JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 20, 236–240 (1994)

A Parallel Algorithm for Traffic Control Problems in Three-Stage Connecting Networks

NOBUO FUNABIKI

System Engineering Division, Sumitomo Metal Industries, Ltd., Amagasaki 660, Japan

AND

YOSHIYASU TAKEFUJI

Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, Ohio 44106, and Faculty of Environmental Information, Keio University, Fujisawa 252, Japan

A parallel heuristic algorithm for traffic control problems in three-stage connecting networks is presented in this paper. A three-stage connecting network consists of an input crossbar switching stage, an intermediate crossbar switching stage, and an output crossbar switching stage. The goal of our algorithm is to quickly and efficiently find a conflict-free switching assignment for communication demands through the network. The algorithm requires $n^2 \times m$ processing elements for the network composed of n input/output switches and m intermediate switches, where it runs not only on a sequential machine, but also on a parallel machine with maximally $n^2 \times m$ processors. The algorithm was verified by 1100 simulation runs with the network size from $10^2 \times 7$ to $50^2 \times 27$. The simulation results show that the algorithm can find a solution in nearly constant time with $n^2 \times m$ processors.

1. INTRODUCTION

Since Clos introduced in 1953 the three-stage connecting network shown in Fig. 1, it has been extensively studied and used in digital switching systems such as computer network systems, telephone network systems, and satellite network systems [1, 2, 5, 6, 12–14, 17–19, 26, 27]. The input switching stage in Fig. 1 is composed of n $s \times m$ crossbar switches, where each of s input channels in an input crossbar switch can be connected with one of m intermediate crossbar switches. The intermediate switching stage is composed of m $n \times n$ crossbar switches, where each input crossbar switch can be connected with any output crossbar switch. The output switching stage is composed of n $m \times s$ crossbar switches, where each of m intermediate crossbar switches is connected with one of s output channels in an output crossbar switch. By assigning one of the m intermediate crossbar switches, each input channel can be connected with an output channel.

Point-to-point connections are considered in this paper where one input channel is connected with one output channel. The constraints in the network are that two or more input channels in an input crossbar switch cannot be connected with the same intermediate crossbar switch simultaneously, and that two or more output channels in an output crossbar switch cannot be connected with the same intermediate crossbar switch simultaneously. A pattern of communication demands is described by an $n \times n$ traffic matrix T where each element t_{ij} represents the number of input channels in the ith input crossbar switch to be connected with output channels in the jth output crossbar switch. The traffic control problem is to assign the intermediate crossbar switch for each traffic matrix element.

Several sequential polynomial time algorithms for traffic control problems in three-stage connecting networks have been proposed. In 1962, Paull formalized the problem and proposed an ad hoc algorithm [18]. In 1968, Waksman proposed an algorithm based on a matrix decomposition procedure [27]. In 1974, Tsao-Wu modified the algorithm to improve the convergence speed [26]. In 1979, Ackroyd proposed a "call repacking" algorithm where existing connections are moved to the busiest part of the network [1]. In 1980, Jajszczyk and Rajski proposed four kinds of simple heuristic algorithms and compared their performance [13]. In 1990, Colombo *et al.* proposed an asynchronous control algorithm [6]. In 1981, Lev *et al.* proposed an $O((\log s \times n)^3)$ time parallel algorithm on $s \times n$ processors, where $s \times n$ is the number of input and output channels [14]. Lev's algorithm assumed that s is equal to m, which is the optimum condition to guarantee the rearrangeability of the network. They showed neither any empirical result nor any example.

This paper proposes an efficient, parallel, heuristic algorithm based on the artificial neural network model, which uses $n^2 \times m$ simple processing elements (neurons)

0743-7315/94 $6.00

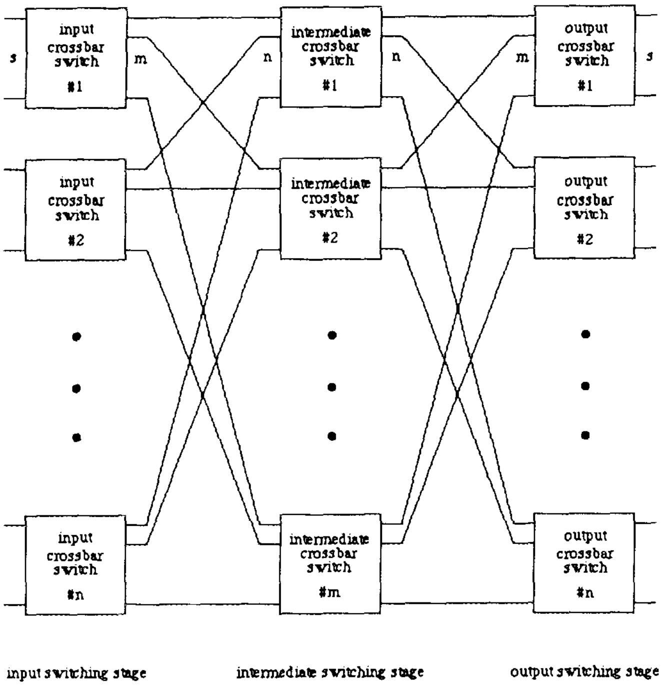

FIG. 1. A three-stage connecting network.

for n input/output crossbar switches and m intermediate crossbar switches. In 1989, Marrakchi and Troudet proposed a Hopfield neural network model [11] for traffic control problems in single crossbar switches [15]. In 1989, Brown proposed a Hopfield neural network model in multistage switches [3]. In 1990, Brown and Liu also proposed a Hopfield neural network model in Banyan networks [4]. However, these models use the decay term, which has been proven to be harmful for the system convergence [25]. They discussed neither the time complexity nor the system convergence, which are always controversial in neural network research.

In order to improve the convergence, the McCulloch–Pitts neural network model [16] has been used to solve optimization problems [7–10, 20–25]. Although the McCulloch–Pitts model sometimes introduces undesirable oscillatory behavior, it has been empirically shown that the hysteresis McCulloch–Pitts model suppresses it [24]. The output V_{ijk} of the ijkth processing element in a three-dimensional neural network model based on the hysteresis McCulloch–Pitts model is

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

where U_{ijk} is the input of the ijkth processing element. The change of U_{ijk} is given by the partial derivatives of the computational energy $E(V_{111}, \ldots, V_{nnm})$ with respect to V_{ijk}, which is called a motion equation. Note that E is given by considering the necessary and sufficient constraints in the problem. It has been proven that the motion equation forces the state of the system to converge to the local minimum [23, 25]. The neural network architecture was described in [10, 25].

2. SYSTEM REPRESENTATION AND SIMULATION RESULTS

Figure 2a shows the problem of a 4×4 traffic matrix and a network with $n = 4$, $m = 3$, and $s = 2$. Figure 2b shows the system representation for the problem. Three processing elements are used to assign a demand of the traffic matrix to one of three intermediate crossbar switches, where a total of 48 ($= 4 \times 4 \times 3$) processing elements are required in this problem. Generally, $n^2 \times m$ processing elements are used to solve the traffic control problem with an $n \times n$ traffic matrix T and m intermediate crossbar switches. Among m processing elements representing the intermediate switch assignment for an element t_{ij} of T, a total of t_{ij} processing elements should have nonzero output. For example, as shown in Fig. 2b, two processing elements among three for t_{11} should have nonzero output. The black square and the white square indicate the nonzero output ($V_{ijk} = 1$) and the zero output ($V_{ijk} = 0$) respectively. The nonzero output means that the demand is assigned on the corresponding intermedi-

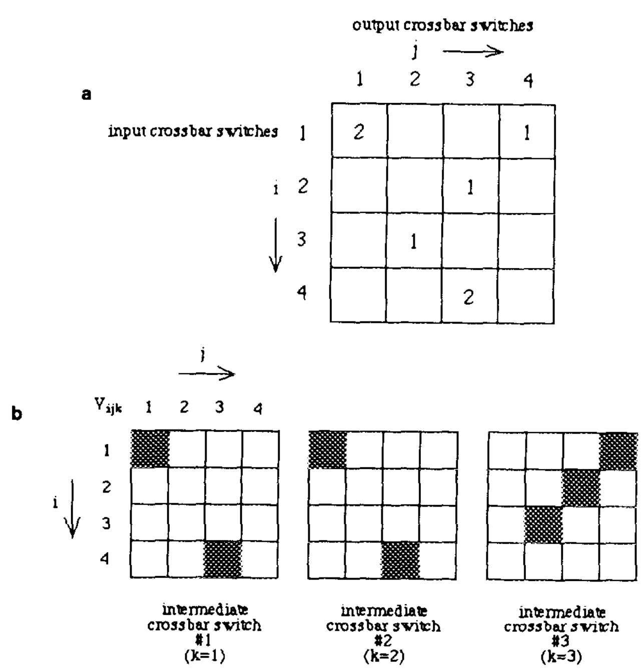

FIG. 2. System representation for a traffic control problem.

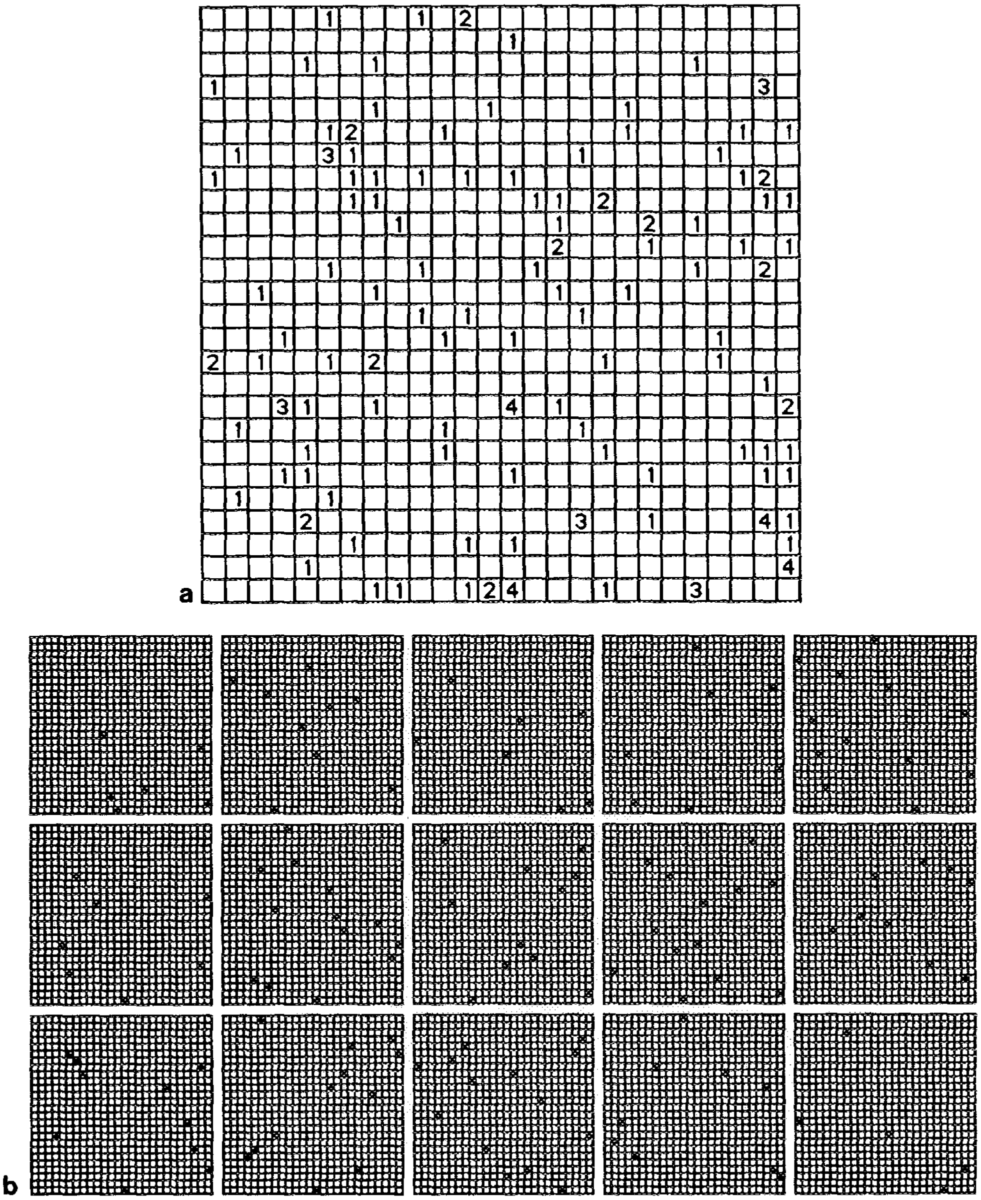

FIG. 3. The traffic matrix for Problem #10 and a solution.

ate switch. Figure 2b shows that demands t_{11} and t_{43} are assigned on the first and second intermediate switch, and demands t_{14}, t_{23}, and t_{32} are on the third intermediate switch.

The constraints in the three-stage connecting network are assumed to be that at most one demand per row and column elements of the traffic matrix can be simultaneously assigned on an intermediate crossbar switch. If other demands of the ith row and/or the jth column in the traffic matrix have been assigned to the kth intermediate crossbar switch, the ijth demand of the traffic matrix must not be assigned to the kth intermediate crossbar switch. These constraints are given by

$$\sum_{\substack{p=1\\p\neq i}}^{n} V_{pjk} + \sum_{\substack{q=1\\q\neq j}}^{n} V_{iqk}. \tag{2}$$

The constraints are nonzero, if other elements in the ith row and/or in the jth column are assigned to the kth intermediate crossbar switch.

The motion equation of the ijkth processing element which represents the assignment of the ijth traffic demand to the kth intermediate crossbar switch is given by

$$\frac{dU_{ijk}}{dt} = -A\left(\sum_{r=1}^{m} V_{ijr} - t_{ij}\right) - B\left(\sum_{\substack{p=1\\p\neq i}}^{n} V_{pjk} + \sum_{\substack{q=1\\q\neq j}}^{n} V_{iqk}\right) + Ch\left(\sum_{r=1}^{m} V_{ijr} - t_{ij}\right). \quad (3)$$

The A-term forces the output of t_{ij} processing elements among m processing elements to be nonzero, where the ijth traffic demand is assigned. The B-term represents the constraints. The C-term provides the hill-climbing which allows the state of the system to escape from the local minimum and increases the frequency to converge to the global minimum. The C-term encourages the output of the ijkth processing element to be nonzero, if the number of nonzero output processing elements for the ijth traffic demand is less than t_{ij}. The function $h(x)$ is 1 if $x < 0$, 0 otherwise.

The simulator has been developed based on the parallel procedure in [7–10] with the motion equation of Eq. (3), where the data set of $A = B = 1$, $C = 5$, UTP = 5, and LTP = −5 is used. Eleven problems in Table I were examined, where Problem #1 was taken from [6] and Problems #2–#11 were newly created. Figure 3 shows Problem #10 and one of the global minimum solutions. The algorithm found several solutions from different initial values of $U_{ijk}(t)$ for the same problem. Table I shows the average number of iteration steps required to converge to solutions, where for each problem, 100 simulation runs were performed from randomly generated initial values. Note that when the system did not converge to the global minimum within 500 steps, a new simulation run was performed from different random values, and the total number of iteration steps was used to calculate the average. The simulation results show that the average number of iteration steps does not depend on the problem size. Although the algorithm does not guarantee the global minimum solution, it can find a solution for a traffic problem with $n \times n$ traffic matrices and m intermediate switches in nearly constant time with $n^2 \times m$ processors. Similar behavior has been observed in other problems [7–10, 20–25], where more than 10,000 examples have been examined.

TABLE I
Summary of Simulation Results

Problem no.	Size of traffic matrix	Number of intermediate crossbar switches	Average number of iteration steps to solutions
Problem #1	12 × 12	7	55.2
Problem #2	10 × 10	7	77.6
Problem #3	12 × 12	8	175.5
Problem #4	14 × 14	9	82.6
Problem #5	16 × 16	10	118.6
Problem #6	18 × 18	11	128.0
Problem #7	20 × 20	12	98.5
Problem #8	22 × 22	13	133.5
Problem #9	24 × 24	14	108.5
Problem #10	26 × 26	15	88.9
Problem #11	50 × 50	27	103.8

3. CONCLUSION

The proposed parallel heuristic algorithm requires $n^2 \times m$ processing elements for three-stage connecting networks with $n \times n$ traffic matrices and m intermediate crossbar switches. The simulation results show that the algorithm can find a solution in nearly constant time with $n^2 \times m$ processors. We conclude that the primary goal of finding the conflict-free traffic demand assignment in parallel processing was successfully achieved in terms of the computation time and the solution quality.

REFERENCES

1. Ackroyd, M. H. Call repacking in connecting networks. *IEEE Trans. Commun.* **COM27,** 3 (Mar. 1979), 589–591.
2. Benes, V. E. Blocking states in connecting networks made of square switches arranged in stages. *Bell System Tech. J.* **60** (Apr. 1981), 511–521.
3. Brown, T. X. Neural networks for switching. *IEEE Commun. Mag.* **27** (Nov. 1989), 72–81.
4. Brown, T. X., and Liu, K. H. Neural network design of a Banyan network controller. *IEEE J. Select. Areas Commun.* **8** (Oct. 1990), 1428–1438.
5. Clos, C. A study of non-blocking switching networks. *Bell System Tech. J.* **32** (Mar. 1953), 406–424.
6. Colombo, G., Scarati, C., and Settimo. F. Asynchronous control algorithms for increasing the efficiency of three-stage connecting networks for multipoint services. *IEEE Trans. Commun.* **38,** 6 (June 1990), 898–905.
7. Funabiki, N., and Takefuji, Y. A parallel algorithm for spare allocation problems. *IEEE Trans. Reliability* **40,** 3 (Aug. 1991), 338–346.
8. Funabiki, N., and Takefuji, Y. A parallel algorithm for channel routing problems. *IEEE Trans. CAD/ICAS* **11,** 4 (Apr. 1992), 464–474.
9. Funabiki, N., and Takefuji, Y. A neural network model for finding a near-maximum clique. *J. Parallel Distrib. Comput.* **14,** 3 (Mar. 1992), 340–344.
10. Funabiki, N., *et al.* A neural network parallel algorithm for clique vertex-partition problems. *Int. J. Electron.* **72,** 3 (Mar. 1992), 357–372.
11. Hopfield, J. J., and Tank, D. W. Neural computation of decisions in optimization problems. *Biol. Cybernet.* **52** (1985), 141–152.
12. Hwang, F. K., and Jajszczyk, A. On nonblocking multiconnection networks. *IEEE Trans. Commun. COM-34,* 10 (Oct. 1986), 1038–1041.
13. Jajszczyk, A., and Rajski, J. The effect of choosing the switches for rearrangements in switching networks. *IEEE Trans. Commun.* **COM-28,** 10 (Oct. 1980), 1832–1834.

14. Lev, G. F., Pippenger, N., and Valiant, L. G. A fast parallel algorithm for routing in permutation networks. *IEEE Trans. Comput.* **C-30,** 2 (Feb. 1981), 93–100.

15. Marrakchi, A., and Troudet, T. A neural net arbitrator for large crossbar packet-switches. *IEEE Trans. Circuits Systems* **36,** 7 (July 1989), 1039–1041.

16. McCulloch, W. S., and Pitts, W. H. A logical calculus of ideas immanent in nervous activity. *Bull. Math. Biophys.* **5,** 115 (1943).

17. Opferman, D. C., and Tsao-Wu, N. T. On a class of rearrangeable switching networks. *Bell System Tech. J.* **50,** 5 (May–June 1971), 1579–1618.

18. Paull, M. C. Reswitching of connection networks. *Bell System Tech. J.* **41** (May 1962), 853–855.

19. Skaperda, N. J. Some architectural alternatives in the design of a digital switch. *IEEE Trans. Commun.* **COM-27,** 7 (July 1979), 961–972.

20. Takefuji, Y., and Lee, K. C. A near-optimum parallel planarization algorithm. *Science* **245** (Sept. 1989), 1221–1223.

21. Takefuji, Y., and Lee, K. C. A parallel algorithm for tiling problems. *IEEE Trans. Neural Networks* **1,** 1 (Mar. 1990), 143–145.

22. Takefuji, Y., Lin, C. W., and Lee, K. C. A parallel algorithm for estimating the secondary structure in ribonucleic acids. *Biol. Cybernet.* **63** (1990), 337–340.

23. Takefuji, Y., and Lee, K. C. A super parallel sorting algorithm based on neural networks. *IEEE Trans. Circuits Systems* **37,** 11 (Nov. 1990), 1425–1429.

24. Takefuji, Y., and Lee, K. C. An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics. *Biol. Cybernet.* **64** (1991), 353–356.

Received October 4, 1990; revised April 26, 1991; accepted June 16, 1992

25. Takefuji, Y., and Lee, K. C. Artificial neural networks for four-coloring map problems and K-colorability problems. *IEEE Trans. Circuits Systems* **38,** 3 (Mar. 1991), 326–333.

26. Tsao-Wu, N. T. On Neiman's algorithm for the control of rearrangeable switching networks. *IEEE Trans. Commun.* **COM-22,** 6 (June 1974), 737–742.

27. Waksman, A. A permutation network. *J. Assoc. Comput. Mach.* **15,** 1 (Jan. 1968), 159–163.

NOBUO FUNABIKI received his B.S. in mathematical engineering and information physics from the University of Tokyo in 1984 and his Ph.D. in 1993. In 1991 he received his M.S. in electrical engineering from Case Western Reserve University. He has published more than 10 papers on channel routing, traffic control in three-stage/multistage connecting networks, time slot assignment in TDM hierarchical switching systems, broadcast scheduling, and spare allocation. He has worked for Sumitomo Metal Industries, Ltd., in Japan since 1984.

YOSHIYASU TAKEFUJI is on the faculty of electrical engineering at Case Western Reserve University. Before joining Case in 1988, 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) in electrical engineering from Keio University (Japan). He received the best paper award in 1980 from IPSJ, and RIA from the National Science Foundation in 1989. He was an editor of the *Journal of Neural Network Computing,* is an associate editor of *IEEE Transactions on Neural Networks* and a guest editor of *JAICSP,* has published more than 80 papers, and has coauthored two books. He is the author of *Neural Network Parallel Computing* (Kluwer, 1992).