Intelligent Optimization and Modelling

696-027
New Ant Colony Algorithm for Continuous Function Optimization in 2D and 3D Search Spaces
Y.H.F. Jbara (Jordan)

696-014
A New Simulation Tool for Testing Smart Home Recognition Algorithms
K. Bouchard, A. Ajroud, B. Bouchard, and A. Bouzouane (Canada)

696-066
Flashlight Detection in Video
L.-H. Chen, B.-C. Hsu, and H.-Y.M. Liao (Taiwan)

696-115
Expert Modeling of Circulating Shaft Current for Fault Diagnosis of Large Alternating Machinery
M.R. Hedayati, S.A. Alavian, and M.S. Ghimati (Iran)

696-028
Ontology-based Concept Map Learning Path Reasoning System using SWRL Rules
K.-K. Chu and C.-I. Lee (Taiwan)

696-045
Evolutionary Neural Networks for Estimating Viscosity and Gas/Oil Ratio Curves
A. Khoukhi, M. Olosa, A. Abdulraheem, and M. Elshafei (Saudi Arabia)

696-107
Stock Price Forecasting using Higher Order Neural Networks
A.M. Kimiagari, M. Keyvanloo, M. Fallahnezhad, M.H. Moradi, and A.B. Farimani (Iran)

696-083
A Modern Neural Network Model to Do Stock Market Timing on the Basis of the Ancient Investment Technique of Japanese Candlestick
A.M. Kimiagari, M. Jasemi (Iran), and S. Kimiagari (Canada)

696-033
Encoding and Identification of Finite State Machines via Recurrent Neural Networks
S.H. Won, I. Song, S.Y. Lee, S. Lee, Y. Lee, and K.Y. Kim (Korea)

696-037
Improved Parallel Genetic Algorithm for Task Scheduling on Heterogeneous Parallel System
G. Singh and J. Singh (India)
ENCODING AND IDENTIFICATION OF FINITE STATE MACHINES VIA RECURRENT NEURAL NETWORKS

Sung Hwan Won*, Iickho Song*, Sun Young Lee*, Seungwon Lee*, Yeonwoo Lee†, and Ki Young Kim†

*Department of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
email: sunghwon@Sejong.kaist.ac.kr, i.song@ieee.org, {sylee, slee}@Sejong.kaist.ac.kr

†Department of Electronics Engineering
Mokpo National University
email: {ylee, danggadu}@mokpo.ac.kr

ABSTRACT
A new class of recurrent neural networks is proposed. The application of the proposed network is addressed in the encoding and identification of finite state machines (FSMs). Simulation results show that the identification of FSMs using the proposed network, trained by the hybrid greedy simulated annealing with a modified cost function in the learning stage, exhibits generally better performance than other conventional identification schemes.

KEY WORDS
Finite state machine, recurrent neural network, hybrid greedy simulated annealing, system identification.

1 Introduction
A finite state machine (FSM) or a finite state automaton is a dynamical system of which state space and time are described in the discrete domain and numbers of input symbols, output symbols, and states are finite. In a number of engineering fields, FSMs have been employed successfully. For example, FSMs are hired to program logic chips, design microprocessors, and run vending machines. In addition, dealing with FSMs provides us a fundamental insight of understanding computation systems since an FSM with an external storage or memory medium can be interpreted as a Turing machine and every computation system could be reduced to a Turing machine.

Meanwhile, it is quite helpful to identify an FSM from pairs of its input-output sequences. In order to identify an FSM via a certain model, it is recommended to first encode an FSM in the model since the encoding scheme will provide a hint on the determination of details of the model in the identification of an FSM. Since 1940s, recurrent neural networks (RNNs) have been used in the studies on FSMs. Several schemes of encoding FSMs in RNNs with hard-limiters and sigmoid units have been developed in [1, 2]. Various attempts on identifying FSMs using RNNs with sigmoid units have been made in [3, 4]: yet, the RNNs could not be made learn the FSM-like behavior successfully from long sequences and only result in a disappointing generalization performance. In order to secure satisfactory generalization performance, identification of FSMs in [5–8] has been considered with multiple pairs of short input-output sequences because the learning algorithm could not adequately deal with long sequences. Recently, online identification with a single pair of input-output sequences has been proposed in [9]: the scheme, nonetheless, requires a pair of overly long input-output sequences to train the RNN in the online fashion.

In this paper, we propose a class of RNNs. The application of the proposed network is addressed in the encoding and identification of FSMs. As the states of the proposed network represent those of the system to be modeled, encoding of an arbitrary FSM in the proposed network becomes quite simple. In the learning stage of the proposed network, we can employ the hybrid greedy simulated annealing (HGSA) [10], as an advantage of which no restriction is imposed on the length and number of input-output sequences in the identification procedure. It is observed from simulation results obtained by employing the HGSA with a modified cost function in the learning stage that the identification of FSMs using the proposed network exhibits generally better performance than other conventional identification schemes.

2 Encoding of FSMs in the Proposed Network
2.1 Finite State Machines
The class of FSMs are usually divided into two types, Mealy and Moore machines. Specifically, a Mealy machine is a 6-tuple denoted usually as $M = (Q, \Sigma, \Gamma, \gamma, \nu, q_0)$, where $Q = \{q_0, q_1, \ldots, q_{|Q|-1}\}$ is the set of states, $\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_{|\Sigma|-1}\}$ is the set of input symbols, $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_{|\Gamma|-1}\}$ is the set of output symbols, $\eta: \Sigma \times Q \to Q$ is the next-state function, $\nu: \Sigma \times Q \to \Gamma$
is the output function, and \( q_i \in Q \) is the initial state. A Moore machine is also a 6-tuple, with the only difference from the Mealy machine being that the output function is described by only the state as \( \nu: Q \rightarrow \Gamma \). It is well-known that, given a Mealy machine, the construction of an equivalent Moore machine is straightforward. Thus, from now on, we will deal only with the Mealy machine without loss of generality.

### 2.2 Proposed Network

Consider the network shown in Fig. 1, which consists of five layers named as layers 0, 1, 2, 3, and 4. Here, layers 1 and 2 can be viewed as the two layers of a multilayer perceptron (MLP) and layers 3 and 4 constitute the second MLP with the number of neurons in layer \( i \) denoted by \( n_i \). In the proposed network, \( \omega_i \) is an \( n_i \times n_{i-1} \) weight matrix of the links from layer \( i - 1 \) to layer \( i \), \( \omega_r \) is an \( n_1 \times n_2 \) weight matrix of the links from layer 2 to layer 1, \( \omega_3 \) is an \( n_3 \times n_4 \) weight matrix of the links from layer 0 to layer 3, \( \beta_i \) is an \( n_i \times 1 \) bias vector of layer \( i \), and \( \varphi_i \) is the transfer function of the \( n_i \) neurons in layer \( i \).

When the input is

\[
\mathbf{u}[t] = \begin{bmatrix} u_1[t] & u_2[t] & \cdots & u_{n_0}[t] \end{bmatrix}^T,
\]

the output of the proposed network is expressed as

\[
\mathbf{y}[t] = \begin{bmatrix} y_1[t] & y_2[t] & \cdots & y_{n_4}[t] \end{bmatrix}^T
\]

\[
= \varphi_4(\mathbf{w}_4\varphi_3(\mathbf{w}_3\mathbf{u}[t] + \mathbf{w}_2\mathbf{x}[t-1] + \mathbf{b}_3) + \mathbf{b}_4),
\]

and the state, output of layer 2, is expressed as

\[
\mathbf{x}[t] = \begin{bmatrix} x_1[t] & x_2[t] & \cdots & x_{n_2}[t] \end{bmatrix}^T
\]

\[
= \varphi_2(\mathbf{w}_2\varphi_1(\mathbf{w}_1\mathbf{u}[t] + \mathbf{w}_0\mathbf{x}[t-1] + \mathbf{b}_1) + \mathbf{b}_2),
\]

with the initial state \( \mathbf{x}[0] = \mathbf{x}_0 \). In (2) and (3), the matrix of an element-wise function is denoted by an underline as,

\[
\underline{\mathbf{w}}_i \underline{\mathbf{r}}_j.
\]

For example, in

\[
\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f(x_{1,1}) & f(x_{1,2}) & \cdots & f(x_{1,n}) \\
\vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
f(x_{m,1}) & f(x_{m,2}) & \cdots & f(x_{m,n}) \end{bmatrix},
\]

where \( \mathbf{x} \) is an \( m \times n \) matrix with \( x_{m,n} \) denoting the entry in the \( m \)-th row and \( n \)-th column, and \( f \) is a scalar function whose domain is one-dimensional.

### 2.3 Encoding of FSMs in the Proposed Network

Now let us describe the way we set the parameters of the proposed network so that it behaves like an FSM. Boolean functions can be expanded by minterms and maxterms. Here, we consider the minterm expansion. There are \( |\Sigma||Q| \) minterms in the next-state function \( \eta \) and the output function \( \nu \) of a Mealy machine. Two functions, \( \eta \) and \( \nu \), can be written as sums of these minterms which can be obtained from performing logical OR. With a single layer, an FNN can perform logical AND or OR. Therefore, with two layers, if 1) one layer performs AND operation (presenting all minterms) and 2) the other layer performs the OR operation (executing summation of minterms), then the MLP can approximate any Boolean function.

#### 2.3.1 Presentation of Minterms

Let the vector

\[
\rho_k = \begin{bmatrix} \sigma_{k/[Q]} \\
q_{k mod |Q|} \end{bmatrix}
\]

be an element of the Cartesian product \( \Sigma \times Q \), and

\[
\mathbf{P} = [\mathbf{P}_0 \mathbf{P}_1 \cdots \mathbf{P}_{|\Sigma||Q|-1}]
\]

be a matrix of size \( (|\log_2|\Sigma|| + |\log_2|Q||) \times |\Sigma||Q| \), where mod is the modulo operation: clearly, \( \rho_k \) is a minterm of Boolean functions \( \eta \) and \( \nu \) whose dimension is \( |\log_2|\Sigma|| + |\log_2|Q|| \). Next, by \( \Phi \) let us denote the set of all monotone-increasing continuous functions \( \varphi \) such that

\[
\lim_{x \to -\infty} \varphi(x) = 0,
\]

\[
\lim_{x \to \infty} \varphi(x) = 1,
\]

and

\[
\varphi(x) = 1 - \varphi(-x).
\]

Now, let \( \varphi_1, \varphi_3 \in \Phi \),

\[
n_0 = |\log_2|\Sigma||,
n_1 = n_3 = |\Sigma||Q|,
n_2 = |\log_2|Q||,
n_4 = |\log_2|\Gamma||,
\]

\[
[w_1, w_r] = \zeta_3 \text{sgn} \left( \mathbf{P}^T - \frac{1}{2} \mathbf{1}_{n_1, n_0 + n_2} \right),
\]

\[
\text{for example, in}
\]

\[
\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f(x_{1,1}) & f(x_{1,2}) & \cdots & f(x_{1,n}) \\
\vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
f(x_{m,1}) & f(x_{m,2}) & \cdots & f(x_{m,n}) \end{bmatrix},
\]

Figure 1. Structure of the proposed network.
\[ [ w_5 \ w_3 ] = \zeta_3 \text{sgn} \left( P^T - \frac{1}{2} I_{n_1, n_0 + n_2} \right), \] (15)

and

\[ b_k = \frac{\zeta_k}{2} 1_{n_k, 1} - \zeta_k (\text{sum}(P))^T, \quad k = 1, 3, \] (16)

where

\[ \text{sgn}(x) = \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{if } x < 0 \end{cases} \] (17)

is the sign function, \( 1_{m,n} \) is the \( m \times n \) matrix of ones, \( \text{sum}(A) \) is a row vector with the element being the sum over each column of a matrix \( A \) and \( \zeta_1 \) and \( \zeta_3 \) are positive real numbers. Then, it is easily obtained that

\[ \lim_{\zeta_1 \to \infty} \varphi_1 \left( [ w_1 \ w_r ] P + b_1 \right) = I_{n_1} \] (18)

and

\[ \lim_{\zeta_3 \to \infty} \varphi_3 \left( [ w_5 \ w_3 ] P + b_3 \right) = I_{n_3}, \] (19)

where \( I_n \) is the \( n \times n \) identity matrix. Hence, for large values of \( \zeta_1 \) and \( \zeta_3 \), the outputs of layer 1 and 3 represent minterms of \( \eta \) and \( \nu \).

### 2.3.2 Summation of Minterms

Let \( \varphi_2, \varphi_4 \in \Phi \),

\[ w_2 = \zeta_2 \left[ \eta(\rho_0) \eta(\rho_1) \cdots \eta(\rho_{n_1-1}) \right], \] (20)

\[ w_4 = \zeta_4 \left[ \nu(\rho_0) \nu(\rho_1) \cdots \nu(\rho_{n_1-1}) \right], \] (21)

\[ b_2 = -\frac{\zeta_2}{2} 1_{n_2,1}, \] (22)

and

\[ b_4 = -\frac{\zeta_4}{2} 1_{n_4,1}, \] (23)

where \( \zeta_2 \) and \( \zeta_4 \) are positive real numbers, then

\[ \lim_{\zeta_1 \to \infty} \varphi_2 \left( [ w_2 \ w_r ] P + b_1 \right) + b_2 \]

\[ = \left[ \eta(\rho_0) \eta(\rho_1) \cdots \eta(\rho_{n_1-1}) \right] \] (24)

and

\[ \lim_{\zeta_3 \to \infty} \varphi_4 \left( [ w_4 \ w_3 ] P + b_3 \right) + b_4 \]

\[ = \left[ \nu(\rho_0) \nu(\rho_1) \cdots \nu(\rho_{n_1-1}) \right]. \] (25)

Thus, for large values of \( \{ \zeta_i \}^4_{i=1} \), the proposed network behaves like the FSM \( M = (Q, \Sigma, \Gamma, \eta, \nu, q_1) \).

### 3 Identification of FSMs

Based on the results in Section 2.3, let us now consider the identification of a system, building a model that matches given input-output sequences of the system without any other information of the system. If we are to identify an FSM with the generalization performance in mind, we should not only consider the relationship of the input-output sequences given but also make the network to have the minimum number of states.

We first obtain \( |\Sigma| \) and \( |\Gamma| \) by observing the input and output sequences provided, respectively. The transfer functions \( \{ \varphi_l \}_1^4 \) of the four layers are arbitrarily selected from \( \Phi \). At the beginning, it is unfortunately not possible to obtain the minimum number \( |Q| \) of states with which the proposed network can match the input and output sequences given. Consequently, we first assume empirically that \( |Q| = 2 |\Sigma| \) and then find the value of \( |Q| \) by a trial and error. Once \( |\Sigma|, |\Gamma|, \) and \( |Q| \) are specified, we can obtain \( \{ n_l \}^4_{l=0} \) from (10)–(13), and all the weight matrices and bias vectors (except for \( w_2 \) and \( w_4 \)) from (14)–(16) by choosing \( \{ \zeta_i \}^4_{i=1} \) sufficiently large.

#### 3.1 HGSA

Next, we determine \( w_2 \) and \( w_4 \) appropriately so that the proposed network yields the output sequence given for the input sequence given. The determination of \( w_2 \) and \( w_4 \) can be accomplished with a proper learning algorithm. In this paper, we apply the HGSA. The HGSA is a hybridization of two optimization algorithms: the Levenberg-Marquardt (LM) algorithm [11, 12] as the local optimization algorithm and the greedy simulated annealing (GSA) as the global optimization algorithm. The LM algorithm has been widely used to train neural networks. The GSA is one of the powerful and robust stochastic optimization algorithms [13], an attractive merit of which is its mathematical global convergence property. The hybridization of the LM and GSA into the HGSA accelerates the convergence speed while guaranteeing the global solution in the HGSA: as is well-known, the global convergence of the HGSA is supported mathematically [10].

With the HGSA, we propose to minimize the regularized sum-of-squares cost function

\[ E(w) = \sum_{i=1}^{\tilde{i}} \sum_{i=1}^{n_i} (\epsilon_4)_i^2[i] + \sum_{i=1}^{\tilde{i}} \sum_{j=1}^{n_2} (\epsilon_2)_j^2[i] \] (26)

with respect to \( w_2 \) and \( w_4 \), where

\[ (\epsilon_4)_i[i] = \mu^\frac{1}{2} (\hat{y}_i[i] - y_i[i]), \] (27)

\[ (\epsilon_2)_j[i] = (1 - \mu)^\frac{1}{2} \left\{ \frac{1}{4} - \left( x_j[i] - \frac{1}{2} \right)^2 \right\}, \] (28)

\[ w = \begin{bmatrix} \text{vec}(w_2) \\ \text{vec}(w_4) \end{bmatrix} \] (29)

is a vector of size \((n_2 + n_4)n_1, \hat{y}_i[i] \) is the \( i \)-th element of the output sequence of an FSM at time \( t \), \( y_i[i] \) and \( x_j[i] \) are the \( i \)-th element of the output and the \( j \)-th element of the state of the proposed network at time \( t \), respectively, \( \tilde{i} \) is the length of the input (output) sequence, and \( 0 < \mu < 1 \). In (29), \( \text{vec}(A) \) is the vectorization of a matrix \( A \): all columns in \( A \) are listed one by one vertically. Here, we assume one
pair of input and output sequences is given: for multiple pairs of input and output sequences, the generalization of the cost function (26) should be straightforward.

When the first term in (26) is minimized to zero, we have \( \hat{y}[t] = y[t] \); when the second term in (26) is minimized to zero, every of the elements of \( x[t] \) is 0 or 1. This observation implies that, when the cost function \( E(w_2, w_4) \) is sufficiently small, the output and states of the proposed network will be quite close to the output and states of the FSA, respectively.

Let us now describe the details of minimizing the cost function (26) with respect to \( w_2 \) and \( w_4 \) with the HGSA.

a. Initialization: Let \( \tau = 1 \), where \( \tau \) is the iteration index. Initialize the \( (n_2 + n_4)n_1 \)-dimensional solution vector \( w \): the first \( n_2n_1 \) elements of \( w \) are chosen randomly in the range of \([0, \zeta_2]\) and the rest are chosen randomly in the range of \([0, \zeta_4]\). Set the initial temperature \( T \) to a sufficiently large value so that the solution vector does not converge to a local minimum.

b. Generation: Generate a new solution vector \( \omega_0 \) from the current solution \( w \) by

\[
\omega_0 = w + \Delta w.
\] (30)

The amount of change \( \Delta w \) is determined by the multidimensional Cauchy generating function [14]: to be more specific let \( \Delta w = T^{\text{rand}}(x) \), where \( x \) is a uniform random vector over \((-\pi/2, \pi/2)\).

c. LM algorithm: Iterate the LM algorithm \( \hat{\tau} \) times to produce \( \omega_1 \) from \( \omega_0 \) where \( \hat{\tau} \in \mathbb{Z}^+ \). Details on the LM algorithm are explained in Section 3.2.

d. Evaluation of costs: Evaluate \( E(w) \) and \( E(\omega_1) \).

e. Selection: Select the solution of the next iteration by the greedy selection rule as

\[
w = \begin{cases} 
\omega_1 & \text{if } E(w) \geq E(\omega_1), \\
w & \text{if } E(w) < E(\omega_1).
\end{cases}
\] (31)

f. Annealing: Let \( \tau = \tau + 1 \) and decrease the temperature by the reciprocal annealing schedule as

\[
T = \frac{T}{\tau}.
\] (32)

g. Termination: If \( \tau \) reaches the maximum number of iteration \( \tau_{\text{max}} \) or \( E(w) \) is smaller than \( E_0 \), stop. Here, \( E_0 \) is a sufficiently small value to make the proposed network behave like the FSM. Otherwise, go to \( b \).

### 3.2 Iteration with LM Algorithm

For convenience, let

\[
\epsilon(w) = \begin{bmatrix} \epsilon_1^T & \epsilon_2^T \end{bmatrix}^T,
\] (33)

where

\[
\epsilon_i = \begin{bmatrix} \epsilon_i^T[1] & \epsilon_i^T[2] & \cdots & \epsilon_i^T[t] \end{bmatrix}^T, \quad i = 2, 4
\] (34)

and

\[
\epsilon[t] = \begin{bmatrix} \epsilon_i[1][t] & \epsilon_i[2][t] & \cdots & \epsilon_i[n][t] \end{bmatrix}^T.
\] (35)

Iterating the LM algorithm \( \hat{\tau} \) times to produce \( \omega_1 \) from \( \omega_0 \) can be symbolized by the procedure of obtaining \( \omega_{i+1}/\hat{\tau} \) from \( \omega_i/\hat{\tau} \) recursively with the LM updating rule for \( i = 0, 1, \cdots, \hat{\tau} - 1 \), where the LM updating rule is [11, 12]

\[
\omega_{i+1}/\hat{\tau} = \omega_i/\hat{\tau} - \left( J^T J + \lambda I_{(n_4+n_2)n_1} \right)^{-1} J^T \epsilon(\omega_i/\hat{\tau}),
\] (36)

where \( J \) is the Jacobian of \( \epsilon(\omega_i/\hat{\tau}) \). In (36), \( \lambda \) is the damping factor which governs the step size: the rule (36) becomes the Newton formula and standard gradient descent for a very small and large values of \( \lambda \), respectively. Initially, the value of \( \lambda \) is set to be very small. If the cost decreases (increases) after taking the step depicted by (36), the new (old) weight vector is stored and the value of \( \lambda \) is divided (multiplied) by a value larger than one, and the process is repeated with \( i \) increased by ones (zero). When \( \lambda \) reaches its pre-fixed maximum value, the procedure stops before \( \hat{\tau} \) iterations.

### 4 Simulation Results

In this section, we demonstrate the performance of the proposed scheme in the identification of FSMs in comparison with other conventional schemes.

#### 4.1 Test Problems

We consider the well-known Tomita automata [7–9]. As in [7], we generated input sequences randomly with an equal number of positive and negative and obtained the output sequences of each FSM. The maximum length of an input sequence varied from 5 to 10: more specifically, Table 1 lists the maximum sequence length and total number of sequences used to learn the Tomita automata.

#### 4.2 Setup

By observing the input-output sequences given, we can figure out \(|\Sigma|\) and \(|\Gamma|\), and set \( n_0 \) and \( n_4 \) accordingly from
Figure 2. The generalization performance of six RNN approaches: no clustering (NC), rigid quantization (RQ), learn then quantize (LQ), dynamic online clustering and state extraction (DOLCE) in unsupervised mode using Forgy’s algorithm (DU), DOLCE in supervised mode using a mixture model (DS), and the scheme proposed in this paper (Proposed). (a) Tomita 1'. (b) Tomita 2'. (c) Tomita 3'. (d) Tomita 4'. (e) Tomita 5. (f) Tomita 6.

(10) and (13), respectively. From the value of $|Q|$ obtained by minimization through trial and error, we set the number of neurons in each layer as in Table 2 from (10)–(13). Let us also assume

$$\varphi_k(x) = \frac{1}{1 + e^{-x}},$$

(37)

$\zeta_k = 20$ for $k = 1, 2, 3, 4$, and the elements of $\omega$ are chosen randomly in $[0, \zeta_2]$.

We use 10 iterations of the LM algorithm for the local optimization, i.e., $\bar{\tau} = 10$. The initial temperature $T$ is set to 10, $\mu$ is set to 0.5, and $\lambda$ is initially set to 0.001 and it is multiplied or divided by 10 as needed until it reaches $10^{10}$. If the cost function is less than 0.01 the HGSA is termi-
the proposed scheme, it took about 10,000, which is derived by trial and error. For higher success rate, we need a larger \( \tau_{\text{max}} \) for the HGSA. If \( \tau_{\text{max}} \to \infty \), then the learning by the HGSA will be always successful. We have chosen \( \tau_{\text{max}} \) to be large enough so that the success rate is acceptable, where the success rate is the rate of the number of trials that have ended meeting the condition \( E(w|\tau) \leq E_0 \) to the total number of trials.

### 4.3 Results

The results are presented in Fig. 2, where the generalization performance of various approaches [5–7] are shown. The vertical axis is the number of misclassified bits for sequences of length 3,000. Each bar is the average result across the 100 repetitions with different initial weights. In the proposed scheme, it took about 0.05 sec per one iteration for a computer with Intel Core2 Quad CPU 2.40GHz and 3.50GB RAM, in which MATLAB and C are used to implement the proposed identification scheme. It is clearly observed that the proposed identification scheme outperforms other conventional schemes in Fig. 2.

### 5 Conclusion

We have proposed a class of RNNs. Thanks to the well-defined structure of the proposed network in relation to FSMs, the proposed network has shown to have advantages over other conventional RNNs in encoding and identifying FSMs. The HGSA with a modified cost function is employed as the learning algorithm in the identification of an FSM with the proposed network. Simulation results have demonstrated that the identification of FSMs using the proposed network exhibits better performance than other conventional schemes.

### Acknowledgements

This work was supported by the National Research Foundation of Korea, with funding from the Ministry of Education, Science and Technology, under Grant 2010-0015175, and by the Information Technology Research Center (ITRC) Program of National IT Industry Promotion Agency, with funding from the Ministry of Knowledge Economy, under Grant NIPA-2010-C1090-1021-0007.

### References


