Development of a Self-Organizing Traffic Control System Using Neural Network Models

Takashi Nakatsuji and Terutoshi Kaku

A multilayer neural network model is introduced in order to realize a self-organizing traffic control system. The neural model inputs split lengths of signal phases and outputs measures of effectiveness such as queue lengths or performance indexes. The operation is separated into two processes, a training process and an optimization process. In the training process, iterations of the training operation by the backpropagation method were effective in forming a steady input-output relationship between splits and measures of effectiveness. In the optimization process, a stepwise method combining the Cauchy machine with a feedback method was proposed. The Cauchy machine is a sort of Monte Carlo method and gives the adjustments in a statistical way. This machine was introduced to urge the convergence and avoid the entrapment into local minimums. The feedback method is based on the steepest descent method and gives the adjustments in a deterministic way. This method has a self-organization ability because it can make adjustments that are closely related to traffic situations. The neural model was applied to a road network consisting of three intersections, and split lengths were optimized in order to minimize the squared sum of queue lengths on inflow links. The neural network model was able to give approximated splits and queue lengths that were in good accordance with analytical ones.

Today, most large cities in industrialized countries are confronted with chronic traffic congestion. With regard to this problem, the Organization for Economic Cooperation and Development (OECD) (*1*) issued a report on traffic management systems in urban areas. It states that future traffic systems should be operated on the self-organizing principle, in which the system would alter the basic form of the control law to respond not only to variations in traffic conditions but also to changes in transportation policies. Moreover, it says that applications of artificial intelligence techniques such as knowledge-based expert systems and fuzzy logic would be effective tools for realizing such intelligent traffic management systems. Because neural network models are also characterized by the ability of self-organization, they would serve to develop future traffic control systems.

Although neural computers have not yet been put into practice, neural network models, which are fundamental concepts of neural computers, have the potential of being able to compute in parallel and being able to learn from past experience. In particular, the self-organization ability is expected to have great effect on future traffic management systems because neural models are able to learn without any knowledge of the system and any logic such as if-then operations in expert systems. In other words, they are able to establish a characteristic input-output relationship without any preliminary information of the system. Therefore, they seem to be applicable even to nonlinear, nonstationary, or nonlogical problems. We are developing a macroscopic traffic simulation program using these characteristics of neural network models. So far, we have applied them to traffic control problems such as short-term prediction of traffic variables, traffic-responsive selection of prestored timing plans, traffic assignment, and split optimization for an isolated intersection under a criterion of the minimum queue length (*2,3*).

With regard to optimization of signal parameters, entrapment into local minimums is a serious and inevitable difficulty. In the hill-climbing method adopted in TRANSYT (*4,5*), a traffic optimization program used throughout the world, escape from local minimums is the major problem. Because some neural network models have the ability to escape from local minimums by introducing some stochastic techniques, they are expected to be effective in overcoming this difficulty. Furthermore, in optimal traffic control, application to a large-scale network is another difficulty because it takes great computation time. A hierarchical technique, first proposed by Singh and Tamura (*6*), is a superseding approach to overcome this difficulty. This method, however, is difficult to understand because it requires some mathematical knowledge. Because neural computers, if they are to be realized in the near future, have the ability of parallel processing, they are potentially applicable to large-scale networks.

This paper is mainly concerned with applications of a neural network model to optimize splits of signal phases. First, we briefly introduce the fundamental ideas of a multilayer neural model and the corresponding training algorithm, the backpropagation method. Second, we formulate optimal traffic control problems using the neural network model. In this formulation, we adopted two kinds of optimization criteria: the minimum queue length and the minimum performance index, which is a weighted sum of delays and stops. To avoid entrapment into a local minimum and urge the convergence to a global minimum, we proposed a stepwise method that combined the Cauchy machine with a feedback method in sequence. Finally, based on numerical analyses, we conclude that the neural network model has a good possibility for the development of future traffic control systems.

Civil Engineering Department, Hokkaido University, Kita 13, Nisha 8, Kita-ku, Sapporo, 060, Japan.

NEURAL NETWORK MODEL

Artificial Neurons

Artificial neurons are designed to emulate the basic mechanism of biological neurons. Figure 1 (left) shows a model that implements this function. A set of outputs ($y_{i1}, y_{i2}, \ldots, y_{iN}$) from other neurons and a bias input (I_i) from itself are applied to a neuron (i). Each output is multiplied by synaptic weights ($W_{i1}, W_{i2}, \ldots, W_{iN}$) and summed up algebraically:

$$x_i = \sum_{j=1}^{N} W_{ji} y_j + I_i \tag{1}$$

The signal x_i is activated by a function, which is called an activation function or a response function, as shown in Figure 1 (right):

$$y_i = F(x_i) \tag{2}$$

We adopted here a sigmoid function, $F(x) = 1/[1 + \exp(-x)]$, as the activation function. This nonlinear function prescribes the fundamental capability, as well as synaptic weights, of neural network models. Details of artificial neurons can be found in Wasserman (7).

Multilayer Neural Network

A multilayer neural network model was used in this analysis, as shown in Figure 2. The neural system consists of several layers: an input layer, some hidden layers, and an output layer. Assume that the neurons in the input layer serve only as distributors. The original input signals are normalized there and transmitted to the next layer. Therefore the first hidden layer, Layer B, has the same number of neurons as the input layer. Neural operations take place at the hidden layers and the output layer. The output layer produces the objective signals.

To obtain precise output signals, the synaptic weights must be adjusted. This adjustment is called the training. The back-propagation method (8) is used for training of multilayer networks. The method is based on the steepest descent method, in other words, the delta rule: Synaptic weights are adjusted so as to minimize the error between the output signals and the target signals, which are desired results determined externally. Letting y_k be the output signal and z_k be the target signal at the kth neuron in the output layer, and letting W_{ij} and W_{jk} be the synaptic weights between the layers shown in Figure 2, the error function is defined:

$$E = \frac{1}{2} \sum_k (y_k - z_k)^2 \tag{3}$$

Differentiating this error function with respect to W_{jk} and W_{ij} in sequence, we obtain the following expressions for adjusting synaptic weights:

$$\delta w_{jk} = \eta(z_k - y_k) y_j y_k (1 - y_k) \tag{4}$$

$$\delta W_{ij} = \eta \sum_k \delta W_{jk} W_{jk} y_i y_j (1 - y_j) \tag{5}$$

where η is the training rate coefficient in the range of 0 to 1. In actual computations, some constants are introduced to smooth the adjustments and urge the convergence. Noting that the error $\Sigma \delta W_{jk} W_{jk}$ in Equation 5 corresponds to $z_k - y_k$ in Equation 4, we can derive the adjustments for the upper layers in sequence.

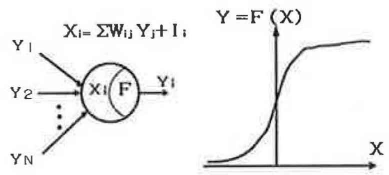

FIGURE 1 Neural network model: *left*, artificial neuron; *right*, activation function.

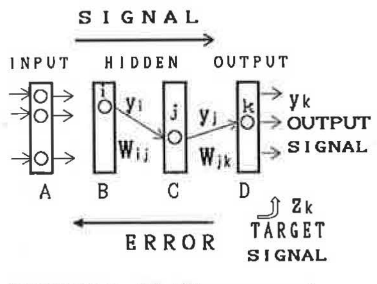

FIGURE 2 Multilayer neural network.

OPTIMAL TRAFFIC CONTROL PROBLEM

Neural Network Model for Estimating Optimal Splits

Figure 3 shows a neural network model for estimating optimal splits. It consists of four layers of neurons, Layers A to D. This neural network model describes the relationship between control variables (splits of signal phases) and objective variables (traffic variables such as queue lengths or performance indexes). That is, it inputs splits into Layer A and outputs

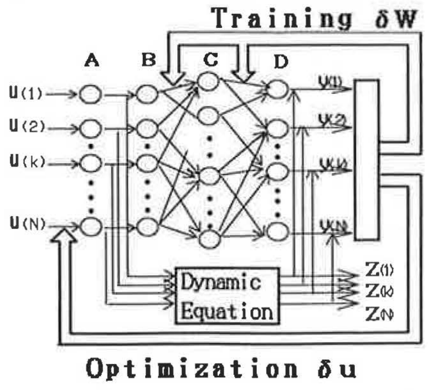

FIGURE 3 Multilayer neural network model for estimating optimal splits.

traffic variables on inflow links from Layer D. Although it is not shown in Figure 3, the traffic volumes on inflow links are also given to the neural system externally. As mentioned, because neurons in Layer A serve only as distributors to Layer B, the number of neurons in Layer B is equal to that of Layer A. The number of neurons in Layer D is the same as the number of inflow links. However, the number of neurons in Layer C used to be determined from numerical manipulation. In this case, we found that equal numbers of neurons in Layers C and D produced acceptable results. Furthermore, it should be noted that by using the time sequence of splits and traffic variables, it is possible to optimize the splits varying with time. For example, suppose an isolated intersection with four arms that is operated by two signal phases. By estimating the splits that vary every cycle, the number of neurons of the input layer is $2 \times N$ and that of the output layer is $4 \times N$, where N is the number of cycle periods.

Dynamic Equation

As mentioned, the neural network model in this analysis requires iterative trainings to adjust synaptic weights. Training signals are given by dynamic equations that are defined by objective variables and control variables. The internal dynamic model in Figure 3 produces those training signals. We formulate two kinds of dynamic equations for a simple road network system: one for queue length and the other for performance index (PI), which is used in the TRANSYT program (*4,5*). In this analysis, we assume for simplicity that the cycle length is common over the network and does not vary with time. Furthermore, we assume that there are no offsets between adjacent intersections.

First, we present the dynamic equation with respect to queue length. Assume a road network that consists of several intersections. Each intersection has inflow links of n_i and signal phases of p_i. We denote the split and the queue length at cycle time k by $\mathbf{y}(k)$ and $\mathbf{u}(k)$, which are column vectors of $N = \Sigma\, n_i$ and $P = \Sigma\, p_i$, respectively. The dynamic equation is given by

$$\begin{aligned}\mathbf{y}(k+1) &= \mathbf{y}(k) + B_0\mathbf{u}(k) + B_1\mathbf{u}(k-1) + \dots \\ &\quad + B_M\mathbf{u}(k-M) + \mathbf{q}(k) \\ &\quad (k = 0, 1, \dots, K-1) \end{aligned} \tag{6}$$

where $\mathbf{q}(k)$ is the input flow vector of N, and B_m is the control weighing matrix of $N \times P$, which is defined by saturation flow rates on inflow links. In this analysis, we adopted an optimization criterion that minimizes the squared sum of queue length:

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} y_i(k)^2 \tag{7}$$

As an example, we suppose a simple road network consisting of two intersections as shown in Figure 4. Each intersection has two phases, one for the eastbound traffic movement and one for the southbound movement. Moreover, we denote the inflow rate at the stop line on link i by $q_i(k)$ and

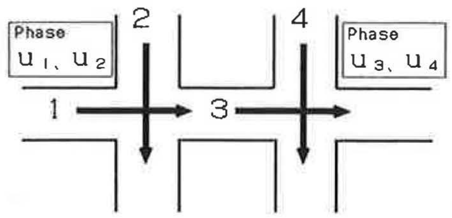

FIGURE 4 Entrance link and internal link.

the saturation flow rate by s_i. For entrance links, such as Links 1, 2, and 4, the dynamic equation is given by

$$y_i(k+1) = y_i(k) + q_i(k) - s_i u_i(k) \qquad (i = 1, 2, 4) \tag{8}$$

For internal links, in this case Link 3 alone, the inflow at stop line depends on outflows from the upstream links and splits. Denoting the inflow at entrance by $p_3(k)$, we can derive the dynamic equation as follows:

$$y_3(k+1) = y_3(k) + \sum_{m=0}^{M} \gamma_{3,m} p_3(k-m) - s_3 u_3(k) \tag{9}$$

$$p_3(k-m) = s_1 u_1(k-m) \tag{10}$$

where $\gamma_{3,m}$ is the dispersion coefficient. The value of M is determined from a correlation analysis between the upstream flows and the downstream flows. Assembling these equations, we obtain a dynamic equation that is identical to Equation 6. For details, refer to Singh and Tamura (*6*).

Next we present the dynamic equation with respect to the performance index. In this case, we have to divide each cycle period into steps of equal duration and formulate the dynamic equation, which is identical to Equation 6, for each time step t. By integrating all of those traffic profiles for each step, we can define some measures of effectiveness, such as delay and stops, for each cycle time. As defined in TRANSYT, the performance index on inflow Link i for Cycle Time k is calculated as follows:

$$PI_i(k) = \mathrm{DLY}_i(k) + \kappa_i \mathrm{STP}_i(k) \tag{11}$$

where

$\mathrm{DLY}_i(k)$ = total delay on Link i for Cycle k,
$\mathrm{STP}_i(k)$ = number of stops on Link i for Cycle k, and
κ_i = stop penalty coefficient.

We took this performance index as the objective variable, $y_i(k)$. Also in this case, the same optimization criterion as Equation 7 was used. The TRANSYT users manuals (*4,5*) provide details of the definition of the delay and stops.

Both the objective and the control variables are subject to constraints for every Cycle Time k:

$$\mathbf{0} \le \mathbf{y}(k) \le \mathbf{Ymax} \tag{12}$$

$$\mathbf{Umin} \le \mathbf{u}(k) \le \mathbf{Umax} \tag{13}$$

$$u_{i,1}(k) + u_{i,2}(k) + \dots + u_{i,p_i}(k) + ls_i = 1 \tag{14}$$

where $u_{i,r}(k)$ is the rth split at intersection i, and ls_i is the ratio of loss time to cycle length.

Computational Procedures

Referring to Maeda (9), we separated the operation of this neural model into two processes, the training process and the optimization process. In the training process, synaptic weights are adjusted so that the output signals from the output layer coincide with those from the internal model as much as possible. This adjustment can be done by the direct use of the backpropagation method. On the other hand, the optimization process performs iterative adjustments of splits to minimize the objective function under given constraints.

Figure 5 shows the block diagram for estimating optimal split lengths. First we have to perform initial training. After preparing a set of traffic volumes that arrive at entry links and scores of split patterns that are randomly generated, we adjust synaptic weights of the neural network model. We repeat the backpropagation operations until the squared sum of the deviations between the output signals and the target signals becomes sufficiently small. We iterate initial training until the neural models satisfy the convergence condition for all split patterns.

Next, we predict traffic volumes on entry links for several cycle periods. There are many prediction methods; however, because the discussion on the methods is beyond the scope of this paper, we assume that precise traffic volumes are already being predicted. Because those traffic volumes are different from those in the initial training process, we have to adjust synaptic weights again. However, traffic volumes do not change drastically, so we can adjust them through several iterations of the backpropagation method.

Establishment of a steady relationship between splits and objective variables makes it possible to estimate optimal splits properly. To do this, we proposed a combined, stepwise method. Theoretically, by repeating the procedures from prediction to optimization in sequence, it might be possible to estimate optimal splits in real time. However, under the present circumstances, the neural approach takes more computation time compared to the conventional analytical methods.

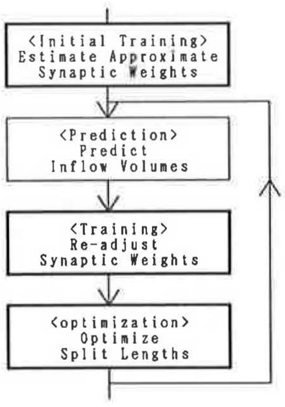

FIGURE 5 Computational procedures for estimating optimal splits.

In addition, because the modeling of offsets is being left unresolved, we analyze the splits for a set of traffic volumes.

Stepwise Method

To optimize split lengths, it is necessary to adjust them iteratively in order to minimize the objective criterion under given constraints. Referring to Wasserman's technique (7), we proposed another combined method consisting of a two-step process. First, we adjust splits based on the Cauchy machine to avoid entrapment into a local minimum and urge the convergence. Next, we adjust the splits using a deterministic technique similar to the backpropagation method. This combined algorithm is called the stepwise method. Entrapment into a local minimum is a serious and inevitable difficulty in some minimum-seeking problems. To overcome this difficulty, some stochastic methods, such as the Boltzman machine, the Gaussian machine and so on, have been proposed in neural network analyses. Szu [Wasserman (7)] developed a stochastic method, called the Cauchy machine, for steady convergence to a global minimum. It is a sort of Monte Carlo method; by adding small changes, which follow the Cauchy distribution, into the present split values, we accept those changes if they improve the objective function, and abandon them otherwise. The probability density function of the Cauchy distribution is given by

$$p(x) = T(t)/[T(t)^2 + x^2] \quad (15)$$

$$T(t) = T_0/(1 + t) \quad (16)$$

where $T(t)$ is the artificial temperature, and T_0 is the initial temperature. Integrating the density function, we obtain the following distribution function:

$$P(x) = \arctan[x/T(t)] \quad (17)$$

Then, resolving for x yields

$$x = \rho T(t) \tan[P(x)] \quad (18)$$

where ρ is a coefficient in the range of 0 to 1. Regarding x in the above equations as split change δu, we can find the change as follows:

1. Select a random value from a uniform distribution over the interval $(-\pi/2, \pi/2)$.
2. Substitute it into $P(x)$ in Equation 18 and calculate the change.
3. Retain it if the adjustment improves the objective function, and return it to the previous value if otherwise.
4. Decrease the deviation of the Cauchy distribution and go back to Step 1 and repeat again.

This algorithm can drastically reduce the computation time because it adopts an annealing scheme in which the temperature is decreased inversely linearly, rather than inversely logarithmically as in the Boltzman machine.

Next we adjust the splits in a deterministic way similar to the backpropagation method in the training process. The steepest descent method is used again. By differentiating the objective function of Equation 7 with respect to $u_i(k)$, we can easily derive the following expression for adjustments of the splits:

$$\delta u_i = \eta \sum_k y_k^2(1 - y_k) \sum_j W_{ij}W_{jk}y_j(1 - y_j) \qquad (19)$$

where η is a coefficient ranging 0 to 1. Because those adjustments in this optimization process are not backpropagated as in the training process, we call such a process the feedback method. The adjustments in Equation 19 are related to synaptic weights, which vary with traffic situations. This means that we are able to alter the parameters for adjusting splits automatically corresponding to the change of traffic situations. This self-organizing ability is a promising feature of neural network models. Furthermore, although in this analysis we adopted the optimization criterion given by the form of Equation 7, we can derive similar expressions to Equation 19 for any criteria only if they are differentiable with respect to $u_i(k)$.

NUMERICAL EXPERIMENTS

Training

The ability of the neural model depends on how precisely the synaptic weights are adjusted. The initial training process requires scores of training operations for each split pattern. Using an isolated intersection as an example, we explain how the synaptic weights were adjusted by the backpropagation method. As shown in Figure 6, the intersection has eight inflow links and is operated with three signal phases. We assume that the cycle length is 120 sec and the simulation period consists of four cycles. Furthermore, we assume for simplicity that inflow rates and split lengths are constant over the simulation period. This assumption is not requisite; a problem for time-variant splits is also presented. Detailed information on the inflow links is shown in Table 1.

First, we discuss the problem of the minimum queue length. We build up a neural network model, shown in Figure 3, in which the neuron in the input layer corresponds to the split length of each signal phase and the one in the output layer to total queue length on each inflow link. That is, the number of neurons in the input layer is three, and that of the output

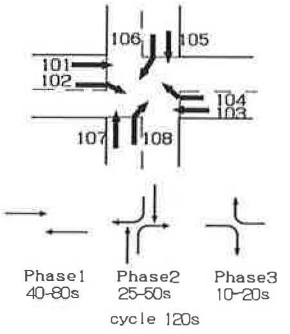

FIGURE 6 Isolated intersection.

TABLE 1 LINK DATA FOR ISOLATED INTERSECTION EXAMPLE

Link No.	Saturation Flow Rate veh./cycle	Inflow Volume veh./cycle	Initial Queue veh.
101	113	26.67	50
102	53	1.53	10
103	113	30.20	100
104	53	3.67	10
105	113	17.67	50
106	53	1.60	10
107	113	11.13	50
108	53	2.87	10

layer is eight. To perform initial training, we prepared in advance 20 randomly generated sets of split patterns that satisfy the constraint conditions. We then calculated the total queue lengths on inflow links for four cycles and made them the training signals for each split pattern.

Figure 7 shows how the estimation error of the synaptic weights would decrease with iterative operations of the backpropagation method for some split patterns, Patterns 1, 2, 11, and 20. Here, the error was calculated by the root mean squared (RMS) value of the deviation between queue lengths by the neural system and those by the dynamic system. We truncated the iteration when the error became less than 10. Roughly speaking, this means an error of 2 percent because both the output and the target signals were normalized by a number of 500. Figure 7 shows that once synaptic weights had been adjusted for the first split pattern, they were easily adjusted for the other ones. However, it also shows that the completion of adjustments for a split pattern brings the deterioration of synaptic weights for the other patterns. Therefore, we have to repeat scores of training operations until the RMS error becomes less than the threshold for all split patterns. Figure 8 shows the variation of the maximum and the average RMS error with iterations of training operations. The average RMS error represents the root mean squared value of the RMS error for each split pattern. The figure shows that the synaptic weights are improved gradually but certainly. In this case, it took 357 iterations to complete the training, and the final average RMS error was 4.97, nearly half of the truncation threshold.

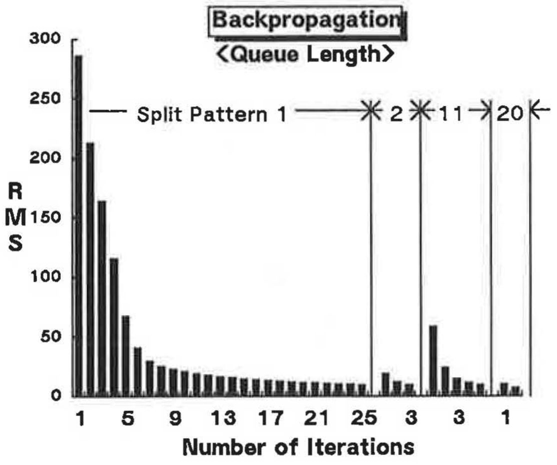

FIGURE 7 Backpropagation operations in the initial training process for some randomly generated split patterns (output variable is queue length).

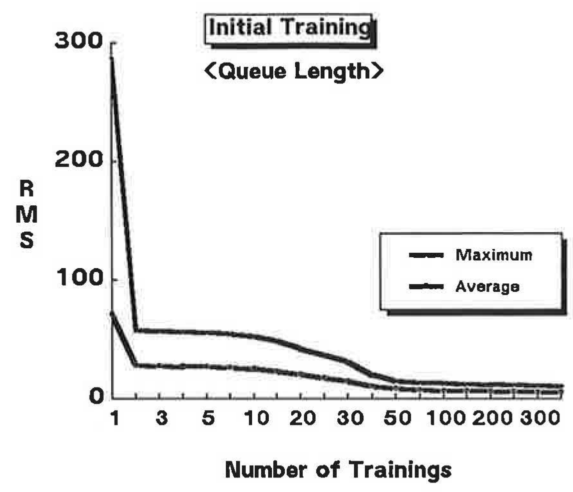

FIGURE 8 Adjustment of synaptic weights by iterations of the training. A training consists of iterative operations of backpropagation method for all split patterns (output variable is queue length).

To examine the ability of the neural system that completed the adjustment of synaptic weights, we prepared other split patterns. We then calculated the total queue lengths on the links using the neural system and compared them with analytical ones, which were given by the dynamic model. Figure 9 is the histogram of the RMS error for 100 sets of split patterns. It shows that the RMS error was less than 5.0 for more than 60 split patterns. For only three patterns, it exceeded the threshold of 10.0. The maximum RMS error was 10.66. This means that the initial training by 20 split patterns was sufficient.

Similarly, for the problem of the minimum performance index, we can build up another neural system that has a steady input-output relationship between split lengths and the corresponding performance indexes on inflow links. The difference lies only in the dynamic model for estimating target signals. We performed the initial training for the same intersection, shown in Figure 6, with the same split patterns as in the previous problem. In this analysis, we divided a cycle length of 120 sec into 60 steps of 2 sec. Parameters to calculate the delay time were the same as those in TRANSYT-7F. The stop penalty of five was used for all links. The output and target signals were normalized by a value of 600. It took 206 iterations of training operations to adjust synaptic weights completely. Figure 10 shows the distribution of the RMS errors for 100 sets of untrained split patterns. Although results in Figure 10 are not as good as in Figure 9, 36 split patterns had RMS errors less than 5.0, and only 6 patterns exceeded 10.0. The average and the maximum RMS errors were 5.06 and 13.42, respectively.

Optimization

Figure 11 shows how the stepwise method worked in the optimization process. We took the same problem in Figure 6. Figure 11a is for the minimum queue length, and Figure 11b is for the minimum performance index. We compared the stepwise method with the feedback method, in which no Cauchy operations were applied. The x-axis represents the number of iterations and the y-axis represents the values of the objective function, the squared sum of queue lengths for Figure 11a and that of performance indexes for Figure 11b. Figure 11 shows that there is little difference between the two methods. The feedback method also reaches the global minimum without being entrapped into a local minimum because the intersection is isolated and operated with simple signal phasing. However, the stepwise method was effective to urge the convergence, particularly for the performance index. Next, we present another example in which the stepwise method was effective to avoid local minimums.

Practical Simulation

As a practical example for real intersections of complicated geometry and phasing, we chose a road network that consists of three intersections, which was analyzed by Singh and Tamura (*6*). The configuration of those intersections and inflow links is given in Figure 12. Every intersection is operated with two phases. That is, the road network has 12 inflow links and six phases in total. The simulation period is three cycles. Detailed information on saturation flow rates and inflow volumes are given along with the initial values in Table 2. In this problem, split lengths are optimized every cycle period so as to minimize the squared sum of queue lengths on the inflow

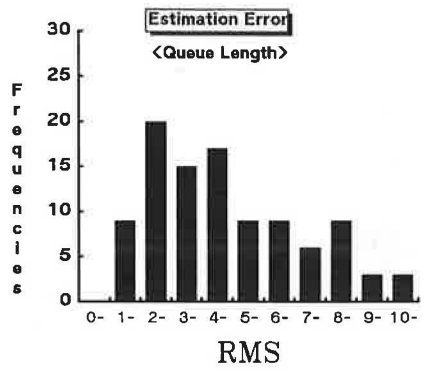

FIGURE 9 Distribution of RMS errors for 100 sets of split patterns (output variable is queue length).

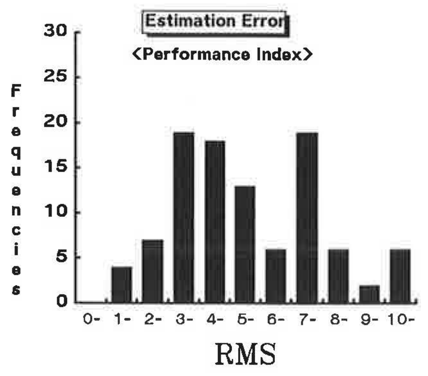

FIGURE 10 Distribution of RMS errors for 100 sets of split patterns (output variable is performance index).

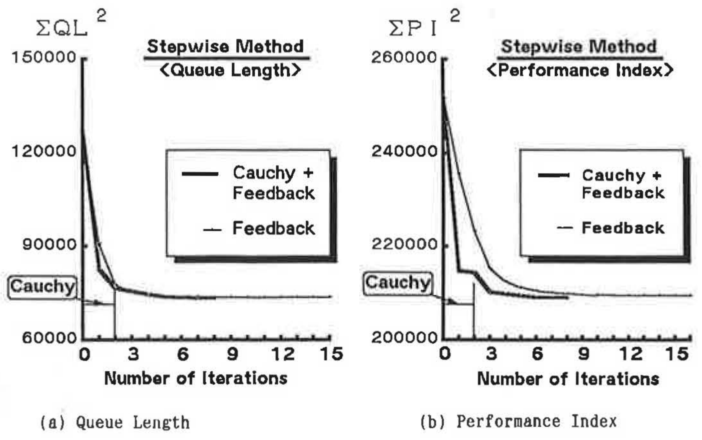

FIGURE 11 Optimization process for an isolated intersection.

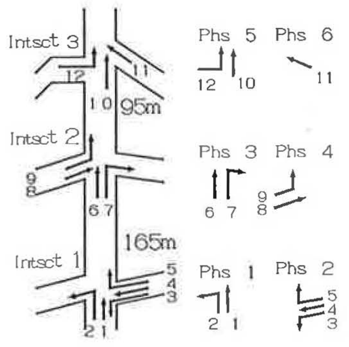

FIGURE 12 Road network for practical simulation (*6*).

TABLE 2 LINK DATA FOR ROAD NETWORK EXAMPLE (*6*)

Link No.	Link Length m	Saturation Flow Rate veh./cycle	Inflow Volume veh./cycle	Initial Queue veh.
1	-	65	7.6	30
2	-	25	3.0	30
3	-	34	23.7	70
4	-	31	21.7	70
5	-	4	27.0	70
6	165	64	Intrnl	40
7	165	26	〃	40
8	-	132	15.0	30
9	-	34	4.0	30
10	95	96	Intrnl	20
11	-	90	2.4	30
12	-	25	2.0	30

links. Referring to Singh and Tamura (*6*), the dynamic equations for this problem reduce to

$$y_1(k+1) = y_1(k) + q_1(k) - s_1u_1(k)$$

$$y_2(k+1) = y_2(k) + q_2(k) - s_2u_1(k)$$

$$y_3(k+1) = y_3(k) + q_3(k) - s_3u_2(k)$$

$$y_4(k+1) = y_4(k) + q_4(k) - s_4u_2(k)$$

$$y_5(k+1) = y_5(k) + q_5(k) - s_5u_2(k)$$

$$y_6(k+1) = y_6(k) + 0.7s_1u_1(k-2) + 0.7s_2u_2(k-2) - s_6u_3(k)$$

$$y_7(k+1) = y_7(k) + 0.3s_1u_1(k-2) + 0.3s_2u_2(k-2) - s_7u_3(k)$$

$$y_8(k+1) = y_8(k) + q_8(k) - s_8u_4(k)$$

$$y_9(k+1) = y_9(k) + q_9(k) - s_9u_4(k)$$

$$y_{10}(k+1) = y_{10}(k) + s_6u_3(k-1) + s_9u_4(k-1) - s_{10}u_5(k)$$

$$y_{11}(k+1) = y_{11}(k) + q_{11}(k) - s_{11}u_6(k)$$

$$y_{12}(k+1) = y_{12}(k) + q_{12}(k) - s_{12}u_5(k) \qquad (20)$$

where

$y_i(k)$ = queue length on Link i at cycle k,
$q_i(k)$ = inflow rate,
s_i = saturation flow rate, and
$u_r(k)$ = split length for signal phase r.

The values of 0.7 and 0.3 represent the dispersion coefficients. All splits were constrained to lie between 0.2 and 0.7 and to satisfy the conditions of $u_1(k) + u_2(k) = u_3(k) + u_4(k) = u_5(k) + u_6(k) = 0.9$.

We built up a neural network model as shown in Figure 3. However, distinct from the one in the previous discussion, it inputs the time sequence of the split lengths and outputs that

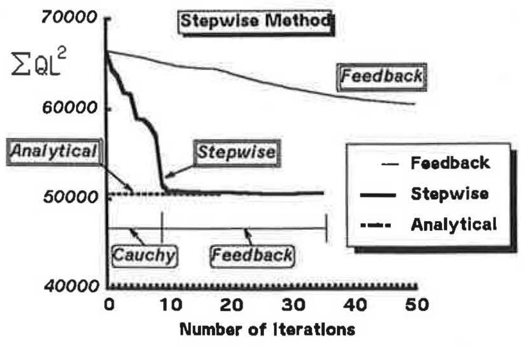

FIGURE 13 Optimization process for a road network.

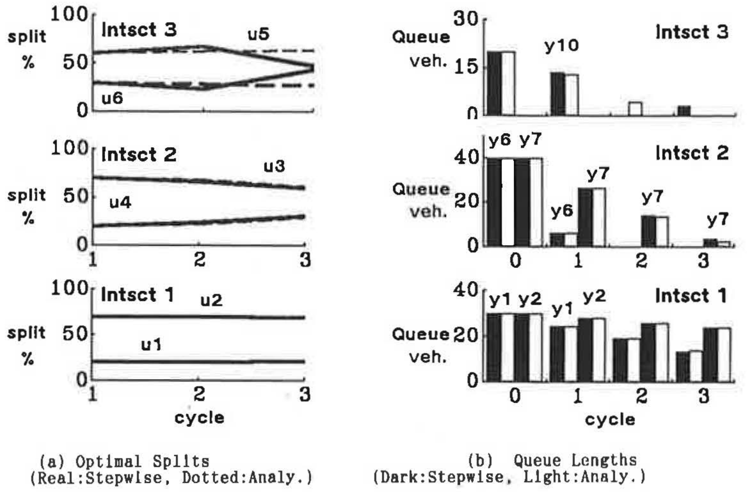

FIGURE 14 Optimal splits and queue lengths.

of the queue lengths because we have to estimate the splits that vary with cycle. Therefore, the number of neurons in the input layer is 6 × 3 and that of the output layer is 12 × 3, where the value of 3 is the number of cycle periods.

Figure 13 shows how the stepwise method effectively optimizes those split lengths. We compared three methods: an analytical method by a hierarchical approach, the feedback method without the Cauchy machine, and the stepwise method. We directly referred to results by Singh and Tamura (*6*) for the analytical method. The x-axis represents the number of iterations and the y-axis represents the squared sum of queue lengths. The comparison shows that although the feedback method was entrapped into a local minimum and took a large number of iteration values, the stepwise method succeeded in reaching the global minimum. Figure 14 shows the optimized control sequence and the corresponding queue lengths on main inflow links, Links 6, 7, and 10. The real lines are for the stepwise method and the dotted ones are for the analytical solutions. They show that solutions by the stepwise method were in good agreement with those of the analytical method.

CONCLUSIONS

Presuming applications to future traffic control systems, we introduce a neural network model, which is characterized by its self-organizing ability, for split optimization problems. First, we built up a multilayer neural network model that inputs split lengths of signal phases and outputs objective variables. We adopted two kinds of control criteria, the minimum queue length and the minimum performance index. Next, we divided the problem into two processes, the training process and the optimization process. In the training process, the backpropagation method was effective to adjust the synaptic weights. We established a steady input-output relationship by scores of iterations of training operations. In the optimization process, we proposed a stepwise method, combining the Cauchy machine and the feedback method, to urge the convergence and avoid entrapment into local minimums. Through numerical analyses, we showed that this method improved the convergence into a global minimum and that solutions by this method were in good accordance with analytical ones.

This paper is only the first step for realization of a self-organizing traffic control system. Many problems must be solved before a neural network model can be applied to an actual road network. One problem is the optimization of offsets. Without modeling the parameters, it is impossible to realize real self-organizing traffic control. The modeling of dispersion phenomena of vehicle platoons is another problem. This modeling is requisite for sophisticated traffic flow simulation. The improvement of computation time is also important. Because we used a conventional digital computer, the neural models presented here required much more computation time than the corresponding analytical method. Some emulation machines that have several parallel processors and are able to realize particular neural algorithms have already been developed. However, the application of a neural network model to an actual road network system would require the development of a neural computer with thousands of parallel processors. We confirm that such neural computers will be realized in the near future.

ACKNOWLEDGMENT

The authors wish to express their thanks to Dr. Tamura of Osaka University for allowing reference to the examples in his paper.

REFERENCES

1. *Dynamic Traffic Management in Urban and Suburban Road Systems*. Organization for Economic Cooperation and Development, Road Transportation Research, 1987.
2. T. Nakatsuji and T. Kaku. Application of Neural Network Models to Traffic Engineering Problems (in Japanese). *Proc., Infrastructure Planning*, Vol. 12, 1989, pp. 297–304.
3. T. Nakatsuji and T. Kaku. Application of Neural Network Models to Traffic Engineering Problems, *Proc., 11th International Symposium on Transportation Traffic Theory*, Yokohama, Japan, July 1990, pp. 291–306.

4. R. A. Vincent, A. I. Mitchell, and D. I. Robertson. *User's Guide to TRANSYT*, Version 8. U.K. Transport and Road Research Laboratory LR888, 1980.
5. *TRANSYT-7F Self-Study Guide*. FHWA, U.S. Department of Transportation, 1986.
6. M. G. Singh and H. Tamura. Modeling and Hierarchical Optimization for Over-Saturated Urban Road Traffic Networks. *International Journal of Control*, Vol. 20, No. 6, 1974, pp. 913–934.
7. P. D. Wasserman. *Neural Computing*. Van Nostrand Reinhold, New York, 1989.
8. D. E. Rumelhart et al. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing*, Vol. 1. MIT Press, Cambridge, Mass., 1986.
9. Y. Maeda, M. Kawato, Y. Uno, and R. Suzuki. *Multi-Layer Neural Network Model Which Learns and Generates Human Multi-Joint Arm Trajectory*. Japan IEICE Technical Report MBE87-133, 1988, pp. 233–240.

Publication of this paper sponsored by Committee on Traffic Signal Systems.