Finding Knight's Tours on an $M \times N$ Chessboard with $O(MN)$ Hysteresis McCulloch–Pitts Neurons

Kuo-Chun Lee and Yoshiyasu Takefuji

Abstract— **How can a knight be moved on a chessboard so that the knight visits each square once and only once and goes back to the starting square? The earliest serious attempt to find a knight's tour on the chessboard was made by L. Euler in 1759 [1]. In this correspondence, a parallel algorithm based on the hysteresis McCulloch–Pitts neurons is proposed to solve the knight's tour problem. The relation between the traveling salesman problem and the knight's tour problem is also discussed. A large number of simulation runs were performed to investigate the behavior of the hysteresis McCulloch–Pitts neural model. The purpose of this correspondence is to present a case study—how to successfully represent the combinatorial optimization problems by means of neural network.**

I. Introduction

A Hamiltonian cycle of a graph G is a cycle that contains every vertex of G. The name Hamiltonian cycle can be regarded as a misnomer, since Hamilton was not the first one to look for cycles which pass through every vertex of a graph [2]. The earliest example of a problem which can be expressed in terms of Hamiltonian cycles is the celebrated knight's tour problem. This problem has interested many great mathematicians: De Moivre, Vandermonde [3], Warnsdorff [4], Pratt [5], Roget, Legendre [6], and De Lavernede [7]. Most of formerly proposed methods are based on either divide-and-conquer or backtracking, especially on the 8×8 chessboard problem. They are all based on sequential computing. No general methods have been given to this problem in the last two centuries [8]. No parallel algorithm has been reported, either. Using the artificial neural network, a parallel algorithm is proposed to solve the knight's tour problem in this correspondence.

Neural network applications may be classified into two types: optimization and associative retrieval/classification [9]. For example, the Hopfield neural network can be used to solve combinatorial optimization problems in which the gradient descent method seeks the local minimum of a given Liapunov energy function E. In general, the Liapunov energy function E is given by constraints and a cost function. The synaptic weights between neurons are determined by the energy function. The mathematical model of the artificial neural networks consists of two important components: neurons and synaptic links. The output signal generated by one neuron propagates to the others through synaptic links. The linear sum of the weighted input signals determines the new state of the neuron. The system dynamics depends on the neuron model as well as the problem representation. With two kinds of problem representations, the practicability of three neuron models, the sigmoid neuron, the McCulloch–Pitts neuron, and the hysteresis McCulloch–Pitts neuron, are tested for solving the knight's tour problem.

II. Neuron Models

Manuscript received October 21, 1990; revised August 22, 1991 and September 5, 1992.

K. Lee is with the R&D Development, Cirrus Logic Inc., Fremont, CA 94538.

Y. Takefuji is with the Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44105.

IEEE Log Number 9206201.

A. Sigmoid Neuron

The neural network for combinatorial optimization problems was first introduced by Hopfield and Tank in 1985 [10]. They used the differentiable and continuous neuron model, sigmoid function. In their paper [10], the symmetric conductance matrix T with zero diagonal elements must be used to guarantee the convergence to the local minimum. For a given energy function E, the motion equation of the ith neuron is given by

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i} \tag{1}$$

where U_i and V_i are the input and the output of the ith neuron and τ is a constant.

The input/output relationship is described by the sigmoid function

$$V_i = g(\lambda U_i) = \frac{1}{2}(1 + \tanh(\lambda U_i)). \tag{2}$$

The parameter λ is a constant gain which damages the steepness of the sigmoid curve. Theorem 1 shows that $-U_i/\tau$ term increases the computational energy E under some conditions when sigmoid neurons are used.

Theorem 1: If the input/output function is continuous and nondecreasing, the U_i/τ term in (1) increases the computational energy E when

$$\left| \sum_i \frac{dV_i}{dt} \frac{U_i}{t} \right| > \left| \sum_i \left(\frac{dV_i}{dU_i} \right) \left(\frac{dU_i}{dt} \right)^2 \right|$$

and if either

$$\left(U_i > 0 \quad \text{and} \quad \frac{dV_i}{dt} < 0 \right)$$

or

$$\left(U_i < 0 \quad \text{and} \quad \frac{dV_i}{dt} > 0 \right)$$

is satisfied.

Proof: See [28]. Theorem 2 states that the computational energy function E monotonically decreases regardless of the condition of the symmetry and diagonal constraints in the conductance matrix as long as the neurons obey a nondecreasing function and the motion equation of the ith neuron is given by

$$\frac{dU_i}{dt} = -\frac{\partial E}{\partial V_i}.$$

Theorem 2: $dE/dt \leq 0$ is satisfied under two conditions such as

$$\frac{dU_i}{dt} = -\frac{\partial E}{\partial V_i}$$

and

$$V_i = f(U_i)$$

where $f(U_i)$ is a nondecreasing function.

Proof:

$$\frac{dE}{dt} = \sum_i \frac{dU_i}{dt} \frac{dV_i}{dU_i} \frac{dE}{dV_i}$$

$$= -\sum_i \left(\frac{dU_i}{dt} \right)^2 \frac{dV_i}{dU_i},$$

$$\text{where } \frac{dE}{dV_i} \text{ is replaced by } -\frac{dU_i}{dt} \text{ (condition 1)}$$

$$\leq 0, \qquad \text{where } \frac{dV_i}{dU_i} > 0 \text{ (condition 2).}$$

Q.E.D

B. McCulloch–Pitts Neuron

McCulloch and Pitts [11] proposed a mathematical model based on the biological computation in 1943. The input/output function is given by $V_i = f(U_i) = 1$ if $U_i \geq 0$ and 0, otherwise where V_i and U_i are the output and the input of the ith neuron, respectively. Takefuji and Lee have successfully used the McCulloch–Pitts neuron and the modified McCulloch–Pitts neuron in neural networks (without the $U - i/\tau$ term in the motion equation) for graph planarization problems [12], tiling problems [13], RNA secondary structure prediction problems [14], [15], maximum independent set problems [15], and sorting problems [16]. The convergence speed of the neural network with McCulloch–Pitts neurons is relatively faster than that of the network with sigmoid neurons. As long as the neural network simulation is performed on the digital computer, Theorem 2 is inadequate to explain the convergence behavior. The reason is that the sigmoid function is no longer continuous on digital computation.

Theorem 3 shows the convergence of the system with the discrete McCulloch–Pitts neurons.

Theorem 3: $\Delta E/\Delta t \leq 0$ is satisfied under two conditions such as

$$\frac{\Delta U_i}{\Delta t} = -\frac{\Delta E}{\Delta V_i}$$

and

$$V_i = f(U_i)$$

where $f(U_i)$ is a binary function: $f(U_i) = 1$ if $U_i \geq 0$, 0 otherwise.

Proof: See [29]. Although the McCulloch–Pitts neuron has the advantage of fast convergence, it causes undesirable oscillatory behavior in neural dynamics. On the contrary, the hysteresis McCulloch–Pitts neurons, discussed below, suppress the oscillatory behavior.

C. Hysteresis McCulloch–Pitts Neuron

In 1986, Hoffman and Benson [17] proposed sigmoid neurons with hysteresis for learning, where any changes in synaptic connection strengths are replaced by hysteresis. Due to the hysteresis property, the system tends to stay in the region of phase space where it is located. They introduced the theory on a role for sleep in learning. In 1985, Segundo and Martinez reported the discovery of dynamic and static hysteresis in Crayfish stretch receptors [18]. They stated that hysteresis might be more widespread than suspected in sensory and perhaps other systems. Yanai and Sawada [19] also presented the associative memory network in which they used the neurons with the hysteresis property. They showed that the recalling ability improves with the hysteretic property. Takefuji and Lee introduced a hysteresis McCulloch–Pitts neuron model [20]. The hysteresis property suppresses the oscillatory behavior of neural dynamics so that the convergence time becomes shorter. The hysteresis McCulloch–Pitts neuron has been successfully used for crossbar switch scheduling [20] and fault-cell allocation [21]. Fig. 1 shows the input/output relation of the hysteresis McCulloch–Pitts neuron model. The output of the ith hysteresis McCulloch–Pitts neuron V_i is given by: $V_i = 1$ if $U_i \geq$ UTP (upper trip point), 0 if $U_i \leq$ LTP (lower trip point), and unchanged otherwise, where U_i is the input of the ith neuron. The output at any moment depends upon the present value of the input as well as the previous values.

Theorem 4 states that a system with hysteresis McCulloch–Pitts neurons can converge to the local minimum.

Theorem 4: $\Delta E/\Delta t \leq 0$ is satisfied under two condition such as

$$\frac{\Delta U_i}{\Delta t} = -\frac{\Delta E}{\Delta V_i}$$

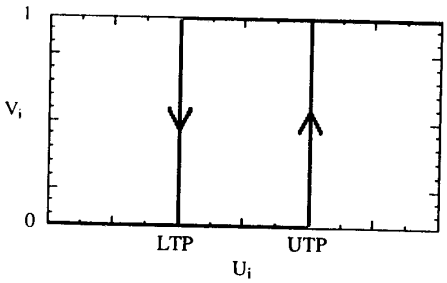


Fig. 1. The input/output relationship of the hysteresis McCulloch–Pitts neuron.

and

$$V_i = f(U_i)$$

where $f(U_i)$ is a hysteresis binary function.

Proof: See [21], [29].

III. Knight 's Tour Problem and City–Order Neural Representation

The neural network for the traveling salesman problem (TSP) has been investigated by Hopfield and Tank. The goal is to find the shortest path where the salesman visits all cities but only once per city and goes back to the starting city, i.e., the shortest Hamiltonian cycle. They formulated the problem using the city-order matrix representation where the row and the column represent the city and the visiting order, respectively. The city–order matrix representation needs N^2 neurons in total where N is the number of cities. Based on the city–order representation, the TSP energy function is as follows:

$$\begin{aligned} E = {} & \frac{A}{2}\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{x,i}V_{x,j} \\ & + \frac{B}{2}\sum_{i=1}^{N}\sum_{x=1}^{N}\sum_{x\neq y}^{N} V_{x,i}V_{y,i} \\ & + \frac{C}{2}\left(\left(\sum_{i=1}^{N}\sum_{x=1}^{N} V_{x,i} - N\right)\right)^2 \\ & + \frac{D}{2}\sum_{x=1}^{N}\sum_{y\neq x}^{N}\sum_{i=1}^{N} d_{x,y}V_{x,i}(V_{y,i+1} + V_{y,i-1}). \end{aligned} \tag{3}$$

The first term and the second term force each row and each column to fire one and only one neuron, respectively. The third term forces the system to fire N neurons in total. As for the last term, it contains the distance information corresponding to a given tour. Neural network investigators have worked on improving the quality and the performance of the Hopfield–Tank neural network through the scaling, normalization, and annealing [22]–[24]. They proposed techniques to alleviate the difficulties for solving large-size problems. However, the solution quality degrades with the problem size rapidly. The state of the global minimum energy of TSP gives the best solution, but no one knows the global minimum unless an exhaustive search is used. If the global minimum energy of the problem is known as a constant, it has no difficulties to test whether the state of the system is in the global minimum or not. Before we solve large-size TSP, it is better to start with the problems which are similar to TSP but with less complexity. The knight's tour problem can be considered as one of the best examples where the global minimum is known.

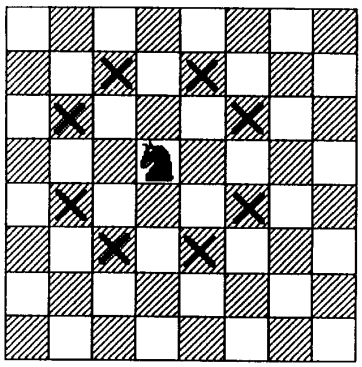


Fig. 2. The legal moves of the knight.

The knight's tour problem is to find a Hamiltonian cycle on a chessboard. On the chessboard, the knight moves in an L shape as shown in Fig. 2. The knight's tour problem is mapped into TSP. There are 8×8 squares on the chessboard. According to the matrix representation proposed by Hopfield and Tank, the knight's tour problem on the 8×8 cheeseboard becomes 64-city TSP. Considering the energy function for this problem in (3), the only difference is the distance between cities. In the knight's tour problem, the distance between two squares is either 0 for the legal move or a large positive number for the illegal move. The legal moves between squares provide the connectivity information. Note that the global minimum of the computational energy of the problem is zero. If the Hamiltonian cycle is found, the state of the system reaches the global minimum of the computational energy.

In order to test the neural network based on the city-order representation, the first three terms were modified in the energy function [25]. The first three terms indicate the constraints for firing only one neuron in each column and row, respectively, and that N neurons are fired in total. The new energy function is

$$\begin{aligned} E = & \frac{A}{2} \sum_{x=1}^{N} \left(\sum_{i=1}^{N} V_{x,i} - 1 \right)^2 \\ & + \frac{A}{2} \sum_{i=1}^{N} \left(\sum_{x=1}^{N} V_{x,i} - 1 \right)^2 \\ & + \frac{B}{2} \sum_{x=1}^{N} \sum_{y \neq x}^{N} \sum_{i=1}^{N} d_{x,y} V_{x,i} (V_{y,i+1} + V_{y,i-1}). \end{aligned} \tag{4}$$

We first experimented a sigmoid neuron as a processing element. The motion equation, time evolution of the system, is given by

$$\begin{aligned} \frac{dU_{x,i}}{dt} = & - A \left(\sum_{j=1}^{N} V_{x,j} - 1 \right) \\ & - A \left(\sum_{j=1}^{N} V_{j,i} - 1 \right) \\ & - B \sum_{y=1}^{N} d_{x,y} (V_{y,i+1} + V_{y,i-1}). \end{aligned} \tag{5}$$

The goal of the knight's problem is to find one of the global minima instead of the local minimum. In TSP, the solutions are examined by checking whether a Hamiltonian cycle is formed whenever the state of the system converges. For the knight's tour problem, the system convergence is defined that $dU_{x,i}/dt$ is 0 for all of the neurons when the system dynamics follows (5). If the $-U_i/\tau$ term is involved in the motion equation, the system convergence cannot be mathematically well defined. It is not hard to find a valid tour, i.e., a Hamiltonian cycle, when coefficient A is larger than B. The quality of the solution depends on how to tune the coefficient parameters. We tested 6×6 and 8×8 chessboard problems where 1000 runs were performed for each case and the parameters A, B, and λ were chosen in a variety of ranges. After the experiments, the city-order neural representation could not find any single solution for the knight's tour problem. In other words, to find one of global minima is a difficult task although the global minimum of the knight's tour problem is known to be zero. We also simulated the behavior of McCulloch–Pitts neural networks with and without hysteresis for the knight's tour problem. We still could not find any solution. It can be concluded that the neural network based on the city–order representation could hardly find any solution.

IV. City– City Representation

The city–order matrix representation fails to find any solution for the knight's tour problem. This does not imply that neural networks always fail. The failures in the city–order neural network experiment have motivated us to find a new neural representation for the problem. A two-dimensional triangular neural network representation is introduced. This new representation is constructed for city–city relations. The city–city representation has been successfully used by Xu and Tsai [26] to solve TSP. In the city–city representation, a $P \times P$ matrix is used for a P-city problem and each element $a_{i,j}$ represents the link between the ith city and the jth city. Fig. 3 shows a city–city representation for a 6-city traveling salesman problem where the black square denotes the link exists. Because the matrix is symmetrical, only $P(P-1)/2$ elements are needed. A 4×4 chessboard example is given as shown in Fig. 4. Each square is assigned a number associated with its coordinate. For instance, the number for the square (3, 2) is 10. The possible legal moves starting from the square with the coordinate (a, b) should be one of the squares with the coordinate $(a-2, b+1), (a-1, b+2), (a+1, b+2), (a+2, b+1), (a+2, b-1), (a+1, b-2), (a-1, b-2), (a-2, b-1)$, if they exist, as shown in Fig. 5. Based on the legal move for the knight, the $d_{i,j}$ matrix is determined. $d_{i,j}$ is 1 if the move from the ith square to the jth square is legal, 0 otherwise. The system consists of $P(P-1)/2$ processing elements or neurons where P is the number of squares in an $M \times N$ chessboard. Based on the city–city representation, the following motion equation for the neuron which represents the move from the ith square to the jth square is used to solve the problem:

$$\begin{aligned} \text{if} \quad d_{i,j} = 1, \quad \text{then} \quad \frac{dU_{i,j}}{dt} = & - \left(\sum_{k=1}^{P} V_{i,k} d_{i,k} - 2 \right) \\ & - \left(\sum_{k=1}^{P} V_{k,j} d_{k,j} - 2 \right) \\ \text{if} \quad d_{i,j} = 0, \quad \text{then} \quad \frac{dU_{i,j}}{dt} = & \ 0. \end{aligned} \tag{6}$$

The upper triangular elements in the two-dimensional array are used: $V_{i,j}$ for $i < j$ and $V_{i,j}$ for $i > j$ is given by $V_{j,i}$. The state of $V_{i,j}$ actually represents a path between the ith and jth square. In other words, the upper triangular neural array represents the nondirected adjacency matrix to find a Hamiltonian cycle. The first and the second term describe that there exist two legal moves for the ith square and the jth square, respectively. According to the (6), it needs no more than $8 \times P$ neurons instead of $P(P-1)/2$ neurons because each square has eight possible legal moves at most.

The system convergence is defined that $dU_{i,j}/dt$ is 0 for all neurons. The following procedure describes the proposed parallel

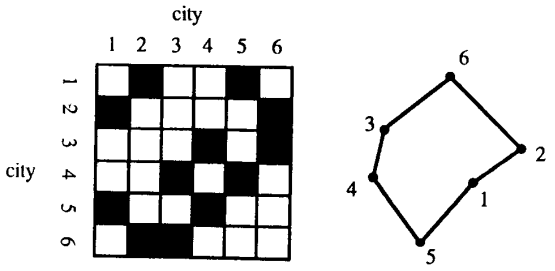


Fig. 3. The city–city representation.

	1	2	3	4
1	1,1	1,2	1,3	1,4
2	2,1	2,2	2,3	2,4
3	3,1	3,2	3,3	3,4
4	4,1	4,2	4,3	4,4

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16

Fig. 4. A 4 × 4 chessboard and the number associated with the respective square.

	a-2,b-1		a-2,b+1	
a-1,b-2				a-1,b+2
		a,b		
a+1,b-2				a+1,b+2
	a+2,b-1		a+2,b+1	

Fig. 5. The eight legal moves from the square with the coordinate (a, b).

algorithm based on the first-order Euler method with hysteresis McCulloch–Pitts neural model.

0 Set $t = 0$.

1. The small negative number is assigned to the initial values of $U_{i,j}(t)$ for $i = 1$ to $P - 1$ and $j = i + 1$ to P.
2. Evaluate values of $V_{i,j}(t)$ based on the hysteresis binary function for $i = 1$ to $P - 1$ and $j = i + 1$ to P.
   $V_{i,j}(t) = 1$, if $U_{i,j}(t) \geq$ UTP
   $V_{i,j}(t) = 0$, if $U_{i,j}(t) \leq$ LTP
   unchanged otherwise.
3. Use the motion equation in (6) to compute $\Delta U_{i,j}(t)$ if $d_{i,j} = 1$ then

$$\Delta U_{i,j}(t) = -\left(\sum_{k=1}^{P} V_{i,k}(t)\, d_{i,k} - 2\right) - \left(\sum_{k=1}^{P} V_{k,j}(t)\, d_{k,j} - 2\right)$$

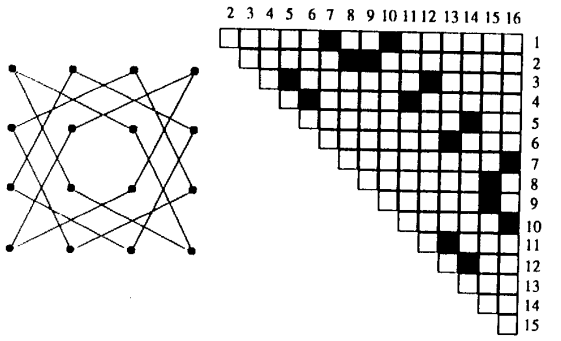


Fig. 6. The 4×4 chessboard with local loops and its final state of the neurons.

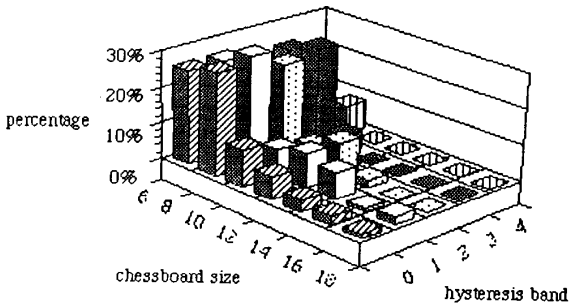


Fig. 7. The relationship among the hysteresis band, the chessboard size, and the percentage to find a valid knight's tour.

if $d_{i,j} = 0$ then $\Delta U_{i,j}(t) = 0$.

4. Compute $U_{i,j}(t)$ based on the first-order Euler method: $U_{i,j}(t+1) = U_{i,j}(t) + \Delta U_{i,j}(t)$ for $i = 1$ to $P - 1$ and $j = i + 1$ to P.
5. Increment t by 1.
6. If $\Delta U_{i,j}(t) = 0$ for $i = 1$ to $P - 1$ and $j = i + 1$ to P then stop this procedure else go to step 2.

There are two kinds of possible solutions when the city–city neural representation is used. One is the valid knight's tour and the other is the local-loop solution. The solution with local loops is unsatisfactory. Fig. 6 shows an example with local loops on the 4 × 4 chessboard and its final state of neurons. The original city–order representation is able to avoid such local loops when the state of the system converges to a solution. However, the city–order representation has never given any successful solution. We have tested the knight's tour problem for $6 \times 6, 8 \times 8, 10 \times 10, 12 \times 12, 14 \times 14, 16 \times 16, 18 \times 18$, and 20×20 chessboards using the city–city neural representation. The simulator was developed in C on DEC 3100 and in Turbo Pascal on Macintosh SE/30 based on the proposed procedure to verify our algorithm. Several hysteresis bands were used and tested. Fig. 7 shows the relationship among the hysteresis band, the chessboard size, and the percentage to find a valid knight's tour solution. The knight's tour is shown in Fig. 8 from 6 × 6 to 14 × 14 chessboard. Fig. 9 shows three solutions for different rectangle chessboards. The required convergence time is no more than 30 iteration steps in our experiments.

Our simulation results show that the different neural respresentations, and neuron models provide the different behaviors. In [10], for the 10-city TSP, the percentage to find a valid tour is about 32%. For the 40-city TSP, the percentage drops to less than 1%. For the 8 × 8 knight's tour problem, it is equivalent

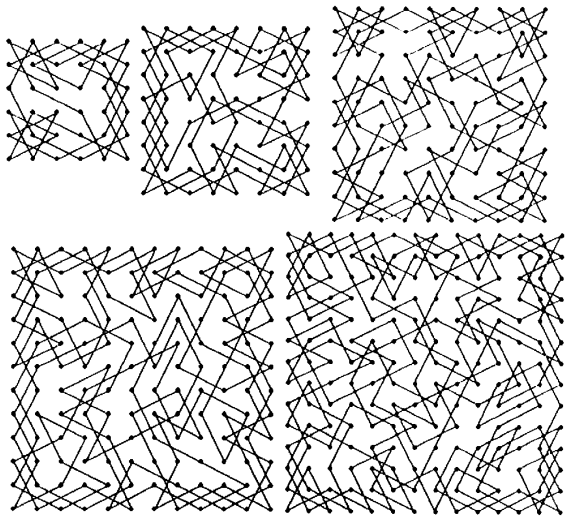


Fig. 8. Knight's tours solutions from 6×6 to 14×14 chessboard.

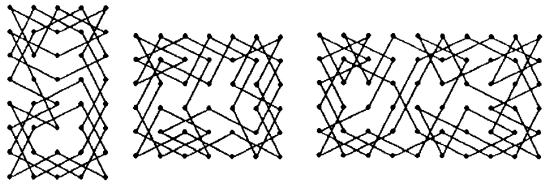


Fig. 9. The knight's tours for three different chessboards.

to the 64-city TSP in terms of the complexity if the city–order representation is used. Using the city–order representation, we have never found any solution to the knight's tour problem. If we use the city–city neural representation, for the 8×8 knight's problem the percentage to find a valid solution is about 30% as shown in Fig. 10. The percentage drops to around 1% for the 20×20 knight's tour problem. (It is equivalent to 400 cities TSP in terms of the complexity.) Note that the percentage number [10] indicates the convergence frequency to the local minimum while our percentage number indicates the convergence frequency to the global minimum. The quality of the solution degrades when the problem size increases in TSP and the knight's tour problems. However, the solution quality in TSP degrades more rapidly than that in the knight's tour problem. Remember that in the knight's tour problem we must find the global minimum instead of the local minimum. Meanwhile, we did not put any effort to tune the parameters in our simulations. We can conclude that the city–city neural representation is better than the city–order neural representation for the knight's tour problem.

V. How to Deal with the Large-Size Knight 's Tour Problem

As the size of the chessboard increases, it becomes very difficult to find a Hamiltonian cycle. However, it is possible to have another alternative to construct a large-size solution based on the small-size solutions. The small-size solutions become cells to form a large cell. For instance, by using knight's tour solutions obtained from the parallel algorithm for the 8×8 chessboard problem, it is possible to construct a solution on an $8N \times 8N$ chessboard where N is any positive integer. Fig. 10 shows a knight's solution on a 40×40 chessboard by the expansion of 25 pieces of the 8×8 knight's tour solutions.

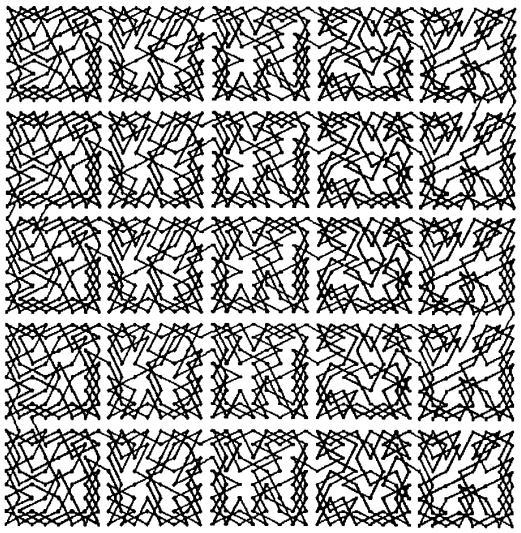
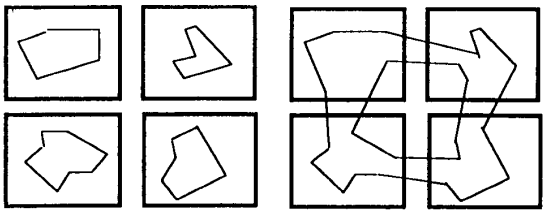


Fig. 10. A knight's tour solution on a 40×40 chessboard.



(a) (b)

Fig. 11. Invalid solution after merging. (a) Before merging. (b) After merging.

When small-size cells are merged together, they are likely to form an invalid knight's tour which is not a Hamiltonian cycle. An example is given to explain the invalid case where four small cells are merged together to form a solution in Fig. 11. Fig. 11(a), each rectangle represents a knight's tour solution for the small-size chessboard. As one cell is merged with the other, both have to open the closed cycle in order to connect themselves. Fig. 11(b) shows the invalid solution after merging. In order to find a solution on the large-size chessboard, we need to provide a proper procedure to guarantee a Hamiltonian cycle. A simple procedure is to cascade small-size solutions one by one in one direction as shown in Fig. 12. This method can guarantee that the final solution is always a valid one. By this procedure, a large-size solution can by easily obtained. The solution in Fig. 10 is a typical example.

The question arises: how to find the cascading paths between the cells? The answer is very simple. If the knight wants to find a closed loop in four moves, the shape of the result must be a parallelogram, as shown in Fig. 13. Parallelograms play a key role in obtaining the cascading paths. To cascade cell A and cell B together successfully, as shown in Fig. 14, we have to find a parallelogram between the right side of cell A and the left side of cell B. In Fig. 14(a), the bold lines indicate a possible parallelogram. Disconnect two bold lines and add the other side lines in this parallelogram. Then cell A and cell B are cascaded successfully. The final result is shown in Fig. 14(b). As long as parallelograms exist between the cells, the cells can be cascaded successfully. For instance, there exists only one path when the knight moves into any corner of the chessboard. This means that

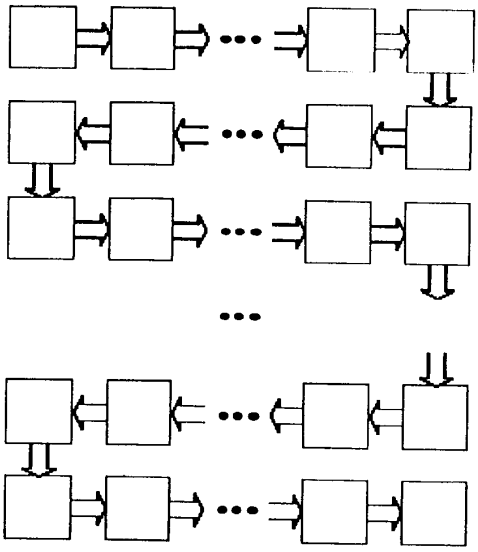


Fig. 12. The cascading method to form a large-size solution in one direction.

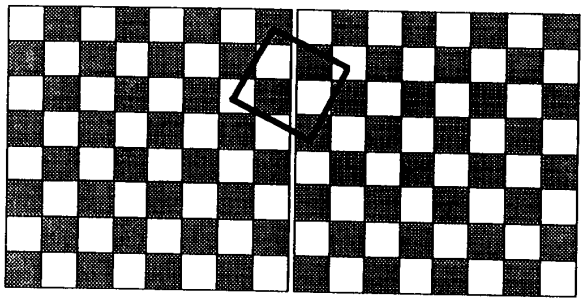


Fig. 13. A parallelogram formed with four knight's path.

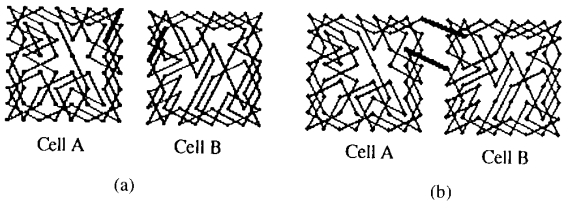


Fig. 14. Cell A and cell B are cascaded sucessfully. (a) Two bold lines on both cells indicate parallelogram. (b) The final result after cascading.

there always exists one side of a possible parallelogram. If the other cell has a path which can form a parallelogram shown in Fig. 13, the cells are cascadable. It is concluded that the $1\,000\,000 \times 1\,000\,000$ chessboard problem or larger problems were solved by our algorithm in several minutes on a Macintosh SE/30.

VI. Conclusion

Hopfield and Tank [10] presented very good examples to show how to map the problem onto neural networks. Wilson and Pawley [27] tried to improve the algorithm but failed. Their failure was caused not by the method of neural computation but by the neural representation and the neuron model. In other words, in order to obtain better solutions we must pay attention to mapping of the problem onto a proper neural network. The hysteresis neurons and city–city neural representation provided better result than the city–order neural representation for the knight's tour problem. The cascading method also gives hope for the solution of the large-size knight's tour problems. Although the cascading method is not a general approach for any large-size problems, the decomposition methodology should play a key role in solving large-size ones. Using the artificial neural network and cascading approach, we can provide the solution for the knight's tour problem which has been investigated for two centuries but no general algorithms have been provided.

Acknowledgment

The authors wish to thank the anonymous referees for their valuable suggestions and comments.

References

[1] L. Euler, in *Memoire de Berlin for 1759,* Berlin, Germany, pp. 310–337, 1766.
[2] R. J. Wilson and J. J. Watkins, *Graphs: An Introductory Approach.* New York: Wiley, 1990, pp. 144–145.
[3] Vandermonde, in *L'Historie de l'Academie des Sciences for 1771.* Paris, France, pp. 566–574, 1774.
[4] H. C. Warnsdorff, *Des Rosselsprunges einfachste und allgemeinste Losung,* Schmalkalden, 1823.
[5] Pratt, in *Studies of Chess,* 6th ed. London, England, 1825.
[6] Legendre, *Theorie des Nombres,* 2nd Ed., vol. 2. Paris, France: 1830, p. 165.
[7] De Lavernede, *Memories de l'Academie Royale du Gard.* Nimes, France: 1839, pp. 151–179.
[8] W. W. Rouse Ball and H. S. M. Coxeter, *Mathematical Recreations & Essays,* 1st ed. 1892; 12th ed. 1974. Toronto, Ont., Canada: Univ. Toronto Press.
[9] S. Y. Kung and J. N. Hwang, "Neural network architectures for robotic applications," *IEEE Trans. Robotics Automat.,* vol. 5, no. 5, pp. 641–657, Oct. 1989.
[10] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.,* vol. 52, pp. 141–152, 1985.
[11] W. S. McCulloch and W. H. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bull. Math. Biophys.,* vol. 5, p. 115, 1943.
[12] Y. Takefuji and K. C. Lee, "A near-optimum parallel planarization algorithm," *Science,* vol. 245, pp. 1221–1223, Sept. 1989.
[13] ——, "A parallel algorithm for tiling problems," *IEEE Trans. Neural Networks,* vol. 1, no. 1, pp. 143–145, Mar. 1990.
[14] Y. Takefuji, C. W. Lin, and K. C. Lee, "A parallel algorithm for estimating the secondary structure in ribonucleic acids," *Biol. Cybern.,* vol. 63, no. 3, 1990.
[15] Y. Takefuji, L. Chen, K. C. Lee, and J. Huffman, "Parallel algorithm for finding a near-maximum independent set of a circle graph," *IEEE Trans. Neural Networks,* vol. 1, no. 3, Sept. 1990.
[16] Y. Takefuji and K. C. Lee, "A super parallel sorting algorithm based on neural networks," *IEEE Trans. Circuits Syst.,* vol. 37, no. 11, 1990.
[17] G. W. Hoffman and M. W. Benson, "Neurons with hysteresis form a network that can learn without any changes in synaptic connection strengths," in *Proc. AIP Conf. on Neural Networks for Computing,* J. S. Denker, Ed., AIP, 1986.
[18] J. P. Segundo and O. D. Martinez, "Dynamic and static hysteresis in crayfish stretch receptors," *Biol. Cybern.,* vol. 52, pp. 291–296, 1985.
[19] H. Yanai and Y. Sawada, "Associate memory network composed of neurons with hysteretic property," *Neural Networks,* vol. 3, pp. 223–228, 1990.
[20] Y. Takefuji and K. C. Lee, "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics," *Biol. Cybern.,* vol. 64, pp. 353–356, 1991.
[21] N. Funabiki and Y. Takefuji, "A parallel algorithm for spare allocation problems," *IEEE Trans. Reliab.,* vol. 40, no. 3, pp. 338–346, 1991.
[22] R. Cuykendall and R. Reese, "Scaling the neural TSP algorithm," *Biol. Cybern.,* vol. 60, pp. 365–371, 1989.
[23] D. E. Van den Bout and T. K. Miller III, "Improving the performance of the Hopfield-Tank neural network through normalization and annealing," *Biol. Cybern.,* vol. 62, pp. 129–139, 1989.
[24] B. Kamgar-Parsi and B. Kamgar-Parsi, "On problem solving with Hopfield neural networks," *Biol. Cybern.,* vol. 62, pp. 415–423, 1990.

[25] H. Szu, "Fast TSP algorithm based on binary neuron output and analog input using the zero-diagonal interconnect matrix and necessary and sufficient constraints of the permutation matrix," in *Proc. IEEE Int. Conf. on Neural Networks,* vol. II, 1988, pp. 59–266.
[26] X. Xu and W. T. Tsai, "Effective neural algorithms for the traveling salesman problem," *Neural Networks,* vol. 4, pp. 193–205, 1991.
[27] G. Wilson and G. Pawley, "On the stability of the traveling salesman algorithm of Hopfield and Tank," *Biol. Cybern.,* vol. 58, pp. 63–70, 1988.
[28] Y. Takefuji and K. C. Lee, "Artificial neural networks for four-coloring map problems and k-colorability problems," *IEEE Trans. Circuits Syst.,* vol. 38, no. 3, pp. 326–333, 1991.
[29] K. C. Lee, Y. B. Cho, Y. Takefuji, and T. Kurokawa, "Convergence theories of sigmoid, McCulloch-Pitts, and hysteresis McCulloch-Pitts neurons," CAISR Tech. Rep. TR90-110.

Efficient Dynamic Simulation of Multiple Manipulator Systems with Singular Configurations

Scott McMillan, P. Sadayappan, and David E. Orin

Abstract—**The paper presents an efficient algorithm for the simulation of a system of m manipulators each having N degrees of freedom that are grasping a common object. Algorithms for such a system have been previously developed by others. In [1], an $O(mN)$ algorithm is presented that does not fully consider the case when one or more of the manipulators are in singular configurations. However, it is stated in [2] that the algorithm has an $O(mN)+O(m^3)$ computational complexity when one or more of the chains are singular. This results because the size of the system of equations to be solved grows linearly with the number of chains in the system [3]. The algorithm presented in this paper significantly reduces the size of the system of equations to be solved to one that grows linearly with the number of singular chains, s, and achieves an $O(mN)+O(s^3)$ complexity. In addition to this result, efficient $O(mN)$ algorithms are also presented for special cases where only one or two chains are in singular configurations. These are particularly useful because it is common to deal with systems consisting of only a few manipulators grasping a common object, and even with more manipulators, it is unlikely that many of them will be singular simultaneously. Finally, by applying the algorithm developed for the case of two singularities to a dual-arm system, an algorithm results that requires fewer computations than that of existing methods, and has the added benefit of being robust in the presence of singular manipulators.**

I. Introduction

In recent years there has been increasing interest in the development of efficient algorithms for the computation of robot dynamics in an effort to achieve real-time computational rates. The desirability of real-time dynamic simulation has been shown in a number of applications such as earth-based teleoperation of remote robotic systems in space [4]. Beyond this goal, *super*-real-time simulation is also desired in some applications such as in advanced control schemes where trajectory planning is used, and seconds of motion trajectory need to be simulated in milliseconds [5]. This is useful in multilegged vehicles [6] when prediction of the action of the present control is used to ensure safety and stability along a desired trajectory.

The major obstacle to achieving the required computational rates, however, is the complexity of the dynamic equations to be simulated. The problem is compounded by increases in the structural and task complexity found in the systems now being considered. These systems may have multiple chains and redundant numbers of degrees of freedom, operating at higher speeds, with the topological structure changing at real-time rates.

To simulate such systems, algorithms for the computation of the open-chain dynamics of each individual chain must be used in addition to equations that describe the interaction between the chains (the closed-chain dynamics). Previous work has been done to develop efficient algorithms and parallel implementations of the open-chain dynamics to reduce its computation time [7]–[11]. In comparison to this body of work, however, much less has been completed to develop efficient dynamics algorithms for multiple closed-chain systems, and still fewer discuss how to handle the cases when one or more of the chains are in singular configurations.

Two papers that have presented algorithms for the computation of multiple-chain dynamics are those by Lilly and Orin [1] and Rodriguez, Jain, and Kreutz-Delgado [2]. Both develop sequential algorithms that have a computational complexity of $O(mN)$ where m is the number of manipulator chains in the system and N is the number of degrees of freedom per chain. While Lilly and Orin discuss the complexity for a parallel implementation, they do not fully consider the case where chains may be in singular configurations. Rodriguez, Jain, and Kreutz-Delgado, on the other hand, go on to state that the computational complexity of the algorithm becomes $O(mN)+O(m^3)$ in the presence of singularities.

The goal of this paper is to develop more efficient algorithms for the simulation of multiple closed-chain manipulators grasping a common object that specifically handle the cases when one or more of the manipulators are in singular configurations. In particular, the algorithm presented in this paper has a computational complexity of $O(mN)+O(s^3)$ where s is the number of chains in singular configurations. This represents a significant improvement over previous results. In addition to this result, efficient $O(mN)$ algorithms are also presented for the cases where only one or two chains are in singular configurations. These are important since it is uncommon, if not undesirable, to deal with systems where many of the manipulators grasping the common object are singular simultaneously. Finally, by applying the algorithm developed for the case of two singularities to a dual-arm system, the resulting algorithm requires fewer total floating point operations than that of the method presented in [1], with the added benefit of being robust in the presence of singular manipulators.

In the next section, the dynamic equations and notation used in the algorithms for simulating systems of multiple manipulators are presented. Particular attention is given to the computational problems introduced when manipulators are in singular configurations. In the third section, an efficient algorithm is developed giving special attention to the special cases where only one or two manipulators are in singular configurations. In the section following, the computational complexity for this algorithm is discussed. An example of special interest in many robotics applications is discussed in the fifth section where the computational requirements for the dual-arm system are examined. Finally, an appendix is included that presents a mathematical analysis that characterizes the singu-

Manuscript received April 20, 1992; revised December 16, 1992. This work was supported in part by a DuPont Fellowship, an AT&T Ph.D. Scholarship, and a grant from CRAY Research, Inc.

S. McMillan and D. E. Orin are with the Department of Electrical Engineering, Ohio State University, Columbus, OH 43210.
P. Sadayappan is with the Department of Computer and Information Science, Ohio State University, Columbus, OH 43210.

IEEE Log Number 9209649.