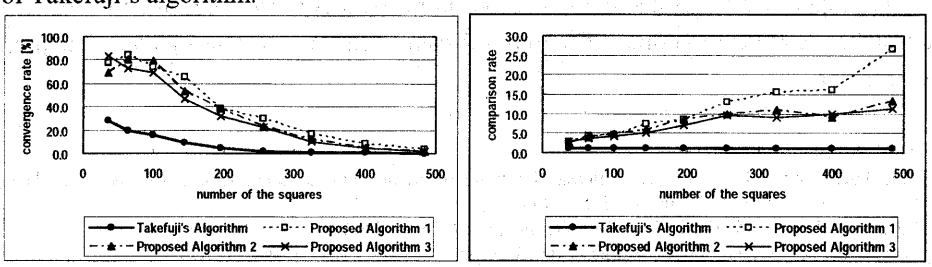

Fig.2. Convergence rate

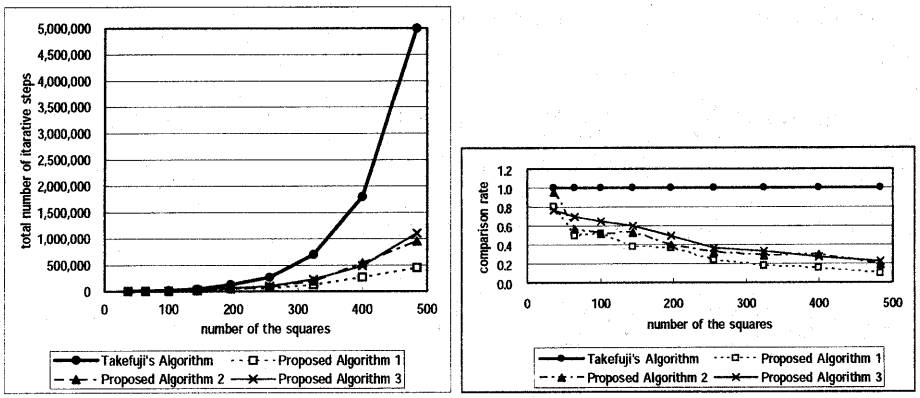

Fig 3. Total number of iterative steps

5. Discussion

Fig.2 and Fig.3 show that the proposed algorithms improve both the convergence rate and the total number of iterative steps especially for large sized chessboards.

In the experimental results, the proposed algorithm 1 is the best method especially when the size of the chessboard is large. We can predict that the proposed algorithm 1 is more efficient when the size of the chessboard becomes larger. We expect the reason for that as follows. When the number of squares is p, the system requires $p(p-1)/2$ neurons. If the system converges to the correct solution or the local minimum, p neurons are fired. In other words, only $2/(p-1)$ of the whole neurons are fired and most of the neurons ($p(p-3)/2$ neurons) are unfired. Thus, in the proposed algorithm 2 and 3, when the system converges to the local minimum, the "connecting subtours" procedure stimulates too many neurons by the motion equations and the state of the network translates further from a solution of the problem than the proposed algorithm 1.

6. Conclusion

Our experimental results show that the proposed algorithms improve the convergence rate and the computation time. The most efficient method is cutting subtours (the proposed algorithm 1).

References

[1] J.A.Shufelt and H.J.Berliner, "Generating Hamiltonian circuits without backtracking from errors", Theoretical Computer Science, Vol.132, pp.347-375, (1994).
[2] L.Euler, "Memoires de Berlin for 1759", Berlin, pp.310-337, (1766).
[3] Vandermonde, "H'Historie de l'Academie des Sciences for 1771", Paris, pp.566-574, (1774).
[4] H.C.Warnsdorff, "Des Rosselsprunges einfachste und allgemeinste Losung", Schmalkalden, (1823).
[5] K.C.Lee and Y.Takefuji, "Finding a knight's tour on an N x M chessboard with O(MN) hysteresis McCulloch-Pitts neurons", IEEE Transaction On System Man and Cybernetics, (1990).