Structural Determination of Multilayered Large-Signal Neural-Network HEMT Model

Kazuo Shirakawa, *Member, IEEE*, Masahiko Shimizu, Naofumi Okubo, and Yosimasa Daido, *Member, IEEE*

***Abstract*—This paper reports on the structure of a large-signal neural-network (NN) high electron-mobility transistor (HEMT) model as determined by a pruning technique and a genetic algorithm. The bias-dependent intrinsic elements of an HEMT's equivalent circuit are described by a generalized multilayered NN whose inputs are the gate-to-source bias ($V_{\rm gs}$) and the drain-to-source bias ($V_{\rm ds}$). Using $C_{\rm gs}$ data as an example, we began by experimentally examining some of the features of the multilayered NN model to obtain rules-of-thumb on choosing training parameters and other information for succeeding studies. We then developed and studied a novel pruning technique to optimize the $C_{\rm gs}$ NN model. Excessively large NN configurations can be reduced to an appropriate size by means of a weight decay, which is based on the analysis of a synaptic connection's activity. Finally, we employed a genetic algorithm for the same purpose. By representing the configuration of a standard multilayered NN as a chromosome, the optimum configuration of a $C_{\rm gs}$ model was obtained through a simulated evolution process. For this approach, the configuration of an NN that simultaneously represents seven intrinsic elements ($C_{\rm gs}$, R_i, $\cdots$, $C_{\rm ds}$) of an equivalent circuit was also shown for comparison to previous work. We successfully obtained simplified NN models using both approaches. The advantages and disadvantages of these two approaches are discussed in the conclusion. To our knowledge, this is the first report to clarify the general process of building an NN device model.**

***Index Terms*— Genetic algorithm, HEMT, large-signal model, neural network, pruning algorithm.**

I. Introduction

THE popularization of high-frequency wireless systems demands an effective method to quickly and accurately design components. Large-signal modeling for active devices, such as high electron-mobility transistors (HEMT's), is the most fundamental technology available to meet this demand. Recently, novel models based on neural networks (NN's) have been reported [1]–[3]. As in closed-form equation models [4], [5], NN's have the adaptability necessary to represent a strong nonlinearity. However, closed-form equation models must be changed to fit the particular device type. NN's can support any number and type of nonlinear features by simply changing their configurations.

As with the look-up table model [6], NN's are highly accurate. However, the look-up table model requires a large database, which can sometimes reach several megabytes. The response of the NN can only be determined by using a few hundred coefficients [3].

The NN model overcomes the drawbacks of the closed-form and look-up table models, while retaining their advantages. However, it is difficult to determine the proper configuration with this attractive NN. As a result, the NN models reported are almost always limited to a three-layer configuration, which can represent either a single large-signal element [2] or all elements with a large number of neurons [1].

In this paper, we report on a pruning approach [7], [8] and a genetic approach [9]–[11] to determine the optimum multilayered large-signal NN model of an HEMT.

To enable its future implementation on standard circuit simulators, we characterized the HEMT's large-signal behavior in terms of the intrinsic elements of a conventional small-signal equivalent circuit [12]. The bias-dependent elements are described by an NN composed of an arbitrary number of neurons and layers (called a generalized multilayered NN).

We first examined some of the features of multilayered NN's for a $C_{\rm gs}$ model experimentally, in order to collect useful information for succeeding studies.

Compared to other significant large-signal elements such as gm, $C_{\rm gs}$ is more difficult to model with the conventional closed-form equation [5]. Thus, the $C_{\rm gs}$ model is preferable for revealing the usefulness of the NN approach.

We then developed and studied a novel pruning technique to determine the structure of an NN. This technique causes NN's with larger than necessary configurations to lose their unimportant synaptic connections (including those of neurons) during back-propagation training to eventually achieve the optimum configuration [7], [8]. The removal of these connections are done by a weight decay based on an analysis of the activity of the synaptic connections.

We then applied a genetic algorithm [9], [10] for the same purpose. The configuration of the NN is assumed to be a characteristic of a virtual creature [11]. By simulating the evolution of a group of these virtual creatures, we obtained not only a $C_{\rm gs}$ model, but also a model that simultaneously

Manuscript received November 21, 1996; revised June 25, 1997.

K. Shirakawa, M. Shimizu, and N. Okubo are with the Wireless Communication Systems Laboratory, Fujitsu Laboratories Ltd., Nakahara-ku, Kawasaki 211, Japan (e-mail: nato@flab.fujitsu.co.jp).

Y. Daido is with the Kanazawa Institute of Technology, Information Theory and Design Core, Nonoichi, Ishikawa 921, Japan.

Publisher Item Identifier S 0018-9480(98)07252-4.

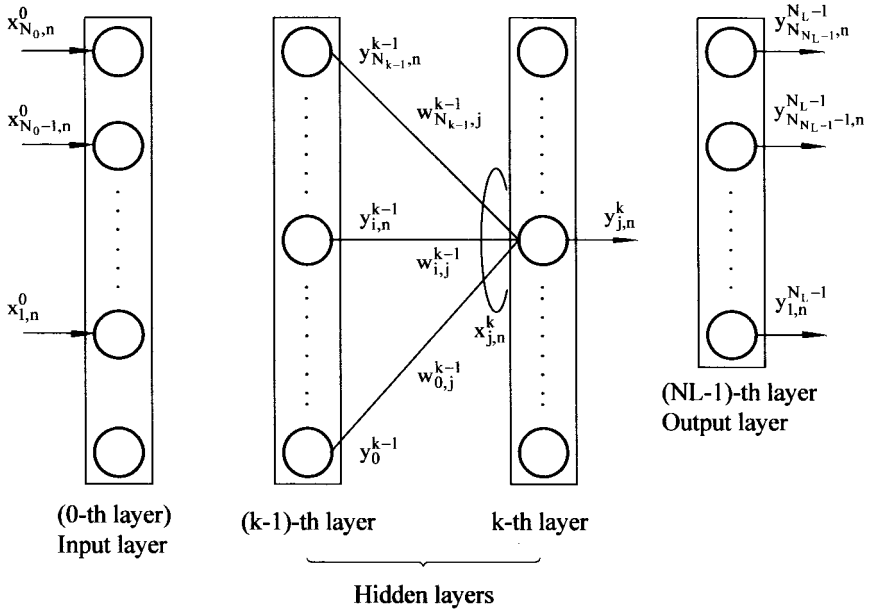

Fig. 1. Multilayered NN. Each circle is a neuron, and the boxes enclosing several neurons are layers. At the far left is the input layer and at the far right is the output layer. The neuron at the bottom of each layer, excluding the output layer, provides bias values to the neurons in the succeeding layer.

represents all the intrinsic elements of an HEMT's equivalent circuit in order to compare it with our previous work [3].

Finally, we evaluated the advantages and disadvantages of the above two methods.

II. Multilayered NN's

For its future implementation on common-circuit simulators, we characterized the large-signal behavior of an HEMT in terms of the intrinsic elements of a conventional small-signal equivalent circuit. The measured data of these elements versus $V_{\rm gs}$ and $V_{\rm ds}$ was obtained from multibias S-parameter measurements [12].

Fig. 1 shows the generalized multilayered NN used to model the data. In this figure, we assume that the number of layers is N_L and that the kth layer includes N_k+1 neurons. The bottom neuron of each layer, excluding the output layer, supplies the bias to the neurons in the succeeding layer.

We abbreviated the network configuration to $(N_0, N_1, \cdots, N_{N_L-1})$ and express an NN having such a configuration as $(N_0, N_1, \cdots, N_{N_L-1})-{\rm NN}$.

We adopted a batch-mode back-propagation algorithm to train this NN [3].

To obtain a training convergence and good extrapolations, we normalized the raw data and bias voltages within a common range of 0–1 throughout this paper [3]. We then defined the set of input data vectors $\mathbf{x}_n^0$ and teaching data vectors $\mathbf{d}_n$ as follows:

$$\begin{aligned}\mathbf{x}_n^0 &= (x_{1,n}^0, x_{2,n}^0)\\ &= \{s(V_{{\rm gs}_n}; a_1, b_1), s(V_{{\rm ds}_n}; a_2, b_2)\} \qquad (1)\\ \mathbf{d}_n &= (d_{1,n}, d_{2,n}, \cdots, d_{7,n})\\ &= \{s[C_{\rm gs}(V_{{\rm gs}_n}, V_{{\rm ds}_n}); \alpha_1, \beta_1], \cdots\\ &\quad s[C_{\rm ds}(V_{{\rm gs}_n}, V_{{\rm ds}_n}); \alpha_7, \beta_7]\} \qquad (2)\end{aligned}$$

where subscript $n(n=0, \cdots, N-1)$ denotes the number of bias points, and s denotes a sigmoid function defined as

$$s(x; a, b) = \frac{1}{1+\exp[-a(x+b)]}. \qquad (3)$$

In (3), a and b are parameters. Then, a_i, $b_i(i=1, 2)$ in (1) and α_i, $\beta_i(i=1, \cdots 7)$ in (2) are determined independently according to the desired data range. For example, if $V_{\rm gs} = -0.8$ to 0.4 V, then $a_1 = 1.0$ and $b_1 = 0.2$ would be good choices.

Generally, when the nth data is supplied to the NN, the input to the jth neuron in the kth layer is described as

$$x_{j,n}^k = \sum_{i=0}^{N_k-1} w_{i,j}^{k-1} y_{i,n}^{k-1}, \qquad k=1, \cdots, N_L-1 \qquad (4)$$

where $w_{i,j}^{k-1}$ denotes the weighting factor that connects the ith neuron in the $(k-1)$th layer with the jth neuron in the kth layer.

The output from this neuron is represented as

$$y_{j,n}^k = s(x_{j,n}^k; 1, 0), \qquad k=0, \cdots, N_L-1. \qquad (5)$$

In the back-propagation algorithm, the network adaptation E is evaluated by

$$E = \sum_{n=0}^{N-1} \left\{ \tfrac{1}{2} \sum_{i=1}^{N_L-1} (y_{i,n}^{N_L-1} - d_{i,n})^2 \right\} = \sum_{n=0}^{N-1} E_n \qquad (6)$$

where $y_i^{N_L-1}(i=1, \cdots, N_L-1)$ is the network output (we fixed the subscript of the bias neuron to zero for convenience in programming, so the subscript of the neurons in the output layer begins with one).

The correction terms in the training iterations are given by

$$\delta w_{i,j}^{k-1}(m) = \eta \frac{\partial E}{\partial w_{i,j}^{k-1}} + \alpha \delta w_{i,j}^{k-1}(m-1) \qquad (7)$$

where m denotes the loop counter [3].

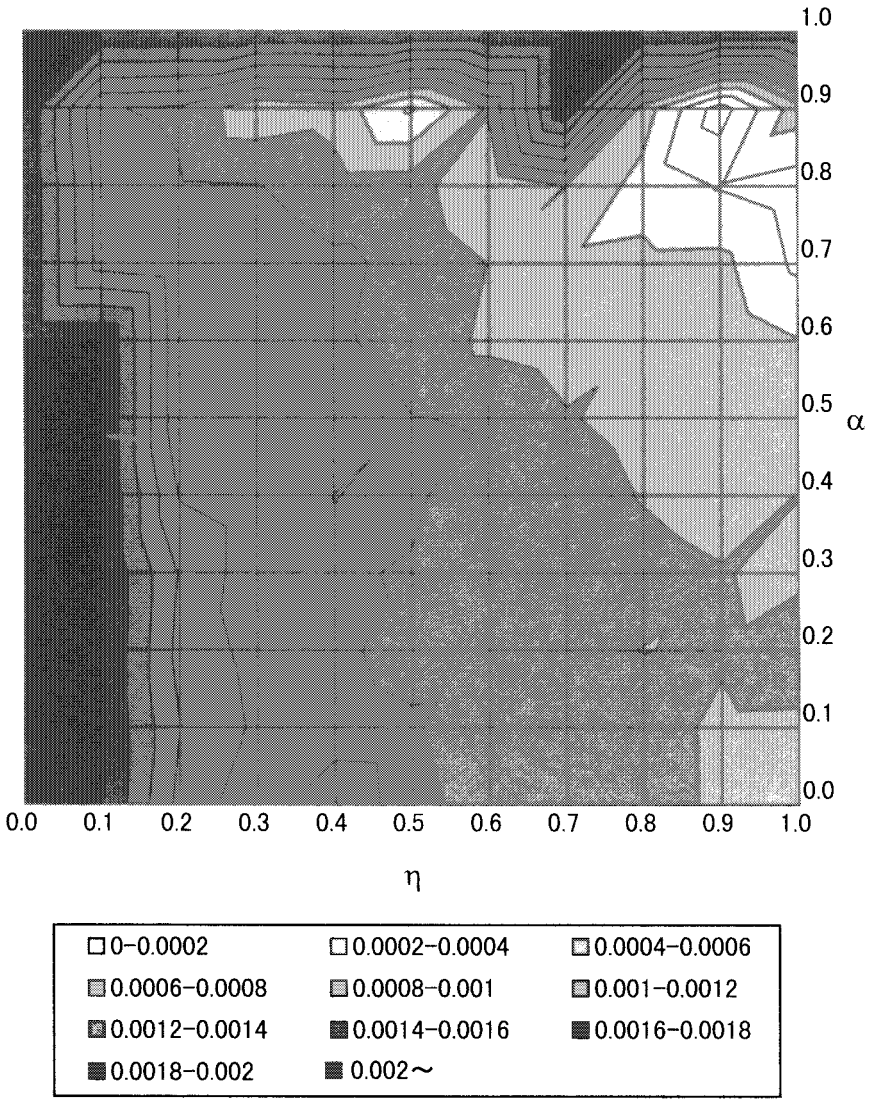

Fig. 2. Error contours in relation to learning rate η and momentum factor α. The range of E is defined above. The network configuration was (2, 5, 5, 1). The number of iterations was fixed at 10 000. A minimum error was stably achieved in the vicinity of $\eta = 0.9$ and $\alpha = 0.8$.

NN's require several parameters, not only for training, but also for specifying the network configuration. However, their effects on the training characteristics have not been sufficiently analyzed, especially in the case of multilayered NN's. Using a C_{gs} model as an example, we experimentally studied various features of a multilayered NN to obtain guidelines for subsequent studies. The data referred to throughout this paper corresponds to an HEMT with a 0.25-μm-long and 100-μm-wide gate.

Fig. 2 shows the fitting error contours versus the parameters η (learning rate) and α (momentum factor) in the case of an NN with a (2, 5, 5, 1) configuration. The number of iterations is 10 000. A minimum number of errors can be obtained in the vicinity of $\eta = 0.9$ and $\alpha = 0.8$. These values generate good results for NN's of different network configurations. Therefore, we fixed them at these values.

Fig. 3 shows the fitting error behavior versus the number of iterations for NN's with configurations of (2, 5, 1), (2, 5, 5, 1), and (2, 5, 5, 5, 1). It should be noted that the training curve of the (2, 5, 5, 1)-NN fluctuates near the end of the iterations. A similar phenomena can be observed with different combinations of network configurations, η, and α, but it is currently difficult to predict its cause. This test was also performed on NN's with other configurations, and we presume that it is acceptable to fix the maximum number of iterations to 10 000.

Moreover, the genetic approach [11] is supposedly much more time-consuming than other approaches [7], [8]. It is helpful to compare the calculation time of one training period against NN's of various configurations. Fig. 4 shows the calculation time required for one iteration relative to the network configuration. The machine we used was an HP-712/80. This graph allows us to predict the total training time and to assign the rational number of layers (and neurons) to our machine from the viewpoint of calculation costs.

III. Pruning Technique

A pruning technique [7], [8] is analogous to the creation of synaptic connections in actual life forms, i.e., many (or highly redundant) synaptic connections are first formed as

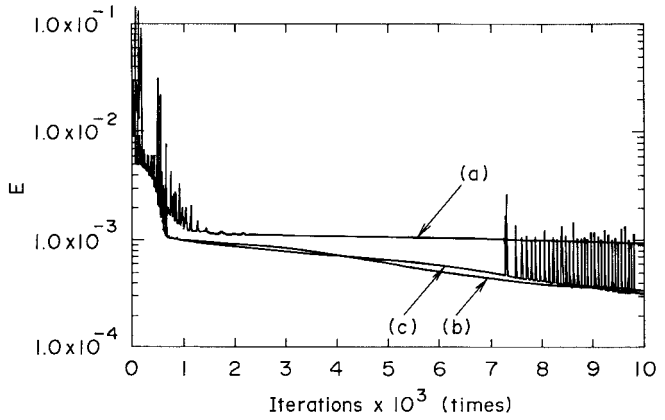

Fig. 3. Error versus number of iterations with $\eta = 0.9$ and $\alpha = 0.8$ used for each case. The graph shows the training curves of (a) a (2, 5, 1)-NN, (b) a (2, 5, 5, 1)-NN, and (c) a (2, 5, 5, 5, 1)-NN. Some fluctuations appear on the curves.

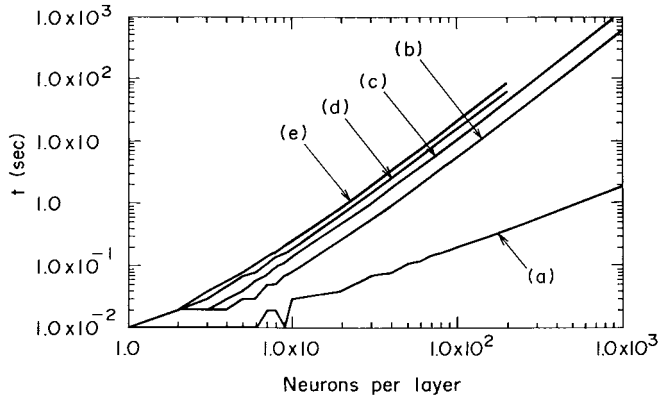

Fig. 4. Time consumption for one training cycle relative to the number of neurons per layer. Lines (a)–(e) represent NN's with 1–5 hidden layers. For example, the rational number of neurons for a five-layer (three hidden layers) NN corresponding to a single-hidden-layer NN composed of 1000 neurons is 40.

a blueprint in a gene. Unnecessary connections are then terminated according to elementary training (infancy) in a particular habitat. This initial training determines the basic response style of an individual.

A. Synaptic Connection Activities

The predominant part of previously proposed pruning techniques commonly focus on the magnitude of the weighting factors, which are minimized by including some cost function in an error used for the original back-propagation [7], [8]. However, since these magnitudes do not necessarily represent the complexity of an NN, the resultant NN can sometimes have only small weighting factors. We thus developed and studied a novel approach that evaluates the importance of synaptic connections according to their activity.

During one training period, the mean information flow through a weighting factor $w_{i,j}^k$ is represented as

$$r_{i,j}^{k+1} = \frac{1}{N}\sqrt{\sum_{n=0}^{N-1}[w_{i,j}^k(y_{i,n}^k - \overline{y_i^k})]^2} \tag{8}$$

where k denotes the number of layers, $y_{i,n}^k$ is the output from the ith neuron in the kth layer, n denotes the amount of data, and

$$\overline{y_i^k} = \frac{1}{N}\sum_{n=0}^{N-1} y_{i,n}^k. \tag{9}$$

At this point, the jth neuron in the kth layer accepts the net information

$$R_j^{k+1} = \frac{1}{N_k}\sqrt{\sum_{i=0}^{N_k-1}(r_{i,j}^{k+1})^2} \tag{10}$$

where N_k denotes the number of active neurons in the kth layer.

We then introduced the following new activity of $w_{i,j}^k$:

$$c_{i,j}^k = \frac{r_{i,j}^{k+1}}{R_j^{k+1}}. \tag{11}$$

B. Implementation and Measurement

Fig. 5 shows the flowchart for our pruning technique. We confined the pruning routine to take place within a normal back-propagation process [3], [7]. An initial NN of excessive size is generated, and the pruning threshold E_p is set to λE_0, where λ $(1.0 > \lambda)$ is a parameter and E_0 is the given objective error level.

To successfully eliminate the inessential connections, they should be removed after pertinent connections have been sufficiently specialized for certain stimulations (elementary training). Parameter λ is adjusted when this specialization is expected as having been completed, and we usually use a convenient value of 2–10.

If the output error E falls below E_p, we calculate the activity of each weighting factor and eliminate or decay it according to the following three rules.

Rule 1: If $|w_{i,j}^k| < 1.0$, then $w_{i,j}^k$ is terminated (for each neuron).

This means that the output of a sigmoid function is confined to a range between zero and one, and if a weighting factor smaller than one only acts to reduce the significance of the transmitted information, then it should be removed. This is done by assuming that "1 is small for $w_{i,j}^k$" and that "a significant $w_{i,j}^k$ is typically larger than unity after elementary training has finished."

Rule 2: If $c_{i,j}^k < 0.1$, then $w_{i,j}^k$ is decayed (for each neuron, except the biasing neuron). The decaying quantity we used is

$$\delta w_{i,j}^k = -\mu w_{i,j}^k \operatorname{sgn}(w_{i,j}^k) \tag{12}$$

where μ is a parameter to adjust the decaying speed for each iteration. We used 0.2 for micron. This decaying rate is kept until the next round of pruning occurs.

Rule 2 evaluates the activity of connection $w_{i,j}^k$ to the jth neuron in the kth layer from the viewpoint of net information flow.

Since Rule 2 is no longer effective for biasing neuron connections or neurons having a single connection, the output

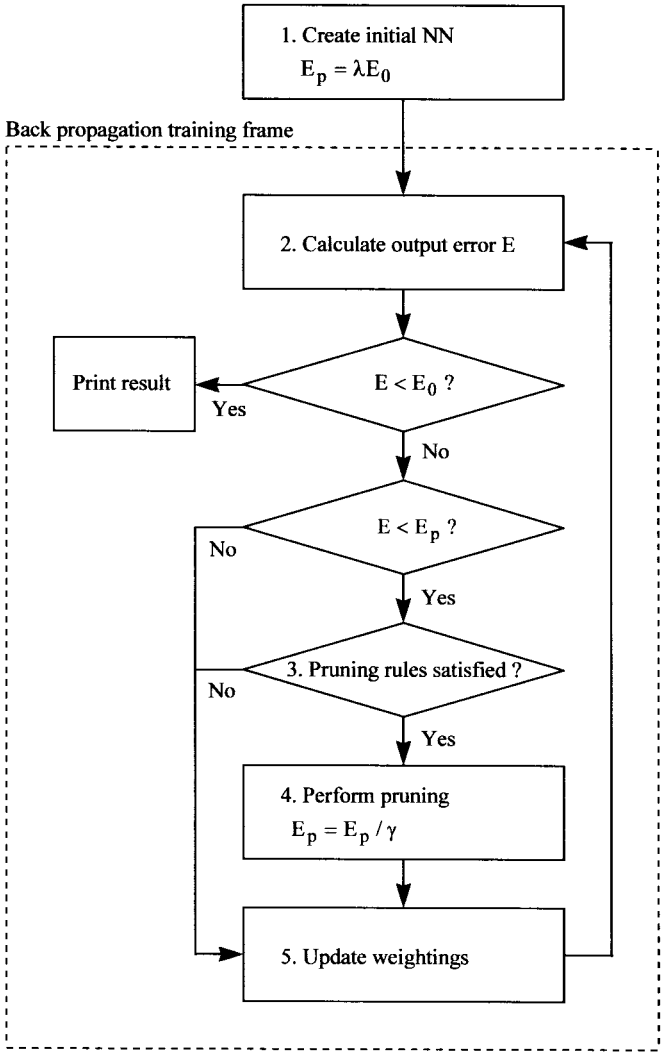

Fig. 5. Flowchart of pruning algorithm.

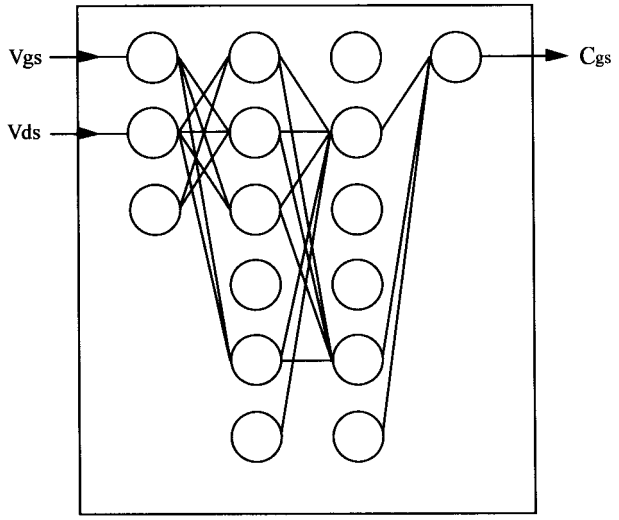

Fig. 6. NN model for $C_{\rm gs}$ obtained by pruning. Unnecessary connections and neurons have been terminated.

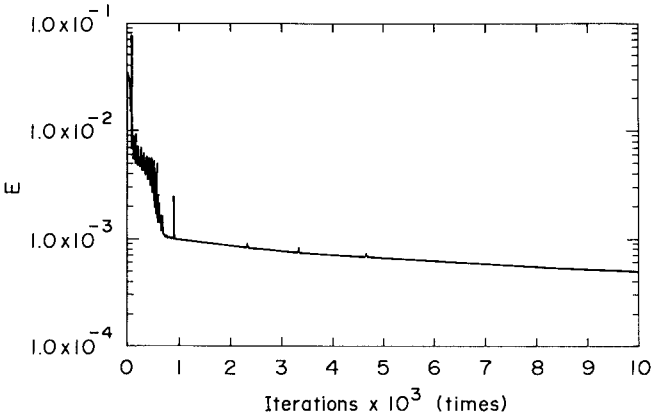

Fig. 7. Training curve of $C_{\rm gs}$ model with pruning.

error in which the weighting factor in question is forced to zero is used as an alternative to Rule 2. Thus, we defined Rule 3 as

$$w_{i,j}^k = \begin{cases} 0, & E(w_{i,j}^k = 0) < E_p \\ w_{i,j}^k, & E(w_{i,j}^k = 0) \geq E_p. \end{cases}$$

We then updated E_p to

$$E_p = \frac{E_p}{\gamma} \tag{13}$$

where $\gamma (1 < \gamma < \lambda)$ is the reduction rate of E_p. E_p then approaches E_0 according to how many times the NN underwent the pruning process. The significant weighting factors are updated with (7), and the inessential ones are decayed with (12). If E is not smaller than E_0, steps 2–5 in Fig. 5 are repeated.

We applied this algorithm to a (2, 5, 5, 1)-NN for modeling large-signal $C_{\rm gs}$. Fig. 6 shows the obtained NN. Due to the elementary training, our pruning can generate a sufficiently simplified network, as well as establish a good convergence. Fig. 7 shows its training curve. Even though some peaks resulting from a hard pruning appeared during the training (see Rule 1), elementary-trained NN's exhibit stable training. Moreover, note that no fluctuations appear on the training curve, despite their presence on the original (2, 5, 5, 1)-NN.

IV. Genetic Algorithm

A. Virtual Creatures

A genetic algorithm [9], [10] is a method used to solve problems such as optimizations via a computer simulation of the evolutionary process of the population of virtual creatures. A virtual creature has a chromosome that determines its degree of fitness to a particular habitat. This chromosome is composed of a set of genes.

We assume that the population includes N_p members, and that the ith individual has a chromosome G_i and a fitness value f_i defined as

$$G_i = \{g_0^i, g_1^i, \cdots, g_{N_g-1}^i\} \tag{14}$$

and

$$f_i = f(G_i) \tag{15}$$

where $g_j^i (j = 0, \cdots, N_g - 1)$ denotes each gene, and N_g is the number of genes in one chromosome. Function f in (15)

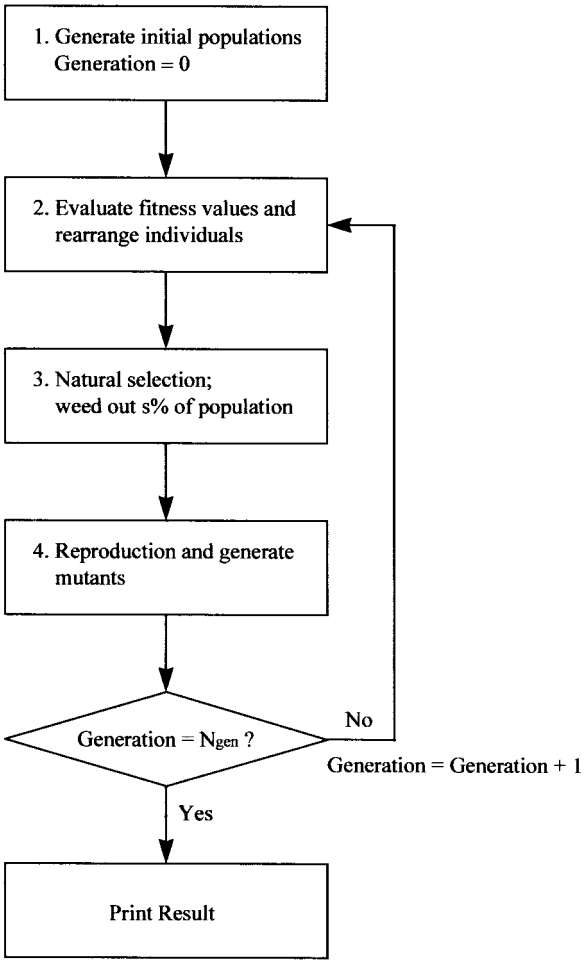

Fig. 8. Flowchart of genetic algorithm.

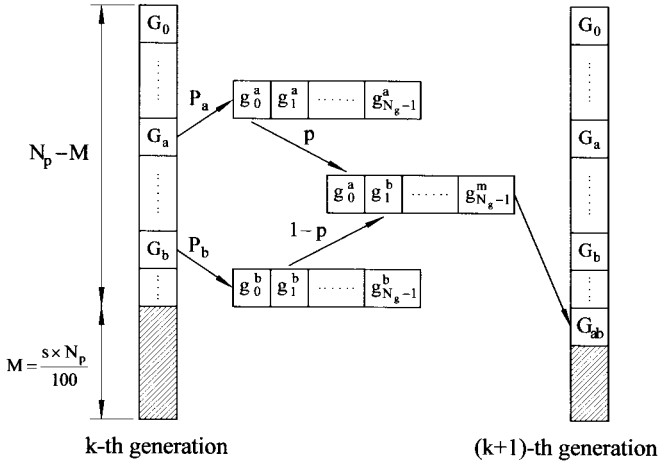

Fig. 9. Natural selection, reproduction (mutation), and generation shift. The block on the right represents the rearranged population in the order of increasing fitness at the kth generation. Individuals a and b are selected at a probability of Pa and Pb from the population at the kth generation, and generate their offspring ab at the $(k+1)$th generation. Hatched areas denote the individuals to be removed.

calculates the fitness value of the chromosome, which changes according to the type of application [11].

B. Algorithm

Fig. 8 is a flowchart of a genetic algorithm [11]. First we generate an initial population composed of N_p individuals and initialize the chromosome of each with random values. We then calculate each individual's fitness and rearrange them in order of increasing fitness. The population undergoes natural selection, reproduction, and the succession of generations. Steps 2–4 are repeated until either the population adequately fits the conditions or the number of generations reaches its maximum value.

Fig. 9 illustrates steps 3 and 4 (from Fig. 8), which are the significant steps in this algorithm. In this figure, the block on the left shows the rearranged population at the kth generation. The individual at the top of the block has the maximum fitness and the individual at the bottom has the minimum fitness.

The block on the right illustrates the population at the succeeding generation in the middle of the generation shift.

In natural selection, we weed out s% of the creatures from the individuals having the lowest fitness value, i.e., $M = N_p \times s/100$ individuals are terminated from the current (kth) population. This process is analogous to natural selection with s being the selection rate. Consequently, $N_p - M$ individuals survived. Among the survivors, and allowing for duplication, we pick M pairs of individuals and generate their offspring.

At this point, the ith individual is randomly selected by the following probability [11]:

$$P_i = \frac{f_i}{\frac{1}{N_p - M} \sum_{k=0}^{N_p - M - 1} f_k}. \tag{16}$$

This means that an individual with a higher fitness is inclined to have more offspring. This process is regarded as being analogous to reproduction.

In the reproduction process, when individuals a and b are selected as one of these pairs, the ith gene of their offspring is chosen from the ith gene in G_a at a probability of p or from the ith gene in G_b at a probability of $1 - p$. p is called the crossover rate. At the same time, each offspring's gene also suffers from a mutation at a probability of m%. This mutation rate should remain rather low because too many mutants will disrupt the evolutionary convergence of the population.

Finally, the offspring's (called ab) chromosome becomes

$$G_{ab} = \{g_0^a, g_1^0, g_2^a, \cdots, g_{n_g-1}^m\} \tag{17}$$

where $g_{N_g-1}^m$ denotes a mutated gene. We then replace the current population by the $N_p - M$ survivors and their M offspring and obtain the population of the next generation.

C. Implementation

To apply this concept to determine an NN model, we defined a translation rule between the chromosome and the configuration of an NN.

Since the number of neurons in both the input and output layers can be determined according to the model type (such as the C_{gs} model in Section III or the seven-intrinsic-element model [3], etc.), we do not have to include them as members of the chromosome. Thus,

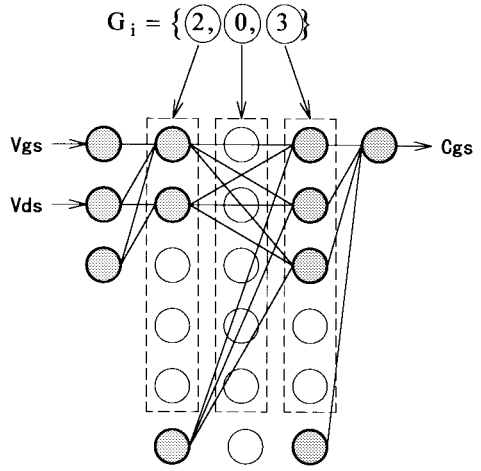

Fig. 10. Correspondence between chromosome G_i and the configuration of an NN. Neurons enclosed within dashed lines are the objects of a genetic algorithm. White neurons are inactive and gray neurons are active. The combination of the gray neurons illustrates an obtained NN model for $C_{\rm gs}$.

Rule 1: The length of the chromosome N_g indicates the maximum number of hidden layers. This shrinks the size of the searching space.

Rule 2: The value of the gene represents the number of neurons in each hidden layer.

Here, we allowed a gene to take multilevel integer values in the range of, e.g., 0–N_n–1. This multilevel integer notation is more compact and convenient than the binary gene notations that commonly appear in texts. Moreover, as the bias neuron is automatically added by our program, we can exclude its count from the value of a gene.

Rule 3: A gene of zero value indicates a null layer.

Fig. 10 illustrates a concrete example of these rules; an NN having two inputs, one output, and a chromosome described as $G_i = \{2, 0, 3\}$ corresponds to a four-layer NN. A gene of two in G_i represents two gray neurons.

Rule 4: A chromosome including n genes of zero value has $N_g C_n = N_g!/n!(N_{g-n})$ possible ways of describing an NN of the same configuration. In the above example, $\{2, 3, 0\}$ and $\{0, 2, 3\}$ are equivalent to G_i. We carefully avoided creating equivalent individuals in the initial populations to achieve as wide a variety of individuals as possible.

Rule 5: We defined the fitness value as

$$f_i = \begin{cases} \dfrac{1}{E_i}, & E_i \geq E_0 \\ \dfrac{1}{E_i}\left(1 + \dfrac{N - N_a}{N}\right), & E_i \leq E_0 \end{cases} \tag{18}$$

where E_i denotes the fitting error of the NN generated from chromosome G_i, and N_a is the total number of active neurons, i.e.,

$$N_a = \sum_{j=1}^{N_{L-2}} N_j, \qquad 0 \leq N_j \leq N_n. \tag{19}$$

This shows that we looked for the simplest configuration possible for an NN generating error values below E_0. In the case of Fig. 10, $N_n = 15$ and $N_a = 5$.

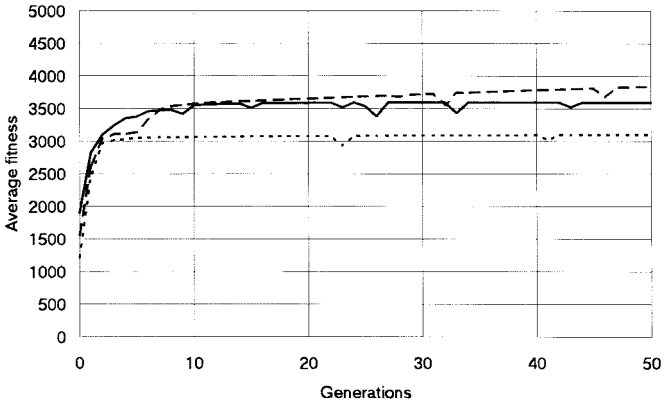

Fig. 11. Average fitness values relative to generation for a one-element model. The solid line is for $(N_n, N_p) = (10, 50)$, the dotted line denotes (7, 30), and the dashed line denotes (5, 10). Regardless of differences in the initial conditions, the values commonly approach the same configuration within 20 generations.

For every generation, we temporarily saved each individual's chromosome and synaptic weighting factors on a disc. If an NN in the next generation has a chromosome equivalent to one of its ancestors, we trained it with the inherited weighting factors. These saved weighting factors worked as the evolutionary gains of the former occupants and promoted the convergence [11]. As these gains are updated every generation, the chance to leave inheritance is open to each individual.

D. Measurement

We presumed that N_n represents the maximum number of neurons in one layer, N_L denotes the maximum number of hidden layers, and $N_{\rm gen}$ denotes the maximum number of generations. The parameters $s = 40\%$, $p = 50\%$, and $m = 1\%$, are used here.

We first determined the structure of $C_{\rm gs}$ model. The convergence stability was studied by supplying different N_n and N_p values. We assumed $E_0 = 0.000\,75$ and $N_{\rm gen} = 50$. Considering previous studies [3], it seemed adequate to set $N_L = 3$ to fit only one equivalent-circuit element. Fig. 11 charts the average fitness versus generations. In this figure, the lines correspond to the conditions of $(N_n, N_p) = (10, 50)$, (7, 30), and (5, 10). In all cases, the populations converge to the same configuration of $\{2, 3\}$. The NN for this configuration is also shown in Fig. 10. It is clearly seen that the populations approach convergence in the early stages of evolution. From these results, we can obtain an optimum configuration within 20 generations.

The fitting results for $C_{\rm gs}$ by using the predetermined NN is shown in [11]. A difference in error values of less than 0.0002 (3.0% of the root-mean-square (rms) error) between the measured and calculated values can be achieved.

Through experiment, we previously found an NN which simultaneously models the seven intrinsic elements ($C_{\rm gs}$, $R_i, \cdots, C_{\rm ds}$) of the equivalent circuit [3]. Its configuration is represented as $\{3, 5, 7\}$, and has 4% of the rms errors.

We then assumed $E_0 = 0.0008$ with $N_{\rm gen} = 20$, $N_L = 4$, $N_n = 10$, and $N_p = 50$. Fig. 12 shows the average fitness versus the generation. These populations also converge in the

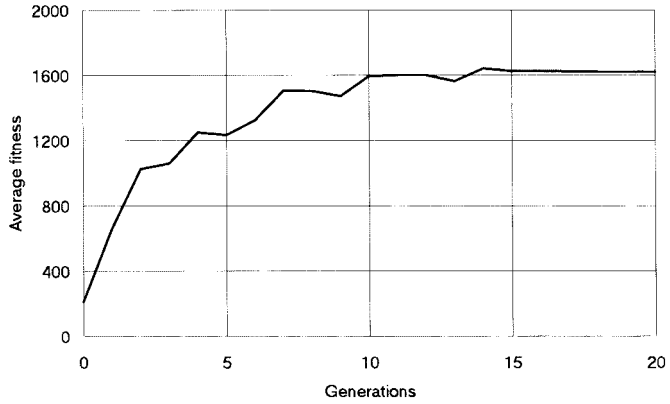

Fig. 12. Average fitness values relative to generation for a seven-element model.

early stages of evolution, and we obtained the configuration $\{3, 5, 9\}$. The rms error is 2.8%. This result is better than that obtained in our previous work [3].

V. Conclusion

We reported on two different techniques to determine the optimum multilayered large-signal NN model of an HEMT. This is the first report regarding techniques for building a general NN device model.

An HEMT's bias-dependent intrinsic elements were represented by a generalized multilayered NN. We began by experimentally studying the influence of the training parameters on the fitting errors by using $C_{\rm gs}$ data as an example. The best values were found to be in the vicinity of $\eta = 0.9$ and $\alpha = 0.8$. We then checked the behavior of the training error in relation to the number of iterations for NN's of different configurations. The output error was drastically reduced after a few thousand training cycles. This indicated that the elementary training had finished and that the better part of the synaptic connections were specialized for certain data. In addition, the calculation times needed for one training iteration were checked under various network configurations in order to find the rational NN's size.

We first studied a pruning technique. Pruning was adopted to sufficiently specialized synaptic connections after several elementary training iterations. By considering the flow of net information through each connection, we terminated the less active connections which carry little information.

We confined this pruning routine to within the normal back propagation and obtained good results.

We then applied a genetic algorithm for the same purpose. A network configuration can be described as the chromosomes of a virtual creature. To achieve as fast a convergence as possible, we introduced the concept of the evolutionary gains of the former occupants in the evolution process. We successfully determined the optimum NN $C_{\rm gs}$ model by simulating the evolution of virtual creatures. Regardless of different initial conditions, we could obtain the same network configuration for a $C_{\rm gs}$ model. This verifies the reliability of our approach.

We also obtained a single NN model that simultaneously represents seven equivalent-circuit elements. These NN's produced a closer agreement with the actual data than those determined experimentally.

The pruning approach tries to arrive at an optimum configuration through training on a single NN of excessive size. We can obtain a simplified NN within a few minutes in this manner. However, if an NN becomes bogged down with unpredictable fluctuations during the training, we have to find and adjust suspicious parameters and restart the training. Moreover, it is difficult to estimate the excessive size to start with for an NN of a particular type of data.

On the other hand, the genetic approach requires a few days on a workstation to obtain a final result. However, this approach has two significant advantages. First, due to the competitive training of many individuals, if one of them becomes stuck in a fluctuating trap, the other individuals will still be able to escape from the trap. This is also suggested by the fact that the occurrence of fluctuations on the training curve results from a slight change in the initial conditions. Second, in this approach, the selection of the optimum NN is done according to the explicit and easy-to-comprehend figures-of-merit (fitness).

The calculation load is very heavy when using only one computer, but if we consider the genetic approach as a distributed system, such as a client-server system, we are sure we can shrink the computational costs to within several dozen minutes.

As a result, the genetic approach is more convenient for operators without NN backgrounds. We are now trying to implement an NN program as a server running on many workstations, and the genetic evolution program as a client, in order to avoid computational problems.

References

[1] A. H. Zaabab, Q. Zhang, and M. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349–1358, June 1995.

[2] J. Rousset, Y. Harkouss, J. M. Collantes, and M. Campovecchio, "An accurate neural network model of FET for intermodulation and power analysis," in *Proc. 26th European Microwave Conf.*, Prague, Czech Republic, Sept. 1996, pp. 16–19.

[3] K. Shirakawa, M. Shimizu, N. Okubo, and Y. Daido, "A large-signal characterization of an HEMT using a multilayered neural network," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 1630–1633, Sept. 1997.

[4] J. W. Bandler, R. M. Biernacki, S. H. Chen, J. Song, S. Ye, and Q. Zhang, "Analytical unified dc/small-signal/large-signal circuit design, *IEEE Trans. Microwave Theory Tech.*, vol. 39, pp. 1076–1082, July 1991.

[5] K. Shirakawa, M. Shimizu, Y. Kawasaki, Y. Ohashi, and N. Okubo, "A new empirical large-signal HEMT model," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 622–624, Apr. 1996.

[6] D. E. Root, S. Fan, and J. Meyer, "Technology independent large signal non quasi static FET models by direct construction from automatically characterized device data," in *Proc. 21st European Microwave Conf.*, Stuttgart, Germany, Sept. 1991, pp. 927–923.

[7] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, Sept. 1993.

[8] D. A. Miller, J. M. Zurada, and J. H. Lilly, "Pruning via dynamic adaptation of the forgetting rate in structural learning," in *IEEE Proc. Int. Conf. Neural Networks*, Washington, DC, June 1996, pp. 448–452.

[9] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning.* New York: Morgan Kaufmann, 1986, vol. 2.

[10] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, pp. 122–128, Jan. 1986.

[11] K. Shirakawa and N. Okubo, "Genetic determination of large-signal HEMT model," in *Proc. 27th European Microwave Conf.*, Jerusalem, Israel, Sept. 1997, pp. 432–436.
[12] K. Shirakawa, H. Oikawa, T. Shimura, Y. Kawasaki, Y. Ohashi, T. Saito, and Y. Daido, "An approach to determining an equivalent circuit of HEMT's," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 499–503, Mar. 1995.

Kazuo Shirakawa (M'95) graduated from the National Defense Academy, Yokosuka, Japan, in 1986.

In 1986, he joined the Wireless Communication Systems Laboratory, Fujitsu Laboratories Ltd., Kawasaki, Japan, where he is currently an Engineer, involved in research and development of monolithic microwave integrated circuits (MMIC's). Since 1993, he has also been engaged in device modeling and millimeter-wave-application system design, such as wireless local area network (LAN) and automobile radar.

Mr. Shirakawa is a member of the IEEE Microwave Theory and Techniques Society and the Institute of Electronics Information and Communication Engineers (IEICE), Japan.

Masahiko Shimizu was born in Tokyo, Japan, in 1963. He received the B.S. degree in mathematics from Chuo University, Tokyo, Japan, in 1987.

In 1987, he joined Fujitsu Laboratories Ltd., Kawasaki, Japan, where he had been engaged in research and development of microwave mixers and FET modeling until 1991. Since 1991, he has been engaged in research and development of mobile radio-communication systems.

Mr. Shimizu is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), Japan.

Naofumi Okubo received the B.E. and M.E. degrees in electronic engineering from Niigata University, Niigata, Japan, in 1976 and 1978, respectively.

In 1978, he joined Fujitsu Laboratories Ltd., Kawasaki, Japan, where he was engaged in research and development of high-power amplifiers and low-loss power dividers/combiners for microwave communications systems, low-noise amplifiers for satellite earth stations, and solid-state power amplifiers (SSPA's) and receivers for satellite transponders. In 1987, he joined the Space Communications Research Corporation, Tokyo, Japan, where he has continued his work on highly efficient power amplifiers for mobile communications satellite. In 1989, he returned to Fujitsu Laboratories Ltd., where he is currently a Senior Researcher in the Wireless Communication Systems Laboratory.

Yosimasa Daido (M'84) received the B.S. and M.S. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1968 and 1970, respectively, and the doctor degree from Tokyo University, Tokyo, Japan, in 1988.

In 1970, he joined Fujitsu Laboratories Ltd., Kawasaki, Japan, where he had been engaged in research on microwave electronics, transmission characteristics of optical fiber, high-speed digital radio systems, and mobile radio communications. Since 1994, he has been a Professor in the Information Theory and Design Core, Kanazawa Institute of Technology, Ishikawa, Japan. His current research interests are mobile radio communications and chaos.