Convergence acceleration of the Hopfield neural network byoptimizing integration step sizes

Convergence Acceleration of the Hopfield Neural Network by Optimizing Integration Step Sizes

Shigeo Abe

***Abstract*—In our previous work we have clarified global convergence of the Hopfield neural network and showed, by computer simulations, improvement of solution quality by gradually decreasing the diagonal elements of the coefficient matrix. In this paper, to accelerate convergence of the Hopfield network, at each time step the integration step size is determined dynamically so that at least one component of a variable vector reaches the surface of the hypercube. The computer simulation for the traveling salesman problem and an LSI module placement problem shows that convergence is stabilized and accelerated compared to integration by a constant step size.**

I. Introduction

Attempting solutions to combinatorial optimization problems using the Hopfield neural network [1] has led to many discussions on positive and negative features of this tool. The major criticisms are:

1) lack of a solid method to determine weights in the energy function, meaning they must be changed by trial and error until a feasible solution is obtained;
2) deterioration of solution quality as the size of problems becomes large;
3) inefficient problem formulation in some cases. For instance, for the n-city traveling salesman problem (TSP), n^2 neurons (variables) are necessary. This is especially prohibitive for a large sized problem both from a memory size requirement and computation time needed to obtain a solution.

As for 1), Aiyer [2], [3] proposed a projection method. In this method minimization of the objective function and projection to the valid space, in which the included hypercube vertices are all feasible, are alternatively iterated until the feasible solution is obtained. In [4], we have clarified the global convergence of the Hopfield model when the diagonal elements of the coefficient matrix are zero and we developed a method to determine the weights so that all the feasible solutions become stable. Although a feasible solution was always obtained by this method, the quality of solution was poor for a large sized problem.

As for 2), Aiyer [2], [3] pointed out that setting the diagonal elements of the coefficient matrix to zero worsens the quality of solutions, and he proposed the MGNC (Matrix Graduated Non-Convexity) method, in which the diagonal elements are increased during integration, so that the solution moves in the direction of the eigenvector whose absolute eigenvalue is the maximum. For the 30-city TSP problem used in [1], he showed that the valid space projection with the MGNC method outperformed the mean field annealing method, on average. Eberhardt *et al.* [5] also showed that the increase of the diagonal elements led to improvement of the solution quality. In [6], we extended our theory to the case where the diagonal elements are nonzero, and showed that, by gradually decreasing the diagonal elements, the quality of solutions was drastically improved. (The sign of the coefficient matrix we used was opposite to that in [2], [3]. Thus, the diagonal elements were "decreased" instead of "increased.")

Manuscript received June 8, 1994; revised February 20, 1995.

The author is with the Hitachi Research Laboratory, Hitachi, Ltd., Hitachi 319-12, Japan.

Publisher Item Identifier S 1083-4419(96)00429-3.

As for 3) if we solve the n-city TSP by the simulated annealing method, we set n variables to represent the order of the visit, while by the Hopfield model we must use n^2 variables. This is unavoidable so long as we use the Hopfield model. Thus, in this paper we discuss acceleration of the convergence within the framework of the Hopfield model. First we summarize the stability condition of the Hopfield model we have derived so far. Then to accelerate convergence of the Hopfield model, at each time step we determine the integration step size so that at least one variable among all the variables reaches the surface of the hypercube. Finally, by computer simulations for the traveling salesman problem and an LSI module placement problem we show that convergence is stabilized and accelerated compared to integration by a constant step size.

II. Stability Condition of the Hopfield Model

In this section we summarize the results, obtained in [6], concerning stability of vertices that are necessary to determine the weights in the energy function. Let the energy function E for an optimization problem be

$$E = \tfrac{1}{2}\mathbf{x}^t T\mathbf{x} + \mathbf{b}^t\mathbf{x} \tag{1}$$

where

1) $\mathbf{x} = (x_1, \cdots, x_n)^t$: the variable vector, $x_i = 1$ or 0, $i = 1, \cdots, n$;
2) $T = \{T_{ij}\}$: the $n \times n$ symmetric matrix; and
3) $\mathbf{b}$: the n dimensional input vector.

Using a piece-wise linear function as an activation function, the Hopfield model becomes [7]

$$\begin{aligned}\frac{d\mathbf{x}}{dt} &= -\frac{\partial E}{\partial \mathbf{x}}\\ &= -T\mathbf{x} - \mathbf{b}, \quad 0 \le x_i \le 1, \quad \text{for} \quad i = 1, \cdots, n.\end{aligned} \tag{2}$$

where the range of $x_i\,(i = 1, \cdots, n)$ is extended to $[0, \ 1]$. As discussed in [4], the results obtained for the piece-wise linear function are also valid for the hyperbolic tangent function used in [1]. In our following study, we assume that T_{ii} are all nonzero.

Let the hypercube and the interior of the hypercube be defined by

$$H = \{x | 0 \le x_i \le 1 \quad \text{for} \quad i = 1, \cdots, n\} \tag{3}$$

and

$$H^- = \{\mathbf{x} | 0 < x_i < 1 \quad \text{for} \quad i = 1, \cdots, n\}, \tag{4}$$

respectively. And for vertex $\mathbf{c} = (c_1, \cdots, c_n)^t$ in H we define n adjacent vertices $\mathbf{c}(i) = (c_1, \cdots, c_{i-1}, 1-c_i, c_{i+1}, \cdots, c_n)^t$ where $c_i = 1$, or 0 for $i = 1, \cdots, n$.

Let H^- be divided into three parts by the hyperplane $T_i\,\mathbf{x} + b_i = 0$ as follows:

$$D^+(i) = \{\mathbf{x} | 0 < x_j < 1, j = 1, \cdots, n, T_i\,\mathbf{x} + b_i > 0\},$$
$$D^-(i) = \{\mathbf{x} | 0 < x_j < 1, j = 1, \cdots, n, T_i\,\mathbf{x} + b_i < 0\},$$

and

$$D^0(i) = \{\mathbf{x} | 0 < x_j < 1, j = 1, \cdots, n, T_i\,\mathbf{x} + b_i = 0\}. \tag{5}$$

For $\mathbf{x} \in D^+(i)$, as time elapses component x_i always decreases so long as $\mathbf{x}$ is in domain $D^+(i)$. If $\mathbf{x} \in D^0(i)$, for $\mathbf{x}' = (x_1, \cdots, x_i + \varepsilon, x_{i+1}, \cdots, x_n)^t$, $\varepsilon > 0$ and $T_{ii} > 0$, $T_i\,\mathbf{x}' + b_i > 0$ holds, namely $\mathbf{x}' \in D^+(i)$. Thus if T_{ii} is positive, $\mathbf{x}'$ moves to the hyperplane $T_i\,\mathbf{x} + b_i = 0$. Likewise if T_{ii} is negative, $\mathbf{x}'$ moves away from the hyperplane $T_i\,\mathbf{x} + b_i = 0$.

We consider stability of vertex $\mathbf{c}$ in which $\mathbf{c} \notin D^0(i)$ for all i holds or even if $\mathbf{c} \in D^0(i)$ for some i holds, $D^0(i)$ and H^- do not intersect. (By the proper selection of the weights in the energy function this assumption holds.) For vertex $\mathbf{c}$ in hypercube H we define domain D_c as follows:

$$D_c = \bigcap_{\substack{i=1,\cdots,n\\ \mathbf{c}\notin D^0(i)}} Z(i) \tag{6}$$

where

$$Z(i) = \begin{cases} D^+(i) & \text{for } \; T_i\mathbf{c} + b_i > 0 \\ D^-(i) & \text{for } \; T_i\mathbf{c} + b_i < 0 \end{cases} \tag{7}$$

For any $\mathbf{x} \in D_c$, $\mathbf{x}$ moves monotonically so long as $\mathbf{x}$ remains in D_c. If for $i = 1, \cdots, n$ vertex $\mathbf{c}$ satisfies

$$T_i\,\mathbf{c} + b_i > 0 \quad \text{for} \quad c_i = 0$$

or

$$T_i\,\mathbf{c} + b_i < 0 \quad \text{for} \quad c_i = 1 \tag{8}$$

or for

$$T_i\,\mathbf{c} + b_i = 0 \quad \text{and} \quad D^0(i) \cap H^- = \emptyset,$$
$$T_i\,\mathbf{c}(i) + b_i > 0 \quad \text{for} \quad c_i = 0$$

or

$$T_i\,\mathbf{c}(i) + b_i < 0 \quad \text{for} \quad c_i = 1 \tag{9}$$

vertex $\mathbf{c}$ is stable since $\mathbf{x}(\in D_c)$ which is in the neighborhood of $\mathbf{c}$ moves to $\mathbf{c}$. Here notice that if $T_i\,\mathbf{c} + b_i = 0$ and $D^0(i) \cap H^- = \emptyset$, the sign of $T_i\,\mathbf{x} + b_i$ is the same on the edge $\mathbf{c}$ and $\mathbf{c}(i)$ except for $\mathbf{c}$.

Now for vertex $\mathbf{c}$ we define the convergent domain D'_c as follows:

$$\begin{aligned} D'_c = \{\mathbf{x} | \mathbf{x} \in D_c \quad &\text{and} \quad (x_1, \cdots, x_{i-1}, x'_i, \\ x_{i+1}, \cdots, x_n)^t \in D_c, \quad &x'_i = \gamma c_i + (1-\gamma)x_i, \\ 0 \le \gamma \le 1 \quad &\text{for} \quad i = 1, \cdots, n\}. \end{aligned} \tag{10}$$

The condition that $(x_1, \cdots, x_{i-1}, x'_i, x_{i+1}, \cdots, x_n)^t \in D_c$ means that a line segment connecting $\mathbf{x}$ and $(x_1, \cdots, x_{i-1}, c_i, x_{i+1}, \cdots, x_n)^t$ excluding the latter point, which is parallel to the x_i axis, does not intersect with hyperplanes $T_i\,\mathbf{x} + b_i = 0 (i = 1, \cdots, n)$. Also it is easy to see that D'_c is a nonempty, connected domain. Therefore, so long as $\mathbf{x}$ is in D'_c, a corrected $\mathbf{x}$ does not go back to $H^- - D'_c$.

From the above definitions the following theorem holds:

Theorem 1: If vertex $\mathbf{c}$ satisfies (8) or (9), for any $\mathbf{x} \in D'_c$, $\mathbf{x}$ converges monotonically to $\mathbf{c}$.

III. Solution of the Hopfield Model

A. Numerical Solution

Let $\mathbf{x}(0)$ denote the variable vector at time $t = 0$. Then, if $\mathbf{x}(0)$ is in the hypercube H, the analytical solution of (2) for the initial vector $\mathbf{x}(0)$ is given by

$$\mathbf{x}(t) = e^{-tT}\mathbf{x}(0) - e^{-tT}\int_0^t e^{\tau T}\mathbf{b}\, d\tau \tag{11}$$

so long as $\mathbf{x}(t)$ remains in the hypercube. But when some component, e.g., x_n, is on the hypercube surface at time 0, and when for $t \ge 0$ the component x_n sticks to the surface, the analytical solution becomes

$$\mathbf{x}'(t) = e^{-tT'}\mathbf{x}'(0) - e^{-tT'}\int_0^t e^{\tau T'}[\mathbf{b}' + \mathbf{v}\, x_n(0)]\, d\tau \tag{12}$$

$$x_n(t) = x_n(0) \tag{13}$$

where

1) $x_n(0) = 1$ or 0,
2) $\mathbf{x}' = (x_1, \cdots, x_{n-1})^t$,
3) $\mathbf{b}' = (b_1, \cdots, b_{n-1})^t$,
4) $\mathbf{v} = (T_{1n}, \cdots, T_{n-1,n})^t$, and
5) $T' = \left[\begin{smallmatrix} T_{11} \cdots T_{1,n-1} \\ T_{n-1,1} \cdots T_{n-1,n-1}\end{smallmatrix}\right]$.

But since we do not know in advance whether some components, which are on the surface of the hypercube, remain on the surface, (12) is a nonrealizable solution.

Expanding the exponentials in (11) into a matrix series gives

$$\begin{aligned} \mathbf{x}(t) = &\left(I - tT + \frac{t^2T^2}{2} - \cdots\right) \\ &\cdot \mathbf{x}(0) - \left(I - tT + \frac{t^2T^2}{2} - \cdots\right) \\ &\cdot \int_0^t \left(I + \tau T + \frac{\tau^2 T^2}{2} + \cdots\right)\mathbf{b}\, d\tau \\ = &\,\mathbf{x}(0) - t[T\mathbf{x}(0) + \mathbf{b}] + \frac{t^2 T}{2}[T\mathbf{x}(0) + \mathbf{b}] \cdots \end{aligned} \tag{14}$$

Then we obtain the first order model

$$\mathbf{x}(t) = \mathbf{x}(0) - t\,[T\mathbf{x}(0) + \mathbf{b}] \tag{15}$$

which is equivalent to the Euler model.

We calculate $\mathbf{x}(t)$ setting a small step size t in (15) even if some of the components of $\mathbf{x}(0)$ are on the surface of the hypercube, namely 0 or 1. When some components are greater than 1 or less than 0 at time t, they are set to 1 or 0 to be used for the next time step calculation. Since there was no good way of optimizing the integration step size, a number of iteration steps was necessary to obtain the solution.

B. Step Size Determination

To accelerate convergence, we define the integration step size at time t as the minimum step size which makes some component among $\mathbf{x}(t)$ reach the surface of the hypercube. Let $t = 0$. If $x_i(0) = 1$ and $[T\mathbf{x}(0) + \mathbf{b}]_i \le 0$, or $x_i(0) = 0$ and $[T\mathbf{x}(0) + \mathbf{b}]_i \ge 0$ where the subscript i denotes the ith component, the ith component moves away from the hypercube H. Thus we need to exclude these variables from determination of the step size. Let I be a set of integers where $x_i(0) = 1$ and $[T\mathbf{x}(0)+\mathbf{b}]_i \le 0$, or $x_i(0) = 0$ and $[T\mathbf{x}(0)+\mathbf{b}]_i \ge 0$. And let $N = \{1, \cdots, n\}$. We calculate

$$t_i = \begin{cases} \dfrac{x_i(0) - 1}{[T\mathbf{x}(0) + \mathbf{b}]_i} & \text{for} \quad [T\mathbf{x}(0) + \mathbf{b}]_i < 0 \\ \dfrac{x_i(0)}{[T\mathbf{x}(0) + \mathbf{b}]_i} & \text{for} \quad [T\mathbf{x}(0) + \mathbf{b}]_i > 0 \end{cases}$$
$$\text{for} \quad i \in N - I \tag{16}$$

where t_i is the step size for $x_i(0)$ to reach the surface of the hypercube. To restrict $\mathbf{x}(t)$ so that only one component of $\mathbf{x}(0)$ reaches the surface, we take the minimum among t_i $(i = 1, \cdots, n)$:

$$\Delta t = \min_{i \in N - I} t_i. \tag{17}$$

[If there is more than one i that satisfies (17), more than one component may reach the surface.]

C. Loop Escape

Loop Detection: Since the determination of the step size is deterministic, there may be cases where an infinite loop occurs during integration. Let $i(j)$ denote the integer in which $t_{i(j)}$ gives the minimum in (17) at time step j. We call the sequence of $i(j)$ the selection sequence. Then the following selection sequence of $i(j)$

$$\cdots \underline{4632}\,\underline{4632}\,\underline{4632}\cdots$$

may result in an infinite loop. To reduce the overhead for loop detection, we judge that an infinite loop occurs when the same pairs of integers are selected consecutively three times. Namely, if

1) first loop condition

$$\begin{aligned} i(j) &= i(k) \\ i(j+1) &= i(k+1) \quad \text{for} \quad j+1<k \\ i(j) &\neq i(m) \quad \text{for} \quad m=j+2,\cdots,k-1, \end{aligned} \tag{18}$$

and

2) second loop condition

$$\begin{aligned} i(k) &= i(l) \\ i(k+1) &= i(l+1) \quad \text{for} \quad k+1<l. \\ i(k) &\neq i(m) \quad \text{for} \quad m=k+2,\cdots,l-1, \end{aligned} \tag{19}$$

are satisfied, we assume that an infinite loop occurs.

To implement this, for each $x_i (i = 1, \cdots, n)$ we prepare two arrays: post(i) which stores the integer selected in the next step after i is selected, and loop(i) which indicates whether two pairs of integers are detected.

And at step j, we do the following:

1) If $i(j) \neq$ post[$i(j-1)$], i.e., if the first loop condition is not satisfied, we set loop(i) = 0 and post[$i(j-1)$] = $i(j)$.
2) If $i(j) =$ post[$i(j-1)$] and loop(i) = 0, i.e., if the first loop condition is satisfied we set loop(i) = 1.
3) If $i(j) =$ post[$i(j-1)$] and loop(i) = 1, i.e., if the second loop condition is satisfied we judge that the infinite loop is detected and set loop(i) = 0.

Also in the extreme case, if the same integer is selected consecutively, i.e., $i(j) = i(j+1)$, we judge that an infinite loop called a self loop has occurred. The condition $i(j) = i(j+1)$ means that the state of $x_{i(j)}$ reaches 0 (or 1) at time step j and then it is reversed to 1 (or 0). When a loop is detected we modify the step size given by (17) so that the loop is resolved.

Step Size Modification: To introduce randomness into the convergence process we modify the step size as follows for step j.

1) If a self loop is detected, we set

$$\Delta t \leftarrow \Delta t \times Rand \tag{20}$$

where $Rand$ is a random value uniformly distributed in [0, 1]. By reducing the step size integer other than $i(j)$ is expected to be selected at step j

2) If (20) does not work, namely, $i(j-2) = i(j-1) = i(j)$, or a loop given by (18) and (19) is detected, we introduce a large perturbation:

$$\Delta t \leftarrow \Delta t(1 + \beta_1 \times Rand) \tag{21}$$

where β_1 is a small positive value. Depending on the value of β_1, more than two components may reach the surface of the hypercube.

3) Otherwise, namely even if there is no loop, we set

$$\Delta t \leftarrow \Delta t(1 + \beta_2 \times Rand) \tag{22}$$

where β_2 is a small positive value. Equation (22) is introduced to make the convergence process nondeterministic. Usually we set $\beta_1 > \beta_2$.

IV. Traveling Salesman Problem

A. Problem Formulation

In the n-city traveling salesman problem, travel to each city is planned so that the total tour length is minimized. For each city n neurons are assigned and when the ith neuron output is one and the remaining $n-1$ neuron outputs are zero, we assume that the city is visited at the ith order. Then the energy function is defined by:

$$\begin{aligned} E =& \frac{A}{2}\sum_x\left(\sum_i V_{xi} - 1\right)^2 \\ &+ \frac{A}{2}\sum_i\left(\sum_y V_{yi} - 1\right)^2 \\ &+ \frac{D}{2}\sum_x\sum_{y\neq x}\sum_i d_{xy}V_{xi}(V_{y,i+1} + V_{y,i-1}) \\ &+ \frac{F}{2}\sum_x\sum_i V_{xi}^2 \end{aligned} \tag{23}$$

where

1) V_{xi}: ith neuron for city x, and $0 \leq V_{xi} \leq 1$;
2) d_{xy}: distance between city x and city y;
3) $A(>0)$: a weight for constraints;
4) $D(>0)$: a weight for the objective function;
5) F: a weight which is decreased during integration; and subscripts $i-1$ and $i+1$ are calculated by modulo n.

Taking a partial derivative of (23) with respect to V_{xi} we get:

$$\begin{aligned} \frac{\partial E}{\partial V_{xi}} =& A\left(\sum_j V_{xj} - 1\right) + A\left(\sum_y V_{yj} - 1\right) \\ &+ D\sum_{y\neq x} d_{xy}(V_{y,i+1} + V_{y,i-1}) + FV_{xi}. \end{aligned} \tag{24}$$

For a feasible solution, the first and second terms of (24) are zero. Thus from (8), the feasible solution is stable if

$$\begin{aligned} D(d_{xy} + d_{xz}) + F &< 0 \quad \text{for} \quad V_{xi} = 1 \\ Dd_{xy} &> 0 \end{aligned} \tag{25}$$

or

$$D(d_{xy} + d_{xz}) > 0 \quad \text{for} \quad V_{xi} = 0 \tag{26}$$

where x, y, and z are all different. Since (26) always holds, if F is 0, feasible solutions are all unstable. Also if

$$F < -D \max_{x,y,z}(d_{xy} + d_{xz}), \tag{27}$$

all the feasible solutions are stable. But according to [4], to obtain good quality of solutions, we may violate this condition. Namely, we need not stabilize all the feasible solutions.

Now we suppress spurious states.

1) Change states of one or more neurons in a feasible solution from one to zero. Let column and row sums corresponding to V_{xi} be zero. Then from (8) and (9) all the infeasible solutions become unstable if

$$A > \frac{D}{2}\max_{x,y,z}(d_{xy} + d_{xz}) \tag{28}$$

where x, y, and z are different.

2) Change states of two or more neurons in a feasible solution from zero to one and one to zero at the same time.

In this case more than one neuron fires at some column or row. Assume $V_{xi} = V_{xj} = 1 (i \neq j)$. If

$$A + F > 0, \tag{29}$$

Equation (24) for V_{xi} is positive. Thus from (25), the infeasible solutions are unstable. If (27)–(29) hold, stable vertices are all feasible.

B. Computer Simulations

To demonstrate the usefulness of the step size optimization over constant step size, we used the 10- and 30-city TSP's in [1]. The constant step size Δt was determined by [4]

$$\Delta t = \frac{\text{const}}{\max_i \left(\frac{1}{2} T\mathbf{1} + b\right)_i} \tag{30}$$

where $\mathbf{1}$ denotes an n-dimensional vector whose elements are 1; i denotes the ith row of a vector; and const is selected as 0.3.

The initial values were set as

$$V_{xi} = 0.5 + \alpha Rand' \tag{31}$$

where α is a small positive value and $Rand'$ is a random value in $[-0.5, 0.5]$.

Let F_i and F_f be the initial and final values of F. In [4] if

$$\sum_x \sum_i |V_{xi}| < 0.0001 \tag{32}$$

is satisfied, we decrease F as follows:

$$F \leftarrow F - \Delta F \tag{33}$$

where ΔF is a positive value. Equation (32) checks whether the solution reaches the surface of the hypercube [2]. But for step size optimization, this is meaningless. So after a predetermined time of integration t_d, we successively decrease F according to (33) until F reaches F_f. For integration by the constant step size, we used the same strategy for decreasing the diagonal elements.

At each time step, we check whether V_{xi} satisfies the constraints, namely whether V_{xi} is feasible, assuming $V_{xi} = 1$ if $V_{xi} \geq 0.5$ and $V_{xi} = 0$ if $V_{xi} < 0.5$ [8]. If V_{xi} is feasible, the integration is terminated. This contributes to a reduction in the number of iterations. Also when all V_{xi} reach 1 or 0, or all the differences in current and previous corrections of V_{xi} are within a specified value, the integration is terminated.

Ten-City TSP: Setting $D = 1$, (27)–(29) become

$$-1.667 > F, \tag{34}$$
$$A > 0.834, \tag{35}$$
$$F > -A. \tag{36}$$

Since, to obtain good quality of solutions, we need not satisfy (34), we set $A = F_i = 1.0$ and $F_f = -0.2$, which satisfy (35) and (36). Also we set

$$\Delta F = 0.001 \tag{37}$$
$$\alpha = 0.0001,$$
$$\beta_1 = 0.001,$$
$$\beta_2 = 0.0001. \tag{38}$$

Table I shows the performance comparison of the step size optimization and the constant step size when the starting time to decrease F_i i.e., t_d was changed. In the table "fixed" means that F_i was set to the final value $F_f = -0.2$. For all 1000 trials, feasible solutions were always obtained. For the "fixed" case and $t_d = 100$, the number of optimal solutions obtained for the step size optimization was a little bit smaller than that by the constant step size, but we could always obtain the optimal solution for t_d larger than 100. Also we could obtain a speedup gain of 2.4 to 5.1 by the step size optimization. Since the overhead caused by the step size optimization was very small, the speedup gain was roughly equal to the ratio of average iteration steps of the step size optimization and constant step size.

Thirty-City TSP: Setting $D = 1$, (27)–(29) become

$$-2.31 > F, \tag{39}$$
$$A > 1.16, \tag{40}$$
$$F > -A. \tag{41}$$

Since we need not satisfy (39), we set $A = 1.2$ and $F_f = -0.2$, which satisfy (40) and (41). This time we set $t_d = 3000$ and changed F_i. The remaining parameters were set by (37) and (38). Table II shows the performance for 100 trials. The step size optimization performed better than the constant step size in average tour length and a speedup gain of 3 to 5 was obtained. Fig. 1 shows the distribution of solutions for $F_i = 1.5$ by the step size optimization which is much better than that in [6] and comparable to that in [2].

TABLE I
NUMBER OF OPTIMAL SOLUTIONS, AVERAGE ITERATION STEPS, AND SPEEDUP RATIO FOR THE 10-CITY TSP BY STEP SIZE OPTIMIZATION AND CONSTANT STEP SIZE (1000 TRIALS, $A = 1.0$, $F_i = 1.0$, $F_f = -0.2$, $\Delta F = 0.001$)

	Optimized		Constant		
t_d	Opt.	Steps	Opt.	Steps	Speedup
Fixed	987	197	997	995	5.1
100	994	298	998	1095	3.7
200	1000	370	999	1139	3.1
300	1000	456	999	1203	2.6
400	1000	543	999	1277	2.4

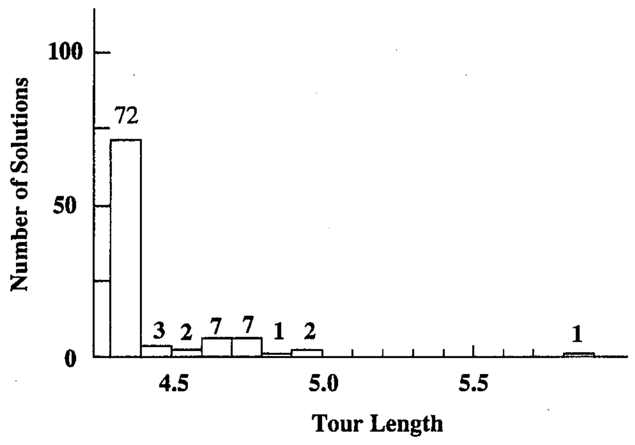

Fig. 1. Number of solutions versus tour length of 30-city TSP by step size optimization (100 trials, $F_i = 1.5$, $F_f = -0.2$, $\Delta F = 0.01$, $A = 1.2$, $\alpha = 0.0001$, optimum tour length = 4.312).

V. LSI MODULE PLACEMENT PROBLEM

A. Problem Formulation

When we implement a logic circuit into a gate array, first we divide the logic circuit into sub-circuits called modules and we divide the chip area of the gate array into sub-areas called slots, which are equal in number to the number of modules. An LSI module

TABLE II
AVERAGE TOUR LENGTH, NUMBER OF FEASIBLE SOLUTIONS, AVERAGE ITERATION STEPS, AND SPEEDUP RATIO FOR THE 30-CITY TSP BY STEP SIZE OPTIMIZATION AND CONSTANT STEP SIZE (100 TRIALS, $A = 1.2$, $F_f = -0.2$, $\Delta F = 0.001$)

	Optimized			Constant			
F_i	Length	Feasible	Steps	Length	Feasible	Steps	Speedup
3	4.55	87	4606	4.67	95	15187	3.3
2.5	4.54	97	3877	4.71	96	15115	3.9
2.0	4.47	95	3904	4.60	95	13667	3.5
1.5	4.45	95	3753	4.60	81	18624	5.0

placement problem is then to place each module with each slot without overlapping so that the total wiring length is minimized. As shown in Fig. 2(a) each module is connected to other modules or terminals by wiring. The slots are classified as external slots to which terminals are assigned and as internal slots to which modules are assigned [see Fig. 2(b)]. As shown in Fig. 2(c), each module is placed to one of the slots so that the wiring length calculated by the Manhattan distance is minimized under the given terminal assignments to external slots. Assume that we have N modules and N slots, and assign N variables x_{ik} $(k = 1, \cdots, N)$ to module i $(i = 1, \cdots, N)$. Assume also that if x_{ik} is 1, module i is assigned to slot k and if 0, it is not. Then similar to the TSP the energy function is defined by

$$\begin{aligned} E = \frac{A}{2}&\left\{\sum_i\left(\sum_k x_{ik}-1\right)^2+\sum_k\left(\sum_i x_{ik}-1\right)^2\right\} \\ &+\frac{B}{2}\sum_j\sum_l\sum_{i\neq j}\sum_{k\neq l} d_{kl}c_{ij}x_{ik}x_{jl} \\ &+B\sum_m\sum_i\sum_k d_{km}c_{im}x_{ik}e_m \\ &+\frac{F}{2}\sum_i\sum_k x_{ik}^2 \end{aligned} \tag{42}$$

where

1) d_{kl}: the Manhattan distance between slot k and slot l;
2) c_{ij}: the number of wirings between module i and module j;
3) c_{im}: the number of wirings between module i and external slot m;
4) $e_m = 1$: external terminal $(m = 1, \cdots, o)$;
5) $A(>0)$: a weight for constraints;
6) $B(>0)$: a weight for the objective function; and
7) F: a weight which is decreased during integration.

Assume that there are I_x internal slots in the x direction and I_y internal slots in the y direction. Thus, $N = I_xI_y$. Let the coordinates of slots k and l be (k_x, k_y) and (l_x, l_y), respectively. Then

$$\begin{aligned} k &= k_x + I_x(k_y - 1) \\ l &= l_x + I_x(l_y - 1). \end{aligned} \tag{43}$$

The Manhattan distance between slots k and l is given by

$$d_{kl} = |k_x - l_x| + |k_y - l_y|. \tag{44}$$

In (42) the first term is for the constraint, the second term is for calculating the Manhattan distance between modules, the third term is for calculating the Manhattan distance between modules and external terminals, and the fourth term is to decrease the diagonal elements during integration. The reason why $\frac{1}{2}$ is added to the second term while it is not added to the third term is that in the second term the distance between the same two modules is counted twice. Taking the partial derivative of (42) with respect to x_{ik} gives

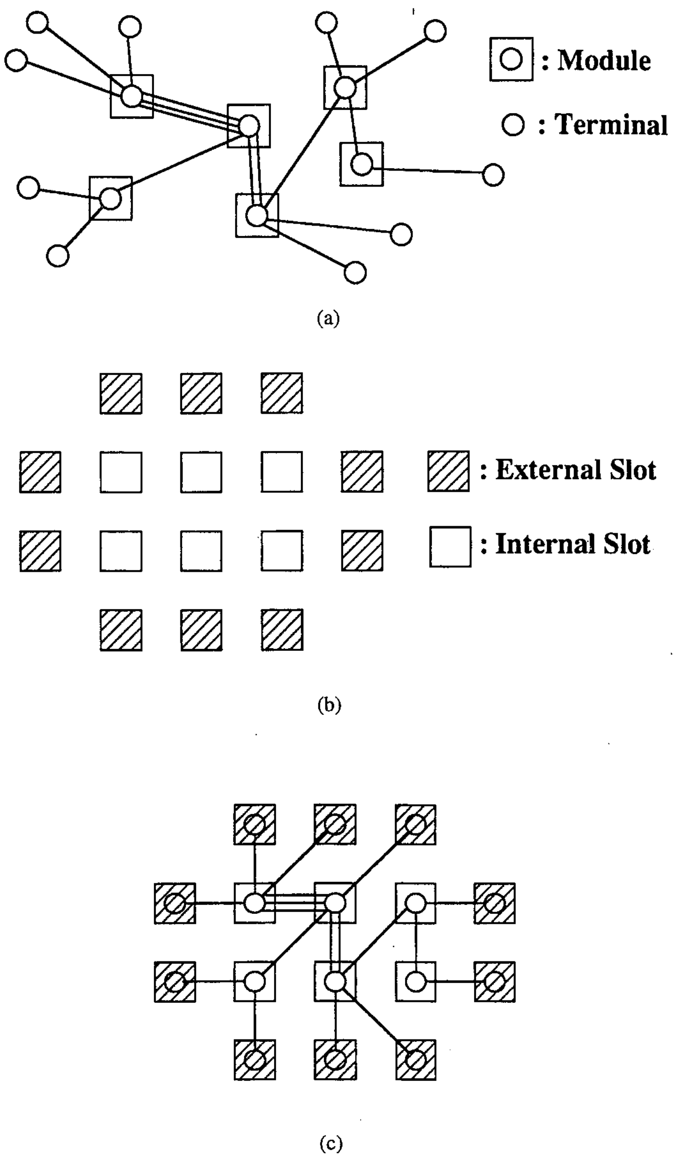

Fig. 2. LSI module placement problem ([8], Fig. 1).

$$\begin{aligned} \frac{\partial E}{\partial x_{ik}} = A&\left\{\left(\sum_j x_{ij}-1\right)+\left(\sum_j x_{jk}-1\right)\right\} \\ &+B\sum_{j\neq i}\sum_{l\neq k} k_{kl}c_{ij}x_{jl}+B\sum_m d_{km}c_{im}e_m \end{aligned} \tag{45}$$

Assuming $B = 1$, all the feasible solutions become stable if

$$\begin{aligned} &\sum_{j\neq i}\sum_{l\neq k} d_{kl}c_{ij}x_{jl}+\sum_m d_{km}c_{im}e_m + C \\ &\quad < 0 \quad \text{for} \quad x_{ik} = 1 \end{aligned} \tag{46}$$

$$\begin{aligned} &\sum_{j\neq i}\sum_{l\neq k} d_{kl}c_{ij}x_{jl}+\sum_m d_{km}c_{im}e_m \\ &\quad > 0 \quad \text{for} \quad x_{ik} = 0 \end{aligned} \tag{47}$$

Now we evaluate the maximum of the first term of (46). For a feasible solution, one element in the jth row or the lth column of x_{jl} is 1 and the remaining elements are 0. Then evaluating the maximum of the first term by the maximum of the jth column of c_{ij},

$$\begin{aligned}
&\max_{i,k} \sum_{j\neq i} \sum_{l\neq k} d_{kl} c_{ij} x_{jl} \\
&\leq \max_{i,k} \left(\max_{j\neq i} c_{ij} \sum_{l\neq k} d_{kl} \right) \\
&\leq \max_{i} \left(\max_{j\neq i} c_{ij} \right) \max_{k} \left(\sum_{l\neq k} d_{kl} \right) \\
&\leq \left(\sum_{l\neq 1} d_{k1} \right) \max_{i} \left(\max_{j\neq i} c_{ij} \right) \\
&= \frac{(I_x + I_y - 2)N}{2} \max_{i} \left(\max_{j\neq i} c_{ij} \right). \qquad (48)
\end{aligned}$$

The fourth term in (48) is derived according to the fact that the summation of the distance between a slot and the remaining slots is maximized when the slot is one of the slots in the four corners. Next, evaluating the maximum of the first term by the maximum of the jth column of d_{kl} we get

$$\begin{aligned}
\max_{i,k} \sum_{j\neq i} \sum_{l\neq k} d_{kl} c_{ij} x_{jl} &\leq \max_{i,k} \left(\sum_{j\neq i} c_{ij} \max_{l\neq k} d_{kl} \right) \\
&\leq (I_x + I_y - 2) \max_{i} \sum_{j\neq i} c_{ij}. \qquad (49)
\end{aligned}$$

From (48) and (49), the maximum of (46) is evaluated by

$$\begin{aligned}
&\max_{i,k} \left\{ \sum_{j\neq i} \sum_{l\neq k} d_{kl} c_{ij} x_{jl} + \sum_{m} d_{km} c_{im} e_m \right\} \\
&\leq \max_{i} \left\{ (I_x + I_y - 2) \left[\min \left(\frac{N}{2} \max_{j} c_{ij}, \sum_{j\neq i} c_{ij} \right) \right] \right. \\
&\quad \left. + \max_{k} \left(\sum_{m} d_{km} c_{im} e_m \right) \right\} \\
&= D_{\max}. \qquad (50)
\end{aligned}$$

Then from (46) and (50) all the feasible solutions become stable if

$$F < -D_{\max}. \qquad (51)$$

Similar to the TSP we can destabilize infeasible solutions as follows.

1) The ith row sum and the kth column sum of x_{ik} are 1 or 0.
This condition corresponds to changing some states from 1 to 0 in a feasible solution. Since the first term of (45) becomes $-2A$ when the ith row sum and kth column sum of x_{ik} are zero, the infeasible solutions become unstable when

$$A > \frac{D_{\max}}{2}. \qquad (52)$$

2) The ith row sum or kth column sum of x_{ik} is more than 1.
When $x_{ik} = 1$ and the kth column sum is more than 1, if

$$A + F > 0, \qquad (53)$$

Equation (45) is positive. Thus from (8) this solution is unstable.

TABLE III
TOTAL WIRING LENGTH OF REAL DATA BY CONVENTIONAL METHODS ([8], TABLE IV)

Data type	SA	Min-Cut	NN
Real Data 1	17875	19864	21576
Real Data 2	17659	19751	19273
Real Data 3	7000	7375	8456
Real Data 4	8014	9059	9046

TABLE IV
PARAMETER SETTING FOR LSI MODULE PLACEMENT ($B = 1$)

Data	D_{max}	A	F_i	F_f	ΔF	t_d
Real data 1	6492	8000	3000	-6500	-50	1000
Real data 2	4611	6500	1000	-5000	-50	1000
Real data 3	2784	4000	2000	-2800	-50	1000
Real data 4	2842	4000	3000	-2900	-50	1000

TABLE V
TOTAL WIRING LENGTH AND THE NUMBER OF INTEGRATION STEPS FOR REAL DATA 4 AGAINST F_f AND A BY THE STEP SIZE OPTIMIZATION AND THE CONSTANT STEP SIZE ($\alpha = 0$)

		Optimized		Constant		
A	F_f	Length	Steps	Length	Steps	Speedup
4000	-2900	8309	2578	8305	4253	1.6
4000	-2000	8018	2533	8435	5820	2.3
4000	-1000	7938	3673	-*	-	-
3000	-1000	8214	3199	8289	14795	4.6

*: A feasible solution was not obtained within 20,000 iterations.

From the above discussion, if from (51) to (53) hold, the stable solutions satisfy constraints.

B. Computer Simulations

We used four sets of real data with 45 modules (slots) as used in [8] to evaluate the step size optimization. Table III [8] shows the best solutions obtained by the simulated annealing method (SA) [10], [11], the min-cut method [12], and neural networks (NN). With neural networks, the diagonal elements were set to zero.

The second column of Table IV shows $D_{\max}$ calculated using (50). Unlike the TSP, if we set A around $D_{\max}/2$ and we set the absolute value of F_f small enough to satisfy (53), convergence was very slow and the quality of solution was not so good. Therefore, using $D_{\max}$ we selected A and F_f so that they satisfy (51)–(53) as shown in the third and fifth columns of the table. Parameters F_i, ΔF, and t_d are also shown in the table. The remaining parameters were set by (38).

Because of the third term in (42) or (45), the degeneration of variables does not occur even if we set $\alpha = 0$ in (31). Table V shows the total wiring length and the number of integration steps for real data 4 by the step size optimization and the constant step size when F_f and A are changed. For step size optimization, performance was better when A was large and the absolute value of F_f was small. The same tendency occurred for the remaining data sets. But with the constant step size, the convergence was unstable and no improvement was obtained when F_f was increased. For $A = 4000$ and $F_f = -1000$,

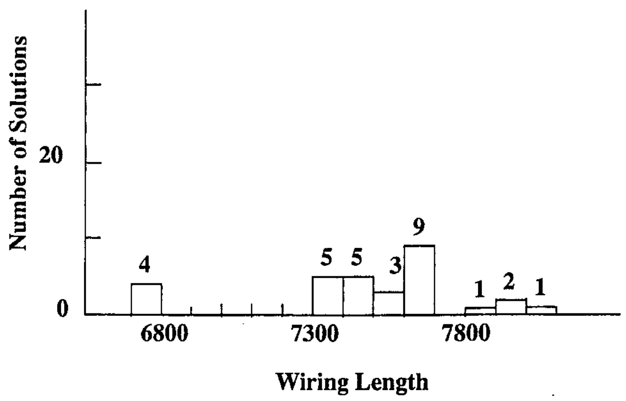

Fig. 3. Number of solutions versus wiring length for real data 3 (30 trials, $F_i = 2000$, $\Delta F = 50$, $F_f = -1000$, $A = 4000$, $\alpha = 0.0001$).

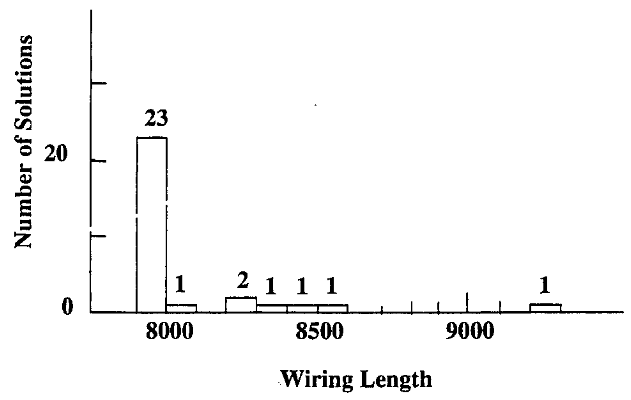

Fig. 4. Number of solutions versus wiring length for real data 4 (30 trials, $F_i = 3000$, $\Delta F = 50$, $F_f = -1000$, $A = 4000$, $\alpha = 0.0001$).

a feasible solution was not obtained within 20,000 iterations. For $A = 4000$ and $F_f = -2900$ vertices which satisfied constraints were all stable. Thus integration was stabilized and the speedup gain was only 1.6.

Setting parameters as shown in Table IV except $F_f = -1000$ we got the solutions by the step size optimization. Table VI shows the wiring lengths and the number of integration steps for $\alpha = 0$ and $\alpha = 0.0001$ with 30 trials. The solutions obtained were all feasible. The quality of the solution by the step size optimization was comparable with that of the SA given by Table III. For real data 1 and 2 the SA performed better while for real data 3 and 4 the step size optimization with $\alpha = 0$ performed better. The minimum wiring length for $\alpha = 0.0001$ was smaller than or equal to that of $\alpha = 0$ for four cases but the average wiring length was inferior. Figs. 3 and 4 show the distribution of solutions for real data 3 and 4, respectively. The distribution of solutions given by Fig. 4 is converged to the minimum while that of Fig. 3 is scattered, which means that for the latter parameter tuning is still necessary to obtain better quality of solutions for a small positive α.

VI. Discussion

For the step size optimization, convergence was accelerated two to five times compared to the constant step size. To further accelerate the convergence we tried the second order model derived from (14):

TABLE VI
Total Wiring Length of Real Data by Step Size Optimization ($F_f = -1000$)

	$\alpha = 0$		$\alpha = 0.0001$, 30 trials		
Data type	Length	Steps	Length		Steps
			Min	Avg	
Real data 1	18032	2420	18018	18209	2555
Real data 2	17929	2620	17773	17932	2649
Real data 3	6719	2426	6719	7481	2896
Real data 4	7938	3673	7886	8057	3188

$$\mathbf{x}(t) = \mathbf{x}(0) - t[T\mathbf{x}(0) + \mathbf{b}] + \frac{t^2 T}{2}[T\mathbf{x}(0) + \mathbf{b}]. \qquad (54)$$

The step size for the ith component of $\mathbf{x}$ to reach the surface of the hypercube is given by solving

$$\left\{\frac{T}{2}[T\mathbf{x}(0) + \mathbf{b}]\right\}_i t^2 - t[T\mathbf{x}(0) + \mathbf{b}]_i + x_i(0) - x_i(t) = 0. \qquad (55)$$

We implemented the above second order model for solving the TSP, but it did not work. With the same parameter setting as the first order model all the components of $\mathbf{x}$ became 1 after some iterations, and a feasible solution could not be reached. By setting a larger value for α, we could obtain feasible solutions but their quality was inferior to that of the first order model. This may mean that the exact calculation of a solution path makes the solution process determinate.

Comparing the TSP and the LSI placement problem, we saw in the former, A and the absolute value of F_f needed to be small enough, while in the latter A did not need to be small to obtain a good quality of solutions. Additionally the convergence process of the latter was much more stable. This may be explained as follows. With the TSP all the components of input vector $\mathbf{b}$ are the same. Thus n neurons assigned to a city behave in a similar manner when their initial values are close, but if we want to obtain good quality of solutions we need to set their initial values very close [6]. On the other hand for the LSI placement problem all the components of $\mathbf{b}$ are different and thus, even if we set the initial values at the center of the hypercube all the neurons behave differently.

VII. Conclusions

To accelerate convergence of the Hopfield network, the integration step sizes were determined so that at least one component of a variable vector reaches the surface of the hypercube. To avoid infinite loops caused by deterministic determination of step sizes, detection, and avoidance of infinite loops were also discussed. The computer simulations for traveling salesman problems and an LSI module placement problem showed that convergence was stabilized and accelerated and the quality of solution was improved compared to integration by a constant step size.

References

[1] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," in *Biolog. Cybern.* Berlin: Springer-Verlag, 1985, vol. 52, pp. 141–152.

[2] S. V. B. Aiyer, "Solving combinatorial optimization problems using neural networks with applications in speech recognition," Department of En-

gineering, University of Cambridge, Tech. Rep. CUED/F-INFENG/TR 89, Oct. 1991.
[3] A. H. Gee, S. V. B. Aiyer, and R. W. Prager, "An analytical framework for optimizing neural networks," *Neural Networks,* vol. 6, no. 1, pp. 79–97, 1993.
[4] S. Abe, "Global convergence and suppression of spurious states of the Hopfield neural networks," *IEEE Trans. Circuits Syst. I,* vol. CAS-40, no. 4, pp. 246–257, Apr. 1993.
[5] S. P. Eberhardt, T. Daud, D. A. Kerns, T. X. Brown, and A. P. Thakoor, "Competitive neural architecture for hardware solution to the assignment problem," *Neural Networks,* vol. 4, pp. 431–442, 1991.
[6] S. Abe and A. H. Gee "Global convergence of the Hopfield neural network with nonzero diagonal elements," *Trans. IEEE Circuits Syst. II,* vol. 41, no. 12, Dec. 1994.
[7] J.-H. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 1405–1422, Nov. 1989.
[8] H. Date, M. Seki, and T. Hayashi, "LSI module placement methods using neural computation networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN-90),* San Diego, CA, June 1990, vol. III, pp. 831–836.
[9] D. E. Van den Bout and T. K. Miller, III, "Graph partitioning using annealed neural networks," *IEEE Trans. Neural Networks,* vol. 1, no. 2, June 1990.
[10] S. Kirkpatrick *et al.,* "Optimization by simulated annealing," *Science,* vol. 220, pp. 671–680, 1983.
[11] H. Date and T. Hayashi, "Parameterization of simulated annealing method for LSI module placement problems," IEICE Tech. Rep., VLD89-45 (in Japanese), 1989-07.
[12] M. A. Breuer, "A class of min-cut placement algorithms," in *Proc. 14th DAC,* 1977, pp. 284–290.

Stability Analysis of Fuzzy Control Systems

S. G. Cao, N. W. Rees, and G. Feng

***Abstract*—A discrete-time fuzzy control system which is composed of a dynamic fuzzy model and a fuzzy state feedback controller is proposed. Stability of the fuzzy control system is discussed and a sufficient condition to guarantee the stability of the system is given in terms of uncertain linear system theory. The results in this paper improve our previous stability results. An example is used to show the proposed method.**

I. Introduction

Recently fuzzy logic control has become a very popular topic in control engineering, especially in industrial control. Fuzzy logic control (FLC) techniques were originally advocated by Zadeh [1] and Mamdani [2] as a means of both collecting human knowledge and experience and dealing with uncertainties in the control process. Since then FLC has attracted a great deal of public attention and a lot of work has been published in the field, e.g., [3]–[10].

Although fuzzy logic control systems have the advantage that no formal mathematical models are needed and the system uncertainties can be coped with, the design of FLC is still based on trial and error procedures because the closed loop system is essentially nonlinear. Recently, some useful stability analysis methods e.g., [6], [10], which are based on Lyapunov stability theory and Hyperstability theory [11], have been reported. But the stability conditions in those papers are conservative and difficult to check for some fuzzy systems. A new stability analysis method for fuzzy control systems based on uncertain linear system theory is proposed in [8], [9]. This method gives a new view on the design of fuzzy controller, but the stability conditions are still conservative.

This paper presents new results on the stability of fuzzy control system based on a fuzzy state space model and using the methods discussed in [8], [9]. The result in this paper improves our previous stability results [8], [9] and gives an easily checked stability condition. The fuzzy system is modeled by a family of linear state space models. Using conventional linear control system theory [12] we can get a state feedback control law for each local linear state space model. Then, we use uncertain linear system theory [13] to determine a global stability condition to guarantee the stability of the global closed-loop system which combines all local subsystems.

This paper is organized as follows. Section II discusses a dynamic fuzzy state space model and derives its global state space form. Section III discusses the analysis and design method of fuzzy control systems. Section IV discusses the design method of a fuzzy state observer. Section V shows a numerical example. Finally, the paper concludes with some brief remarks in Section VI.

II. Dynamic Fuzzy Local Model

We use the following fuzzy model to represent a complex single-input single-output system which includes both approximation inference rules and local analytic models.

$$\begin{aligned} R^l: \quad & \text{IF} \quad z_1 \text{ is } F_1^l \quad \text{AND} \cdots z_{2n} \text{ is } F_{2n}^l \\ & \text{THEN} \quad x(t+1) = A_l x(t) + B_l u(t) \\ & \qquad y_l(t) = C_l x(t) + d^l \\ & \qquad l = 1, 2, \cdots, m \end{aligned} \tag{2.1}$$

where R^l denotes the lth approximation inference rule. m is the number of approximation inference rules, $y_l(t+1)$ is the output from the lth implication; $y(t)$ and $u(t)$ are the output and input variables of the system, $x(t)$ are the state variables, d^l is the constant term of the lth subsystem, and

$$\begin{aligned} z(t) &= [y(t), \cdots, y(t-n+1), u(t), \cdots, u(t-n+1)]. \\ &:= (z_1, z_2, \cdots, z_{2n}). \end{aligned} \tag{2.2}$$

The final output of the system is inferred by taking the weighted average of the $y_l(t+1)$'s. Because the model in (2.1) only represents the properties of the system in a local region, it is called a fuzzy dynamic local model.

Let $\mu_l[z(t)]$ be the normalized membership function of the inferred fuzzy set F^l where $F^l = \prod_{i=1}^{2n} F_i^l$ and let

$$\sum_{l=1}^{m} \mu_l = 1. \tag{2.3}$$

By using the fuzzy inference methods described in [6], [8], [9], that is, using center-average defuzzifier, product inference, and singleton

Manuscript received July 5, 1994; revised February 20, 1995.
The authors are with the Department of Systems and Control, School of Electrical Engineering, University of New South Wales, Sydney 2052, NSW, Australia.
Publisher Item Identifier S 1083-4419(96)00430-X.