Cogn Neurodyn (2007) 1:189–202
DOI 10.1007/s11571-006-9009-2

RESEARCH ARTICLE

Numerically evaluated functional equivalence between chaotic dynamics in neural networks and cellular automata under totalistic rules

Ryu Takada · Daigo Munetaka · Shoji Kobayashi · Yoshikazu Suemitsu · Shigetoshi Nara

Received: 14 June 2006 / Accepted: 25 September 2006 / Published online: 7 December 2006

Abstract Chaotic dynamics in a recurrent neural network model and in two-dimensional cellular automata, where both have finite but large degrees of freedom, are investigated from the viewpoint of harnessing chaos and are applied to motion control to indicate that both have potential capabilities for complex function control by simple rule(s). An important point is that chaotic dynamics generated in these two systems give us autonomous complex pattern dynamics itinerating through intermediate state points between embedded patterns (attractors) in high-dimensional state space. An application of these chaotic dynamics to complex controlling is proposed based on an idea that with the use of simple adaptive switching between a weakly chaotic regime and a strongly chaotic regime, complex problems can be solved. As an actual example, a two-dimensional maze, where it should be noted that the spatial structure of the maze is one of typical ill-posed problems, is solved with the use of chaos in both systems. Our computer simulations show that the success rate over 300 trials is much better, at least, than that of a random number generator. Our functional simulations indicate that both systems are almost equivalent from the viewpoint of functional aspects based on our idea, harnessing of chaos.

Keywords Chaotic dynamics · Recurrent neural network · Cellular automata · Information processing · Complex control · Adaptive function

R. Takada · D. Munetaka
Department of Electronic & Information System Engineering, The Graduate School of Natural Science & Technology, Okayama University, 700-8530 Okayama, Japan

S. Kobayashi
Department of Electrical & Electronic Engineering, Faculty of Engineering, Okayama University, 700-8530 Okayama, Japan

Y. Suemitsu
Kyoto School of Computer Science, Nishikujyo Minami-ku, Kyoto 601-8407, Japan

S. Nara (✉)
Department of Electrical & Electronic Engineering, The Graduate School of Natural Science & Technology, Okayama University, 700-8530 Okayama, Japan
e-mail: nara@elec.okayama-u.ac.jp

Introduction

For the last several decades, biological systems have been extensively investigated because of their excellent functions that work in various environments. Neural networks is one of these that has been developed both experimentally and theoretically. The collection of papers in Anderson and Rosenfeld (1988, 1990) is a recognition of such historical works. Among these developments, the discovery of chaos has had a great impact not only on biology, but also on engineering, brain science, and other fields of science. Now, much interest is focused on how chaotic dynamics is related to excellent information processing or control functioning realized in biological systems, for example, in the brain (Skarda and Freeman 1987; Kay et al. 1996). There has been a considerable number of works about brain functioning from the viewpoint of complex nonlinear dynamical systems. In several pioneering works (Tsuda 2001; Kaneko and Tsuda 2001; Tokuda et al. 1997; Fujii et al. 1996), coupled map systems are models proposed to describe these systems related to the functional roles of chaos. Examples include systems composed of binary map elements, cellular automata (Aizawa and Nishikawa 1986; Kim and Aizawa 2000), or binary neuron networks (Davis and Nara 1990; Nara and Davis 1992; Nara et al. 1995). Also, a globally coupled map lattice (GCM) was proposed by Kaneko (1990), and a chaotic neural network (CNN) was put forward by Aihara et al. (1990). Generally speaking, it is quite plausible that GCM or CCN generates high-dimensional chaos, because a single element itself includes chaotic mapping. However, it is not clear whether a coupled binary map can provide chaos. Therefore, it is surprising that Wolfram claimed to have found chaotic dynamics in cellular automata (Wolfram 1986). It was found, however, by Nara and Davis in a recurrent neural network based on a binary neuron model also (Nara and Davis 1992).

We support the viewpoint that chaotic dynamics are potentially useful in complex functioning and/or controlling in systems with the large but finite degrees of freedom typically observed in biological systems. For instance, Nara and Davis have been studying the functional role of chaotic dynamics in a recurrent neural network of binary neurons (abbreviated as NN hereafter) (Davis and Nara 1990; Nara and Davis 1992; Nara et al. 1995; Mikami and Nara 2003; Nara 2003; Suemitsu and Nara 2004). They have also shown, in their series of papers, that complex dynamics generated by Cellular Automata (abbreviated as CA hereafter) could be useful in the sense that CA can describe or reproduce arbitrarily given or observed complex dynamics with use of the rule dynamics of CA. In fact, they provided two practical examples. First, digital sound signals (music and/or spoken words) can be completely reproduced by a one-dimensional two-state three-neighbors cellular automata (1-2-3 CA) (Nara et al. 1999; Wada et al. 2002; Tamura et al. 2003). Second, two-dimensional motion pictures can also be completely reproduced by two-dimensional two-state cellular automata with totalistic rules assigned to each pixel (2-2-T CA) (Tamura et al. 2003; Miura et al. 2005). In these papers, it is shown that arbitrarily given pattern dynamics in one- and two-dimensional systems with a large number of binary cells can be completely reproduced by extracting appropriate rule(s). These are typical examples of inverse problems.

Noise robustness investigated by Nara et al. indicated that dynamics starting from initial states with including of noise gave the three cases. The first is the converged case into the originally given pattern dynamics (attractor regime) (Kuroiwa et al. 2005), the second is the itinerant chaotic dynamics that are not so far from the original pattern dynamics (weakly chaotic regime), and the third is highly developed chaotic dynamics (strongly chaotic regime) (Tamura et al. 2003). These would surely be related to complex dynamics occurring in a recurrent neural network model extensively investigated by one of the authors in Mikami and Nara (2003), Nara (2003), and Suemitsu and Nara (2004). Such attractor and/or chaotic dynamics in NN and CA give us the complex pattern dynamics sampled from intermediate state points between embedded patterns in the high-dimensional state space. A new idea arises from this: an application for complex control could be possible through the use of these dynamics, in particular, to solve ill-posed problems.

In this paper, we focus on a heuristic application of chaotic dynamics generated by NN and CA for motion control and report the results of our functional simulation with the use of adaptive dynamics of NN and CA. We pay particular attention to a specified motion control under an appropriate coding of complex dynamics in high-dimensional state space into motions in two-dimensional space and to try to solve a maze, where a moving object should reach a set target in two-dimensional space under certain obstacle configurations. It should be noted that this functional example is taken from the behaviors of a female cricket, which can track the directions of males' positions in dark fields where there are a large number of unknown obstacles, by their chirps. We were not able to find an appropriate word to represent the function to solve one of typical ill-posed problems as such tracking. So, let us use the word “maze“ to represent the functional simulation in this paper. Our motivation is that we believe that chaotic dynamics, even in systems with large degrees of freedom, could be governed by a sequence of certain simple deterministic rules, or by a single rule as observed in nonlinear systems with small degrees of freedom. By virtue of our heuristic idea, one could obtain some insight into the dynamical mechanisms of their functions, which could then be applied toward realizing complex controls or complex information processing via a certain simple rule(s).

Embedding designed attractors into a recurrent neural network model and introducing chaotic dynamics

Let us introduce a recurrent neural network model and define the synchronous updating rule as follows:

$$s_i(t+1) = \mathrm{sgn}\left(\sum_{j \in G(r)} W_{ij} \cdot s_j(t) - \theta_i\right), \qquad (1)$$

where $s_i(t) = \pm 1$ ($i = 1, \ldots, N$) represents the firing state of a neuron specified by index i at time t, and the function sgn(x) takes 1 (if $x \geq 0$) or –1 (if $x < 0$). θ_i is the threshold of each neuron and is taken to be zero until the learning rule will be introduced in the discussion in the fourth section. W_{ij} is a connection weight (synaptic weight) from the neuron s_j to the neuron s_i, where W_{ii} is taken to be 0. $G(r)$ denotes a connectivity configuration set, where each neuron has r connectivities (randomly located fan-in number for each neuron), and the transmission of signals from the other ($N - r$) connectivities are assumed to be blocked by certain inhibitory action. It should be noted that configuration set $G(r)$ indicates spatial configurations of connectivity r, where there are ${}_NC_r$ different combinations for determining a specified $G(r)$. In our model, long-term behavior of $\boldsymbol{s}(t)$ is determined depending on a given set of connection matrices W_{ij}, and as is well known, an appropriately determined $\{W_{ij}\}$ enables us to make arbitrary chosen state vectors $\{\xi\}$ be multiple stationary states in the time development of $\boldsymbol{s}(t)$, which is equivalent to storing memory states in the functional context. In our study, W_{ij} are taken as follows:

$$W_{ij} = \sum_{\mu=1}^{K} \sum_{\lambda=1}^{M} (\xi_\mu^{\lambda+1})_i \cdot (\xi_\mu^{\lambda\dagger})_j, \qquad (2)$$

where M is the number of states included in a cycle ($\xi_\mu^{M+1} = \xi_\mu^1$) and K is the number of cyclic memories ($MK << N$). If connectivity r is large, with $r \simeq N$, the sequences of patterns used to construct the memory matrix are attracting sequences. Therefore, in the absence of external input, which will be introduced in later sections, the network can then function as a conventional associative memory. If $\boldsymbol{s}(t)$ is one of the memory patterns, ξ_μ^λ say, then $\boldsymbol{s}(t + 1)$ will be the next memory pattern in the cycle, $\xi_\mu^{\lambda+1}$. If $\boldsymbol{s}(t)$ is near one of the memory patterns ξ_μ^λ, then the sequence $\boldsymbol{s}(t + kM)$ (k = 1, 2, 3, ...) generated by the M-step map will converge to the memory pattern ξ_μ^λ. More specifically, for each memory pattern ξ_μ^λ, there is a set of states $B_{\mu\lambda}$, called a memory basin, such that if $\boldsymbol{s}(t)$ is in $B_{\mu\ \lambda}$, then $\boldsymbol{s}(t + kM)$ (k = 1, 2, 3, ...) will converge to ξ_μ^λ. In Eq. (2), $\xi_\mu^{\lambda\dagger}$ is the conjugate vector of ξ_μ^λ that satisfies $\xi_\mu^{\lambda\dagger} \cdot \xi_{\mu'}^{\lambda'} = \delta_{\mu\mu'}\delta_{\lambda\lambda'}$ and is introduced to enable us to avoid increasing spurious memories, and is defined as follows. If we choose arbitrary patterns as $\{\xi_\mu^\lambda\}$, then the patterns usually have strong overlap between them, where an overlap between the patterns is defined by

$$o^{\alpha\beta} = \frac{1}{N}\xi^\alpha \cdot \xi^\beta = \frac{1}{N}\sum_{i=1}^{N} \xi_i^\alpha \xi_i^\beta \quad (\alpha, \beta = 1, \ldots, MK) \qquad (3)$$

and, for case of description, the suffixes are changed to the total numbering MK. Using this $MK \times MK$ overlap matrix, let us define a as the inverse matrix of o, i.e., $a = o^{-1}$. Then $\xi^{\lambda\dagger}$ is defined as $\xi^{\lambda\dagger} = \sum_{\gamma=1}^{MK} a_{\lambda\gamma}\xi^\gamma$. It is almost trivial that $\{\xi_\mu^{\lambda\dagger}\}$ and $\{\xi_{\mu'}^{\lambda'}\}$ satisfy the orthogonal relation $\xi_\mu^{\lambda\dagger} \cdot \xi_{\mu'}^{\lambda'} = \delta_{\mu\mu'}\delta_{\lambda\lambda'}$ when the MK numbering of the vectors $\{\xi^{\nu(\dagger)} | \ \nu = 1, \cdots, MK\}$ is divided into (M, K) numbering as $\{\xi_\mu^{\lambda(\dagger)} | \ \mu = 1, \ldots, K, \lambda = 1, \cdots, M\}$ again.

Next, we introduce a certain system parameter and destabilize these multi-stable attractors. The idea is to reduce the number of synaptic connectivity r, in other words, the fan-in number to each neuron. When r becomes smaller and smaller, each basin volume decreases gradually. Finally, when r reaches some critical connectivity r_c ($r < r_\mathrm{c} \cong N/10$), each basin vanishes and the attractor becomes unstable. Thus, if the number of connectivity r is sufficiently reduced, updated network states do not converge to any cycle even if they have been updated for a long time. Since it can be observed that network dynamics becomes itinerant in N-dimensional state space consisting of the 2^N points of an N-dimensional hypercube, we will call such dynamics "chaotic wandering". To investigate the dynamical structure, we can calculate the basin visiting distribution in a certain updating step interval (see Suemitsu and Nara 2004).

It should be noted that the basin visiting measures are different from the relative basin volumes, which means that the orbits are not distributed uniformly in the state space but have certain dynamical structures. Considering this result, when connectivity r is sufficiently reduced, all the memory cycle attractors become unstable and the network shows highly developed chaotic dynamics. Empirically speaking, we have found that, depending on the selections of connection numbers and their spatial configurations, there are many cases where dynamics are itinerant but stay in each attractor basin for considerably longer time steps. In quantitative sense, however, it is quite difficult to obtain concrete statistical data due to the enormous variety of complex dynamics even in this model system with a finite but not so small number of degrees of freedom. Therefore, it is not yet clear whether these chaotic dynamics could be called Chaotic Itinerancy (CI) or not. The name CI was introduced by Kaneko, Tsuda, and Ikeda.

Now, to demonstrate the destabilizing processes in our model, let us simply show a bifurcation diagram of an overlap, where an overlap is a one-dimensional projection of state vector $\boldsymbol{s}(t)$ onto a certain reference vector. Thus, let us define an overlap $m(t)$ as an inner product with an initial state (the reference vector) $\boldsymbol{s}(0)$ and the state $\boldsymbol{s}(t)$ at time step t:

$$m(t) = \frac{1}{N}\sum_{i=1}^{N} s_i(0)s_i(t). \tag{4}$$

We have calculated long-term behaviors of overlap $m(t)$ for various connectivities r. In consequence, with reducing connectivity r, each M period attractor (taking $K = 4$ and $M = 6$ in the present example) becomes unstable, and periodic motions longer than M period are generated. Finally, non-periodic dynamics occurs (Fig. 1).

Complete description of arbitrarily given pattern dynamics with two-dimensional CA governed by totalistic rules

Let us begin with a brief introduction of one-dimensional CA under totalistic rules. We assume many cells, where each cell is arranged on a one-dimensional chain. We employ the variables a_i^t (=0 or 1, $i = 1,\ldots, N$), which indicate the state of the ith site in the chain at time step t. The state of the ith site at time step $t + 1$, a_i^{t+1} is determined by its state and those of its neighboring sites at time step t, so that the updating rule can be represented as

$$a_i^{t+1} = f(\ldots, a_{i-1}^t, a_i^t, a_{i+1}^t, \ldots), \tag{5}$$

where the function $f(\cdot)$ is called a transition function, which updates the state of a_i^t to a_i^{t+1}. If we employ the transition function:

$$a_i^{t+1} = f(a_{i-1}^t + a_i^t + a_{i+1}^t) \tag{6}$$

where $f(n) =$ 0 or 1 ($n =$ 0, 1, 2, 3), and introduce the rule number as

$$\begin{aligned}\text{Rule number (0–15)} &= f(0) + 2f(1) + 2^2 f(2) \\ &\quad + 2^3 f(3),\end{aligned} \tag{7}$$

then it is called the one-dimensional two-state three-neighbor CA under *totalistic rule*. We abbreviate this as 1-2-3-T-CA hereafter. It means that the state of a cell at a certain time step is determined by the total sum of the states of the three neighboring cells.

It is quite straightforward to extend this rule to two dimensions by introducing two-dimensional variables a_{ij}^{t+1} with the following formula:

$$a_{ij}^t = \quad 0 \text{ or } \quad 1. \tag{8}$$

The three neighbors in one-dimension are changed to, for example, five neighbors in two-dimension:

$$a_{ij}^{t+1} = f(a_{i-1,j}^t + a_{i,j-1}^t + a_{i,j}^t + a_{i,j+1}^t + a_{i+1,j}^t) \tag{9}$$

$$\begin{aligned}\text{Rule number} &= f(0) + 2f(1) + 2^2 f(2) + 2^3 f(3) \\ &\quad + 2^4 f(4) + 2^5 f(5),\end{aligned} \tag{10}$$

Table 1 and Fig. 2 show some examples of the rules and the time developing patterns given by the rule numbers 30, 47, and 48. The patterns are completely different to each other in spite of the same initial conditions.

Our idea is that we regard a sequence of arbitrarily given binary patterns as the time development of a corresponding 2-2-T-CA, where the number of neighbors should be chosen in an appropriate way so as to reproduce the given pattern dynamics. Now, an *inverse problem* arises: how to find, for each pixel, a 2-2-T-CA rule that can reproduce the sequence of given (designed) bit patterns, for instance, shown in Fig. 2.

Let us briefly describe our method of extracting the rules from the arbitrarily given cyclic pattern dynamics.

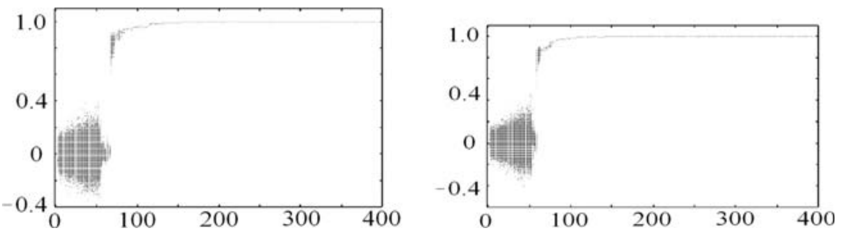


Fig. 1 The long-term behaviors of overlap $m(t)$ at M-step mappings as the functions of connectivity r, where the embedded attractor patterns are different to each other. One can observe that in both cases, the reduction in connectivity causes non-periodic dynamics (chaos)

Table 1 The relation between the rule number and the updated state of the cell

$n = \sum_{i,j} a_{ij}$	Rule number: $\sum_{n=0}^{n=5} 2^n f(n)$				
	0	30	47	48	63
0	0	0	1	0	1
1	0	1	1	0	1
2	0	1	1	0	1
3	0	1	1	0	1
4	0	1	0	1	1
5	0	0	1	1	1

0 ∼ ‘5 in the table correspond to $f(0)$~$f(5)$ in Eq. (10)

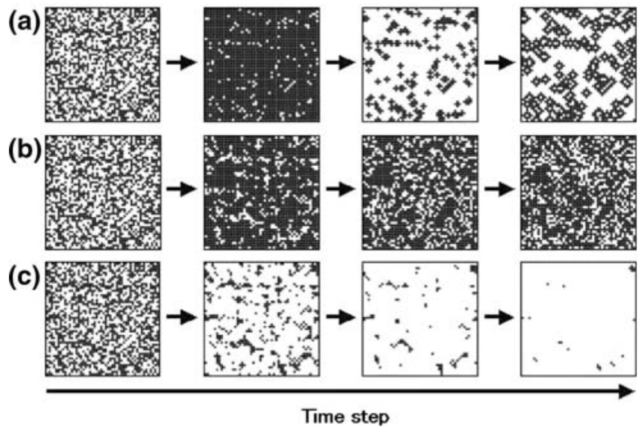


Fig. 2 The time developing patterns of 2-2-5-T-CA for four time steps of 2,500 cells (50 × 50). (**a**) rule number 30; (**b**) rule number 47; (**c**) rule number 48. As a one-dimensional CA, there are various time developing patterns

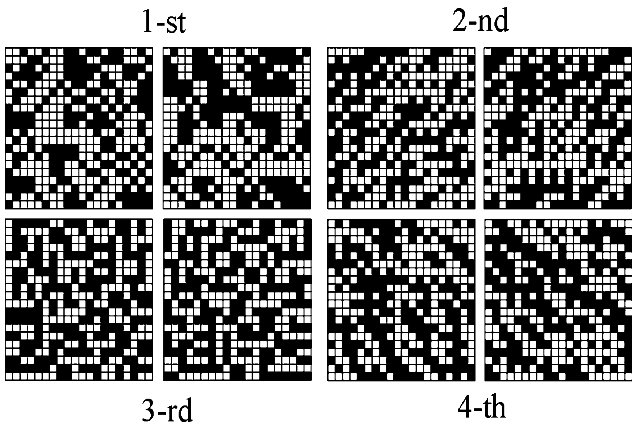


Fig. 3 The *K*-cycle patterns, each of which consists of *M* patterns per cycle. Note that each pattern has a certain internal structure as stated later. In the present case, *M* = 2 and *K* = 4, giving a total of *L* = *MK* = eight patterns. Let us call them cycles C_1, C_2, C_3, and C_4

Supposing certain cyclic pattern sequences, consisting of $N \times N$ pixels (cells) and *K*-cycles with *M*-patterns per cycle, so with a total of $L(= MK)$ patterns, we try to determine a totalistic rule of 2-2-T-CA to each cell (pixel) so as to reproduce the given periodic sequence of cell states perfectly, first, cycle by cycle and next, *K*-cycles simultaneously(Fig. 3). Each rule is represented by *Rule*[*i*, *j*], where (*i*, *j*) indicates the two-dimensional position in the frames. In relation to the 2-2-T-CA used in this paper, let us write the updating equation of the cell states as

$$a_{ij}^{t+1} = f_{ij}\left(\sum_{k,l \in G_{ij}(r)} a_{kl}^t\right), \tag{11}$$

where (*i*, *j*) represents the coordinate of the cells and $G_{ij}(r)$ represents the configuration of neighboring cells to calculate the total summation of states. Here, *r* denotes the numbering that represents the specified configuration so as to perfectly reproduce the given binary sequence. Furthermore, f_{ij} takes 0 or 1 depending on the value $\sum_{k,l \in G_{ij}(r)} a_{kl}^t$. To specify the *Rule*[*i*, *j*] in more detail, let us represent the rule of the cell located at (*i*,*j*) as

$$Rule[i, j] = (r : f_{ij}(0), f_{ij}(1), \ldots, f_{ij}(m(r)), \tag{12}$$

where $m(r)$ represents the maximum number of neighboring configurations in a chosen configuration specified by *r*. In finding a rule set { *Rule*[*i*, *j*] } to update cell patterns according to Eq. (11) for all (*i*, *j*), we must specify $\{G_{ij}(r)\}$ that represents the configuration of neighboring cells to calculate the total summation of states. So, let us take $G_{ij}(r)$ as shown in Fig. 4, where *r* = 1, 2, 3, ... and *r* should be increased until the given sequence of the cell is perfectly reproduced by applying *Rule*[*i*,*j*]. For computer simulation in the present paper, we employ *N* = 400, *K* = 4, and *M* = 2, so *L* = 8 (see Fig. 2). A detailed algorithm to determine *Rule*[*i*,*j*] = (*r* : $f_{ij}(0)$, $f_{ij}(1)$, ... ,$f_{ij}(m(r))$ is described and explained in Miura et al. (2005). After extracting the rules from the given cyclic patterns, we obtain, for each

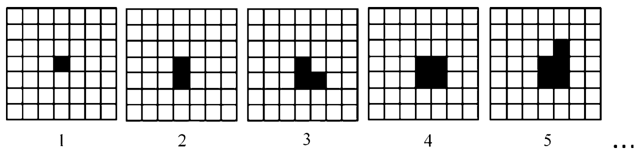


Fig. 4 The configurations of neighboring cells that will be necessary to reproduce the given bit-pattern sequence starting from the initial configuration. Numbers 1, 2, 3, ... , represent the configuration numbering *r* that specifies the spatial configuration

set of cyclic patterns, (cycles C_1, C_2, C_3, and C_4), the four corresponding rule sets (set R_1, R_2, R_3, and R_4) each of which consists of the 400-rules, and $\{ Rule[i, j] \}$, which corresponds to the $20 \times 20 = 400$ pixels and reproduces the periodic sequences in each cycle. The next task is to determine the rule set R_C, where it includes the 400-rules that corresponds to the $20 \times 20 = 400$ pixels and reproduces *all of the given periodic pattern sequences*. This means that once a pattern in the total $L(=MK)$ patterns belonging to any cycle is given as an initial pattern, the corresponding cyclic pattern sequence is reproduced by applying the extracted rules to each pixel. The rules belonging to rule set R_C are called the chaos rules, and the meaning of "chaos" is explained below.

The final task in this section is to consider the case that initial states include noise. An important point is that if noise is introduced in an initial pattern or during the updating of patterns, non-used cases in the possible values of $\sum_{k,l \in G_{ij}(r)} a^t_{kl}$ occur due to the existence of noise. Thus, one needs to determine the rules, which state whether value 1 or 0 should be employed in those cases. For instance, at each time step, there could occur $m(r) + 1$ cases of $\sum_{k,l \in G_{ij}(r)} a^t_{kl}$ in determining each cell state at the next step, which in a large number of cases would not occur under the non-existence of noise. There would be many choices because a large number of cases in $m(r) + 1$ are not used in reproducing the given pattern sequences. In the present paper, we employ the following interpolated extension of totalistic rules. We propose an idea that used cases (1 or 0 at certain specified cells) should be extended to the neighboring non-used cases one by one until they are fully interpolated. The result would be that any of the $m(r) + 1$ cases in $\sum_{k,l \in G_{ij}(r)} a^t_{kl}$ become available, even in the presence of noise during updating. Computer experiments indicate that the effect of noise brings the time development not random pattern dynamics but, depending on the five rule sets (R_1, R_2, R_3, R_4, and R_C), itinerant orbits that are, for the rule set R_1, R_2, R_3, R_4, quite close to the cycles that we designed (a weakly chaotic regime), and for the rule set R_C, highly delocalized chaotic orbits (a strongly chaotic regime) in the high-dimensional state space. These results suggests that the rules could drive new dynamics without the complex mechanism that is observed, say, in the neurodynamics of biological systems. If a large deviation from the original patterns occur, it becomes crucial to solve the maze designed, as shown in a later section.

In the next section, we observe both chaos in NN and CA using a certain visualizing method.

Observation of chaos in both NN and CA using transformation of dynamics into two-dimensional motion via a certain coding

Motion function defined from firing patterns in NN

The detailed content in this section was previously reported in our paper (Suemitsu and Nara 2004). Now, in two-dimensional space, an object is assumed to move from position $(q_x(t), q_y(t))$ to $(q_x(t + 1), q_y(t + 1))$ via a set of motion functions. With use of network state (t), the motion functions are defined, for instance, by $f_x(\boldsymbol{s}(t)) = \frac{4}{N}\sum_{i=1}^{\frac{N}{4}} s_i(t) \cdot s_{i+\frac{N}{2}}(t)$ and $f_y(\boldsymbol{s}(t)) = \frac{4}{N}\sum_{i=1}^{\frac{N}{4}} s_{i+\frac{N}{4}}(t) \cdot s_{i+\frac{3}{4}N}(t)$. Here, f_x and f_y range from –1 to +1 because of the normalization by $4/N$. Note that each motion function is calculated using a self inner-product between two parts of the network state $s(t)$. In two-dimensional space, actual motion of the object is given by $q_x(t + 1) = q_x(t) + f_x(\boldsymbol{s}(t + 1))$ and $q_y(t + 1) = q_y(t) + f_y(\boldsymbol{s}(t + 1))$. In the present example, we take $N = 400$, $M = 6$, $K = 4$, and two-dimensional space is digitized with a resolution of 0.02 because of the binary state vector $\boldsymbol{s}(t)$ with 400 elements and the definitions of f_x and f_y. A set of attractor patterns is determined as follows. For case of description, let us represent a designed state vector as $\xi^\lambda_\mu = \{\xi^{\mu,\lambda}_i | i = 1, \ldots, N\}$, where ($\mu = 1, \ldots ,K, \lambda = 1, \ldots ,M$). Each attractor pattern is divided into two parts. One is a random pattern part, where each state becomes + 1 or –1 with probability 0.5 ($\xi^{\mu,\lambda}_i = \pm 1 : i = 1,\ldots,N/2$). The other, ($\xi^{\mu,\lambda}_i = \pm 1 : i = N/2 + 1 ,\ldots, N$), is determined so as the relations $(f_x(\xi^\lambda_1), f_y(\xi^\lambda_1)) = (-1, -1)$, $(f_x(\xi^\lambda_2), f_y(\xi^\lambda_2)) = (-1, +1)$, $(f_x(\xi^\lambda_3), f_y(\xi^\lambda_3)) = (+1, -1)$, $(f_x(\xi^\lambda_4), f_y(\xi^\lambda_4)) = (+1, +1)$ are satisfied. When we choose $M = 6$ and $K = 4$, the four limit cycle attractors ($K = 4$), each of which has M (=6) patterns, are embedded in the synaptic connection matrix of the network.

Figure 5 shows the used patterns in the present experiment. Each limit cycle attractor corresponds to a constant motion of the object toward one of the four directions (+1, +1), (+1, –1), (–1, +1), (–1, –1). Let us call these attractor patterns to be "prototype patterns", which means they drive monotonic motions by introduced coding of motion functions.

Figure 6 shows an example of two motions that are generated by the first cycle attractor while maintaining the full connectivity ($r = 400$) and under chaotic dynamics with a low connectivity ($r = 10$).

Motion function defined from cell patterns in CA

According to the definition in the previous subsection, an object is assumed to move from the position $(q_x(t),$

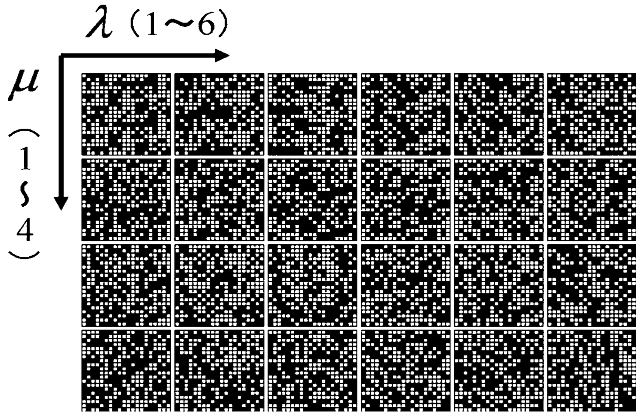


Fig. 5 The embedded cycles, where each cycle consists of six patterns. The first cycle $\{\xi_1^\lambda | \lambda = 1, \ldots, 6\}$ is chosen so as $(f_x(\xi_1^\lambda), f_y(\xi_1^\lambda)) = (-1, -1)$ are satisfied. The others correspond to (–1, +1), (+1, –1), (+1, +1), respectively

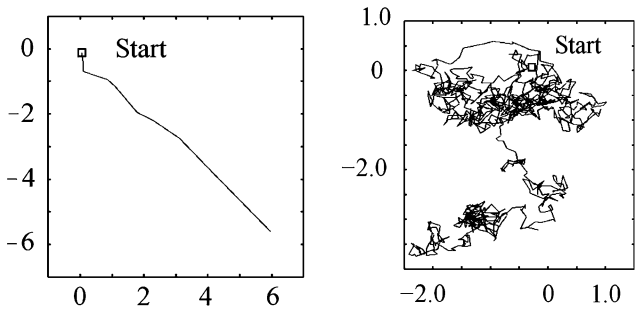


Fig. 6 (*Left*) An example of the object orbit from start point (0, 0) in two-dimensional space in an attractor regime (r = 400). One of the monotonic motions embedded in synaptic connection matrix W_{ij} is shown. (*Right*) An example of the object orbit from start point (0, 0) in two-dimensional space in a chaotic regime (r = 10). Note that the chaotic motions strongly depend on connectivity r

$q_y(t)$) to ($q_x(t+1)$, $q_y(t+1)$) via a set of motion functions. Converting of cell states $\{a_{i,j}^t = 0, 1\}$ to $\mathbf{s}(t)$ defined as $\mathbf{s}(t) = \{s_{ij}(t) = 2a_{ij}^t - 1 \mid i, j = 1\text{–}20\}$, and employing a slightly different definition from the previous section, the motion functions are defined by, for instance, by

$$f_x(\mathbf{s}(t)) = \frac{1}{100} \sum_{(i,j)}^{(10,10)} s_{2i-1,2j-1}(t) \cdot s_{2i,2j} \tag{13}$$

$$f_y(\mathbf{s}(t)) = \frac{1}{100} \sum_{(i,j)}^{(10,10)} s_{2i-1,2j}(t) \cdot s_{2i,2j-1}(t), \tag{14}$$

where we take N = 400, so f_x and f_y range from –1 to +1 because of normalization. In two-dimensional space, actual motion of the object is given by

$$q_x(t+1) = q_x(t) + f_x(\mathbf{s}(t+1)), \tag{15}$$

$$q_y(t+1) = q_y(t) + f_y(\mathbf{s}(t+1)). \tag{16}$$

In our computer simulations, two-dimensional space is digitized with a resolution of 0.02 because of the minimum variation of the motion function, f_x and f_y, defined with use of the state vector $\mathbf{s}(t)$ with 400 elements. Taking K = 4 and M = 2 in the present simulation, K (=4) cycles, each of which has M (=2) patterns, are embedded in the state space. In other words, they are described by the extracted CA rules. Each cycle pattern (prototype pattern) corresponds to a monotonic motion of the object toward one of the four directions (+1, +1), (+1, –1), (–1, +1), (–1, –1) in two-dimensional space.

In our model, each embedded pattern cycle corresponds to one of four prototypical motions in two-dimensional space. When we give a random binary pattern as an initial state and update it by applying a rule set, weakly chaotic output of cell pattern dynamics is generated and the rule sets R_1, R_2, R_3, R_4 provide monotonic motions of the object in two-dimensional space in the four directions. Figure 7 shows the motion of the object to move in four directions monotonically within 20 time steps. Furthermore, if an initial cell pattern (a random binary pattern) is updated by the chaos rule set R_C, chaotic motion of the object occurs. Figure 8 depicts the chaotic orbit generated from chaotic dynamics of the chaos rule set.

In closing this section, it should be noted that different choices of M, the number of patterns per cycle, give qualitatively the same results for the simulation stated in the next section. Therefore, our results can be described as typical. However, regarding the various

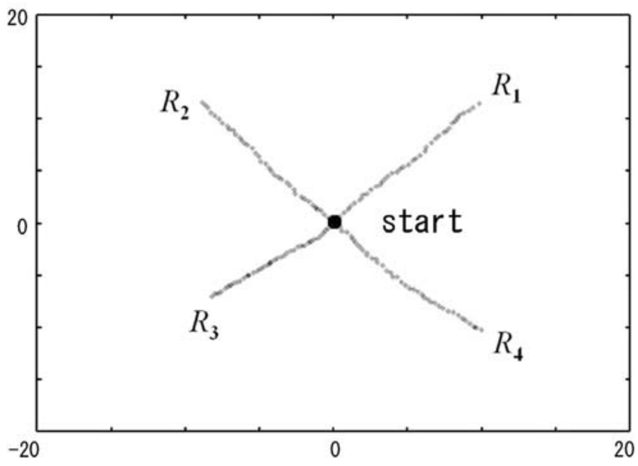


Fig. 7 An example of the object orbit driven by the four rule sets R_1, R_2, R_3, and R_4, starting from the point (0,0) in two-dimensional space: the four motions within 20 time steps

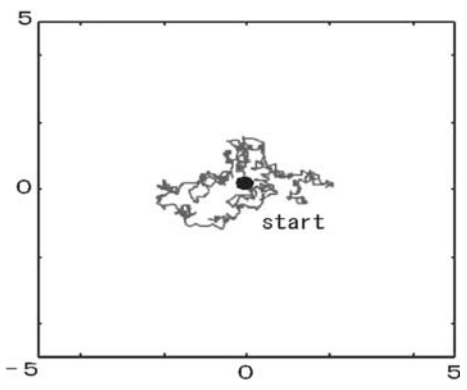


Fig. 8 An example of a chaotic orbit driven by the chaos rule set, R_C, starting from the point (0,0) in two dimensional space, where the orbit is taken up to 500 time steps

choices of prototype patterns that satisfy $(f_x(\xi_\mu^\lambda), f_y(\xi_\mu^\lambda)) = (-1,-1), (-1,+1), (+1,-1), (+1,+1)$, different rule sets, $R = \{R_1, R_2, R_3, R_4, R_C\}$ perform differently in solving the maze. Let us call $R = \{R_1, R_2, R_3, R_4, R_C\}$ a ''Motion-Rule-Set'', where we use the set numbering α (α = 1, 2, 3, …) if we employ many Motion-Rule-Sets and call them R^α. Various choices of different prototype patterns gave different performance when we conducted simulations to solve the maze. Thus, we tested a hundred of Motion-Rule-Sets ($\{ R^\alpha \mid \alpha = 1\text{–}100 \}$ and the number of corresponding designed patterns is 800 in total) in our simulations executed in the next section, to find Motion-Rule-Sets that provide better performance.

Motion control with adaptive switching between a weakly chaotic regime and a strongly chaotic regime in NN and CA

Generally speaking, from a controlling viewpoint, chaos is considered to spoil control systems. A large number of methods have been proposed to avoid the emergence of chaos. It was surprising when the OGY method was proposed (Ott et al. 1990) to stabilize an arbitrary periodic orbit in chaos, but their successful treatment was restricted to a system with a few degrees of freedom.

Now, as stated in the ''Introduction'' section, we consider chaos to be useful not only in solving ill-posed problems but also in controlling of systems with many but finite degrees of freedom. To demonstrate potential capabilities of chaotic dynamics generated by NN and CA, let us try to apply it to control functioning and select, as an example, a maze in two-dimensional space. An object is assumed to move in the two-dimensional maze and approaches the target via the use of chaotic dynamics. One reason why we consider the maze is that the process to solve a maze can be easily visualized. That is, we can understand how the dynamical structures are effectively utilized in controlling. An object is assumed to move in the two-dimensional maze and approaches the target with use of chaotic dynamics. This functional simulation corresponds to the behaviors of a female cricket, which can track the directions of males' positions in dark fields where there are a large number of unknown obstacles, by their chirps. Our idea, therefore, is as follows:

(1) The weakly chaotic regime in NN/CA gives monotonic motion to the quadrant where the target belongs.
(2) The strongly chaotic regime in NN/CA gives non-monotonic motion, provided that there is an obstacle in the target direction and/or the direction of the previous one-step motion does not coincide with the target direction.
(3) A roving robot can always know the quadrant where the target belongs, just as a female cricket can know the directions of males' positions in dark fields where there are a large number of unknown obstacles, from their chirps.

Computer simulations of motion control using chaos in NN

In this section, we propose a control method for the object through switching the connectivity r with a simple evaluation. A target is assumed to be set in two-dimensional space. Note that an exact coordinate value of the target (Q_{x0}, Q_{y0}) is not introduced in the control method proposed below. As an example of an ambiguous external response, a rough direction of the target $D_1(t)$ with a certain tolerance is defined and is recognized by the object. For example, $D_1(t)$ becomes 1 if the direction from the object to the target is observed between an angle from 0 to $\pi/2$ (the first quadrant), where the angle is defined by the x-axis in two-dimensional space. Similarly, $D_1(t)$ becomes n (= 1,2,3,4) if the direction belongs to the angle $(n-1)\pi/2$ and $n\pi/2$. Second, the direction of the object's motion $D_2(t)$ from time t–1 to t is defined as

$$D_2(t) = \begin{cases} 1 & (c_x(t) = +1 \text{ and } c_y(t) = +1) \\ 2 & (c_x(t) = -1 \text{ and } c_y(t) = +1) \\ 3 & (c_x(t) = -1 \text{ and } c_y(t) = -1) \\ 4 & (c_x(t) = +1 \text{ and } c_y(t) = -1) \end{cases},$$

where $c_x(t)$ and $c_y(t)$ are given as

$$c_x(t) = \frac{q_x(t) - q_x(t-1)}{|q_x(t) - q_x(t-1)|} \tag{17}$$

$$c_y(t) = \frac{q_y(t) - q_y(t-1)}{|q_y(t) - q_y(t-1)|}. \tag{18}$$

Finally, with use of $D_1(t)$ and $D_2(t)$, a time-dependent connectivity $r(t)$ in the network is determined:

$$r(t) = \begin{cases} R_L(= N) & \text{if } D_1(t-1) = D_2(t-1) \\ R_S(\ll N) & \text{otherwise} \end{cases}, \tag{19}$$

where R_L is sufficiently high connectivity (N in our experiments) and R_S is low connectivity. Therefore, after determination of the connectivity $r(t)$, the motion of the object is calculated from the network state updated with $r(t)$. This motion provides a further calculation for $D_1(t)$ and $D_2(t)$. By repeating this process, where the connectivity is switched between R_L and R_S, the object moves in two-dimensional space. In this control method, the high connectivity is maintained with $r(t) = R_L$ if the object moves toward the target with a tolerance of $\pi/2$. This provides stable motion toward the target.

Figure 9 shows an example of the orbit of the object with the control method proposed above. In the figure, the object moves along the horizontal axis, which is not embedded as the prototypical attractor in the network. Furthermore, Figs. 10 and 11 show examples where the walls, which the object is not permitted to go through, are set in two-dimensional space like a maze. We have assumed that the object can know $D_1(t)$ as each time step without depending on the existence of the wall. In our experiments, both $f_x(\mathbf{s}(t))$ and $f_y(\mathbf{s}(t))$ are taken to be zero if the moving object hits the wall. This means that the object cannot penetrate the wall when it hits the wall. Various solutions to the problem that the object approaches the target in the two-dimensional maze can be observed.

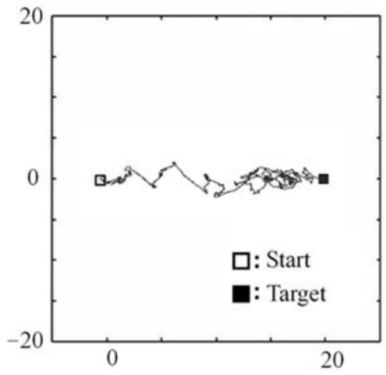


Fig. 9 An example of the object orbit in two-dimensional space with a simple control method from start point (0, 0) to target point (20, 0): using simple switching between R_S and R_L, motion toward the correct direction occurs, although that motion is NOT embedded in the synaptic connection matrix

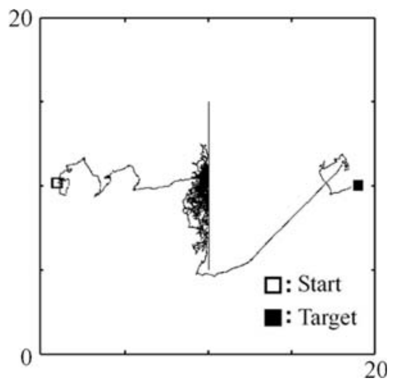


Fig. 10 Example of an object orbit in two-dimensional space with a wall between start point (0, 0) and target point (20, 0): after colliding with the wall, the object escapes and moves to the target

Computer simulations of motion control using chaos in CA

Now, let us introduce a control method for the object that operates by switching the rule sets based on simple evaluation at each time step (Fig.12). The set situations are the same as those of NN, the only difference is the method of changing the system parameters (connectivity in NN). Because Connectivity Switching in NN is replaced by Rule Switching in CA, using $D_1(t)$ and $D_2(t)$, the time-dependent rule $R(t)$ for updating the cell pattern is chosen as

$$R(t) = \begin{cases} R_i & \text{if } D_1(t-1) = D_2(t-1) = i \\ R_C & \text{otherwise} \end{cases}, \tag{20}$$

where R_i ($i =$ 1–4) is one of the rule sets defined in the previous section and R_C is the chaos rule set, also defined in the previous section. Therefore, after selecting the rules $R(t)$, the motion of the object is calculated from the cell states updated by $R(t)$. The motion provides a further calculation for $D_1(t)$ and $D_2(t)$. Repeating this process, where the rule set is switched between R_i and R_C, moves the object in two-dimensional space. In this control method, the rule set is kept as $R(t) = R_i$ while the object moves toward the target with a tolerance of $\pi/2$ and provides monotonic motion toward the target. Figure 13 shows a block diagram of the control method proposed above.

Now, based on this control method, let us show our computer simulations to solve the maze. In our

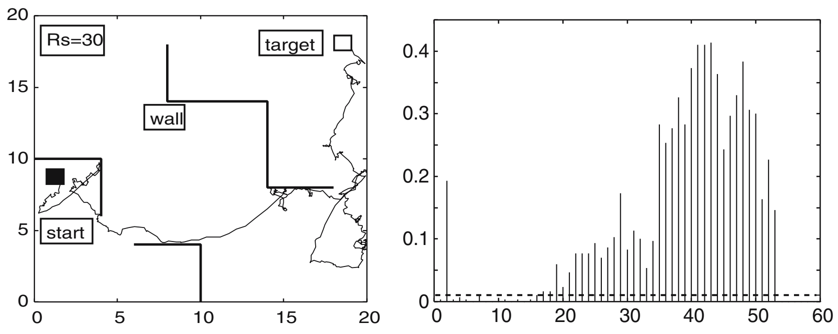


Fig. 11 (*Left*) Example of an object orbit in a two-dimensional maze with obstacles (walls) between start point (1,9) and target point (19,19). (*Right*) The horizontal axis represents R_S and the vertical axis is the success rate. The dotted line represents the result by means of a random pattern generator

experiments, both $f_x(\mathbf{s}(t))$ and $f_y(\mathbf{s}(t))$ are taken to be zero if the moving object hits the wall, meaning that the object cannot penetrate the wall when it hits the wall (Fig. 14).

Functional equivalence of chaos in NN and CA

In this section, let us show the functional equivalence of chaos in NN and CA in the simulations to solve a maze with respect to the three kinds of maze structures shown in Fig. 15: (1) Ω-type, (2) S-type, and (3) V-type. We prepared four variations of each maze type, which are the mirror symmetries shown in Fig. 16

Figure 16 shows an example of successful cases. Various solutions in which the object approaches the target in two-dimensional maze structures can be observed. The equivalent cases solved using CA are depicted in Fig. 17. These simulations were also executed using the maze structures shown in Fig. 15 and their mirror-symmetric cases. We do not indicate the results, however, since all cases give successful approaches to the targets, being the same as in the Ω-type.

These results indicate that NN and CA can solve various mazes. Now, to evaluate the functional equivalence in more detail, let us investigate the functional performance. To show the performance of the control method proposed above regarding solving a two-dimensional maze from a quantitative viewpoint, we calculate the a success rate for 300 random initial states. The result of the control method is compared with that of a random method (random walk), where

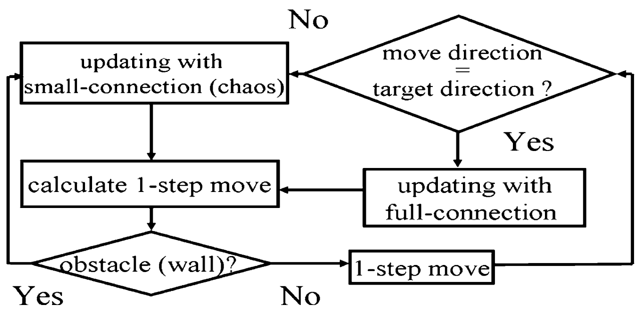

No
updating with small-connection (chaos)
move direction = target direction ?
Yes
calculate 1-step move
updating with full-connection
obstacle (wall)?
1-step move
Yes
No

Fig. 12 Block diagram of the controlling method using our neural network model

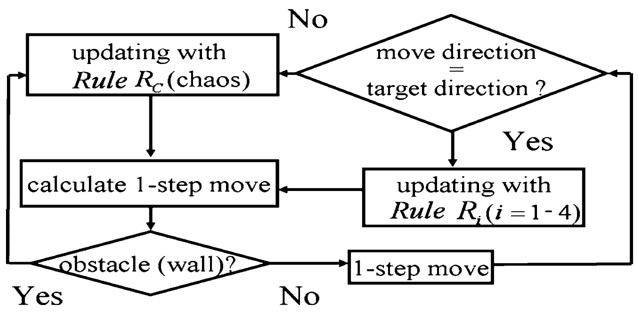

No
updating with Rule R_C (chaos)
move direction = target direction ?
Yes
calculate 1-step move
updating with Rule R_i (i = 1 - 4)
obstacle (wall)?
1-step move
Yes
No

Fig. 13 Block diagram of the control method with use of CA proposed in the present paper. Note that it differs from that of NN only from the point that the connectivity switching in NN is replaced by rule switching in CA

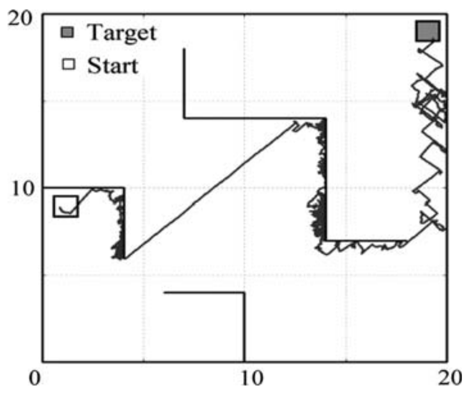


Fig. 14 Example of solving a maze from the start point (1,9) to the target point (19,19), where the object's orbit in two-dimensional space with walls is shown

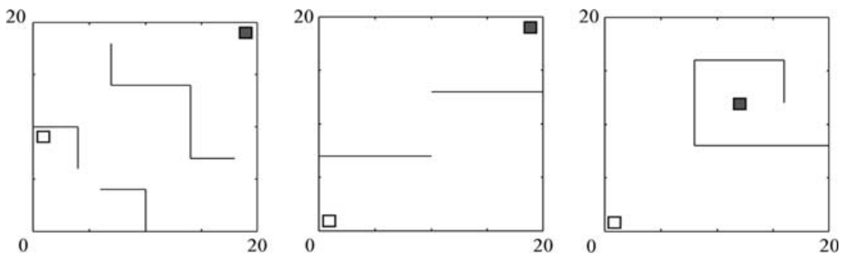


Fig. 15 The three kinds of mazes (Ω-type, S-type, and V-type) taken in our simulations. The open square is the starting position and the solid square is the target position

Fig. 16 The four types of mazes taken with mirror symmetries, and the successful orbit in each case, obtained using NN

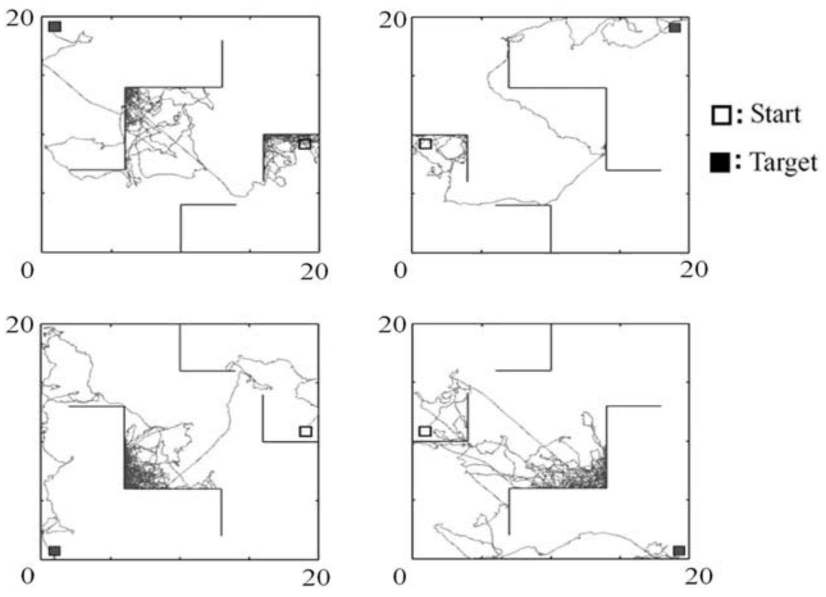
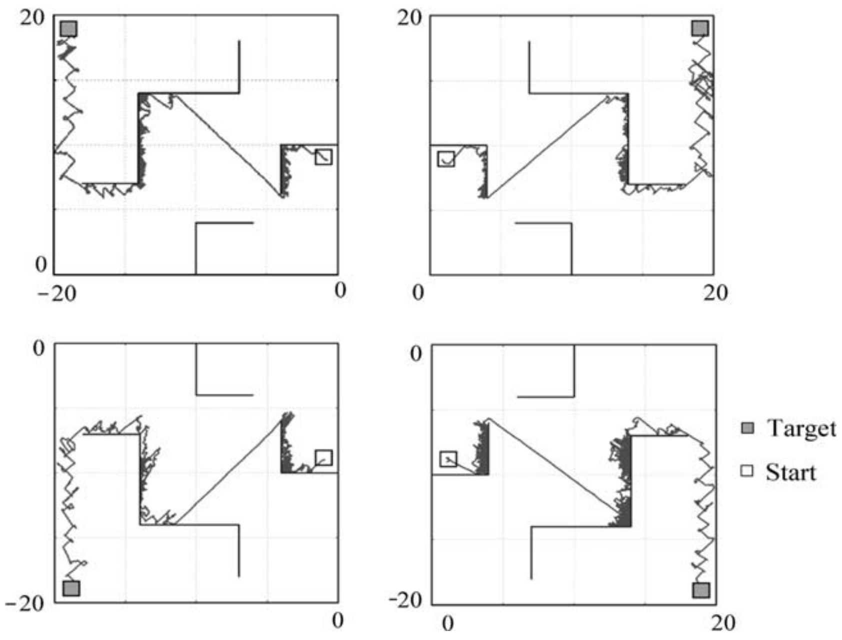


Fig. 17 Four solutions using CA, which are mirror-symmetric to each other to investigate the validity of chaos for solving various mazes



the random bit-pattern generator is used instead of chaotic dynamics. At every time step, the network state $\boldsymbol{s}(t)$ is replaced by a random pattern generator in the case $D_1(t–1) \neq D_2(t–1)$ instead of chaotic dynamics. If $D_1(t–1) = D_2(t–2)$, then $r(t)$ is switched onto $R_L = N$. The problem in the actual computer simulation is the maze shown in Fig. 15, where 300 random patterns are set as the initial states of the network. The rate of successful approaches to the target within 5,000 time steps is calculated as the success rate in Fig. 18 for NN. The success rate for the same problem with random walk is 0.01 (Fig. 19). The success rate of the control method with chaotic dynamics in the network between $R_S = 40$ and $R_S = 50$ is significantly larger than that with random walk. The configuration of $\varepsilon_{r;ij}$'s is randomly chosen with the condition $\sum_j \varepsilon_{r;ij} = r$. The difference of configuration $\varepsilon_{r;ij}$'s give various results even if the same initial condition is given, as shown in Fig. 20.

To explain one of the reasons why our method performs better than random walk, let us discuss time hysteresis of the network state $\boldsymbol{s}(t)$ from a dynamical viewpoint. In the proposed controlling method, for instance in NN, the network's connectivity is set to a low connectivity $r(t) = R_S$ when the object collides with a wall in two-dimensional space while moving toward

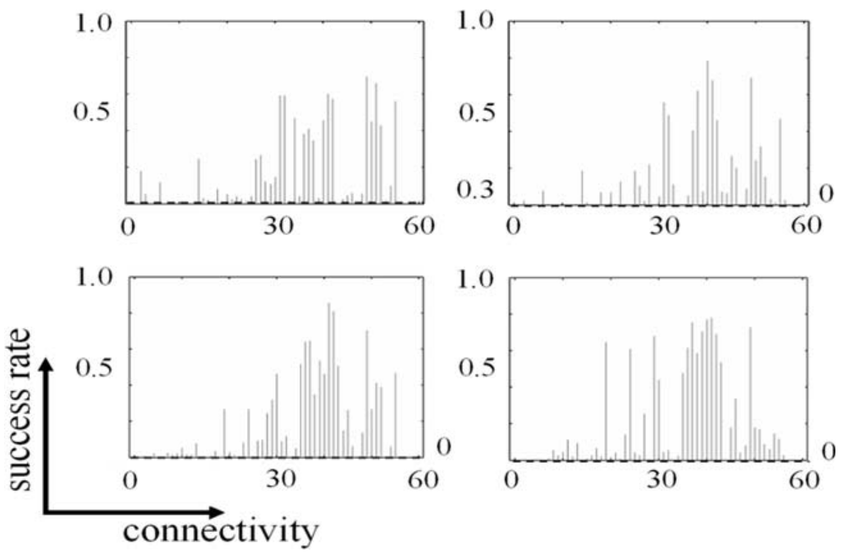


Fig. 18 The success rate of solving mazes with NN, where each figure corresponds to the maze type in Fig. 16. The results from using a random-number generator are shown by dotted lines in the figures but they are too small for comparison with those by NN

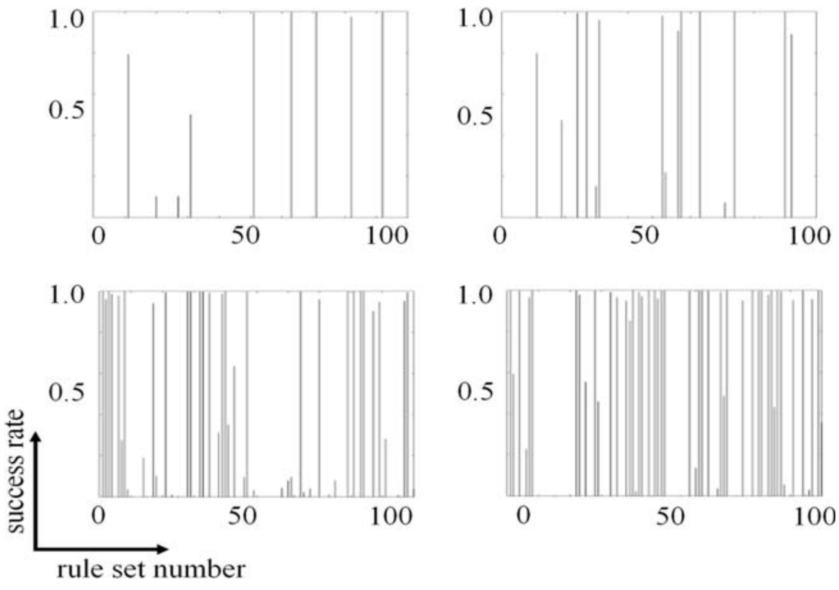


Fig. 19 The success rate of solving mazes with CA, where each figure corresponds to the maze-type in Fig. 17. The results using a random number generator are too small for comparison with those of CA. Note that, instead of connectivity in NN, the horizontal axis indicates that we employ various Motion-Rule-Sets R^{α}, and the numbering α (α = 1–100) corresponds to each choice of variation of the prototype patterns

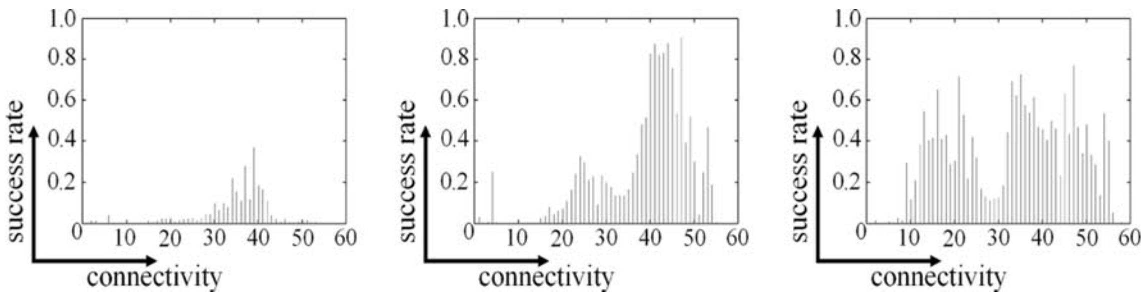


Fig. 20 A performance comparison in NN depending on connectivity, where the three kinds of configurations $\varepsilon_{r;ij}$'s with respect to the same connectivity number ($r =$ 40) are taken. The maze type is the one shown in Fig. 11

the target. Then, in making a detour to approach the target, the object must move in a direction that does not take it directly toward the target. This implies that, with low connectivity R_S, chaotic wandering in the network should remain for a while in a certain attractor basin until the associated motion in maze can find a detouring route to the target that avoids the wall, which means that "attractor ruin" still remains in chaos, and it gives a better performance than a completely random number generator. Actually, Fig. 21 shows an example of itinerating in attractor ruins in the chaos of NN.

Finally, let us comment about a random method where a random bit-pattern generator is used instead of chaotic dynamics. At every time step, the pattern $\mathbf{s}(t)$ is replaced by a random pattern generator in the

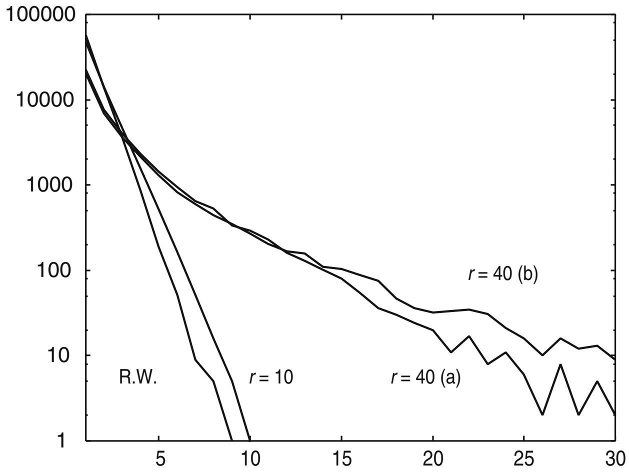


Fig. 21 Log plot of the frequency distribution of staying-time step length t: The horizontal axis represents the successive basin staying length and the vertical axis is each observed frequency up to 10^5. $r = 10$ is the case of highly developed chaos with connectivity $r = 10$. $r = 40$(a) is the case with $r = 40$ and that gives good performance for the maze shown in Fig. 11. $r = 40$(b) is different from $r = 40$(a) for the configuration $\varepsilon_{r;ij}$ and gives bad performance. R.W. is where a random number generator is replaced with chaotic dynamics in the neural network model

case $D_1(t-1) \neq D_2(t-1)$ instead of chaotic dynamics. If $D_1(t\text{–}1) = D_2(t\text{–}2) = i$, then $R(t)$ is switched onto R_i ($i =$ 1–4) in NN and the rule R_i in { R_1, R_2, R_3, R_4 } in CA. The success rate for the same problem with random walk is 0% up to 50,000 trials of initial random cell patterns, where a trial is regarded as unsuccessful if it does not reach the target within 5,000 time steps of updating. This result indicates that our idea of using chaotic dynamics in the present method is more useful, at least, than that using random number generator.

Concluding remarks

The following is a summary of our work.

(1) We proposed a novel method of motion control using high-dimensional chaotic dynamics in a recurrent neural network model and totalistic cellular automata.
(2) Actual examples were given using computer simulations with respect to binary bit-patterns consisting of multi-cyclic patterns that were designed for motion control.
(3) Utilizing adaptive switching of connectivity in NN and/or in CA, the five rule sets between a weakly chaotic regime (generating the four monotonic motions) and a strongly chaotic regime (generating chaotic motion), we obtained better performance than a random number generator in solving two-dimensional mazes.
(4) We successfully performed functional computer simulations of a roving robot to solve mazes using chaotic dynamics in a recurrent neural network model, and in totalistic cellular automata. An important point is that high-dimensional firing patterns in state space should be appropriately coded into simple motions, then after introducing chaotic dynamics, attractor ruins give useful motions to solve the maze.

(5) The adaptive parameter switching between a converging regime (or weakly chaotic regime) and a strongly chaotic regime, depending on a simple evaluation which includes uncertainty, is the key idea in realizing complex functions using chaos.
(6) The results of simulations indicate that both are equivalent with respect to functional potentialities.

Acknowledgement This work has been supported partly by a Grant-in-Aid for the Promotion of Science #16500131 from the Japan Society for the Promotion of Science.

References

Aihara K, Takabe T, Toyoda M (1990) Chaotic neural networks. Phys Lett A 114:333–340

Aizawa Y, Nagai Y (1987) Dynamics on patterns and rules—rule dynamics. Bussei Kenkyu 48:316–320 (private communications in Japanese)

Aizawa Y, Nishikawa I (1986) In: Ikegami G (ed) Dynamical systems and nonlinear oscillators. World scientific, singapore, pp. 210–222

Anderson JA, Rosenfeld E (eds) (1988) Neurocomputing. The MIT Press, Cambridge, MA

Anderson JA, Rosenfeld E (eds) (1990) Neurocomputing 2. The MIT Press, Cambridge, MA

Davis P, Nara S (1990) Chaos and neural networks. Proc. First Symp. Nonlinear Theory and Its Application, p 97

Fujii H, Itoh H, Ichinose N, Tsukada M (1996) Dynamical cell assembly hypothesis – Theoretica possibility of spatio-temporal coding of the cortex. Neural Netw 9:1303

Kaneko K (1990) Clustering, coding, switching, hierarchical ordering and control in a network of chaotic elements. Physica D 41:137

Kaneko K, Tsuda I (2001) Complex systems: chaos and beyond. Springer-Verlag, Berlin/Heidelberg/New York

Kay L, Lancaster L, Freeman WJ (1996) Reafference and attractors in the olfactory system during odor recognition. Int J Neural Syst 7:489

Kim S, Aizawa Y (2000) Cluster formations in rule dynamical systems: emergency of non-local effects. Prog Theor Phys 104:289–305

Kuroiwa J, Nara S, Ogura H (2005) RISP Int. workshop on nonlinear circuits and signal processing, Honolulu, Hawai, USA, March 4–6, 2005

Mikami S, Nara S (2003) Dynamical responses of chaotic memory dynamics to weak input in a recurrent neural network model. Neurocomput Applic 11(3&4):129–136 (double issue)

Miura T, Tanaka T, Suemitsu Y, Nara S (2005) Complete and compressive description of motion pictures by means of two-dimensional cellular automata. Phys Lett A 346(4):296–304

Nara S (2003) Can potentially useful dynamics to solve complex problems emerge from constrained chaos and/or chaotic itinerancy? Chaos 13(3):1110–1121

Nara S, Davis P (1992) Chaotic wandering and search in a cycle-memory neural network. Prog Theor Phys 88:845–855

Nara S, Davis P, Kawachi, Totsuji H (1995) Chaotic memory dynamics in a recurrent neural network with cycle memories embedded by pseudo-inverse method. International Journal of Bifurcation and Chaos 5(4):1205–1212

Nara S, Wada M, Abe N, Kuroiwa J (1999) A novel method of sound data description by means of cellular automata and its application to data compression. Int J Birurcation Chaos 9(6):1211–1217

Nicolelis MAL (2001) Actions from thoughts. Nature 409:403–407

Ott E, Grebogi C, Yorke JA (1990) Controlling chaos. Phys Rev Lett 64:1196

Skarda CA, Freeman WJ (1987) How brains make chaos in order to make sense of the world. Behav Brain Sci 10:161–195

Suemitsu Y, Nara S (2004) A solution for two dimensional maze with use of chaotic dynamics in a recurrent neural network model. Neural Comput 16:1943–1957

Tamura T, Kuroiwa J, Nara S (2003) Errorless reproduction of given pattern dynamics by means of cellular automata. Phys Rev E 68:036707-1–036707-8

Tokuda I, Nagashima T, Aihara K (1997) Global bifurcation structure of chaotic neural networks and its application to traveling salesman problems. Neural Netw 10(9):1673–1690

Tsuda I (2001) Towards an interpretation of dynamic neural activity in terms of chaotic dynamical systems. Behav Brain Sci 24:793–847

Wada M, Kuroiwa J, Nara S (2002) Completely reproducible coding of digital sound data with Cellular Automata. Phys Lett A 306:110–115

Wessberg J, Stambaugh CR, Kralik JD, Beck PD, Chapin JK, Kim J, Biggs SJ, Srinivasan MA, Nicolelis MAL (2000) Real-time prediction of hand trajectory by cortical neurons in primates. Nature 408:361–365

Wolfram S (1986) Theory and application of cellular automata. World Scientific Singapore. Also see the recent publication, A new kind of science, Wolfram Media Inc., 2002