Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 7158507, 14 pages
http://dx.doi.org/10.1155/2016/7158507

Research Article

A Self-Organizing Incremental Spatiotemporal Associative Memory Networks Model for Problems with Hidden State

Zuo-wei Wang

Department of Computer Science and Software, Tianjin Polytechnic University, Tianjin 300387, China

Correspondence should be addressed to Zuo-wei Wang; wangzuowei@126.com

Received 31 May 2016; Revised 23 July 2016; Accepted 27 July 2016

Academic Editor: Manuel Graña

Identifying the hidden state is important for solving problems with hidden state. We prove any deterministic partially observable Markov decision processes (POMDP) can be represented by a minimal, looping hidden state transition model and propose a heuristic state transition model constructing algorithm. A new spatiotemporal associative memory network (STAMN) is proposed to realize the minimal, looping hidden state transition model. STAMN utilizes the neuroactivity decay to realize the short-term memory, connection weights between different nodes to represent long-term memory, presynaptic potentials, and synchronized activation mechanism to complete identifying and recalling simultaneously. Finally, we give the empirical illustrations of the STAMN and compare the performance of the STAMN model with that of other methods.

1. Introduction

The real environment in which agents are is generally an unknown environment where there are partially observable hidden states, as the large partially observable Markov decision processes (POMDP) and hidden Markov model (HMM) literatures attest. The first problem for solving a POMDP is hidden states identifying. In many papers, the method of using the k-step short-term memory to identify hidden states has been proposed. The k-step memory is generally implemented through tree-based models, finite state automata, and recurrent neural networks.

The most classic algorithm of tree-based model is U-tree model [1]. This model is a variable length suffix tree model; however, this method can only obtain the task-related experiences rather than general knowledge of the environment. A feature reinforcement learning (FRL) framework [2, 3] is proposed, which considers maps from the past observation-reward-action history to an MDP state. Nguyen et al. [4] introduced a practical search context trees algorithm for realizing the MDP. Veness et al. [5] introduced a new Monte-Carlo tree search algorithm integrated with the context tree weighting algorithm to realize the general reinforcement learning. Because the depth of the suffix tree is restricted, these tree-based methods cannot efficiently handle long-term dependent tasks. Holmes and Isbell Jr. [6] first proposed the looping prediction suffix trees (LPST) in the deterministic POMDP environment, which can map the long-term dependent histories onto a finite LPST. Daswani et al. [7] extended the feature reinforcement learning framework to the space of looping suffix trees, which is efficient in representing long-term dependencies and perform well on stochastic environments. Daswani et al. [8] introduced a squared Q-learning algorithm for history-based reinforcement learning; this algorithm used a value-based cost function. Another similar work is by Timmer and Riedmiller [9], who presented the identify and exploit algorithm to realize the reinforcement learning with history lists, which is a model-free reinforcement learning algorithm for solving POMDP. Talvitie [10] proposed the temporally abstract decision trees to learning partially observable models. These k-step memory representations based on multidimensional tree required additional computation models, resulting in poor time performances and more storage space. And these models have poor tolerance to fault and noise because of the accurate matching of each item.

More related to our work, finite state automata (FSA) has been proved to approximate the optimal policy on belief

states arbitrarily well. McCallum [1] and Mahmud [11] both introduced the incremental search algorithm for learn probabilistic deterministic finite automata, but these methods learn extremely slowly and with some other restrictions. Other scholars use recurrent neural networks (RNN) to acquire memory capability. A well known architecture for RNN is Long Short-term Memory (LSTM) proposed by Hochreiter and Schmidhuber [12]. Deep reinforcement learning (DRL) [13] first was proposed by Mnih et al., which used deep neural networks to capture and infer hidden states, but this method still apply to MDP. Recently, deep recurrent Q-learning was proposed [14], where a recurrent LSTM model is used to capture the long-term dependencies in the history. Similar methods were proposed to learn hidden states for solving POMDP [15, 16]. A hybrid recurrent reinforcement learning approach that combined the supervised learning with RL was introduced to solve customer relationship management [17]. These methods can capture and identify hidden states in an automatic way. Because these networks use common weights and fixed structure, it is difficult to achieve incremental learning. These networks were suited to resolve the spatiotemporal pattern recognition (STPR), which is extraction of spatiotemporal invariances from the input stream. For the temporal sequence learning and recalling in more accurate fashion, such as the trajectory planning, decision making, robot navigation, and singing, special neural network models for temporal sequence learning may be more suitable.

Biologically inspired associative memory networks (AMN) have shown some success for this temporal sequence learning and recalling. These networks are not limited to specific structure, realizing incremental sequence learning in an unsupervised fashion. Wang and Yuwono [18] established a model to recognize and learn complex sequence which is also capable of incremental learning but need to provide different identifiers for each sequence artificially. Sudo et al. [19] proposed the self-organizing incremental associative memory (SOIAM) to realize the incremental learning. Keysermann and Vargas [20] proposed a novel incremental associative learning architecture for multidimensional real-valued data. However, these methods cannot address temporal sequences. By using time-delayed Hebb learning mechanism, a self-organizing neural network for learning and recall of complex temporal sequences with repeated items and shared items is presented in [21, 22], which was successfully applied to robot trajectory planning. Tangruamsub et al. [23] presented a new self-organizing incremental associative memory for robot navigation, but this method only dealt with simple temporal sequences. Nguyen et al. [24] proposed a long-term memory architecture which is characterized by three features: hierarchical structure, anticipation, and one-shot learning. Shen et al. [25, 26] provided a general self-organizing incremental associative memory network. This model not only leaned binary and nonbinary information but realized one-to-one and many-to-many associations. Khouzam [27] presented a taxonomy about temporal sequences processing methods. Although these models realized heteroassociative memory for complex temporal sequences, the memory length k still is decided by designer which cannot vary in self-adaption fashion. Moreover, These models are unable to handle complex sequence with looped hidden state.

The rest of this paper is organized as follows: In Section 2, we introduce the problem setup, present the theoretical analysis for a minimal, looping hidden state transition model, and derive a heuristic constructing algorithm for this model. In Section 3, STAMN model is analyzed in detail, including its short-term memory (STM), long-term memory (LTM), and the heuristic constructing process. In Section 4, we present detailed simulations and analysis of the STAMN model and compare the performance of the STAMN model with that of other methods. Finally, a brief discussion and conclusion are given in Sections 5 and 6 separately.

2. Problem Setup

A deterministic POMDP environment can be represented by a tuple $E = \langle S, A, O, f_s, f_o \rangle$, where S is the finite set of hidden world states, A is the set of actions that can be taken by the agent, O is the set of possible observations, f_s is a deterministic transition function $f_s(S, A) \to S$, and f_o is a deterministic observation function $f_o(S, A) \to O$. In this thesis, we only consider the special observation function $f_o(S) \to O$ that solely depends on the state s. A history sequence h is defined as a sequence of past observations and actions $\{o_1, a_1, o_2, a_2, \ldots, a_{t-1}, o_t\}$, which can be generated by the deterministic transition function f_s and the deterministic observation function f_o. The length $|h|$ of a history sequence is defined as the number of observations in this history sequence.

The environment that we discuss is deterministic and the state space is finite. We also assume the environment is strongly connected. However the environment is deterministic, which can be highly complicated, and nondeterministic at the level of observation. The hidden state can be fully identified by a finite history sequence in a deterministic POMDP, which is proved in [2]. Several notations are defined as follows:

$\text{trans}(h, a) = \{o \mid o$ is a possible observation following h by taking action $a\}$.

$\gamma(s_i, d_{ij})$ denotes the observations sequence $o_1, o_2, \ldots, o_n$ generated by taking actions sequence d_{ij} from state s_i.

Our goal is to construct a minimal, looping hidden state transition model by use of the sufficient history sequences. First we present the theoretical analysis showing any deterministic POMDP environment can be represented by a minimal, looping hidden state transition model. Then we present a heuristic constructing algorithm for this model. Corresponding definitions and lemmas are proposed as follows.

Definition 1 (identifying history sequence). A identifying history sequence h_i is considered to uniquely identify the hidden state s_i. In the rest of this paper, the hidden state s_i is equally regarded as its identifying history sequence h_i, so s_i and h_i can replace each other.

A simple example of deterministic POMDP is illustrated in Figure 1. We conclude easily that an identifying history

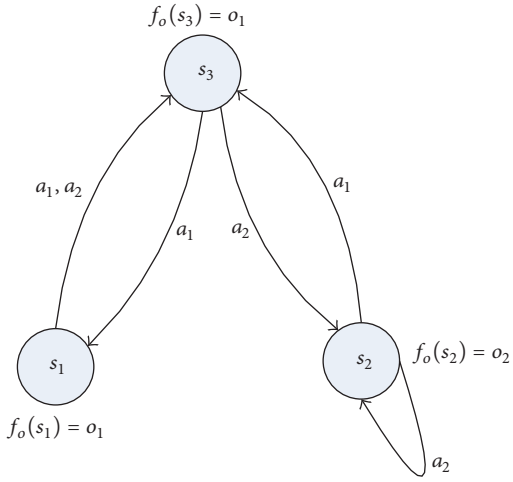

Figure 1: A simple example of deterministic POMDP.

sequence h_2 for s_2 is expressed by $\{o_2\}$ and an identifying history sequence h_1 for s_1 is expressed by $\{o_2, a_1, o_1, a_1, o_1\}$. Note that there may exist infinitely many identifying history sequences h_i for s_i because the environment is strongly connected and may exist as unbounded long identifying history sequences h_i for s_i because of uninformative looping. So this leads us to determine the minimal identifying history sequences length k.

Definition 2 (minimal history sequences length k). The minimal identifying history sequences length k for the hidden state s_i is defined as follows: The minimal identifying history sequence h for the hidden state s_i is the identifying history sequence h such that no suffix h' of h is also identified for s_i. However, the minimal identifying history sequence may have unbounded length, because the identifying history sequence can include looping arbitrarily many times, which merely lengthens the history sequence. For example, two identifying history sequences h_1 and h_1' for s_1 are separately expressed to $\{o_2, a_1, o_1, a_1, o_1\}$ and $\{o_2, a_1, o_1, a_1, o_1, a_1, o_1, a_1, o_1\}$. In this situation, $\{o_1, a_1, o_1, a_1, o_1\}$ is treated as the looped item. So the minimal identifying history sequences length k for the hidden state s_i is the minimal length value of all identifying history sequences by excising the looping portion.

Definition 3 (a criterion for identifying history sequence). Given a sufficient history sequence, for the set of history sequences for the hidden state s, if each history sequence h for the hidden state s exists, which satisfies that trans(h, a) is the same observation for each action, thus we consider the history sequence h as identifying history sequence. Definition 3 is correct iff a full sufficient history sequence is given.

Lemma 4 (backward identifying history sequence). *We assume an identifying history sequence h is for a single hidden state s, if h' is a backward extension of h by adding an action $a \in A$ and an observation $o \in O$; then h' is a new identifying history sequence for the single state $s' = f_s(s, a)$. s' is called the prediction state of s.*

Proof. We assume h' is a history sequence $\{o_1, a_1, \ldots, a_{t-1}, o_t\}$ generated by the transition function f_s and observation function f_o. And h is a history sequence $\{o_1, a_1, \ldots, a_{t-2}, o_{t-1}\}$ as a prefix of h'. Since h is identifying history sequence for s, it must be that $s_{t-1} = s$. Because the environment is deterministic POMDP, thus, the next state s' is uniquely determined by $f_s(s_{t-1}, a_{t-1}) = s_t$ after the action a_{t-1} has been executed. Then h' is a new identifying history sequence for the single state $s_t = f_s(s_{t-1}, a_{t-1}) = f_s(s, a_{t-1}) = s'$. □

Lemma 5 (forward identifying history sequence). *We assume an identifying history sequence h' is for a single hidden state s', if h is a prefix of h' by removing the last action $a \in A$ and the last observation $o \in O$; then h is the identifying history sequence for the single state s such that $s' = f_s(s, a)$. s is called the previous state of s'.*

Proof. We assume h' is a history sequence $\{o_1, a_1, \ldots, a_{t-1}, o_t\}$ generated by the transition function f_s and observation function f_o. And h is a history sequence $\{o_1, a_1, \ldots, a_{t-2}, o_{t-1}\}$ as a prefix of h by removing the action a_{t-1} and the observation o_t. Since h' is an identifying history sequence for the state s', it must be that $s_t = s'$. Because the environment is deterministic POMDP, thus, the current state s' is uniquely determined by the previous state s such that $f_s(s_{t-1}, a_{t-1}) = s'$. Then h is the identifying history sequence for the state s such that $f_s(s_{t-1}, a_{t-1}) = f_s(s, a_{t-1}) = s'$. □

Theorem 6. *Given the sufficient history sequences, a finite, strongly connected, deterministic POMDP environment can be represented soundly by a minimal, looping hidden state transition model.*

Proof. We assume $E = \langle S, A, O, f_s, f_o \rangle$, where S is the finite set of hidden world states and $|S|$ is the number of the hidden states. First, for all hidden states, at least a finite length identifying history sequence h for one state s exists; because the environment is strongly connected, there must exist a transition history h' from s to s', according to Lemma 4, by backward extending of h by adding h', which is the identifying history sequence for s'. Since the hidden state transition model can identify the looping hidden state, there exists the maximal transition history sequence length from s to s' which is $|S|$. Thus, this model has minimal hidden state space with $|S|$.

Since the hidden state transition model can correctly realize the hidden state transition $s' = f_s(s, a)$, this model can correctly express all identifying history sequences for all hidden state s and possesses the perfect transition prediction capacity, and this model has minimal hidden state space with $|S|$. This model is a k-step variable history length model, and the memory depth k is a variable value for the different hidden state. This model is realized by the spatiotemporal associative memory networks in Section 3.

If we want to construct correct hidden state transition model, first we necessary to collect sufficient history

sequences to perform statistic test to determine the minimal identifying history sequence length k. However, in practice, it can be difficult to obtain a sufficient history. So we propose a heuristic state transition model constructing algorithm. Without the sufficient history sequence, the algorithm may produce a premature state transition model, but this model at least is correct for the past experience. We realize the heuristic constructing algorithm by way of two definitions and two lemmas as follows. □

Definition 7 (current history sequence). A current history sequence is represented by the k-step history sequence $\{o_{t-k+1}, a_{t-k+1}, o_{t-k+2}, \ldots, a_{t-1}, o_t\}$ generated by the transition function f_s and the observation function f_o. At $t = 0$, the current history sequence h_0 is empty. o_t is the observation vectors at current time t, and o_{t-k+1} is the observation vectors that precede k-step at time t.

Definition 8 (transition instance). The history sequence associated with time t is captured as a transition instance. The transition instance is represented by the tuple $h_{t+1} = (h_t, a_t, o_{t+1})$, where h_t and h_{t+1} are current history sequences occurring at times t and $t + 1$ on episode. A set of transition instances will be denoted by the symbol F, which possibly contains transitions from different episodes.

Lemma 9. *For any two hidden states s_i and s_j, $s_i \neq s_j$ iff* $\text{trans}(s_i, a) \neq \text{trans}(s_j, a)$.

This lemma gives us a sufficient condition to determine $s_i \neq s_j$. However, this lemma is not a necessary condition.

Proof by Contradiction. We assume that $s_i = s_j$; thus for all actions sequences $d_{ij} = a_1, a_2, \ldots, a_n$, there exists $\gamma(s_i, d_{ij}) = \gamma(s_j, d_{ij})$. $\gamma(s_i, d_{ij}) = \gamma(s_j, d_{ij})$ is contradicting with $\text{trans}(s_i, a) \neq \text{trans}(s_j, a)$. Thus, original proposition is true.

Lemma 10. *For any two identifying history sequences h_i and h_j separately for s_i and s_j, there exists $h_i \neq h_j$ iff $s_i \neq s_j$. However, Lemma 10 is not a necessary condition.*

Proof by Contradiction. Since an identifying history sequence h is considered to uniquely identify the hidden state s, the identifying history sequences h_i uniquely identify the hidden state s_i and the identifying history sequences h_j uniquely identify the hidden state s_j. We assume $h_i = h_j$; thus there must exit $s_i = s_j$, which is contradicting with $s_i \neq s_j$. Thus, original proposition is true.

However, the necessary condition for Lemma 10 is not always true, because there maybe exist several different identifying history sequences for the same hidden state.

Algorithm 11 (a heuristic constructing algorithm for the state transition model). The initial state transition model is constructed making use of the minimal identifying history sequence length $k = 1$. And the model is empty initially. The transition instance h is empty, and $t = 0$:

(1) We assume the given model can perfectly identify the hidden state. The agent makes a step in the environment (according to history sequence definition, the first item in h is the observation vector). It records the current transition instance on the end of the chain of transition instance. And at each time t, for the current history sequence h_t, execute Algorithm 12, if the current state s is looped to the hidden state s_i; then go to step (2); otherwise a new node s_i is created in the model; go to step (4).

(2) According to Algorithm 13, if $\text{trans}(h_t, a_j) \neq \text{trans}(h_i, a_j)$, then the current state s is distinguished from the identifying sate s_i; then go to step (3). If there exists $\text{trans}(h_t, a_j) = \text{trans}(h_i, a_j)$, then go to step (4).

(3) According to Lemma 10, for any two identifying history sequences h_i and h_j separately for s_i and s_j, there exists $h_i \neq h_j$ iff $s_i \neq s_j$. So the identifying history sequence length for h_i and h_j separately for s_i and s_j must increase to $k+1$ until h_i and h_j are discriminated. We must reconstruct the model based on the new minimal identifying history sequence length k, and go to step (1).

(4) If the action node and corresponding Q function for the current identifying state node exist in the model, the agent chooses its next action node a_j based on the exhaustive exploration Q function. If the action node and corresponding Q function do not exist, the agent chooses a random action a_j instead, and a new action node is created in the model. After the action control signals to delivery, the agent obtains the new observation vectors by $\text{trans}(s_i, a_j)$; go to step (1).

(5) Steps (1)–(4) continue until all identifying history sequences h for the same hidden state s can correctly predict the $\text{trans}(h, a)$ for each action.

Note, we apply the k-step variable history length from $k = 1$ to n, and the history length k is a variable value for the different hidden state. We adopt the minimalist hypothesis model in constructing the state transition model process. This constructing algorithm is a heuristic. If we adopt the exhaustiveness assumption model, the probability of missing looped hidden state increases exponentially, and many valid loops can be rejected, yielding larger redundant state and poor generalization.

Algorithm 12 (the current history sequence identifying algorithm). In order to construct the minimal looping hidden state transition model, we need to identify the looped state by identifying hidden state. Current history sequence with k-step is needed. According to Definition 7, current k-step history sequence at time t is expressed by $\{o_{t-k+1}, a_{t-k+1}, o_{t-k+2}, \ldots, a_{t-1}, o_t\}$. There exist three identifying processes.

(1) k-Step History Sequence Identifying. If the identifying history sequence for s_i is $\{o_{i-k+1}, a_{i-k+1}, o_{i-k+2}, \ldots, a_{i-1}, o_i\}$, satisfy

$$\begin{aligned} o_t &= o_i, \\ a_{t-1} &= a_{i-1}, \\ &\vdots \\ o_{t-k+2} &= o_{i-k+2}, \\ a_{t-k+1} &= a_{i-k+1}, \\ o_{t-k+1} &= o_{i-k+1}. \end{aligned} \tag{1}$$

Thus, the current state s is looped to the hidden state s_i. In the STAMN, the k-step history sequence identifying activation value $m_j^s(t)$ is computed.

(2) Following Identifying. If the current state s is identified as the hidden state s_i, the next transition history h_{t+1} is represented by $h_{t+1} = (h_t, a_t, o_{t+1})$ through trans(s_i, a_t). According to Lemma 4, the next transition history h_{t+1} is identified as the transition prediction state $s_j = f_s(s_i, a_t)$.

In the STAMN, the following activation value $q_j^s(t)$ is computed.

(3) Previous Identifying. If there exists a transition instance $h_{t+1} = (h_t, a_t, o_{t+1})$, h_{t+1} is an identifying history sequence for state s_j. According to Lemma 5, the previous state s_i is uniquely identified by h_t such that $s_j = f_s(s_i, a_t)$. So if there exists a transition prediction state s_j, and $s_j = f_s(s_i, a_t)$, then the previous state s_i is uniquely identified.

In the STAMN, the previous activation value $d_j^s(t)$ is computed.

Algorithm 13 (transition prediction criterion). If the model correctly represents the environment model as Morkov chains, then, for all state s, for all identifying history sequence h for the same state, it is satisfied that trans(h, a) is the same observation for the same action.

If h_1 and h_2 are the identifying history sequences with k-step memory for the hidden state s, there exist trans$(h_1, a) \neq$ trans(h_2, a), according to Lemma 9; thus s_i is discriminative with s_j.

In the STAMN, the transition prediction criterion is realized by the transition prediction value $P_j^s(t)$.

3. Spatiotemporal Associative Memory Networks

In this section, we want to relate the spatiotemporal sequence problem to the idea of identifying the hidden state by a sequence of past observations and actions. An associative memory network (AMN) is expected to have several characteristics: (1) An AMN must memorize incrementally, which has the ability to learn new knowledge without forgetting the learned knowledge. (2) An AMN must be able to not only record the temporal orders but also record sequence items duration time with continuous time. (3) An AMN must be able to realize the heteroassociation recall. (4) An AMN must be able to process the real-valued feature vectors in bottom-up method, not the symbolic items. (5) An AMN must be robust and must be able to recall the sequence correctly with incomplete or noisy input. (6) An AMN can realize learning and recalling simultaneously. (7) An AMN can realize interaction between STM and LTM; dual-trace theory suggested that the persistent neural activities of STM can lead to LTM.

The first thing to be defined is the temporal dimension (discrete or continuous). The previous researches are mostly based on a regular intervals of Δt, and few AMN have been proposed to deal with not only the sequential characteristic but also sequence items duration time with continuous time. However, this characteristic is important for many problems. In speech production, writing, music generation and motor-planning, and so on, the sequence item duration time and sequence item repeated emergence have essential different meaning. For example, "B^4" and "*B-B-B-B*" are exactly different, the former represents that item B sustains for 4 timesteps, and the latter represents that item B repeatedly emerges for 4 times. STAMN can explicitly distinguish these two temporal characteristics. So a history of past observations and actions can be expressed by a special spatiotemporal sequence h. $h = \{o_{t-k+1}, a_{t-k+1}, o_{t-k+2}, \ldots, a_{t-1}, o_t\}$, where k is the length of sequence of h; the items of h include the observation items and the action items, where o_t denotes the real-valued observation vectors generated by taking action a_{t-1}. a_t denotes the action taken by the agent at time t, where t does not represent temporal discrete dimension by sampling at regular intervals Δt but represents the tth step in the continuous-time dimension, which is the duration time between the current item and the next one.

A spatiotemporal sequence can classified as simple sequence and complex sequence. Simple sequence is a sequence without repeated items; for example, the sequence "*B-A-E-L*" is a simple sequence, whereas those containing repeated items are defined as the complex sequence. In complex sequence, the repeated item can be classified as looped items and discriminative items by identifying the hidden state, for example, in the history sequence "*X-Y-Z-Y-Z*", "*Y*," and "*Z*" maybe the looped items or discriminative items. Identifying the hidden state needs to introduce the contextual information resolving by the k-step memory. The memory depth k is not fixed and k is a variable value in different parts of state space.

3.1. Spatiotemporal Associative Memory Networks Architecture. We build a new spatiotemporal associative memory network (STAMN). This model makes use of neuron activity decay of nodes to achieve short-term memory, connection weights between different nodes to represent long-term memory, presynaptic potentials and neuron synchronized activation mechanism to realize identifying and recalling, and a time-delayed Hebb learning mechanism to fulfil the one-shot learning.

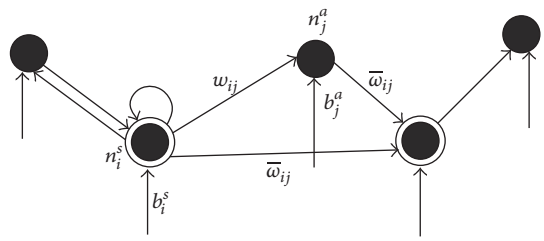

Figure 2: STAMN structure diagram.

STAMN is an incremental, possibly looping, nonfully connected, asymmetric associative memory network. Non-homogeneous nodes correspond to hidden state nodes and action nodes.

For the state node j, the input values are defined as follows: (1) the current observation activation value $b_j^s(t)$, responsible for matching degree of state node j, and current observed value, which is obtained from the preprocessing neural networks (if the matching degree is greater than a threshold value, then $b_j^s(t) = 1$); (2) the observation activation value $b_j^s(t)$ of the presynaptic nodes set τ_j of current state node j; (3) the identifying activation value $n_i^s(t)$ of the previous state node i of state node j; (4) the activation value $n_i^a(t)$ of the previous action node i of state node j. The output values are defined as follows: (1) The identifying activation value $n_j^s(t)$ represents whether the current state s is identified to the hidden state s_j or not. (2) The transition prediction value $p_j^s(t)$ represents whether the state node j is the current state transition prediction node or not.

For the action node j, the input values are defined as the current action activation value $b_j^a(t)$, responsible for matching degree of action node j and current motor vectors. The output value is activation value of the action nodes $n_j^a(t)$ and indicates that the action node j has been selected by agent to control the robot's current action.

For the STAMN, all nodes and connections weight do not necessarily exist initially. The weights $\omega_{ij}(t)$ and $\overline{\omega}_{ij}(t)$ connected to state nodes can be learned by a time-delayed Hebb learning rules incrementally representing the LTM. The weight $w_{ij}(t)$ connected to action nodes can be learned by reinforcement learning. All nodes have activity self-decay mechanism to record the duration time of this node representing the STM. The output of the STAMN is the winner state node or winner action node by winner takes all.

STAMN architecture is shown in Figure 2, where black nodes represent action nodes and concentric circles nodes represent state nodes.

3.2. Short-Term Memory. We using self-decay of neuron activity to accomplish short-term memory, no matter whether observation activation $b_i^s(t)$ or identifying activation $n_i^s(t)$. The activity of each state node has self-decay mechanism to record the temporal order and the duration time of this node. Supposing the factor of self-decay is $\gamma_i^s \in (0,1)$ and the activation value $b_i^s(t), n_i^s(t) \in [0,1]$ the self-decay process is shown as

$$b_i^s(t), n_i^s(t) = \begin{cases} 0 & b_i^s(t), n_i^s(t) \leqslant \eta \\ \gamma_i^s b_i^s(t-1), \gamma_i^s n_i^s(t-1) & \text{else.} \end{cases} \quad (2)$$

The activity of action node also has self-decay characteristic. Supposing the factor of self-decay is $\delta_i^a \in (0,1)$ and the activation value $b_i^a(t), n_i^a(t) \in [0,1]$ the self-decay process is shown as

$$b_i^a(t), n_i^a(t) = \begin{cases} 0 & b_i^a(t), n_i^a(t) \leqslant \mu \\ \delta_i^a b_i^a(t-1), \delta_i^a n_i^a(t-1) & \text{else,} \end{cases} \quad (3)$$

wherein η and μ are active thresholds and γ_i^s and δ_i^a are self-decay factors. Both determine the depth of short-term memory k, where t is a discrete time point by sampling at regular intervals Δt, and Δt is a very small regular interval.

3.3. Long-Term Memory. Long-term memory can be classified into semantic memory and episodic memory. Learning of history sequence is considered as the episodic memory, generally adopting the one-shot learning to realize. We use the time-delayed Hebb learning rules to fulfil the one-shot learning.

(1) The k-Step Long-Term Memory Weight $\omega_{ij}(t)$. The weight $\omega_{ij}(t)$ connected to state nodes j represents k-step long-term memory. This is a past-oriented behaviour in the STAMN. The weight $\omega_{ij}(t)$ is adjusted according to

$$\omega_{ij}(t) \underset{\substack{i \in \tau_j \\ n_j^s(t)=1}}{=} \begin{cases} n_j^s(t)\, n_i^X(t) & \omega_{ij}(t-1) = 0,\ X \in \{s,a\} \\ \omega_{ij}(t-1) + \alpha\left(n_i^X(t) - \omega_{ij}(t-1)\right) & \text{else,} \end{cases} \quad (4)$$

where j is the current identifying activation state node, $n_j^s(t) = 1$. Because the identifying activation process is a neuron synchronized activation process, when $n_j^s(t) = 1$, all nodes whose $n_i^s(t)$ and $n_i^a(t)$ are not zero are the contextual information related to state node j. These nodes whose $n_i^s(t)$ and $n_i^a(t)$ are not zero are considered as the presynaptic node

set of current state node j. The presynaptic node set of current state node j is expressed by τ_j, where $n_i^s(t), n_i^a(t)$ represent the activation value at time t, and $n_i^s(t), n_i^a(t)$ record not only the temporal order but also the duration time because of the self-decay of neuron activity. If $n_i^s(t)$, $n_i^a(t)$ are smaller, the node i is more early to current state node j. $\omega_{ij}(t)$ is activation weight between presynaptic node i and state node j. $\omega_{ij}(t)$ records context information related to state node j to be used in identifying and recalling, where $\omega_{ij}(t) \in [0, 1]$, $\omega_{ij}(0) = 0$, and α is learning rate.

The weight $\omega_{ij}(t)$ is time-related contextual information; the update process is shown as in Figure 3.

(2) One-Step Transition Prediction Weight $\overline{\omega}_{ij}(t)$. The weight $\overline{\omega}_{ij}(t)$ connected to state nodes j represents one-step transition prediction in LTM. This is a future-oriented behaviour in the STAMN. Using the time-delayed Hebb learning rules, the weight $\overline{\omega}_{ij}(t)$ is adjusted according to

$$\underset{\substack{i=\arg\max n_i^X(t)\\ n_j^s(t)=1}}{\overline{\omega}_{ij}(t)} = \begin{cases} n_j^s(t)\, n_i^X(t) & \overline{\omega}_{ij}(t-1) = 0,\ X \in \{s, a\} \\ \overline{\omega}_{ij}(t-1) + \beta\left(n_i^X(t) - \overline{\omega}_{ij}(t-1)\right) & \text{else.} \end{cases} \tag{5}$$

The transition activation of current state node j is only associated with the previous winning state node and action node, where j is the current identifying activation state node, $n_j^s(t) = 1$. The previous winning state node and action node are presented by $i = \arg\max_i^X(t)$, $X \in \{s, a\}$, where $\overline{\omega}(t) \in [0, 1]$ and $\overline{\omega}(0) = 0$, β is learning rate.

The weight $\overline{\omega}_{ij}(t)$ is one-step transition prediction information; the update process is shown as in Figure 4.

(3) The Weight $w_{ij}(t)$ Connected to Action Nodes. The activation of action node is only associated with the corresponding state node which selects this action node directly. We assume the state node with maximal identifying activation value is the state node i at the time t, so the connection weight $w_{ij}(t)$ connected to current selected action node j is adjusted by

$$\underset{\substack{i=\arg\max n_i^s(t)\\ n_j^a(t)=1}}{w_{ij}(t)} = \begin{cases} Q_0\left(s_i, a_j\right) & w_{ij}(t-1) = 0 \\ -Q\_\max\_\text{const} & a_j \text{ is a invalid action} \\ Q_t\left(s_i, a_j\right) & \text{else,} \end{cases} \tag{6}$$

where $w_{ij}(t)$ is represented by Q function $Q_t(s_t, a_j)$. This paper only discusses how to build generalized environment model, not learning the optimal policy, so this value is set to be the exhaustive exploration Q function based on the curiosity reward. The curiosity reward is described by (7). When action a_j is an invalid action, $w_{ij}(t)$ is defined to be a large negative constant, which is avoided going to the dead ends

$$r(t) = \begin{cases} C - \dfrac{n_t}{n_{\text{ave}}} & n_{\text{ave}} > 0 \\ C & n_{\text{ave}} = 0, \end{cases} \tag{7}$$

where C is a constant value, n_t represents the count of exploration to the current action by the agent, and n_{ave} is the average count of exploration to all actions by the agent; n_t/n_{ave} represents the degree of familiarity with the current selected action. The curiosity reward is updated when each action is finished. $Q_t(s_i, a_j)$ update equation is showed by (8), where λ is learning rate

$$Q_t\left(s_i, a_j\right) = \begin{cases} (1-\lambda)\, Q_{t-1}\left(s_i, a_j\right) + \lambda r(t) & i = \arg\max n_i^s(t),\ n_j^a(t) = 1 \\ Q_{t-1}\left(s_i, a_j\right) & \text{else.} \end{cases} \tag{8}$$

The update process is shown as in Figure 5.

The action node j is selected by $Q_t(s_t, a_j)$ according to

$$a_j = \underset{\substack{j \in A_i^{\text{vl}}\\ n_i^s(t)=1}}{\arg\max} Q\left(s_i, a_j\right), \tag{9}$$

where A_i^{vl} is the valid action set of state node i. $n_i^s(t) = 1$ represent that the sate node i is identified, and the action node j was selected by agent to control the robot's current action; set $n_i^a(t) = 1$.

3.4. The Constructing Process in STAMN. In order to construct the minimal looping hidden state transition model, we need to identify the looped state by identifying hidden state. There exist identifying phase and recalling phase (transition prediction phase) simultaneously in the constructing process in STAMN. There is a chicken and egg problem during the constructing process: the building of the STAMN depends on state identifying; conversely, state identifying depends on the current structure of the STAMN. Thus, exhaustive exploration and k-step variable memory length (depends on the prediction criterion) are used to try to avoid local minima that this interdependent causes.

According to Algorithm 12, The identifying activation value of state node s_j depends on three identifying processes: k-step history sequence identifying, following identifying, and previous identifying. We provide calculation equations for each identifying process.

(1) k-Step History Sequence Identifying. The matching degree of current k-step history sequence with the identifying

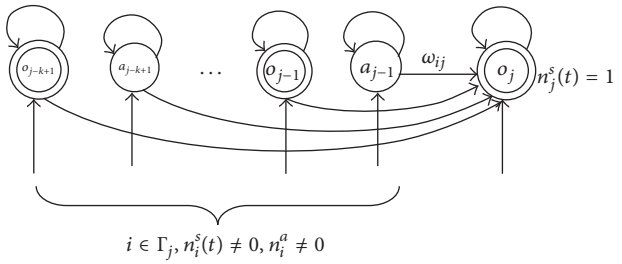

Figure 3: Past-oriented weights $\omega_{ij}(t)$ update process associated with state node j.

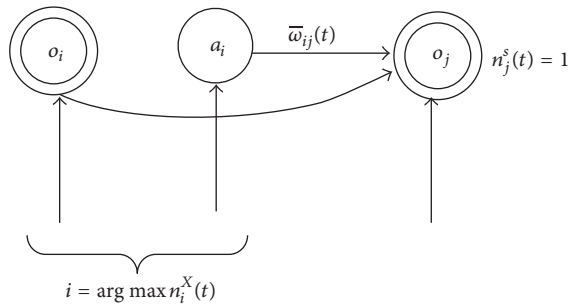

Figure 4: Future-oriented weights $\overline{\omega}_{ij}(t)$ update process associated with state node j.

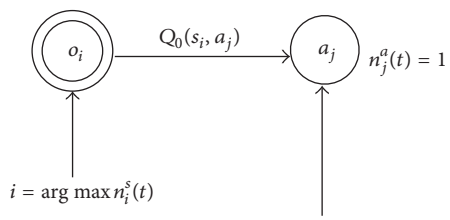

Figure 5: Weights $w_{ij}(t)$ updating process associated with action node j.

history sequence h_j for s_j is computed to identify the looping state node j. First, we compute the presynaptic potential $V_j(t)$ for the state node j according to

$$V_j(t) = \sum_{i \in \tau_j} C_{ij} \varnothing \left(b_i^X(t), \omega_{ij} \right) \quad X \in \{s, a\}, \tag{10}$$

where $b_i^X(t)$ is the current activation value in the presynaptic node set τ_j. C_{ij} is the confidence parameter which means the node i's importance degree in the presynaptic node set τ_j. The value of C_{ij} can be set in advance, and $\sum_{i \in \tau_j} C_{ij} = 1$. The function $\varnothing$ represents the similar degree between activation value of the presynaptic node and the contextual information ω_{ij} in LTM. $\varnothing(x, y) \in [0, 1]$; the similar degree is high between x and y; thus $\varnothing(x, y)$ is close to 1. According to "winner takes all," among all nodes whose presynaptic potentials exceed the threshold ϑ, the node with the largest presynaptic potential will be selected.

The presynaptic potential $V_j(t)$ represents the synchronous activation process of presynaptic node set of τ_j, which represents the previous $k-1$ step contextual information matching of state node j. To realize all k-step history sequence matching, the k-step history sequence identifying activation value of the state node j is given below:

$$\begin{aligned} m_j^s(t) &= H\left(V_j(t), \vartheta\right) \\ &\quad \cdot H\left(V_j(t), \max_{k:1 \to N^c}\left(V_k(t)\right)\right) b_j^s(t) \\ H(x, y) &= \begin{cases} 0 & x < y \\ 1 & x \geq y, \end{cases} \end{aligned} \tag{11}$$

where $\max_{k:1 \to N^c}(V_k(t))$ is the maximum potential value of node k. $H(V_j(t), \vartheta) \cdot H(V_j(t), \max_{k:1 \to N^c}(V_k(t)))$ means the node with the largest potential value is selected among all nodes whose synaptic potentials exceed the threshold ϑ. $b_j^s(t)$ represents the matching degree of state node j and current observed value. If $m_j^s(t) = 1$, the current state s is identified to looped state node j by the k step memory.

(2) Following Identifying. If the current state s is identified as the state s_i, then the next transition prediction state is identified as the state $s_j = f_s(s_i, a_i)$. First, we compute the transition prediction value for the state node j according to

$$p_j^s(t) = \sum_{i = \arg\max n_i^X(t)} D_{ij} \phi\left(n_i^X(t), \overline{\omega}_{ij}\right) \quad X \in \{s, a\}, \tag{12}$$

where $n_i^X(t)$ is the identifying activation value of the state node and action node at time t. $i = \arg\max n_i^X(t)$ indicates state node i and action node i are the previous winner nodes. D_{ij} is the confidence parameter which means the node i's importance degree. The value of D_{ij} can be set in advance, and $\sum_{i = \arg\max n_i^X(t)} D_{ij} = 1$. $\overline{\omega}_{ij}$ records one-step transition prediction information related to state node j to be used in identifying and recalling phase. If the current state s is identified as the hidden state s_i, $p_j^s(t)$ represent the probability of the next transition prediction state is j.

If the next prediction state node j is the same as the current observation value, the current history sequence is identified as the state node j. the following identifying value of the state node j is given below:

$$\begin{aligned} q_j^s(t) &= H\left(p_j^s(t), \vartheta\right) \\ &\quad \cdot H\left(p_j^s(t), \max_{k:1 \to N^c}\left(p_k^s(t)\right)\right) b_j^s(t). \end{aligned} \tag{13}$$

If $q_j^s(t) = 1$, the current history sequence h_t is identified to looped state node j by the following identifying. If $p_j^s(t) \geq \max_{k:1 \to N^c}(p_k^s(t))$, $p_j^s(t) \geq \vartheta$, and $q_j^s(t) \neq 1$, then there exists $b_j^s(t) \neq 1$, representing mismatching of state node j and current observed value. There exist $\text{trans}(h_t, a_i) \neq \text{trans}(h_i, a_i)$. According to transition prediction criterion (Algorithm 13), the current history sequence h_t is not identified by the hidden

sate s_i, so the identifying history sequence length for h_t and h_i separately must increase to $k+1$.

(3) Previous Identifying. If the current history sequence is identified as the state s_j, then the previous state s_i is identified such that $s_j = f_s(s_i, a_i)$. First, we compute the identifying activation value for all states s. if there exists $n_j^s(t) = 1$, then the current state s_t is identified as state s_j, and the previous state s_{t-1} is identified as the previous state s_i of state s_j. The previous identifying value of the state node j is defined as $d_i^s(t)$. The previous state s_i is $i = \arg\max(b_i^s(t))$ satisfying condition 1 and condition 2. Then we set $d_i^s(t)$:

Condition 1:

$$n_j^s(t) = 1 \tag{14}$$

Condition 2:

$$\phi\left(b_i^s(t), \overline{\omega}_{ij}\right) > \vartheta \tag{15}$$

According to above three identifying processes, the identifying activation value of the state node j is defined as follows:

$$n_j^s(t) = \max\left(m_j^s(t), q_j^s(t), d_j^s(t)\right). \tag{16}$$

According to Algorithm 11, we give Pseudocode 1 for Algorithm 11.

The pseudocode for Algorithm 12 is as follows: the current history sequence identifying algorithm (identifying phase). Compute the identifying activation value according to (16).

The pseudocode for Algorithm 13 is in Pseudocode 2.

4. Simulation

4.1. A Simple Example of Deterministic POMDP. We want to construct the looping transition model as in Figure 1. First, we obtain the transition instance h_t : $\{o_1, a_1, o_1, a_2, o_2, a_2, o_2, a_1, o_1, a_1, o_1\}$. According to Algorithm 11, set the initial identifying history sequence length $k = 1$. The STAMN model is constructed incrementally using h_t as in Figure 6.

When $h_t = \{o_1, a_1, o_1, a_2, o_2, a_2, o_2, a_1, o_1, a_1, o_1, a_2, o_1\}$, because of $k = 1$, the looped state is identified by the current observation vector. Thus $h_1 = \{o_1\}$ and $h_2 = \{o_1\}$ are the identifying history sequences for the state s_1. Since $o_1 = \text{tans}(h_1, a_2)$ and $o_2 = \text{tans}(h_2, a_2)$ in h_t exist, so as to $\text{trans}(h_1, a_2) \neq \text{trans}(h_2, a_2)$, according to Lemma 9, h_1 and h_2 are the identifying history sequences for the states s_i and s_j, respectively, and $s_i \neq s_j$. So o_1 is distinguished as o_1' and o_1''. And $h_t = \{o_1, a_1, o_1', a_2, o_2, a_2, o_2, a_1, o_1, a_1, o_1'', a_2, o_1\}$.

According the Lemma 10, for any two identifying history sequences h_i and h_j separately for o_1' and o_1'', there exists $h_i \neq h_j$ iff $s_i \neq s_j$. So the identifying history sequence length for h_i and h_j separately for s_i and s_j must increase to 2. For o_1', the identifying history sequence is $\{o_1, a_1, o_1'\}$, and for o_1'', the identifying history sequence is $\{o_1, a_1, o_1''\}$. These two identifying history sequences are identical, so longer transition instance is needed.

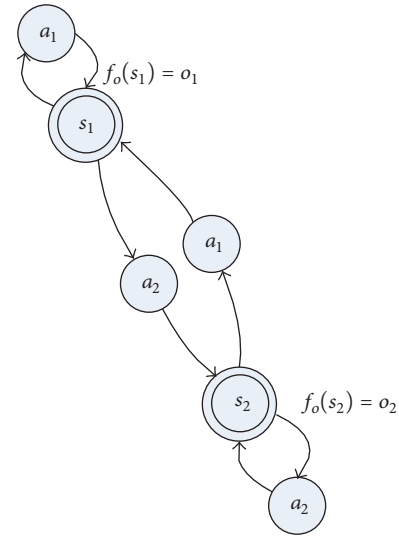

Figure 6: The STAMN model with memory depth 1.

Table 1: The identifying history sequences for o_1' and o_1''.

Hidden state	o_1' $(k=2)$	o_1'' $(k=3)$
Identifying history sequence	o_1, a_1, o_1' o_1'', a_2, o_1' o_2, a_1, o_1'	$o_2, a_1, o_1, a_1, o_1''$

When $h_t = \{o_1, a_1, o_1', a_2, o_2, a_2, o_2, a_1, o_1, a_1, o_1'', a_2, o_1', a_2, o_2, a_1, o_1', a_2, o_2, a_1, o_1, a_1, o_1, a_1, o_1', a_2, o_2\}$. In h_t, we obtain the identifying history sequence $h_i \neq h_j$ for states o_1' and o_1''. For $k = 2$, the distinguished identifying history sequence for o_1' is $\{o_1'', a_2, o_1'\}$ and $\{o_2, a_1, o_1'\}$. However, the distinguished identifying history sequence for o_1'' is $\{o_1, a_1, o_1''\}$, which is not a discriminated history sequence from the identifying history sequence for o_1'. So the identifying history sequence length for o_1'' must increase to 3. The identifying history sequences for o_1' and o_1 are described as in Table 1.

According to k-step history identifying, following identifying and previous identifying, h_t can be represented as $h_t = \{o_1'', a_1, o_1', a_2, o_2, a_2, o_2, a_1, o_1', a_1, o_1'', a_2, o_1', a_2, o_2, a_1, o_1', a_2, o_2, a_1, o_1', a_1, o_1'', a_1, o_1', a_2, o_2\}$.

According to Pseudocode 1, the STAMN model is constructed incrementally using h_t as in Figure 7(a), and the LPST is constructed as Figure 7(b). The STAMN has the fewest nodes because the state nodes in STAMN represent the hidden state, and the state nodes in LPST represent the observation state.

To illustrate the difference between the STAMN and LPST, we present the comparison results in Figure 8.

After 10 timesteps, the algorithms use the current learned model to realize the transition prediction, and the transition prediction criterion is expressed by the prediction error. The data point represents the average prediction error over 10 runs. We ran three algorithms: the STAMN with exhaustive exploration, the STAMN with random exploration, and the LPST.

Set initial memory depth $k = 1$; $t = 0$;
All nodes and connections weight do not exist initially in STAMN;
The transition instance $h_0 = \phi$;
A STAMN is constructed incrementally through interaction with the environment by the agent, expanding the SATAMN until the transition prediction is contradicted with the current minimal hypothesis model, reconstruct the hypothesis model by increase the memory depth k. in this paper, the constructing process includes the identifying and recalling simultaneously.
While one pattern o_t or a_t can be activated from the pattern cognition layer **do**
$h_t = (h_{t-1}, o_t, a_t)$
for all existed state nodes
compute the identifying activation value by executing Algorithm 12
If exist the $n_j^s(t) = 1$ **then**
the looped state is identified, and the weight $\omega_{ij}\overline{\omega}_{ij}$, is adjusted according to (4), (5)
else
a new state node j will be created, set $n_j^s(t) = 1$. and the weight $\omega_{ij}\overline{\omega}_{ij}$, is adjusted according to (4), (5)
end if
end for
for all existed state nodes
compute the transition prediction criterion by executing Algorithm 13.
If IsNotTrans(s_j) **then**
for the previous winner state node i of the state node j
set memory depth $k = k + 1$ until each h_i are discriminated
end for
reconstruct the hypothesis model by the new memory depth k according to Algorithm 11
end if
end for
for all existed action nodes
compute the activation value according to (9).
if exist the $n_j^a(t) = 1$ **then**
the weight $w_{ij}(t)$ is adjusted according to (6).
else
a new action node j will be created, set $n_j^a(t) = 1$. and the weight $w_{ij}(t)$ is adjusted according to (6).
end if
end for
The agent obtains the new observation vectors by $o_{t+1} = \text{trans}(s_t, a_t)$
$t = t + 1$
end While

Pseudocode 1: Algorithm for heuristic constructing the STAMN.

IsNotTrans(s_j)
if the $p_j^s(t) > \vartheta$ according to (12) and the $q_j^s(t) = 1$ according to (13) **then**
return False
else
return True
end if

Pseudocode 2: Transition prediction criterion algorithm (recalling phase).

Figure 8 shows that three algorithms all can produce the correct model with zero prediction error at last because of no noise. However, the STAMN with exhaustive exploration has the better performance and the faster convergence speed because of the exhaustive exploration Q-function.

4.2. Experimental Comparison in $4 * 3$ *Grid Problem.* First, we use the small POMDP problem to test the feasibility of the STAMN. $4 * 3$ gird problem is selected, which is shown in Figure 9. The agent wanders inside the grid and has only left sensor and right sensor to report the existence of a wall in the current position. The agent has four actions: forward, backward, turn left, and turn right. The reference direction of the agent is northward.

In the $4 * 3$ gird problem, the hidden state space size $|S|$ is 11, the action space size $|A|$ is 4 for each state, and

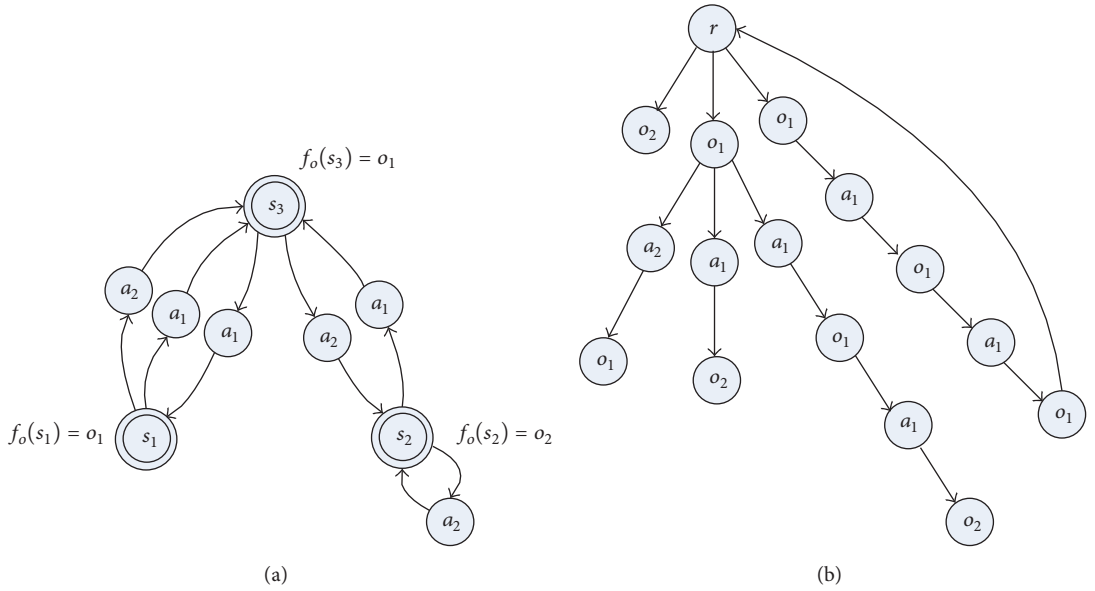

Figure 7: (a) The STAMN model. (b) The LPST model.

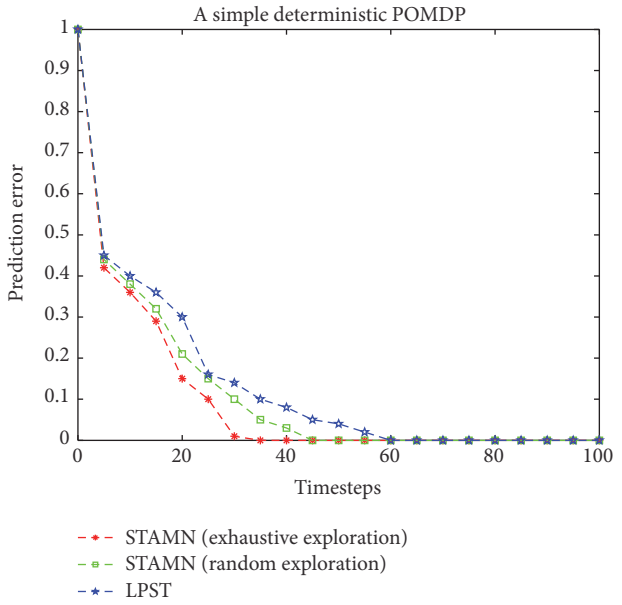

Figure 8: Comparative performance results for a simple deterministic POMDP.

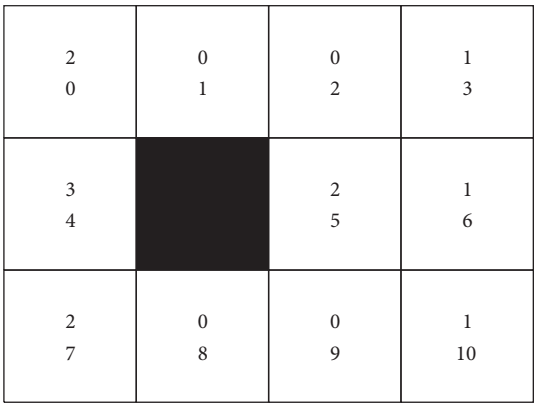

Figure 9: 4 ∗ 3 gird problem.

Table 2: The number of state nodes and action nodes in STAMN and LPST.

Algorithm	The number of state nodes	The number of action nodes
LPST	70	50
STAMN	11	44

the observation space size $|O|$ is 4. The upper figure in the grid represents the observation state and the bottom figure in the grid represents the hidden state. The black grid and the wall are regarded as the obstacles. We present a comparison between the LPST and the STAMN. The results are shown in Figure 10. Figure 10 shows that the STAMN with exhaustive exploration still has the better performance and the faster convergence speed in the 4 ∗ 3 gird problem.

The number of state nodes and action nodes in STAMN and LPST is described in Table 2. In STAMN, the state node represents the hidden state, but in LPST, the sate node represents observation value, and the hidden state is expressed by k-step past observations and actions; thus, more observation nodes and action nodes are repeated created, and most of the observation nodes and action nodes are the same. LPST has the perfect observation prediction capability and is the same as the STAMN but is not the state transition model.

In STAMN, The setting of parameters $\mu, \eta, \gamma_i^s, \delta_i^a$ is very important which determines the depth k of short-term

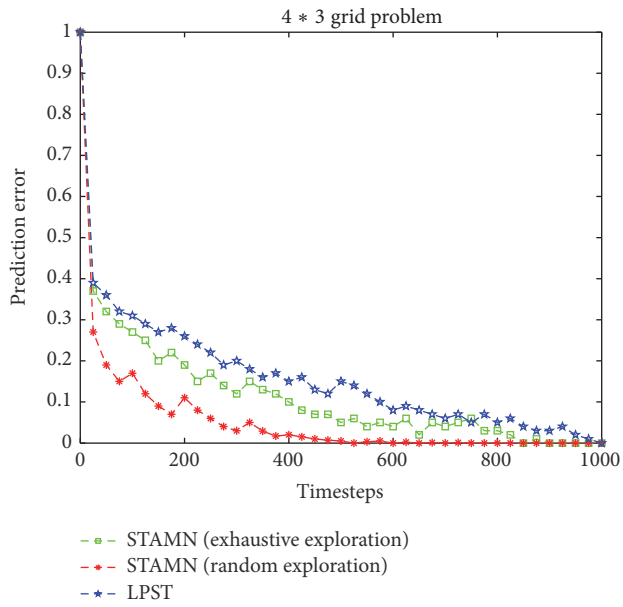

FIGURE 10: Comparative performance results for 4 ∗ 3 gird problem.

memory. We assume $\gamma_i^s = \delta_i^a = 0.9$ and $\mu = \eta = 0.9$ initial representing $k = 1$. When we need to increase k to $k + 1$, we only need to decrease μ, η according to $\eta = \eta - 0.2, \mu = \mu - 0.1$. The other parameters are determined relative easily. We set learning rates $\alpha = \beta = \gamma = 0.9$ and set the confidence parameters $C_{ij} = 1/|\tau_j|$, $D_{ij} = 1/2$ representing the same importance degree. Set constant values $C = 5$, $\vartheta = 0.9$.

4.3. Experimental Results in Complex Symmetrical Environments. The symmetrical environment in this paper is very complex, which is shown in Figure 11(a). The robot wanders in the environment. By following wall behaviour the agent can recognize the left wall, right wall, and corridor landmarks. These observation landmarks are different from the observation vector in the previous grid problem; every observation landmark has different duration time. For analysis of the fault tolerance and robustness of STAMN, we assume the disturbed environment is shown in Figure 11(b).

The figures in Figure 11(a) represent the hidden states. The robot has four actions: forward, backward, turn left, and turn right. The initial orientation of the robot is shown in Figure 11(a). Since the robot follows the wall in the environment, the robot has only one optional action in each state. The reference direction of action is the robot current orientation. Thus, the path 2-3-4-5-6 and path 12-13-14-15-16 are identical, and the memory depth $k = 6$ is necessary to identify hidden states reliably. We present a comparison between the STAMN and the LPST in noise-free environment and disturbed environment. The results are shown as in Figures 12(a) and 12(b). Figure 12(b) shows that STAMN has better noise tolerance and robustness than LPST because of the neuron synchronized activation mechanism, which bears the fault and noise in the sequence item duration time, and can realize the reliable recalling. However, the LPST cannot realize the correct convergence in reasonable time because of accurate matching.

5. Discussion

In this section, we compare the related work with the STAMN model. The related work mainly includes the LPST and the AMN.

5.1. Looping Prediction Suffix Tree (LPST). LPST is constructed incrementally by expanding branches until they are identified or become looped using the observable termination criterion. Given a sufficient history, this model can correctly capture all predictions of identifying histories and can map the all infinite identifying histories onto a finite LPST.

The STAMN proposed in this paper is similar to the LPST. However, the STAMN is looping hidden state transition model, so in comparison with LPST, STAMN have less state nodes and action nodes because these nodes in LPST are based on observation, not hidden state. Furthermore, STAMN has better noise tolerance and robustness than LPST. The LPST realizes recalling by successive accurate matching, which is sensitive to noise and fault. The STAMN offers the neuron synchronized activation mechanism to realize recalling. Even if in noisy and disturbed environment, STAMN can still realize the reliable recalling. Finally, the algorithm for learning a LPST is an additional computation model, which is not a distributed computational model. The STAMN is a distributed network and uses the synchronized activation mechanism, where performance cannot become poor with history sequences increasing and scale increasing.

5.2. Associative Memory Networks (AMN). STAMN is proposed based on the development of the associative memory networks. In existing AMN, firstly, almost all models are unable to handle complex sequence with looped hidden state. The STAMN realizes the identifying the looped hidden state indeed, which can be applied to HMM and POMDP problems. Furthermore, most AMN models can obtain memory depth determined by experiments. The STAMN offers a self-organizing incremental memory depth learning method, and the memory depth k is variable in different parts of state space. Finally, the existing AMN models general only record the temporal orders with discrete intervals Δt rather than sequence items duration with continuous-time. STAMN explicitly deals with the duration time of each item.

6. Conclusion and Future Research

POMDP is the long-standing difficult problem in the machine learning. In this paper, SATMN is proposed to identify the looped hidden state only by the transition instances in deterministic POMDP environment. The learned STAMN is seen as a variable depth k-Morkov model. We proposed the heuristic constructing algorithm for the STAMN, which is proved to be sound and complete given sufficient history sequences. The STAMN is real self-organizing incremental unsupervised learning model. These transition instances can

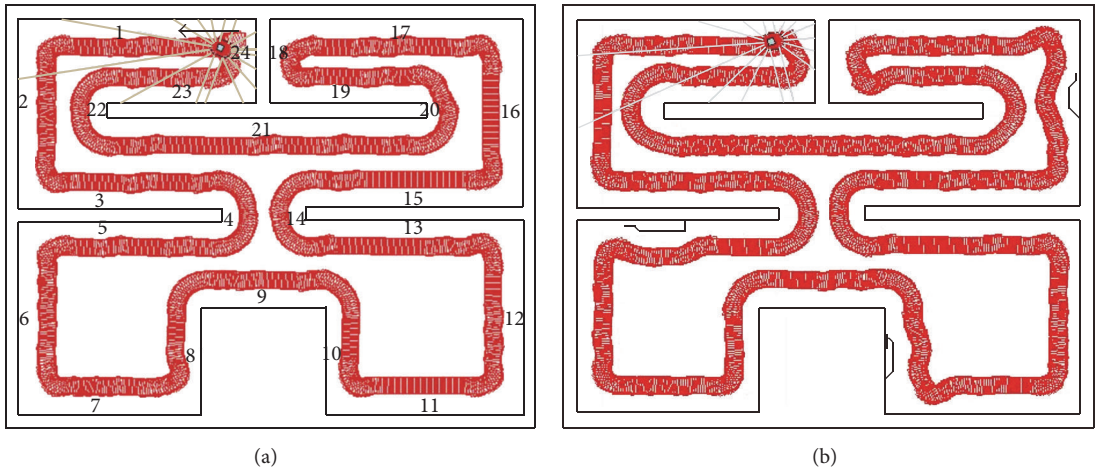

Figure 11: (a) Complex symmetrical simulation environment. (b) The disturbed environment.

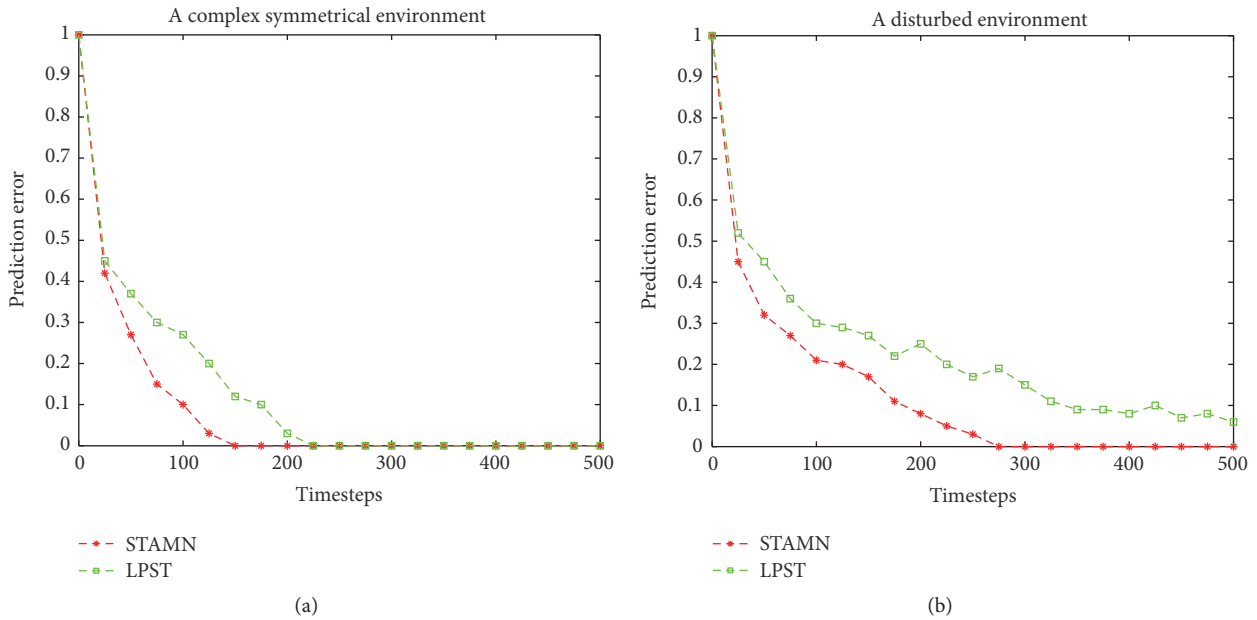

Figure 12: (a) Comparative results for a noise-free environment. (b) Comparative results for a disturbed environment.

be obtained by the interaction with the environment by the agent and can be obtained by a number of training data that does not depend on the real agent. The STAMN is very fast, robust, and accurate. We have also shown that STAMN outperforms some existing hidden state methods in deterministic POMDP environment. The STAMN can generally be applied to almost all temporal sequences problem, such as simultaneous localization and mapping problem (SLAM), robot trajectory planning, sequential decision making, and music generation.

We believe that the STAMN can serve as a starting point for integrating the associative memory networks with the POMDP problem. Further research will be carried out on the following aspects: how to scale our approach to the stochastic case by heuristic statistical test; how to incorporate with the reinforcement learning to produce a new distributed reinforcement learning model; and how it should be applied to robot SLAM to resolve the practical navigation problem.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this article.

References

[1] A. K. McCallum, *Reinforcement learning with selective perception and hidden state [Ph.D. thesis]*, University of Rochester, Rochester, NY, USA, 1996.

[2] M. Hutter, "Feature reinforcement learning: part I. unstructured MDPs," *Journal of Artificial General Intelligence*, vol. 1, no. 1, pp. 3–24, 2009.

[3] M. Daswani, P. Sunehag, and M. Hutter, "Feature reinforcement learning: state of the art," in *Proceedings of the Workshops at the 28th AAAI Conference on Artificial Intelligence*, 2014.

[4] P. Nguyen, P. Sunehag, and M. Hutter, "Context tree maximizing reinforcement learning," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1075–1082, Toronto, Canada, July 2012.

[5] J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver, "A Monte-Carlo AIXI approximation," *Journal of Artificial Intelligence Research*, vol. 40, no. 1, pp. 95–142, 2011.

[6] M. P. Holmes and C. L. Isbell Jr., "Looping suffix tree-based inference of partially observable hidden state," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 409–416, ACM, Pittsburgh, Pa, USA, 2006.

[7] M. Daswani, P. Sunehag, and M. Hutter, "Feature reinforcement learning using looping suffix trees," in *Proceedings of the 10th European Workshop on Reinforcement Learning (EWRL '12)*, pp. 11–24, Edinburgh, Scotland, 2012.

[8] M. Daswani, P. Sunehag, and M. Hutter, "Q-learning for history-based reinforcement learning," in *Proceedings of the Conference on Machine Learning*, pp. 213–228, 2013.

[9] S. Timmer and M. Riedmiller, *Reinforcement learning with history lists [Ph.D. thesis]*, University of Osnabrück, Osnabrück, Germany, 2009.

[10] E. Talvitie, "Learning partially observable models using temporally abstract decision trees," in *Advances in Neural Information Processing Systems*, pp. 818–826, 2012.

[11] M. Mahmud, "Constructing states for reinforcement learning," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, pp. 727–734, June 2010.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," http://arxiv.org/abs/1507.06527.

[15] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," https://arxiv.org/abs/1506.08941.

[16] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," http://arxiv.org/abs/1507.08750.

[17] X. Li, L. Li, J. Gao et al., "Recurrent reinforcement learning: a hybrid approach," http://arxiv.org/abs/1509.03044.

[18] D. Wang and B. Yuwono, "Incremental learning of complex temporal patterns," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1465–1481, 1996.

[19] A. Sudo, A. Sato, and O. Hasegawa, "Associative memory for online learning in noisy environments using self-organizing incremental neural network," *IEEE Transactions on Neural Networks*, vol. 20, no. 6, pp. 964–972, 2009.

[20] M. U. Keysermann and P. A. Vargas, "Towards autonomous robots via an incremental clustering and associative learning architecture," *Cognitive Computation*, vol. 7, no. 4, pp. 414–433, 2014.

[21] A. F. R. Araújo and G. D. A. Barreto, "Context in temporal sequence processing: a self-organizing approach and its application to robotics," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 45–57, 2002.

[22] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A Case Study," in *Proceedings of the 6th Latin American Robotics Symposium (LARS '09)*, pp. 1–6, October 2009.

[23] S. Tangruamsub, A. Kawewong, M. Tsuboyama, and O. Hasegawa, "Self-organizing incremental associative memory-based robot navigation," *IEICE Transactions on Information and Systems*, vol. 95, no. 10, pp. 2415–2425, 2012.

[24] V. A. Nguyen, J. A. Starzyk, W.-B. Goh, and D. Jachyra, "Neural network structure for spatio-temporal long-term memory," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 971–983, 2012.

[25] F. Shen, H. Yu, W. Kasai, and O. Hasegawa, "An associative memory system for incremental learning and temporal sequence," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '10)*, pp. 1–8, IEEE, Barcelona, Spain, July 2010.

[26] F. Shen, Q. Ouyang, W. Kasai, and O. Hasegawa, "A general associative memory based on self-organizing incremental neural network," *Neurocomputing*, vol. 104, pp. 57–71, 2013.

[27] B. Khouzam, "Neural networks as cellular computing models for temporal sequence processing," Suplec, 2014.