JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 14, 340–344 (1992)

A Neural Network Model for Finding a Near-Maximum Clique

NUBUO FUNABIKI

Systems Engineering Division, Sumitomo Metal Industries, Ltd., Japan

YOSHIYASU TAKEFUJI

Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, Ohio 44106

AND

KUO-CHUN LEE

R&D Department, Cirrus Logic, Inc., Fremont, California 94538

A parallel algorithm based on the neural network model for finding a near-maximum clique is presented in this paper. A maximum clique of a graph G is a maximum complete subgraph of G where any two vertices are adjacent. The problem of finding a maximum clique is NP-complete. The parallel algorithm requires n processing elements for an n-vertex graph problem. The algorithm is verified by solving 230 different graph problems. The simulation results show that our computation time on a Macintosh IIfx is shorter than that of two better known algorithms on a Cray 2 and an IBM 3090 while the solution quality is similar. The algorithm solves a near-maximum clique problem in nearly constant time on a parallel machine with n processors.

I. INTRODUCTION

A parallel algorithm based on the neural network model for finding a near-maximum clique of an arbitrary graph is presented. When a graph G with a set of vertices and a set of edges has an edge between two vertices, the vertices are called adjacent. A clique of G is a complete subgraph of G where any two vertices are adjacent to each other. The maximum clique is a clique with the largest number of vertices among cliques of G. The problem of finding a maximum clique of an arbitrary graph is known to be NP-complete [4]. This means that in the worst case the computing time for solving the maximum clique problem grows exponentially with the graph size. Figure 1a shows an 8-vertex 17-edge graph [2]. The maximum cliques are {2, 3, 4, 7} and {2, 4, 6, 7} as shown in Figs. 1b and 1c.

Several sequential algorithms for the maximum clique problem, in which the computation time not only depends on the number of vertices in the graph but also on the number of edges, have been reported. They can deal with neither large-sized graphs nor high-density graphs. Note that the density means the ratio between the number of edges in the n-vertex graph and that in the n-vertex complete graph $n(n-1)/2$. In 1986 Balas and Yu proposed an $O(n+m)$ time branch and bound algorithm for an n-vertex m-edge graph that was tested only by small-sized graphs with maximally 400 vertices and 30,000 edges [1]. In 1990 Pardalos and Phillips formulated the maximum clique problem as a linearly constrained indefinite quadratic global optimization problem [11]. Although the supercomputer Cray 2 was used, their algorithm could not solve larger than 75-vertex graph problems. It required more than 1 h even for the 75-vertex 91%-edge-density graph problem on the Cray 2. In 1990 Carraghan and Pardalos proposed an algorithm based on a partial enumeration [2]. Although it could solve up to 3000-vertex and over 1,000,000-edge graph problems on the mainframe IBM 3090, it required a prohibitively long computation time even for a middle-sized graph problem. For example, it took more than 1 h for a 1000-vertex 40%-density graph problem to be solved on the IBM 3090. In 1980 Mead and Conway proposed a parallel algorithm for the maximum clique problem that requires $O(n^2)$ time with maximum $O(2^n)$ processors for an n-vertex graph problem [9].

Several applications of the maximum clique problem for real-world problems have been reported. Balas and Yu introduced four applications: information retrieval, experimental design, signal transmission, and computer vision [1]. Ogawa proposed an application for the labeled point pattern matching problem [10]. Lecky *et al.* proposed an application for the PLA folding problem [7]. Horaud and Skordas proposed an application for the stereo vision correspondence problem [6].

0743-7315/92 $3.00

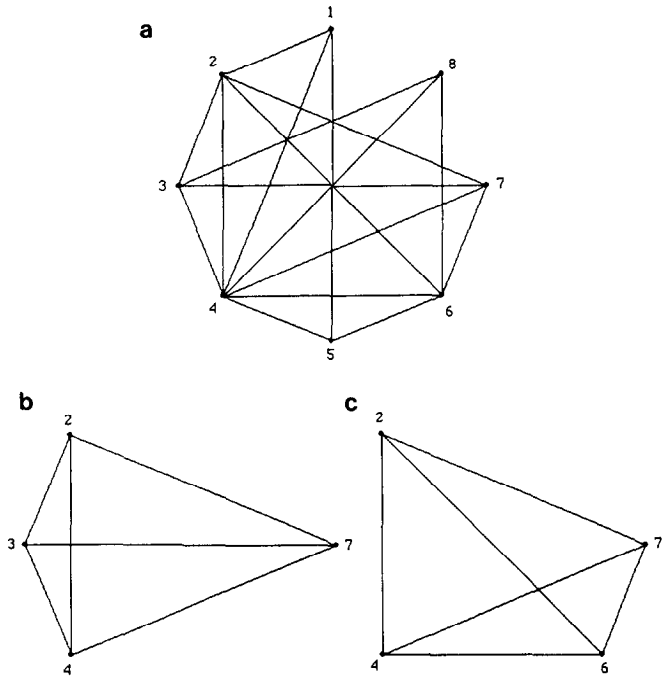

FIG. 1. An 8-vertex 17-edge graph and the maximum cliques. (a) The original graph. (b) The maximum clique No. 1. (c) The maximum clique No. 2.

Whereas existing algorithms require long computation times on expensive machines, our algorithm can solve large-sized problems both on an inexpensive machine such as a personal computer and on a parallel machine. It can also be realized on an analog circuit [14]. The computation time and the number of required processing elements in our algorithm do not depend on the graph density.

II. NEURAL NETWORK APPROACH

The neural network model is composed of n processing elements (neurons) for an n-vertex graph problem. Processing element i has an input U_i and an output V_i. The McCulloch–Pitts neuron model [8] is adopted in this paper where the input/output function is given by

$$V_i = 1 \quad \text{if } U_i > 0,\ V_i = 0 \text{ otherwise.} \tag{1}$$

The change of U_i is given by the motion equation in order to minimize the energy function $E(V_1, \ldots, V_n)$ which is determined by considering the necessary and sufficient constraints in the problem [3, 5, 12–14]. The motion equation is given by

$$\frac{dU_i}{dt} = -\frac{\partial E(V_1, \ldots, V_n)}{\partial V_i}. \tag{2}$$

It is proven that the motion equation forces the state of the neural network system to converge to the local minimum [14].

Figure 2 shows the neural network representation for the maximum clique problem in Fig. 1a where a total of eight processing elements are required. The output of processing element i represents whether vertex i belongs to the clique (solution) or not. The nonzero output ($V_i = 1$) means that vertex i belongs to the clique while the zero output ($V_i = 0$) means that it does not. Figure 2 shows the solution state corresponding to Fig. 1b where the black (white) square indicates the nonzero (zero) output. Our approach to the maximum clique problem is to maximize the number of vertices of a selected complete subgraph. The output of processing element i is zero if vertex i is not adjacent to a vertex in a complete subgraph and the output is nonzero if vertex i is adjacent to all vertices in the subgraph. The motion equation for processing element i in the n-vertex graph problem is composed of two terms, a negative force and a positive force, and is given by

$$\frac{dU_i}{dt} = -A \sum_{j=1}^{n} (1 - d_{ij}) V_j + Bh\left(\sum_{j=1}^{n} (1 - d_{ij}) V_j + V_i\right), \tag{3}$$

where d_{ij} is 1 if two vertices i and j are adjacent to each other, 0 otherwise. Note that $d_{ij} = d_{ji}$ and $d_{ii} = 1$ are always satisfied. The A-term discourages processing element i to have the nonzero output if vertex i is not adjacent to a vertex chosen in a clique. The B-term encourages processing element i to have the nonzero output if vertex i is adjacent to all vertices in the clique and the output of processing element i is zero. The function $h(x)$ is 1 if $x = 0$, 0 otherwise. A and B are constant coefficients. Because a vertex which has many adjacent vertices is more likely to belong to the maximum clique, the coefficient B is changed by the number of adjacent vertices in our algorithm:

$$B = \frac{\text{the number of adjacent vertices}}{n \times (\text{density of the graph})} \times 20$$
$$A = 1 \text{ is used in our simulation.} \tag{4}$$

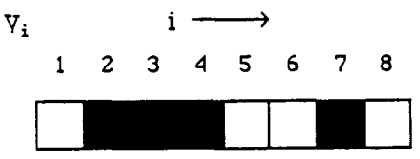

FIG. 2. Neural network representation for the maximum clique problem in Fig. 1.

TABLE I
Comparison of Simulation Results for 10-Vertex Graphs

Pardalos and Phillips's algorithm on Cray 2			Our algorithm on Macintosh IIfx		
Average density	Average clique size	Average comp. time	Average density	Average clique size	Average comp. time
0.10	2.1	0.3	0.06	2.0	0.03
0.22	2.9	0.5	0.20	2.5	0.03
0.50	3.7	0.6	0.53	3.9	0.03
0.75	5.5	0.4	0.76	5.4	0.03
0.88	7.0	0.2	0.93	7.5	0.03

TABLE III
Comparison of Simulation Results for 50-Vertex Graphs

Pardalos and Phillips's algorithm on Cray 2			Our algorithm on Macintosh IIfx		
Average density	Average clique size	Average comp. time	Average density	Average clique size	Average comp. time
0.10	2.7	93.6	0.09	2.7	0.12
0.25	3.6	98.1	0.25	3.9	0.13
0.50	5.9	86.3	0.51	6.7	0.13
0.75	10.4	78.7	0.77	11.7	0.15
0.90	18.8	69.2	0.89	18.9	0.16

III. SIMULATION AND CONCLUSION

The following procedure describes the parallel algorithm for finding a near-maximum clique of an n-vertex graph based on the first-order Euler method.

0. Set $t = 0$.
1. The initial values of $U_i(t)$ for $i = 1, \ldots, n$ are uniformly randomized between 0 and -20, and the initial values of $V_i(t)$ for $i = 1, \ldots, n$ are assigned to 0.

2. $$\Delta U_i(t) = -A \sum_{j=1}^{n} (1 - d_{ij}) V_j(t) + Bh\left(\sum_{j=1}^{n} (1 - d_{ij}) V_j(t) + V_i(t)\right). \quad (5)$$

3. $$U_i(t+1) = U_i(t) + \Delta U_i(t). \quad (6)$$

4. $$V_i(t+1) = 1 \quad \text{if } U_i(t+1) > 0,$$
$$V_i(t+1) = 0 \quad \text{otherwise.} \quad (7)$$

5. If $V_i(t) = 1$ and $\sum_{j=1}^{n} (1 - d_{ij}) V_j(t) = 0$

or $V_i(t) = 0$ and $\sum_{j=1}^{n} (1 - d_{ij}) V_j(t) \neq 0$

for $i = 1, \ldots, n$, then terminate this procedure else increment t by 1 and goto step 2.

The simulator has been developed on the Macintosh IIfx in order to verify our algorithm. Four graph size problems (10, 25, 50, and 75 vertices) with five edge densities (10, 25, 50, 75, and 90%) were simulated to compare our algorithm with Parados's algorithm on the Cray 2 [11] where 10 different graphs were randomly generated for each size–density and 10 simulation runs were performed with different initial values of $U_i(t)$ for each graph problem. Tables I–IV show the results on the solution quality and the computation time (seconds). Our computation time on the Macintosh IIfx is shorter than theirs on the Cray 2 while the solution quality is about the same. Note that the speed of the Cray 2 is 125 MIPS while that of the Macintosh IIfx is 7 MIPS. Ten randomly generated graph size problems (100–1000 vertices) with three edge densities (25, 50, and 75%) were simulated to compare our algorithm with Carraghan's algorithm on the IBM 3090 [2] where 100 simulation runs were performed with different initial values of $U_i(t)$ for each problem. Table V shows the average number of iteration steps to converge to solutions, and the average/maximum numbers of vertices in cliques found by our simulator. Table VI compares the computation time (seconds) between our algorithm on the Macintosh IIfx and Carraghan's algorithm on the IBM 3090. Tables I–IV and VI show that our algorithm is

TABLE II
Comparison of Simulation Results for 25-Vertex Graphs

Pardalos and Phillips's algorithm on Cray 2			Our algorithm on Macintosh IIfx		
Average density	Average clique size	Average comp. time	Average density	Average clique size	Average comp. time
0.09	2.6	6.7	0.09	2.4	0.04
0.24	3.4	7.3	0.25	3.3	0.04
0.49	5.5	7.3	0.53	5.3	0.05
0.75	8.7	6.7	0.78	9.3	0.06
0.91	14.2	4.7	0.91	13.8	0.06

TABLE IV
Comparison of Simulation Results for 75-Vertex Graphs

Pardalos and Phillips's algorithm on Cray 2			Our algorithm on Macintosh IIfx		
Average density	Average clique size	Average comp. time	Average density	Average clique size	Average comp. time
0.90	3.0	2283.0	0.10	2.8	0.24
0.24	4.0	3690.1	0.25	4.0	0.24
0.49	6.0	2837.1	0.51	6.9	0.30
0.75	11.0	2941.4	0.76	13.1	0.30
0.91	22.0	3879.6	0.90	23.0	0.36

TABLE V
Summary of Simulation Results for 30 Graph Problems

	25% density		50% density		75% density	
Graph size	Ave. steps	Sol. quality	Ave. steps	Sol. quality	Ave. steps	Sol. quality
100	16.0	4.2/5	15.8	8.0/9	19.3	14.1/16
200	18.8	4.9/7	23.9	8.5/10	23.4	15.9/18
300	28.9	5.1/6	19.1	8.9/11	26.8	15.9/18
400	25.7	4.9/6	25.7	8.9/11	26.8	17.7/20
500	26.7	6.2/7	33.3	9.4/11	31.3	18.9/22
600	33.8	5.7/7	29.4	9.9/12	30.4	19.5/23
700	39.8	5.4/6	32.4	9.9/12	31.3	20.1/23
800	41.1	6.0/7	38.2	9.9/12	31.3	20.5/24
900	47.9	5.8/7	38.8	10.2/12	34.6	21.0/24
1000	49.9	5.8/7	45.9	10.4/12	34.4	21.4/25

TABLE VI
Comparison of the Computation Time (Seconds) for 50%-Edge-Density Graph Problems

Graph size	Carraghan's algorithm on IBM 3090	Our algorithm on Macintosh IIfx
100	0.14	0.52
200	4.16	2.7
300	46.04	4.8
400	235.68	11.2
500	1114.78	22.3
600	—	28.1
700	—	45.0
800	—	69.9
900	—	88.8
1000	—	129.8

superior to the best existing algorithms in terms of computation time with about the same solution quality. We also conclude that the proposed parallel algorithm finds a near-maximum clique of an n-vertex graph in nearly constant time on a parallel machine with n processors. The computation time depends neither on the graph size nor on the graph density on a parallel machine.

REFERENCES

1. Balas, E., and Yu, C. S. Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.* **15,** 4 (Nov. 1986), 1054–1068.
2. Carraghan, R., and Pardalos, P. M. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **9** (Nov. 1990), 375–382.
3. Funabiki, N., and Takefuji, Y. A parallel algorithm for spare allocation problems. *IEEE Trans. Reliability* **40,** 3 (Aug. 1991), 338–346.
4. Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.
5. Hopfield, J. J., and Tank, D. W. Neural computation of decisions in optimization problems. *Biol. Cybernet.* **52** (1985), 141–152.
6. Horaud, R., and Skordas, T. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern Anal. Mach. Intell.* **11,** 11 (Nov. 1989), 1168–1180.
7. Lecky, J. E., Murphy, O. J., and Absher, R. G. Graph theoretic algorithms for the PLA folding problem. *IEEE Trans. CAD/CAS* **8,** 9 (Sept. 1989), 1014–1021.
8. McCulloch, W. S., and Pitts, W. H. A logical calculus of ideas immanent in nervous activity. *Bull. Math. Biophys.* **5** (1943), 115–133.
9. Mead, C., and Conway, L. *Introduction to VLSI systems.* Addison–Wesley, Reading, MA, 1980.
10. Ogawa, H. Labeled point pattern matching by Delaunay triangulation and maximal cliques. *Pattern Recognition* **19,** 1 (1986), 35–40.
11. Pardalos, P. M., and Phillips, A. T. A global optimization approach for solving the maximum clique problem. *Int. J. Comput. Math.* **33** (1990), 209–216.
12. Takefuji, Y., and Lee, K. C. A near-optimum parallel planarization algorithm. *Science* **245** (Sept. 1989), 1221–1223.
13. Takefuji, Y., and Lee, K. C. A parallel algorithm for tiling problems. *IEEE Trans. Neural Networks* **1,** 1 (Mar. 1990), 143–145.
14. Takefuji, Y., and Lee, K. C. Artificial neural networks for four-coloring map problems and K-colorability problems. *IEEE Trans. Circuits and Systems* **38,** 3 (Mar. 1991), 326–333.

NOBUO FUNABIKI received his B.S. in mathematical engineering and information physics from the University of Tokyo in 1984 and his M.S. in electrical engineering from Case Western Reserve University in 1991. 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 Ind., Ltd., in Japan since 1984.

KUO-CHUN LEE received his B.S. from National Tsing-Hua University in Taiwan (1984), his M.S. from National Chiao-Tung University in Taiwan (1986), and his Ph.D. in electrical engineering from Case Western Reserve University (1991). Before joining Cirrus Logic, Inc., in 1991, he taught at Chinese Naval Academy Taiwan (1986–1988). He has published 20 journal papers and conference papers with Dr. Takefuji on neural networks. His current interests include placement and routing in CAD and static timing analysis.

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).

Received August 19, 1991; revised August 29, 1991; accepted September 12, 1991