JOURNAL OF ARTIFICIAL NEURAL NETWORKS, 1(3), 371–401 (1994)

A Parallel Algorithm for Traffic Control Problems in Multistage Interconnection Networks

Nobuo Funabiki

Sumitomo Metal Industries, Ltd.
Amagasaki, Japan

Yoshiyasu Takefuji

Case Western Reserve University
Cleveland, Ohio
and
Keio University
Fujisawa, Japan

A parallel algorithm for traffic control problems in multistage interconnection networks is presented. Since Goke and Lipovski [1] defined the class of Banyan networks in 1973, multistage interconnection networks have been extensively studied and used in many applications such as telephone switching networks, parallel processing computer networks, and integrated service digital network (ISDN). The multistage interconnection networks have two advantages over simple crossbar switches: one is that the number of switching devices increases by $O(n \cdot \log n)$, instead of $O(n^2)$, as it does in crossbar switches for n-input/n-output switching systems. The other advantage is that the fanout of switching devices is constant, as opposed to $O(n)$ in crossbar switches. A disadvantage, however, is that the multistage interconnection networks not only have output blocking (which is unavoidable in crossbar switches as well) but also have internal blocking, which degrades the network throughput. The goal of the proposed parallel algorithm is to find conflict-free traffic flows in order to realize the maximum or near-maximum throughput for a given I/O traffic demand in a given multistage interconnection network. The algorithm requires n^2 processing elements for the traffic control

Correspondence and requests for reprints should be sent to Yoshiyasu Takefuji, Department of Electrical Engineering and Applied Physics, Case Western University, Cleveland, OH 44106.

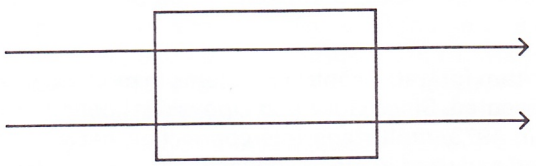
problem in an $n \times n$ multistage interconnection network. The algorithm runs not only on a sequential machine but also on a parallel machine with maximally n^2 processors. The algorithm is verified by solving 40 problems where the size of networks is varied from 4×4 to 32×32 and the traffic demand density is varied from 10 to 100%. In massive simulation runs, the algorithm finds conflict-free traffic flows in a nearly constant time with n^2 processors.

Banyan networks, multistage interconnection,
integrated service digital network, crossbar switch, traffic control

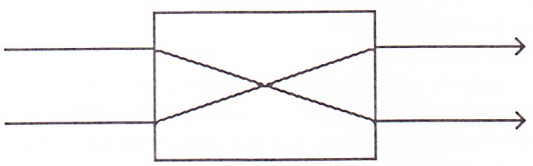
INTRODUCTION

Since Goke and Lipovski defined the class of Banyan networks in 1973 [1], multistage interconnection networks have been extensively studied and used in many applications such as telephone switching networks, parallel processing computer networks, and integrated service digital networks (ISDN) [2–29]. The multistage interconnection networks have two major advantages over simple crossbar switches. One is that the number of switching devices increases by $O(n \cdot \log n)$ instead of $O(n^2)$ as it does in crossbar switches for n-input/n-output switching systems. The other is that the fanout of switching devices is constant as opposed to $O(n)$ in crossbar switches.

The multistage interconnection network considered in this article consists of $\log_2$



(a) Direct connection state



(b) Crossed connection state

Figure 1. Two states of a 2 × 2 switching element.

n stages of 2×2 nonblocking switching elements where n inputs and n outputs are connected and n is the power of 2. The network is called an $n \times n$ multistage interconnection network. Each stage is composed of $n/2$ 2×2 switching elements. As shown in Figure 1, each switching element has two states, a direct connection state and a crossed connection state. The multistage interconnection networks have a self-routing function where each bit of the destination address of a packet of data decides the state of the corresponding switching element [4]. This function implies that one and only one path through the network between an input–output exists, and that the path is predetermined. Several variations in the class of multistage interconnection networks such as data manipulator (modified version) [3], omega network [4], flip network [5], indirect binary n-cube network [6], regular SW banyan network (with spread and fanout of 2), baseline network, and reverse baseline network [8] have been proposed. It has been proven that they are topologically equivalent [7, 8].

In this article, we use the reverse baseline network as the benchmark in the

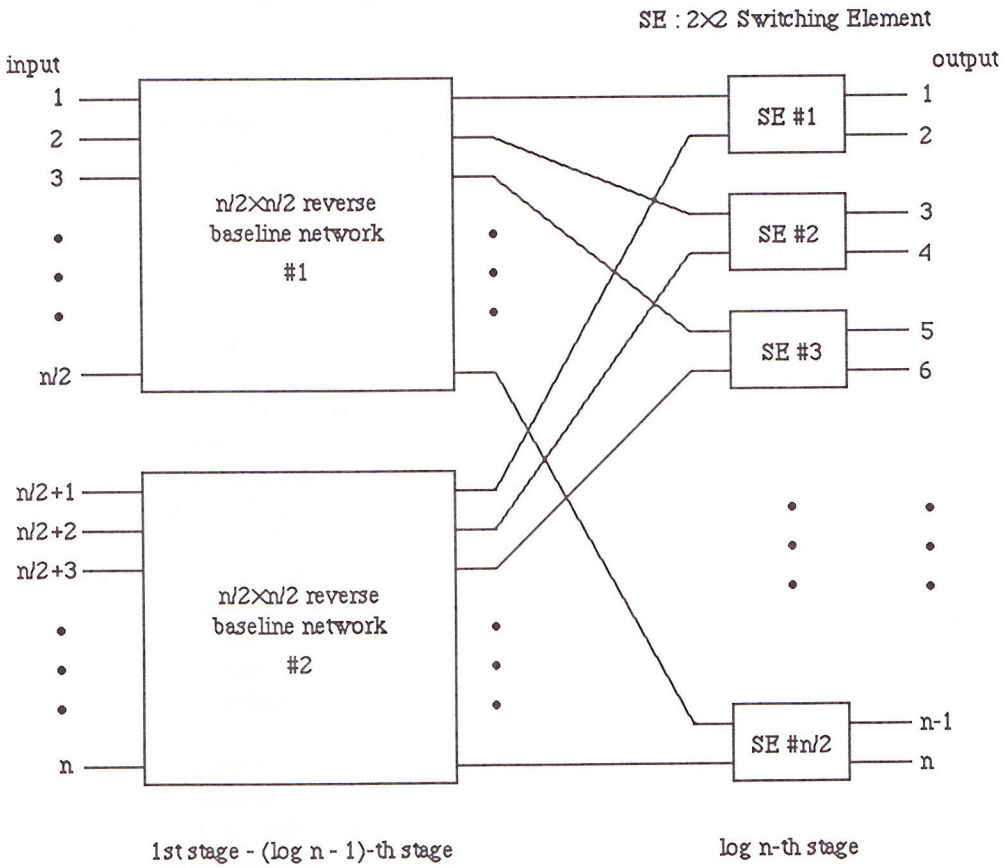


Figure 2. An $n \times n$ reverse baseline network generated from two $n/2 \times n/2$ baseline networks and $n/2$ 2×2 switching elements.

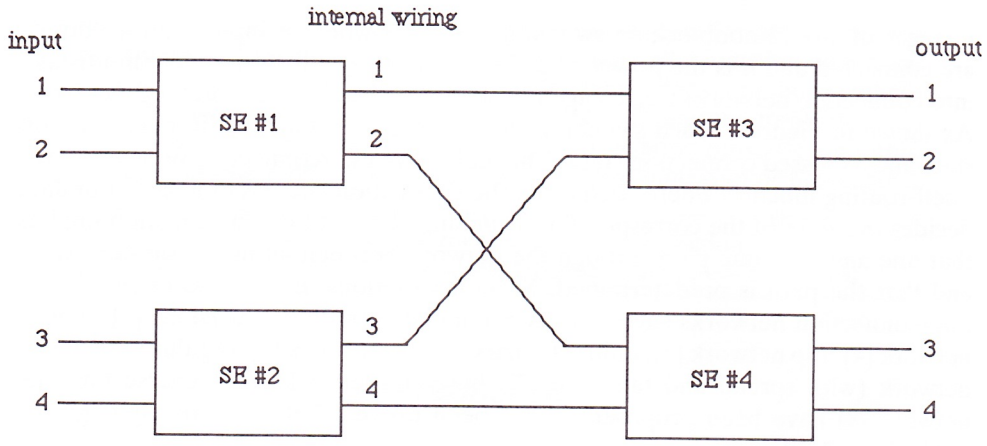


Figure 3a. A 4 × 4 reverse baseline network.

multistage interconnection networks because it has the simplest wiring structure. The topology of the reverse baseline network can be generated in a recursive way. Figure 2 shows an $n \times n$ reverse baseline network that is generated from two $n/2 \times n/2$ reverse baseline networks at up to (log n − 1)th stages and $n/2$ 2 × 2 switching elements at the (log n)th stage. One input of each switching element is wired with one of $n/2$ outputs of an $n/2 \times n/2$ network, and another input of the switching element is wired with one of $n/2$ outputs of the other $n/2 \times n/2$ network. Figures 3a and 3b show a 4 × 4 reverse baseline network and a 32 × 32 reverse baseline network, respectively.

The multistage interconnection network including the reverse baseline networks usually has two kinds of constraints for data transmission: the output-blocking constraint and the internal blocking constraint [27]. The output blocking constraint means that more than one input cannot be connected with the same output simultaneously because it causes data collision at the output. This output-blocking constraint is also unavoidable in crossbar switches. The internal blocking constraint means that more than one path from inputs to outputs cannot simultaneously share the same internal wiring in the network because it also causes data collision in the wiring. In addition to these two constraints, we assume that the network provides only point-to-point connections where one input cannot be connected with more than one output. In this article this is referred to as the input-blocking constraint. The point-to-point connection has a severer restriction than the multipoint connection, which allows one input to be connected with more than one output simultaneously. Unless a reasonable traffic control scheme is adopted, these three constraints degrade the network throughput.

The reverse baseline network considered here has input buffers for switching configuration, but has no internal buffers [17]. The data in the network is transmitted in the fixed length packet form. The network is also operated in cycles synchronized by a single clock. In the first step of a communication cycle, conflict-free traffic flows or packets are determined by the proposed algorithm. In the second step of the cycle, the states of all switching elements are selected so that

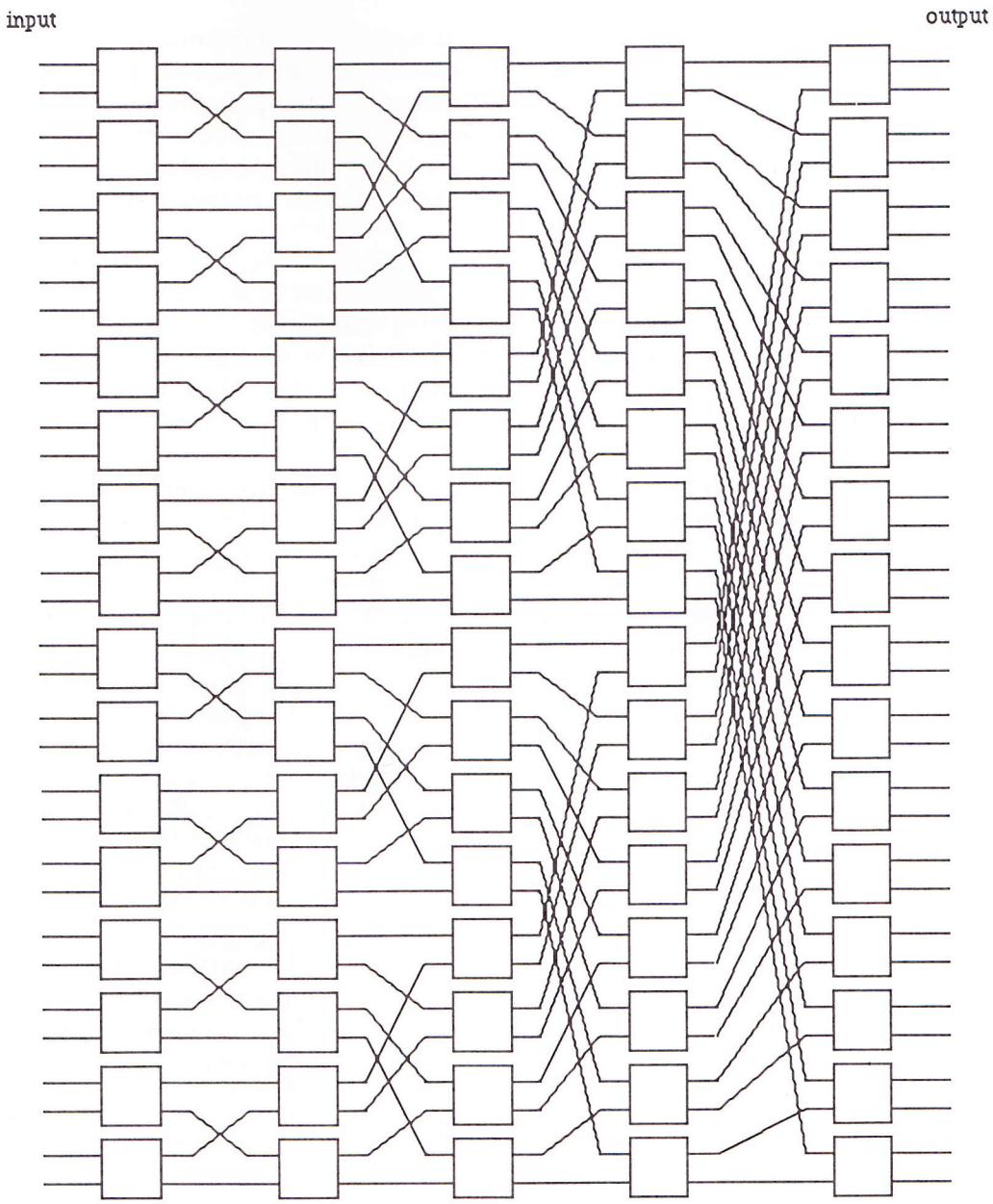


Figure 3b. A 32 × 32 reverse baseline network.

the packets can be transmitted through the network. In the third step of this cycle, the packets are finally transmitted from inputs to outputs.

A request for the packet transmission through an $n \times n$ reverse baseline network is described by an $n \times n$ traffic matrix T where n inputs are connected with n outputs. In the traffic matrix T, each element t_{ij} represents a request for packets

to be transmitted from the ith input to the jth output; $t_{ij} = 0$ means that there is no packet to be transmitted, and $t_{ij} = 1$ means that there is at least one packet to be transmitted from the ith input to the jth output. The traffic matrix usually has a random distribution of 0–1 elements, because the incoming traffic in the network is usually randomly generated. In order to maximize the throughput of the network, we must find the possible conflict-free traffic flows, given the requests in the traffic matrix.

Figure 4 shows an example of a 4 × 4 traffic matrix corresponding to the 4 × 4 reverse baseline network in Figure 3a. The traffic matrix shows that there are a total of seven packet requests to be transmitted from inputs to outputs in this network. There are packets to be transmitted from first input to first output, from first input to fourth output, from second input to third output, from second input to fourth output, from third input to second output, from fourth input to second output, and from fourth input to third output.

The problem of finding conflict-free traffic flows for the given traffic matrix in the given multistage interconnection networks has been reported in several articles. In 1968, Batcher proposed the first approach to this problem, an approach that is now called the Batcher network [2]. In the Batcher network, the traffic requests are sorted before transmission so that they can have conflict-free paths in the network. However, the Batcher network has some drawbacks. It needs more switching elements than a multistage interconnection network itself, and it cannot avoid input and output blockings. In 1980, Wu and Feng proposed a sequential algorithm for this problem [8]. The Wu–Feng algorithm deletes requests that have the most conflicts with other requests until all remaining requests can be transmitted without conflict. Their algorithm, however, does not guarantee discovery of the maximum throughput solution. In 1983, Agrawal proposed a sequential algorithm for the same problem based on the graph theory [12]. In 1990, Brown and Liu proposed a parallel algorithm and a circuit design for the same problem in Banyan networks based on the Hopfield neural network model [27]. Although their approach is similar to ours, their algorithm has the following disadvantages: (a) they use sigmoid functions that result in a slower convergence time; and (b) they use the decay term in the Hopfield neural network model, which has been proven to decrease the probability of convergence for the system [30]; and (c) they do not discuss the time

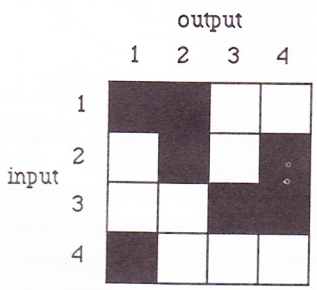


Figure 4. A 4 × 4 traffic matrix.

complexity and the convergence probability of the systems, matters that are always problematic in neural network research.

NEURAL NETWORK APPROACH

In this article we propose a parallel algorithm based on the two-dimensional artificial neural network model for finding conflict-free traffic flows for the traffic matrix in the multistage interconnection network. The reverse baseline network is used as a benchmark. The artificial neural network model is composed of a large number of simple processing elements that are massively interconnected. The processing elements are called neurons because they perform the function of the simplified biological neuron. The interconnections between processing elements are determined by the motion equation:

$$\frac{dU_{ij}}{dt} = -\frac{\partial E(V_{11}, V_{12}, \ldots, V_{nn})}{\partial V_{ij}} \tag{1}$$

where U_{ij} and V_{ij} are the input and the output of the ijth processing element, respectively. Note that E is called the computational energy function given by considering the necessary and sufficient constraints of the problem. The goal of using the artificial neural network model is to minimize the fabricated energy function, E. Theorem 1 in the Appendix shows that the motion equation uses a gradient descent method to minimize the energy function, E.

In 1943 McCulloch and Pitts proposed the first mathematical neuron model [31]. The input/output function of the ijth processing element in the McCulloch–Pitts neuron model is given by:

$$\begin{aligned} V_{ij} &= 1 \text{ if } U_{ij} > 0 \\ &= 0 \text{ otherwise.} \end{aligned} \tag{2}$$

The sigmoid neural network model for solving combinatorial optimization problems was first introduced by Hopfield and Tank [31]. The input/output function of the ijth processing element in the sigmoid neuron model is given by:

$$V_{ij} = \frac{1}{2}[1 + \tan h(\lambda U_{ij})], \tag{3}$$

where λ is a gain parameter. In 1989, Marrakchi and Troudet proposed the Hopfield neural network model for the crossbar switching systems, where it was verified only in an 8×8 crossbar switching system [33]. In 1989, Brown proposed the Hopfield neural network model for the multistage crossbar switching systems [34], and in 1990, Brown and Liu also proposed the Hopfield neural network model for the Banyan network systems [27]. However, all the previous investigators did not discuss the time complexity and the convergence probability of the systems, matters

that are always problematic in neural network research. Takefuji and Lee proved that the decay term in the Hopfield neural network model decreases the convergence probability of the system [30]; in other words, under some conditions, the decay term hinders the system convergence. Therefore, the decay term is not used in our algorithm.

In order to enhance the convergence speed, the McCulloch–Pitts neural network model has been used for finding near-optimum solutions of several NP-complete or optimization problems [30, 35–48]. However, the McCulloch–Pitts model sometimes introduces undesirable oscillatory behavior. A hysteresis McCulloch–Pitts neural network model, which was proposed by Takefuji and Lee, has been shown to suppress the oscillatory behavior of neural dynamics, consequently shortening the convergence time [40]. The input/output function of the ijth processing element in the hysteresis McCulloch–Pitts neuron model is given by:

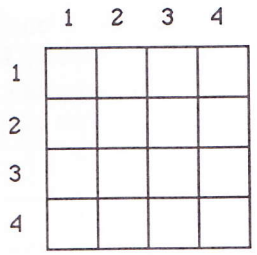
$$\begin{aligned} V_{ij} &= 1 \text{ if } U_{ij} > \text{UTP (Upper Trip Point)} \\ &= 0 \text{ if } U_{ij} < \text{LTP (Lower Trip Point)} \\ &\quad \text{unchanged otherwise} \end{aligned} \tag{4}$$

where UTP is always larger than LTP and the initial value of V_{ij} must be assigned as 1 or 0. Theorem 2 in the Appendix shows that a discrete motion equation based on the first-order Euler method forces a system composed of the hysteresis McCulloch–Pitts neurons to converge to the local minimum [41].

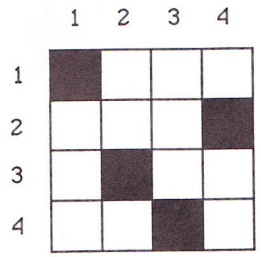
We verified our parallel algorithm by solving 40 problems where the size of the reverse baseline networks was varied from 4×4 to 32×32. The traffic matrices are randomly generated where the density of nonzero elements is also varied from 10 to 100%. The simulation results shown and discussed in the following.

SYSTEM REPRESENTATION

Figure 5a shows the two-dimensional system representation for the traffic control problem of the 4×4 traffic matrix in Figure 4. A total of 16 ($=4 \times 4$) processing elements are used to find conflict-free traffic flows in this problem. Generally, a total of n^2 processing elements are required for the traffic control problem in a $n \times n$ multistage interconnection network, including the reverse baseline network. The state of the ijth processing element describes whether the request from the ith input to the jth output has been selected for transmission in a communication cycle or not. The processing element has two states: nonzero output and zero output. A nonzero output signifies that the corresponding traffic request has been selected for transmission in the cycle. A zero output signifies that the traffic request has not been selected for transmission in the cycle and remains in the input buffer. In Figure 5b, a black square indicates the nonzero output of a processing element and a white square indicates the zero output of a processing element. Figure 5b



(a) 4×4 processing elements for the traffic control problem in Fig. 3



(b) The convergence of the 4×4 processing elements to a solution

Figure 5. System representation for the traffic control problem in Figure 4.

shows that the four requests of the traffic matrix elements, t_{11}, t_{24}, t_{32}, and t_{43}, are selected for transmission in the current cycle.

As mentioned before, the reverse baseline network has three kinds of constraints: input-blocking constraint, output-blocking constraint, and internal blocking constraint. The input-blocking constraint is that each input can be simultaneously connected with only one output. In other words, if one and only one traffic request among the requests from the same input is selected for transmission in a communication cycle, it can both satisfy the input-blocking constraint and maximize the network throughput. This condition for the ijth processing element in an $n \times n$ traffic matrix problem is given by:

$$\left(\sum_{k=1}^{n} V_{ik} - 1\right). \tag{5}$$

The condition is zero if one processing element among n processing elements for the ith input has nonzero output in the cycle.

The output-blocking constraint is that each output can be simultaneously con-

nected with only one input. In other words, if one and only one traffic request among the requests to the same output is selected for transmission in a communication cycle, it can both satisfy the output-blocking constraint and maximize the network throughput. This condition for the ijth processing element in an $n \times n$ traffic matrix problem is given by:

$$\left(\sum_{k=1}^{n} V_{kj} - 1\right). \tag{6}$$

The condition is zero if one processing element among n processing elements for the jth output has nonzero output in this cycle.

The internal blocking constraint is that more than one path from inputs to outputs cannot share the same internal wiring in the network. In other words, a traffic request must not be selected for transmission in a communication cycle if another request sharing the same internal wirings has been selected previously in the cycle. This condition for the ijth processing element in an $n \times n$ traffic matrix problem is given by:

$$\sum_{\substack{p=1 \\ p \neq i}}^{n} \sum_{\substack{q=1 \\ q \neq j}}^{n} s_{ijpq} V_{pq} \tag{7}$$

where s_{ijpq} is an element of the internal blocking matrix S_{ij}, which describes the internal wiring sharing states of the path from the ith input to the jth output. If the path shares an internal wiring with the path from the pth input to the qth output, s_{ijpq} is 1, otherwise, it is 0. If the processing elements sharing internal wirings with the ijth processing element have nonzero output in the cycle, the condition in Equation 7 is nonzero.

The internal blocking matrix S_{ij} is determined by the network topology. The element s_{ijpq} in an $n \times n$ network is calculated by the following two-step procedure. In the first step, the internal wiring number W_{kij} at the kth stage for the communication path from the ith input to the jth output for $i = 1, \ldots, n$, $j = 1, \ldots, n$, and $k = 1, \ldots, (\log n - 1)$ is given by:

$$W_{kij} = \left(\left\lceil \frac{i}{2^k} \right\rceil - 1\right) \cdot 2^k + \left\lceil \frac{j}{\left(\frac{n}{2^k}\right)} \right\rceil \tag{8}$$

where the function $\lceil x \rceil$ gives the minimum integer that is greater than or equal to x. In the second step, the element s_{ijpq} of the internal blocking matrix S_{ij} for $i = 1, \ldots, n$, $j = 1, \ldots, n$, $p = 1, \ldots, n$, and $q = 1, \ldots, n$ is given by:

$$\begin{aligned} s_{ijpq} &= 1 \text{ if } W_{kij} = W_{kpq} \text{ for } {}^{\exists}k \in \{1, \ldots, (\log n - 1)\} \text{ and for } i \neq p \text{ and } j \neq q \\ &= 0 \text{ otherwise.} \end{aligned} \tag{9}$$

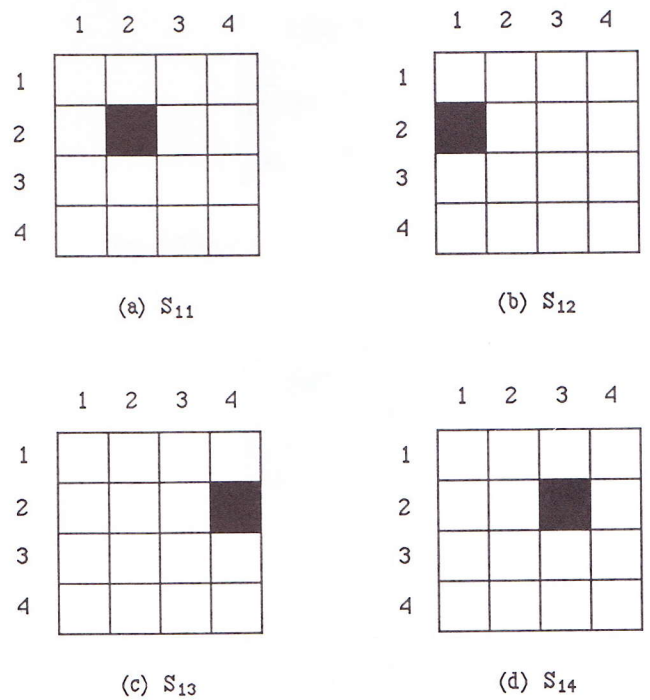


Figure 6. Four of the internal blocking matrices for the 4 × 4 reverse baseline network in Figure 3.

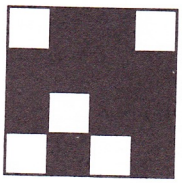
Figure 6 shows four of the 16 matrices, S_{11}, S_{12}, S_{13}, and S_{14}, for the 4 × 4 reverse baseline network in Figure 3a where a black square indicates the nonzero element and a white square indicates the zero element. For example, s_{1122} is 1 (black square) because the path from the first input to first output shares the first internal wiring with the path from the second input to the second output.

The motion equation for the ijth processing element corresponding to the ijth traffic matrix element t_{ij} in an $n \times n$ traffic matrix problem is given by:

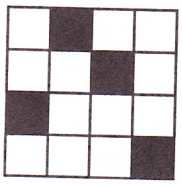
$$\frac{dU_{ij}}{dt} = -A\left(\sum_{k=1}^{n} V_{ik} - 1\right) - A\left(\sum_{k=1}^{n} V_{kj} - 1\right) - B\sum_{\substack{p=1 \\ p\neq i}}^{n}\sum_{\substack{q=1 \\ q\neq j}}^{n} s_{ijpq}V_{pq}$$

$$+ Ch\left(\sum_{k=1}^{n} V_{ik}\right) + Ch\left(\sum_{k=1}^{n} V_{kj}\right). \tag{10}$$

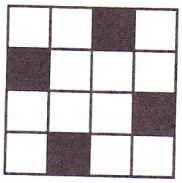
The first term in Equation 10 forces the output of one processing element among n processing elements for the ith input to be nonzero. The second term forces the output of one processing element among n processing elements for the jth output to be nonzero. The third term is the inhibitory force, that is, it discourages the output of the ijth processing element from being nonzero if the other processing elements sharing internal wirings with the ijth processing element have nonzero



(a) A 4×4 traffic matrix with 50% density



(b) Solution #1



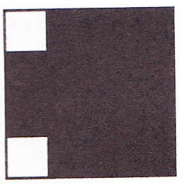
(c) Solution #2

Figure 7. The problem of a 4 × 4 traffic matrix with 50% density and two of its solutions.

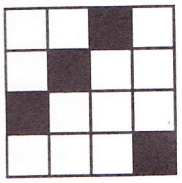
output. The last two terms provide a procedure, known as "hill climbing," which allows the state of the system to escape from the local minimum, and thus increases the probability of convergence to the global minimum. The last two terms encourage the output of the ijth processing element to be nonzero if none of n processing elements for the ith input and/or none of n processing elements for the jth output have nonzero output. The function $h(x)$ is 1 if $x = 0$, otherwise, it is 0. A, B, and C are constant coefficients.

PARALLEL ALGORITHM

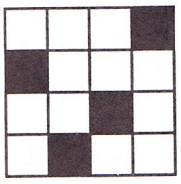
The following procedure describes the proposed parallel algorithm based on the first-order Euler method for the traffic control problem in an $n \times n$ reverse baseline network. The data set of the coefficients A, B, and C, UTP, LTP, U_max, U_min, T_max1, and T_max2 are empirically determined. The procedure must be per-



(a) A 4×4 traffic matrix with 80% density



(b) Solution #1

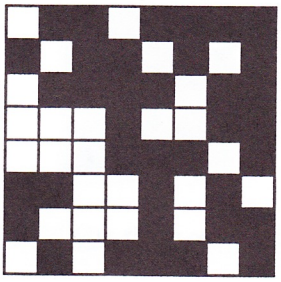


(c) Solution #2

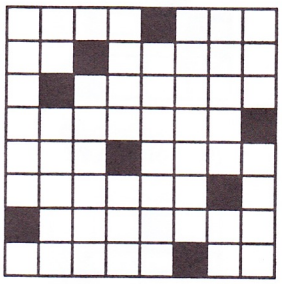
Figure 8. The problem of a 4 × 4 traffic matrix with 80% density and two of its solutions.

formed only for the processing elements corresponding to nonzero traffic elements where at least one packet has been requested to be transmitted.

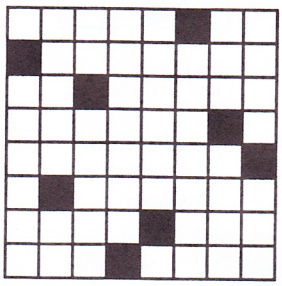
0. Set $t = 0$, $A = B = 1$, $C = 2$, UTP $= 5$, LTP $= -5$, $U\_\max = 50$, $U\_\min = -200$, $T\_\max1 = 500$, and $T\_\max2 = 1000$. ($U\_\max$ and $U\_\min$ are the constant upper and lower limits of $U_{ij}(t + 1)$, respectively; $T\_\max2$ is the maximum number of iteration steps.)
1. $U_{ij}(t)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$ are assigned uniformly randomized initial values between 0 and $U\_\min$, and $V_{ij}(t)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$ are assigned initial values of 0.
2. Use the motion equation in Equation 10 to compute $\Delta U_{ij}(t)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$. If $(t < T\_\max1)$ and $[(t \bmod 10) < 5]$, then



(a) An 8×8 traffic matrix with 50% density



(b) Solution #1

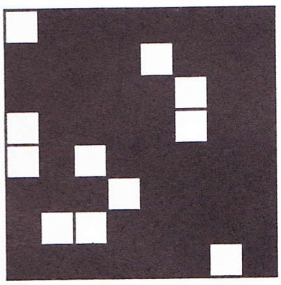


(c) Solution #2

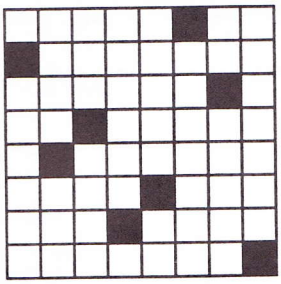
Figure 9. The problem of an 8 × 8 traffic matrix with 50% density and two of its solutions.

$$\Delta U_{ij}(t) = -A\left(\sum_{k=1}^{n} V_{ik}(t) - 1\right) - A\left(\sum_{k=1}^{n} V_{kj}(t) - 1\right) - B\sum_{\substack{p=1 \\ p\neq i}}^{n}\sum_{\substack{q=1 \\ q\neq j}}^{n} s_{ijpq}V_{pq}(t)V_{ij}(t)$$

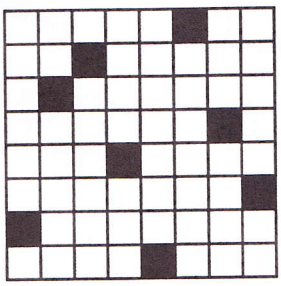
$$+ Ch\left(\sum_{k=1}^{n} V_{ik}(t)\right) + Ch\left(\sum_{k=1}^{n} V_{kj}(t)\right); \tag{11}$$



(a) An 8×8 traffic matrix with 80% density



(b) Solution #1



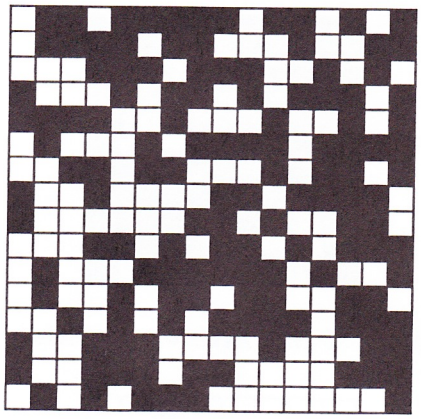
(c) Solution #2

Figure 10. The problem of an 8 × 8 traffic matrix with 80% density and two of its solutions.

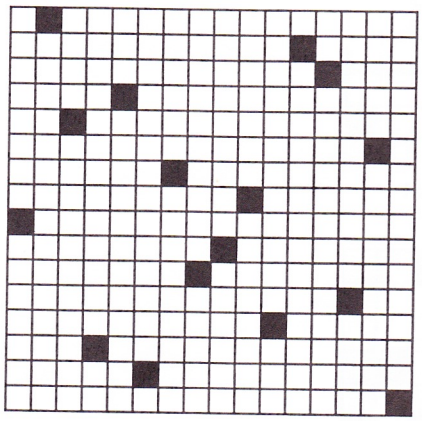
If $(t < T\_\max 1)$ and $[(t \bmod 10) \geq 5]$, then

$$\Delta U_{ij}(t) = -A\left(\sum_{k=1}^{n} V_{ik}(t) - 1\right) - A\left(\sum_{k=1}^{n} V_{kj}(t) - 1\right) - B\sum_{\substack{p=1 \\ p\neq i}}^{n}\sum_{\substack{q=1 \\ q\neq j}}^{n} s_{ijpq}V_{pq}(t)$$

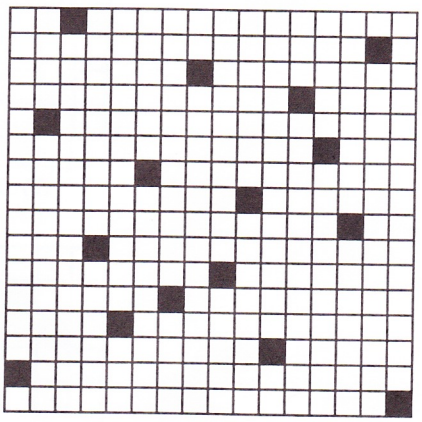
$$+ Ch\left(\sum_{k=1}^{n} V_{ik}(t)\right) + C\left(h\sum_{k=1}^{n} V_{kj}(t)\right); \tag{12}$$



(a) A 16×16 traffic matrix with 50% density



(b) Solution #1



(c) Solution #2

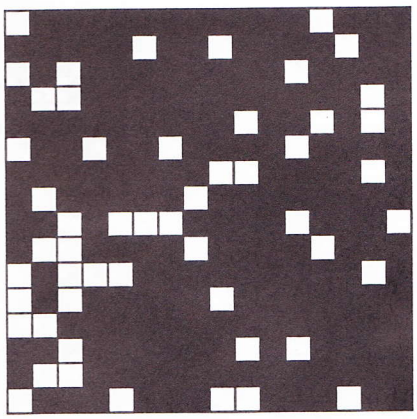
Figure 11. The problem of a 16 × 16 traffic matrix with 50% density and two of its solutions.

If $(t \geq T\_\max 1)$ and $[(t \bmod 10) < 5]$, then

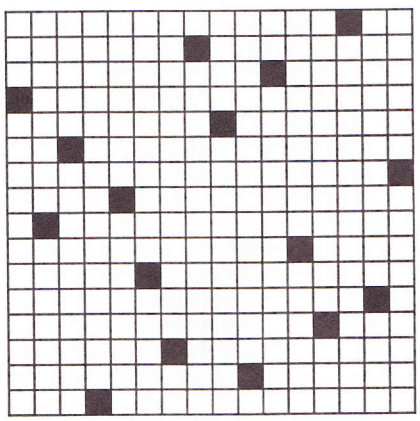
$$\Delta U_{ij}(t) = -A\left(\sum_{k=1}^{n} V_{ik}(t) - 1\right) - A\left(\sum_{k=1}^{n} V_{kj}(t) - 1\right) - B\sum_{\substack{p=1 \\ p\neq i}}^{n}\sum_{\substack{q=1 \\ q\neq j}}^{n} s_{ijpq}V_{pq}(t)V_{ij}(t); \tag{13}$$

If $(t \geq T\_\max 1)$ and $[(t \bmod 10) \geq 5]$, then

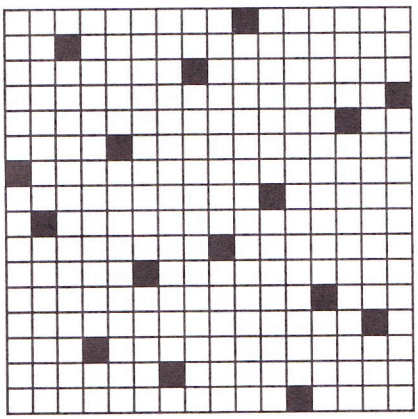
$$\Delta U_{ij}(t) = -A\left(\sum_{k=1}^{n} V_{ik}(t) - 1\right) - A\left(\sum_{k=1}^{n} V_{kj}(t) - 1\right) - B\sum_{\substack{p=1 \\ p\neq i}}^{n}\sum_{\substack{q=1 \\ q\neq j}}^{n} s_{ijpq}V_{pq}(t). \tag{14}$$



(a) A 16×16 traffic matrix with 80% density



(b) Solution #1



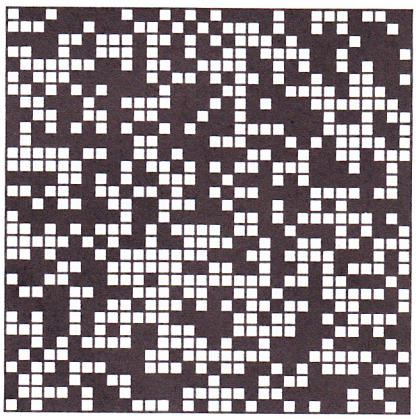
(c) Solution #2

Figure 12. The problem of a 16 × 16 traffic matrix with 80% density and two of its solutions.

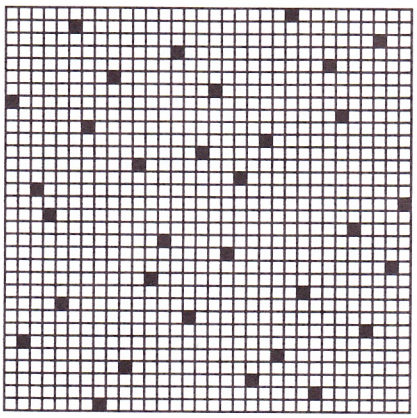
3. Compute $U_{ij}(t + 1)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$ based on the first-order Euler method:

$$U_{ij}(t + 1) = U_{ij}(t) + \Delta U_{ij}(t). \tag{15}$$

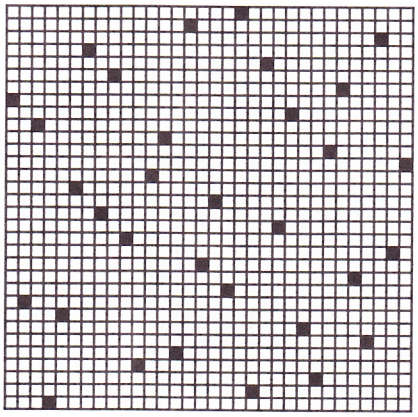
4. If $U_{ij}(t + 1) > U\_\max$ then $U_{ij}(t + 1) = U\_\max$. (16)
   If $U_{ij}(t + 1) < U\_\min$ then $U_{ij}(t + 1) = U\_\min$. (17)
5. Evaluate the values of $V_{ij}(t + 1)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$:



(a) A 32×32 traffic matrix with 50% density



(b) Solution #1



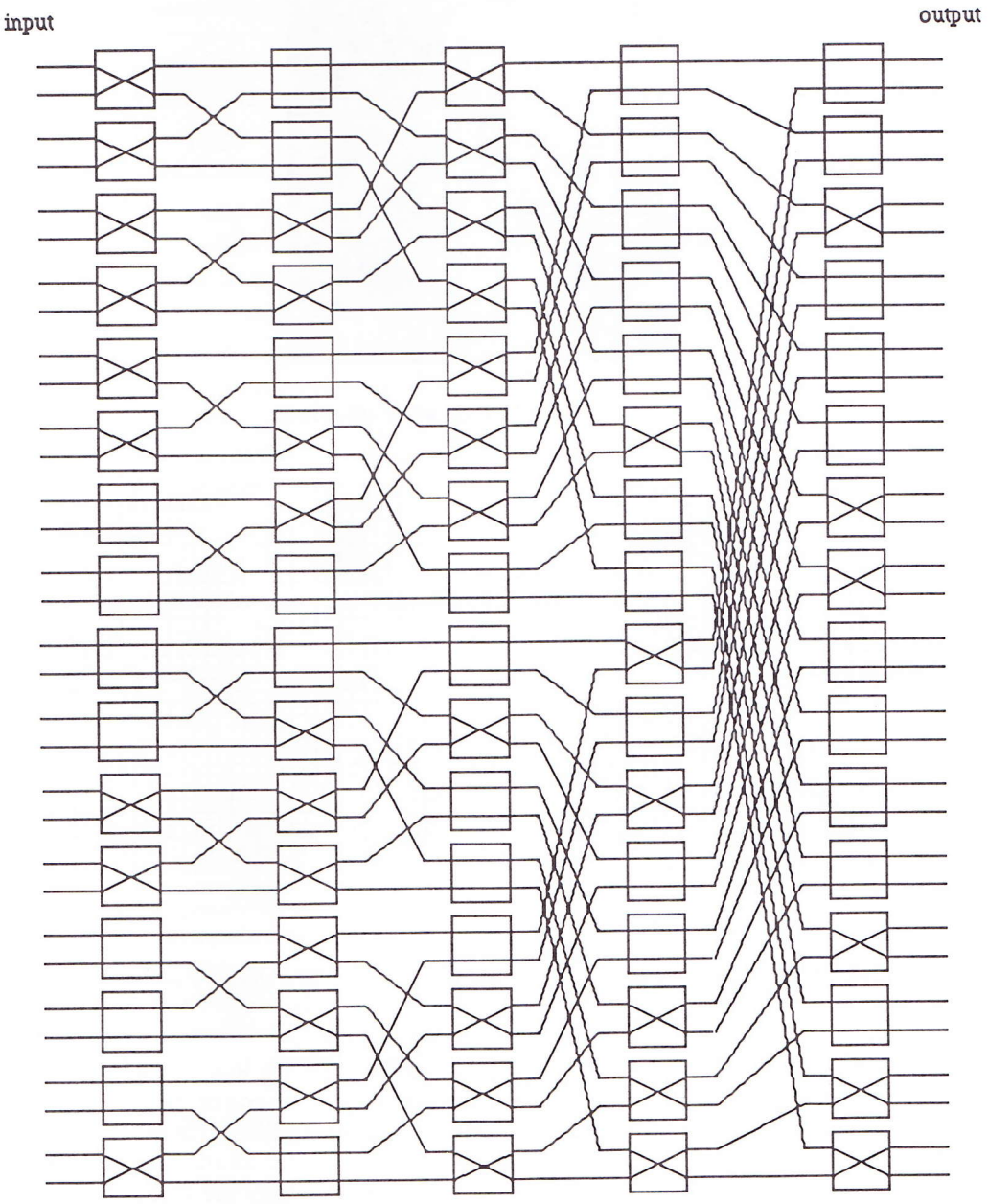
(c) Solution #2

Figure 13. The problem of a 32 × 32 traffic matrix with 50% density and two of its solutions. (d) appears on the facing page number 389.

$$\begin{aligned} V_{ij}(t+1) &= 1 \text{ if } U_{ij}(t+1) > \text{UTP} \\ &= 0 \text{ if } U_{ij}(t+1) < \text{LTP} \\ &\quad \text{unchanged otherwise} \end{aligned} \tag{18}$$

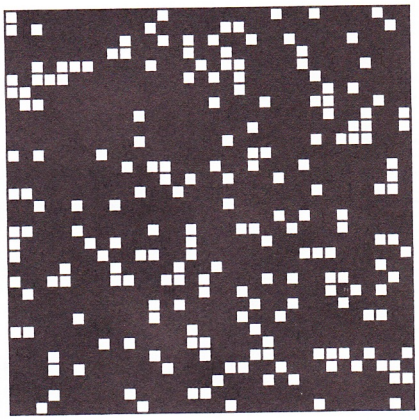
6. If all conflicts are resolved or $t = T\_\max 2$, then terminate this procedure; else increment t by 1 and return to Step 2.

The modified motion equations in Step 2 and the range limitation of the input $U_{ij}(t+1)$ in Step 4 empirically improve the convergence frequency to the global

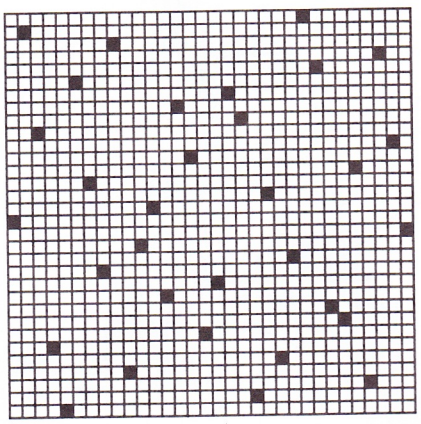


(d) The switching configuration corresponding to the solution #1

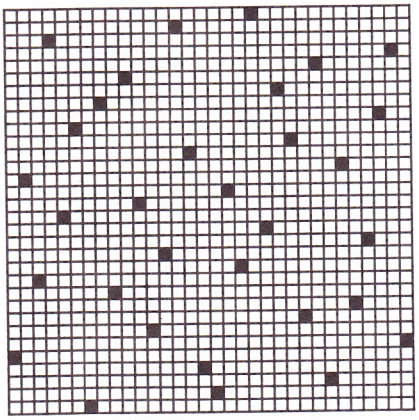
Figure 13. (*Continued*).



(a) A 32×32 traffic matrix with 80% density



(b) Solution #1



(c) Solution #2

Figure 14. The problem of a 32 × 32 traffic matrix with 80% density and two of its solutions.

minimum [37]. When the density of the given traffic matrix is low, it may happen that none of the processing elements for some inputs and/or none of the processing elements for some outputs have nonzero output without conflicts. Therefore, after T_max1 iteration steps, this procedure terminates in any valid solution, which does not always give a maximum throughput, but which does satisfy the three blocking constraints. Even though this procedure does not always provide a maximum solution, the average computation time is shorter than that of other methods.

The state of n^2 processing elements for the traffic control problem in an $n \times n$ reverse baseline network can be updated synchronously or asynchronously. In the synchronous parallel system, the states of all processing elements are updated simultaneously. In the asynchronous parallel system, the states of all processing

Table 1. Summary of Simulation Results for the 4 × 4 Network Problems

traffic matrix density	A	B	C	D
10%	-	0%	503.3	94%
20%	-	0%	505.6	100%
30%	-	0%	503.1	100%
40%	55.6	100%	55.6	100%
50%	67.1	100%	67.1	100%
60%	67.1	100%	67.1	100%
70%	54.5	100%	54.5	100%
80%	46.4	100%	46.4	100%
90%	38.8	100%	38.8	100%
100%	38.8	100%	38.8	100%

A: the average number of iteration steps required to converge to optimum solutions
B: the probability of convergence to optimum solutions
C: the average number of iteration steps required to converge to valid solutions
D: the probability of convergence to valid solutions

elements are updated randomly. In this article the synchronous parallel system is simulated on a sequential machine where the synchronous parallel system can be performed on maximally n^2 processors. An outline for the sequential program for simulating the synchronous parallel system follows:

```
Program parallel-simulator-on-a-sequential-machine
.....
initialization of Uij and Vij for i:=1 to n and for j:=1 to n;
.....
/*** Main Program ***/
while (a set of conflicts is not empty) do
begin
.....
/*** Updating all input values ***/
   for i:=1 to n
```

```
   for j:=1 to n
            Uij:=Uij+ΔUij;
/*** End of the first loop ***/
.....
/*** Updating all output values ***/
   for i:=1 to n
   for j:=1 to n
            If Uij>UTP then Vij:=1 else if Uij<LTP then Vij:=0;
/*** End of the second loop ***/
.....
end;
/*** Main Program end ***/
```

It is quite simple to simulate such a synchronous parallel model on a sequential machine. In the first loop, all input values U_{ij} are sequentially updated whereas all

Table 2. Summary of Simulation Results for the 8 × 8 Network Problems

traffic matrix density	A	B	C	D
10%	-	0%	506.2	100%
20%	-	0%	505.9	100%
30%	-	0%	503.5	100%
40%	144.8	84%	202.2	100%
50%	129.9	98%	137.6	100%
60%	79.4	100%	79.4	100%
70%	76.3	100%	76.3	100%
80%	68.0	100%	68.0	100%
90%	66.3	99%	70.7	100%
100%	62.8	100%	62.8	100%

A: the average number of iteration steps required to converge to optimum solutions
B: the probability of convergence to optimum solutions
C: the average number of iteration steps required to converge to valid solutions
D: the probability of convergence to valid solutions

Table 3. Summary of Simulation Results for the 16 × 16 Network Problems

traffic matrix density	A	B	C	D
10%	-	0%	512.2	92%
20%	-	0%	515.8	100%
30%	238.6	44%	389.0	99%
40%	207.5	56%	340.5	100%
50%	175.3	94%	195.2	100%
60%	143.8	98%	151.0	100%
70%	129.7	100%	129.7	100%
80%	107.4	100%	107.4	100%
90%	97.3	99%	101.3	100%
100%	84.5	99%	84.5	99%

A: the average number of iteration steps required to converge to optimum solutions
B: the probability of convergence to optimum solutions
C: the average number of iteration steps required to converge to valid solutions
D: the probability of convergence to valid solutions

output values V_{ij} are fixed. Then, in the second loop, all output values V_{ij} are sequentially updated whereas all input values U_{ij} are fixed. This is equivalent to updating simultaneously the values of all inputs and outputs. In a future article, we hope to discuss the use of an asynchronous parallel system simulator on a sequential machine.

SIMULATION RESULTS AND DISCUSSION

The simulator based on the proposed algorithm was developed on a Macintosh SE/30 and a Macintosh IIfx in order to verify the algorithm. We applied the simulator to 40 problems derived by varying the sizes of reverse baseline networks (4×4, 8×8, 16×16, and 32×32), and by varying the densities of nonzero

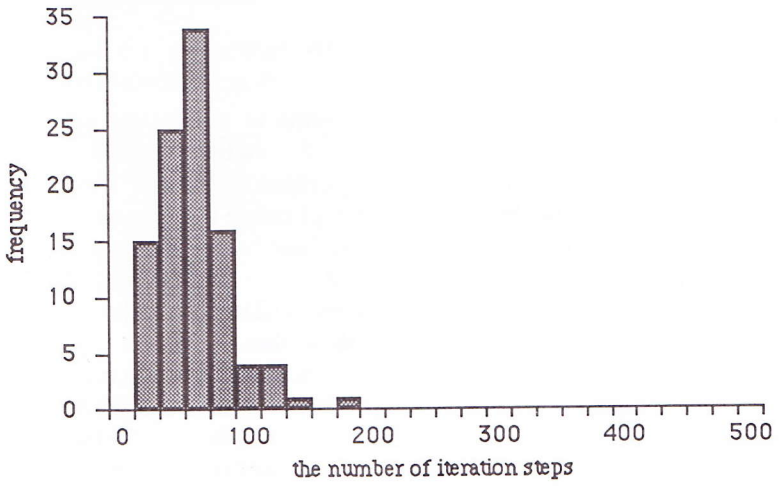
Table 4. Summary of Simulation Results for the 32 × 32 Network Problems

traffic matrix density	A	B	C	D
10%	-	0%	668.0	91%
20%	-	0%	562.6	97%
30%	277.6	18%	473.1	98%
40%	267.1	73%	332.0	100%
50%	233.6	88%	266.2	100%
60%	172.8	100%	172.8	100%
70%	160.5	96%	174.5	100%
80%	139.8	100%	139.8	100%
90%	125.5	100%	125.5	100%
100%	114.2	98%	122.1	100%

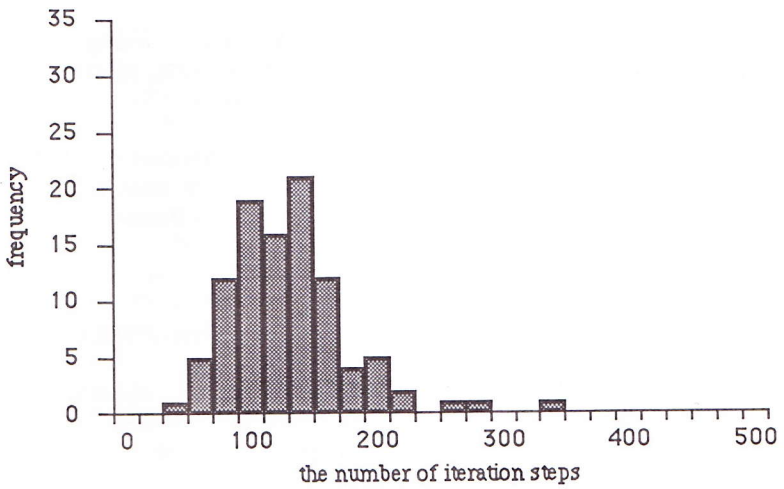
A: the average number of iteration steps required to converge to optimum solutions
B: the probability of convergence to optimum solutions
C: the average number of iteration steps required to converge to valid solutions
D: the probability of convergence to valid solutions

elements in the traffic matrices from 10 to 100% in increments of 10% for each network size. The matrix elements are randomly generated.

Figures 7–14 show the traffic matrices and their solutions for eight of the problems. The algorithm found several solutions for each problem. Tables 1–4 show the average number of iteration steps and the probability of convergence both to optimum solutions and to any valid solutions; 100 simulation runs were performed for each of the 40 problems. For each simulation run, the different initial values of $U_{ij}(t)$ were randomly generated. Figure 15 shows the distribution of the number of iteration steps required to converge to the optimum solutions for two of the problems. The simulation results show that the average number of iteration steps and the probability of convergence to solutions are not determined by the problem size. Based on our simulation results, we conclude that the proposed algorithm with n^2 processors finds solutions in a nearly constant time for traffic control



(a) The problem of an 8×8 traffic matrix with 80% density



(b) The problem of an 32×32 traffic matrix with 80% density

Figure 15. The distribution of the number of iteration steps required to converge to the optimum solutions.

problems in $n \times n$ multistage interconnection networks, including reverse baseline networks.

CONCLUSION

A parallel algorithm for traffic control problems in multistage interconnection networks is presented in this article. The reverse baseline network is used as the benchmark network. The proposed parallel algorithm requires n^2 simple processing elements for traffic control problems in $n \times n$ multistage interconnection networks. The algorithm runs not only on a sequential machine but also on a parallel machine with maximally n^2 processors. In 40 simulated problems, the algorithm finds solutions in a nearly constant time with n^2 processors. The simulation results support the consistency of the algorithm. They also show that the primary goal of finding conflict-free traffic flows in parallel processing can be successfully achieved in terms of the computation time and the solution quality. The algorithm is so flexible that it can easily be modified and extended to solve traffic control problems that have multipoint connections and/or have other types of multistage connection networks.

REFERENCES

[1] L.R. Goke and G.J. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," in *Proc. 1st Annual Intl. Symp. Comp. Arch.*, 1973, pp. 21–28.

[2] K.E. Batcher, "Sorting Networks and Their Applications," in *Proc. Spring Joint Computer Conf.*, 1968, pp. 307–314.

[3] T.Y. Feng, "Data Manipulating Functions in Parallel Processors and Their Implementations," *IEEE Trans. Comp.*, Vol. C-23, pp. 309–318, Mar. 1974.

[4] D.H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Comp.*, Vol. C-24, No. 12, pp. 1145–1155, Dec. 1975.

[5] K.E. Batcher, "The Flip Network in STARAN," in *Proc. Intl. Conf. Parallel Processing*, 1976, pp. 65–71.

[6] M.C. Pease, "The Indirect Binary *n*-Cube Microprocessor Array," *IEEE Trans. Comp.*, Vol. C-26, No. 5, pp. 458–473, May 1977.

[7] H.J. Siegel and S.D. Smith, "Study of Multistage SIMD Interconnection Networks," in *Proc. 5th Annual Intl. Symp. Comp. Arch.*, 1978, pp. 223–229.

[8] C.L. Wu and T.Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Comp.*, Vol. C-29, No. 8, pp. 694–702, Aug. 1980.

[9] D.M. Dias and J.R. Jump, "Analysis and Simulation of Buffered Delta Networks," *IEEE Trans. Comp.*, Vol. C-30, No. 4, pp. 273–282, Apr. 1981.

[10] J.H. Patel, "Performance of Processor–Memory Interconnections for Multiprocessors," *IEEE Trans. Comp.*, Vol. C-30, No. 10, pp. 771–780, Oct. 1981.

[11] T.Y. Feng, "A Survey of Interconnection Networks," *IEEE Comp.*, Vol. 14, pp. 12–27, Dec. 1981.

[12] D.P. Agrawal, "Graph-Theoretical Analysis and Design of Multistage Interconnection Networks," *IEEE Trans. Comp.*, Vol. C-32, No. 7, pp. 637–648, Jul. 1983.

[13] C.P. Kruskal and M. Snir, "The Performance of Multistage Interconnection Networks

for Multiprocessors," *IEEE Trans. Comp.*, Vol. C-32, No. 12, pp. 1091–1098, Dec. 1983.

[14] Y.C. Jenq, "Performance Analysis of a Packet Switch Based on Single-Buffered Banyan Network," *IEEE J. Select. Areas Comm.*, Vol. SAC-1, No. 6, pp. 1014–1021, Dec. 1983.

[15] M. Lee and C.L. Wu, "Performance Analysis of Circuit Switching Baseline Interconnection Networks," in *Proc. 11th Annual Intl. Symp. Comp. Arch.*, 1984, pp. 82–90.

[16] R.J. McMillen, "A Survey of Interconnection Networks," in *Proc. IEEE Globecom '84*, 1984, pp. 105–113.

[17] D.M. Dias and M. Kumar, "Packet Switching in *N* Log *N* Multistage Networks," in *Proc. IEEE Globecom '84*, 1984, pp. 114–120.

[18] A. Huang and S. Knauer, "Starlite: A Wideband Digital Switch," in *Proc. IEEE Globecom '84*, 1984, pp. 121–125.

[19] V. Cherkassky and M. Malek, "On Permuting Properties of Regular Rectangular SW-Banyans," *IEEE Trans. Comp.*, Vol. C-34, No. 6, pp. 542–546, Jun. 1985.

[20] J.Y. Hui and E. Arthurs, "A Broadband Packet Switch for Integrated Transport," *IEEE J. Select. Areas Comm.*, Vol. SAC-5, No. 8, pp. 1264–1273, Oct. 1987.

[21] U. Grag and Y.P. Huang, "Decomposing Banyan Networks for Performance Analysis," *IEEE Trans. Comp.*, Vol. 37, No. 3, pp. 371–376, Mar. 1988.

[22] M. J. Narasimha, "The Batcher-Banyan Self-Routing Network: Universality and Simplification," *IEEE Trans. Comm.*, Vol. 36, No. 10, pp. 1175–1178, Oct. 1988.

[23] H. Uematsu and R. Watanabe, "Architecture of a Packet Switch Based on Banyan Switching Network with Feedback Loops," *IEEE J. Select. Areas Comm.*, Vol. 6, No. 9, pp. 1521–1527, Dec. 1988.

[24] H. Yoon and K.Y. Lee, "B-Banyan and D-Delta Networks for Multiprocessor Systems," *Parallel Distributed Computing*, Vol. 7, pp. 570–582, 1989.

[25] T.H. Lee, "Simple Implementation of Load-Sharing Banyan Network and Its Throughput Performance," *Electronics Letters*, Vol. 26, No. 1, pp. 79–80, Jan. 1990.

[26] A. Youssef and B.W. Arden, "Equivalence Between Functionality and Topology of Log *N*-Stage Banyan Networks," *IEEE Trans. Comp.*, Vol. 39, No. 6, pp. 829–832, Jun. 1990.

[27] T.X. Brown and K.H. Liu, "Neural Network Design of a Banyan Network Controller," *IEEE J. Select. Areas Comm.*, Vol. 8, pp. 1428–1438, Oct. 1990.

[28] M. Dècina, "Progress Towards User Access Arrangements in Integrated Services Digital Networks," *IEEE Trans. Comm.*, Vol. COM-30, No. 9, pp. 2117–2130, Sep. 1982.

[29] J.S. Turner, "New Directions in Communications (Or Which Way to the Information Age?)," *IEEE Comm. Mag.*, Vol. 24, No. 10, pp. 8–15, Oct. 1986.

[30] Y. Takefuji and K.C. Lee, "Artificial Neural Networks for Four-Coloring Map Problems and *K*-Colorability Problems," *IEEE Trans. Circuits Systems*, Vol. 38, No. 3, pp. 326–333, Mar. 1991.

[31] W.S. McCulloch and W.H. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, 5, p. 115, 1943.

[32] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, pp. 141–152, 1985.

[33] A. Marrakchi and T. Troudet, "A Neural Net Arbitrator for Large Crossbar Packet-Switches," *IEEE Trans. Circuits Systems*, Vol. 36, No. 7, pp. 1039–1041, Jul. 1989.

[34] T.X. Brown, "Neural Networks for Switching," *IEEE Comm. Mag.*, Vol. 27, pp. 72–81, Nov. 1989.

[35] Y-P.S. Foo, Y. Takefuji, and H. Szu, "Binary Neurons with Analog Communication Links for Solving Large-Scale Optimization Problems," in *Proc. Intl. Neural Network Society Meeting*, 1988.

[36] Y. Takefuji and K.C. Lee, "A Near-Optimum Parallel Planarization Algorithm," *Science*, Vol. 245, pp. 1221–1223, Sep. 1989.

[37] Y. Takefuji and K.C. Lee, "A Parallel Algorithm for Tiling Problems," *IEEE Trans. Neural Networks*, Vol. 1, No. 1, pp. 143–145, Mar. 1990.

[38] Y. Takefuji, C.W. Lin, and K.C. Lee, "A Parallel Algorithm for Estimating the Secondary Structure in Ribonucleic Acids," *Biological Cybernetics*, Vol. 63, No. 5, pp. 337–340, 1990.

[39] Y. Takefuji, L.L. Chen, K.C. Lee, and J. Huffman, "Parallel Algorithms for Finding a Near-Maximum Independent Set of a Circle Graph," *IEEE Trans. Neural Networks*, Vol. 1, No. 3, pp. 263–267, Sep. 1990.

[40] Y. Takefuji and K.C. Lee, "An Artificial Hysteresis Binary Neuron: A Model Suppressing the Oscillatory Behaviors of Neural Dynamics," *Biological Cybernetics*, Vol. 64, pp. 353–356, 1991.

[41] Y. Takefuji and K.C. Lee, "A Super Parallel Sorting Algorithm Based on Neural Networks," *IEEE Trans. Circuits Systems*, Vol. 37, No. 11, pp. 1425–1429, Nov. 1990.

[42] N. Funabiki and Y. Takefuji, "A Parallel Algorithm for Spare Allocation Problems," *IEEE Trans. Reliability*, Vol. 40, No. 3, pp. 338–346, 1991.

[43] N. Funabiki and Y. Takefuji, "A Parallel Algorithm for Solving the "Hip" Games," *Neurocomputing*, Vol. 3, pp. 97–106, Jul. 1991.

[44] N. Funabiki and Y. Takefuji, "A Parallel Algorithm for Channel Routing Problems," *IEEE Trans. CAD/ICAS*, Vol. 11, No. 4, pp. 464–474, Apr. 1992.

[45] N. Funabiki and Y. Takefuji, "A Parallel Algorithm for Traffic Control Problems in Three-Stage Connecting Networks," *Journal of Parallel and Distributed Computing*, in press.

[46] K.C. Lee, N. Funabiki, and Y. Takefuji, "A Parallel Improvement Algorithm for the Bipartite Subgraph Problem," *IEEE Trans. Neural Networks*, Vol. 3, No. 1, pp. 139–145, Jan. 1992.

[47] N. Funabiki, Y. Takefuji, K.C. Lee, and Y.B. Cho, "A Neural Network Parallel Algorithm for Clique Vertex-Partition Problems," *International Journal of Electronics*, Vol. 72, No. 3, pp. 357–372, Mar. 1992.

[48] N. Funabiki, Y. Takefuji, and K.C. Lee, "A Neural Network Model for Finding a Near-Maximum Clique," *Journal of Parallel and Distributed Computing*, Vol. 14, No. 3, pp. 340–344, Mar. 1992.

APPENDIX

Theorem 1. The system always satisfies $\frac{dE}{dt} \leq 0$ under two conditions:

$$\text{(Condition 1)} \quad \frac{dU_{ij}}{dt} = -\frac{\partial E}{\partial V_{ij}}$$

$$\text{(Condition 2)} \quad V_{ij} = f(U_{ij})$$

where E is the computational Liapunov energy function and $f(U_{ij})$ is a nondecreasing function.

Proof. Consider the derivatives of the computational energy E with respect to time t:

$$\frac{dE}{dt} = \sum_i \sum_j \frac{dV_{ij}}{dt} \frac{\partial E}{\partial V_{ij}}$$

$$= \sum_i \sum_j \frac{dU_{ij}}{dt} \frac{dV_{ij}}{dU_{ij}} \frac{\partial E}{\partial V_{ij}}$$

$$= -\sum_i \sum_j \left(\frac{dU_{ij}}{dt}\right)^2 \frac{dV_{ij}}{dU_{ij}} \qquad \text{where } \frac{\partial E}{\partial V_{ij}} \text{ is replaced by } -\frac{dU_{ij}}{dt} \text{ (Condition 1)}$$

$$\leq 0 \qquad \text{where } \frac{dV_{ij}}{dU_{ij}} \geq 0 \text{ (Condition 2).}$$

Q.E.D.

Theorem 2. The system always satisfies $\dfrac{\Delta E}{\Delta t} \leq 0$ under two conditions:

$$\text{(Condition 1)} \quad \frac{\Delta U_{ij}}{\Delta t} = -\frac{\Delta E}{\Delta V_{ij}}$$

$$\text{(Condition 2)} \quad V_{ij} = f(U_{ij})$$

where E is the computational Liapunov energy function and $f(U_{ij})$ is the hysteresis McCulloch–Pitts binary function:

$$f(U_{ij}) = 1 \qquad \text{if } U_{ij} > UTP$$

$$= 0 \qquad \text{if } U_{ij} < LTP$$

$$\text{unchanged otherwise.}$$

Proof. Consider the derivatives of the computational energy E with respect to time t:

$$\frac{\Delta E}{\Delta t} = \sum_i \sum_j \frac{\Delta V_{ij}}{\Delta t} \frac{\Delta E}{\Delta V_{ij}}$$

$$= \sum_i \sum_j \frac{\Delta V_{ij}}{\Delta t} \left(-\frac{\Delta U_{ij}}{\Delta t}\right) \qquad \text{where } \frac{\Delta E}{\Delta V_{ij}} \text{ is replaced by } \left(-\frac{\Delta U_{ij}}{\Delta t}\right)$$

$$= -\sum_i \sum_j \left(\frac{\Delta U_{ij}}{\Delta t} \frac{\Delta V_{ij}}{\Delta U_{ij}}\right) \left(\frac{\Delta U_{ij}}{\Delta t}\right)$$

$$= -\sum_i \sum_j \left(\frac{\Delta V_{ij}}{\Delta U_{ij}}\right) \left(\frac{\Delta U_{ij}}{\Delta t}\right)^2.$$

Let $\dfrac{\Delta U_{ij}}{\Delta t}$ be $\dfrac{U_{ij}(t + \Delta t) - U_{ij}(t)}{\Delta t}$. Let $\dfrac{\Delta V_{ij}}{\Delta U_{ij}}$ be $\dfrac{V_{ij}(t + \Delta t) - V_{ij}(t)}{U_{ij}(t + \Delta t) - U_{ij}(t)}$. It is necessary and sufficient to consider the following four regions:

Region 1: $U_{ij}(t) > \text{UTP}$ and $V_{ij}(t) = 1$,
Region 2: $\text{LTP} \leq U_{ij}(t) \leq \text{UTP}$ and $V_{ij}(t) = 1$,
Region 3: $\text{LTP} \leq U_{ij}(t) \leq \text{UTP}$ and $V_{ij}(t) = 0$, and
Region 4: $U_{ij}(t) < \text{LTP}$ and $V_{ij}(t) = 0$

In Region 1, we must consider four possible cases for $U_{ij}(t + \Delta t)$:

(a) $U_{ij}(t + \Delta t) > U_{ij}(t)$,
(b) $\text{LTP} \leq U_{ij}(t + \Delta t) < U_{ij}(t)$,
(c) $U_{ij}(t + \Delta t) < \text{LTP} < U_{ij}(t)$, and
(d) $U_{ij}(t + \Delta t) = U_{ij}(t)$.

In (a) and (b), $V_{ij}(t + \Delta t) = V_{ij}(t) = 1 \Rightarrow \dfrac{\Delta V_{ij}}{\Delta U_{ij}} = 0$. Therefore, $\dfrac{\Delta E}{\Delta t} = 0$.

In (d), $\dfrac{\Delta U_{ij}}{\Delta t} = 0 \Rightarrow \dfrac{\Delta E}{\Delta t} = 0$.

In (c), $V_{ij}(t + \Delta t) = 0 \Rightarrow \dfrac{\Delta V_{ij}}{\Delta U_{ij}} = \dfrac{0 - 1}{\text{negative number}} > 0$ and $\dfrac{\Delta U_{ij}}{\Delta t} < 0$.Therefore,

$\dfrac{\Delta E}{\Delta t} < 0$.

It is concluded that $\dfrac{\Delta E}{\Delta t} \leq 0$ is always satisfied in Region 1.

Similarly, in Regions 2, 3, and 4, $\dfrac{\Delta E}{\Delta t} \leq 0$ is always satisfied. This completes the proof.

Q.E.D.

Nobuo Funabiki received the B.Sc. degree in mathematical engineering and information physics from the University of Tokyo, Japan, in 1984, and the M.Sc. degree in electrical engineering from Case Western Reserve University, Ohio, in 1991. He is currently a senior engineer at the System Engineering Division, Sumitomo Metal Industries, Ltd. Amagasaki 660, Japan. His research interests include neural network applications for optimization problems and process control problems, and the industrial use of Petri net theory. He is a member of the IEEE Computer Society.

Yoshiyasu Takefuji is an associate professor on the faculty of Environmental Information at Keio University (Fujisawa, 252, Japan) since 1992 and also on the faculty of Electrical Engineering at Case Western Reserve University (Cleveland, Ohio 44106) since 1988. Before joining Case, he taught at the University of South Florida and the University of South Carolina. He received his B.S. (1978), M.S. (1980), and Ph.D. (1983) from Electrical Engineering from Keio University (Japan). His research interests focus on neural network parallel computing for solving real-world problems. He is interested in VLSI applications and silicon architecture. He received the National Science Foundation/Research Initiation Award in 1989 and is an NSF advisory panelist. A member of the IEEE Computer Society, ACM, International Neural Network Society, and American Association for the Advancement of Science, he received the Information Processing Society of Japan's best paper award in 1980. He wrote a book entitled "Neural Network Parallel Computing," published by Kluwer in January 1992 and coauthored two books *Digital Circuits*, (Ohm-Sha Publishers) in 1984 and *Neural Network Computing*, (Baifukan Publishers) in 1992. He was an editor of the Journal of Neural Network Computing and is an associate editor of IEEE Transactions on Neural Networks and Neurocomputing, and a guest editor of Journal Analog Integrated Circuits and Signal Processing in the special issue on analog VLSI neural networks and also guest editor of Neurocomputing in the special issue on neural network based optimization. He has published more than 100 articles.