INT. J. ELECTRONICS, 1992, VOL. 72, NO. 3, 357-372

A neural network parallel algorithm for clique vertex-partition problems

NOBUO FUNABIKI†, YOSHIYASU TAKEFUJI‡,
KUO CHUN LEE§ and YONG BEOM CHO‡

A parallel algorithm based on a neural network model for solving clique vertex-partition problems in arbitrary non-directed graphs is presented in this paper. A clique of a graph $G=(V,E)$ with a set of vertices V and a set of edges E is a complete subgraph of G where any pair of vertices is connected with an edge. A clique vertex-partition problem of a graph G is to partition every vertex in V into a set of disjointed cliques of G. The clique vertex-partition problem with the minimum number of cliques in an arbitrary graph is known to be NP-complete. The algorithm requires nm processing elements for the n vertex m partition problem. A total of 10 different problems with 8 vertex to 300 vertex graphs were examined where the algorithm found a solution in nearly constant time. The circuit diagram of the neural network model is also proposed in this paper.

1. Introduction

A clique of a graph $G=(V,E)$ with a set of vertices V and a set of edges E is a complete subgraph of G where any pair of vertices is connected with an edge in E. A clique vertex-partition problem of a graph G is to partition every vertex in V into a set of disjointed cliques of G. It is known that the problem of partitioning vertices with the minimum number of disjointed cliques in an arbitrary graph is NP-complete (Garey and Johnson 1979)

Figure 1(*a*) shows an 8-vertex 17-edge graph. It is known that the graph is partitioned with the minimum number of three cliques where the two different solutions are shown in Figs 1(*b*) and 1(*c*) respectively.

Tseng and Siewiorek (1983) proposed an *ad hoc* algorithm for the clique vertex-partition problem in an arbitrary graph. They showed three important applications of the clique vertex-partition problem in digital systems: the storage element allocation problem, the data operator allocation problem, and the interconnection unit allocation problem. Unfortunately they neither discussed the time complexity nor the solution quality of their algorithm.

Gregory *et al.* (1986) discussed the lower and upper bounds of the minimum number of cliques for the clique edge-partition problem in a cocktail party graph. In the clique edge-partition problem in a graph $G=(V,E)$, every edge in E is partitioned into a set of disjointed cliques of G. Erdös *et al.* (1988) discussed the lower and upper bounds of the minimum number of cliques for the clique edge-partition problem in an arbitrary graph. McGuinness and Rees (1990) discussed the minimum number of cliques for the clique edge-partition problem in a line graph.

Received 3 June 1991; accepted 20 August 1991.

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

‡Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, U.S.A.

§R&D Department, Cirrus Logic Inc., Fremont, California, U.S.A.

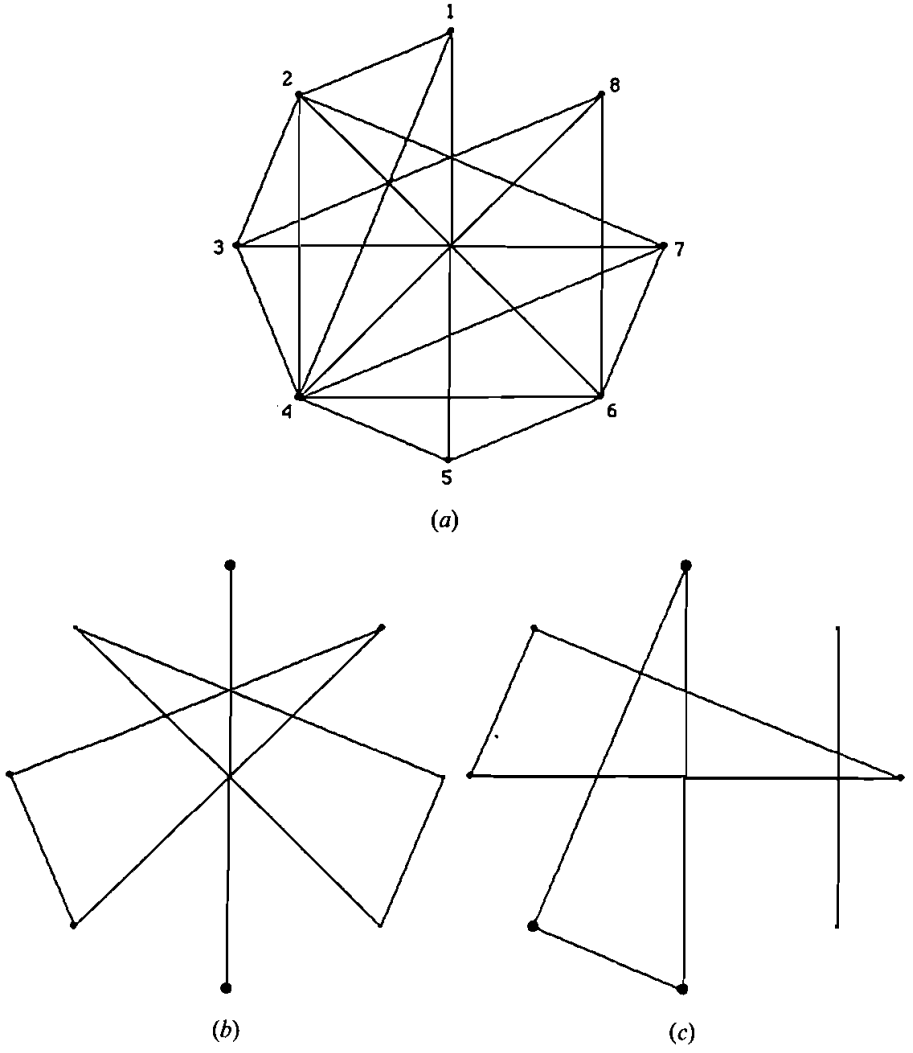


Figure 1. An 8-vertex 17-edge graph and the minimum clique vertex partitions: (*a*) an 8-vertex 17-edge graph; (*b*) minimum clique vertex-partition number 1; (*c*) minimum clique vertex-partition number 2.

Grötschel and Wakabayashi (1990) discussed the minimum weight clique vertex-partition problem in a complete graph where each edge in the group has a weight. Few algorithms have been reported for the clique vertex-partition problem in an arbitrary graph within our knowledge.

This paper proposes the first parallel algorithm for solving a clique vertex-partition problem in an arbitrary graph where it is based on the artificial neural network model. The algorithm requires nm processing elements for an n-vertex m-partition problem where n vertices of a graph are partitioned into m disjointed cliques. Due to neural network computing, the algorithm not only runs on a sequential machine but also on a parallel machine with maximally nm processors without a rigorous synchronization procedure.

The processing element is called a neuron in the neural network model because it performs the function of a simplified biological neuron model. The processing element has an input and an output. Among several proposed neuron models, the simplext McCulloch-Pitts (1943) binary neuron model is adopted in this paper:

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

where U_i and V_i are the input and the output of the ith processing element respectively. The change of the input U_i is given by the partial derivatives of a computational energy function $E(V_1, \ldots, V_n)$ which is called a motion equation:

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

Note that n is the number of processing elements required in the problem. The energy function E represents the distance between the current state of the neural network system and the solution state. The energy function is given by considering the necessary and sufficient constraints in the problem. The goal of the neural network model for solving optimization problems is to minimize the energy function. A theorem in the Appendix states that the motion equation forces the state of the neural network system composed of the McCulloch-Pitts neurons to converge to a local minimum (Takefuji and Lee 1991).

A neural network model for solving an optimization problem was first introduced by Hopfield and Tank (1985, 1986). Wilson and Pawley (1988), and Paielli (1988) criticized the stability and the solution quality of the Hopfield neural network model. Takefuji and Lee (1991) pointed out that the decay term in the Hopfield neural network model is harmful for the system convergence. They have also used the McCulloch-Pitts neuron model instead of the sigmoid neuron model in the Hopfield neural network model in order to improve the convergence speed. More than 15 successful neural network applications for solving NP-complete and optimization problems have been reported in the last two years (Takefuji and Lee, 1989, 1990a, b, 1991a, b, Kurokawa *et al.* 1990, Takefuji *et al.* 1990a, b, Funabiki and Takefuji 1991a, b, c, d, e, f, Takefuji 1991, 1992, Takefuji *et al.* 1991).

2. Neural network representation

A two-dimensional neural network model is used for solving a clique vertex-partition problem in this paper. Figure 2 shows the neural network representation for solving the 8-vertex 17-edge graph problem in Fig. 1 where 8 vertices be partitioned into 3 cliques. Because each vertex has three candidates (cliques), three processing elements are required for each vertex. As shown in Fig. 2(*a*) the total of 24 ($=8 \times 3$) processing elements is required for the 8-vertex 3-partition problem. Generally the total of nm processing elements is required for solving the n-vertex m-partition problem. The output of the ijth processing element represents whether the ith vertex be partitioned into the jth clique or not. When the output of the ijth processing element is non-zero ($V_{ij}=1$), the ith vertex is partitioned into the jth clique. When the output of the ijth processing element is zero ($V_{ij}=0$), the ith vertex is not partitioned into the jth clique. Figure 2(*b*) shows the solution state of the 24 processing elements where the black square and the white square indicate the non-

zero output and the zero output respectively. In Fig. 2(*b*) the second, sixth and seventh vertices are partitioned into the first clique, the third, fourth and eighth vertices into the second clique, and the first and fifth vertices into the third clique. The solution state of the neural network system corresponds to Fig. 1(*b*).

One and only one processing element among m candidates for the ith vertex must have non-zero output. This constraint is given by:

$$\sum_{k=1}^{m} V_{ik} - 1 \tag{3}$$

It is zero if and only if one processing element among m candidates has non-zero output.

Any pair of vertices in each clique must be connected with an edge which is given in the original graph. In other words, the ith vertex must not be partitioned into the jth clique if the ith vertex has no edge with a vertex which has been already partitioned into the jth clique. This constraint is given by:

$$\sum_{\substack{k=1 \\ k \neq i}}^{n} (1 - d_{ik}) V_{kj} \tag{4}$$

where d_{ik} is 1 if the original graph has an edge between the ith vertex and the kth vertex, 0 otherwise, and it is always satisfied that $d_{ik} = d_{ki}$ (non-directed graph) and $d_{ii} = 0$. It is non-zero if a vertex in the jth clique has no edge with the ith vertex.

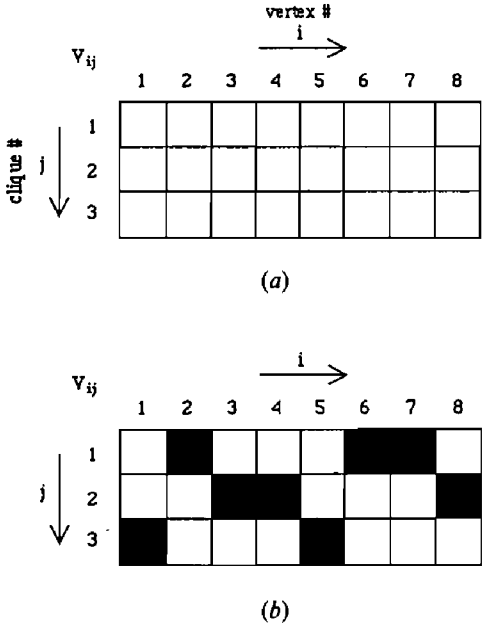


Figure 2. Neural network representation for the clique vertex partition problem in Fig. 1: (*a*) 8×3 processing elements for the problem in Fig. 1; (*b*) convergence of the processing elements to a solution.

The motion equation for the ijth processing element in the n-vertex m-partition problem is given from (3) and (4):

$$\frac{dU_{ij}}{dt} = -A\left(\sum_{k=1}^{m} V_{ik} - 1\right) - B\left(\sum_{\substack{k=1 \\ k\neq i}}^{n} (1-D_{ik})V_{kj}\right) \tag{5}$$

The A term forces one and only one processing element among m candidates for the ith vertex to have non-zero output where the ith vertex is partitioned. The B term performs the inhibitory force if a vertex in the jth clique has no edge with the ith vertex. A and B are constant coefficients. The energy function E for this problem is given by considering (2) and (5):

$$E = \frac{A}{2}\left(\sum_{k=1}^{m} V_{ik} - 1\right)^2 + B\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}\right)V_{ij} \tag{6}$$

As shown in the Appendix only the local minimum convergence is guaranteed in the neural network model although we must consider the global minimum convergence. In order to increase the frequency of the global minimum convergence, the following three heuristics have been empirically introduced:

(1) The hill climbing heuristic: the following two terms are added to the motion equation in (5):

$$+Ch\left(\sum_{k=1}^{m} V_{ik}\right) + Dh\left(\sum_{k=1}^{m} V_{ik} + \sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}\right) \tag{7}$$

where $h(x)$ is 1 if $x=0$, 0 otherwise. C and D are constant coefficients. The C term encourages the ijth processing element to have non-zero output if no processing elements for the ith vertex have non-zero output. The D term encourages the ijth processing element to have non-zero output if no processing elements for the ith vertex have non-zero output and the ith vertex can be partitioned into the jth clique without violations.

(2) The omega function heuristic: two forms of the B term are used periodically in the motion equation:

$$-B\left(\sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}\right)V_{ij} \quad \text{if} \quad (t \bmod T) < \omega$$

$$-B\left(\sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}\right) \qquad \text{otherwise} \tag{8}$$

where t is the number of iteration steps, and T and ω are constant parameters.

(3) The input saturation heuristic: the input of the processing element is confined between two values:

$$U_{ij} = U\_\max \quad \text{if } U_{ij} > U\_\max$$

$$U_{ij} = U\_\min \quad \text{if } U_{ij} < U\_\min \tag{9}$$

where $U\_\max$ and $U\_\min$ are the constant upper and lower bounds of the input value U_{ij} respectively.

3. Parallel algorithm

The following procedure describes the parallel algorithm for the n-vertex m-partition problem based on the motion equation with three heuristics. The first-order Euler method is used to solve the partial differential equations. The data set of coefficients and parameters are empirically determined.

(0) Set $t=0$, $A=B=1$, $C=2$, $D=4$, $T=10$, $\omega=7$, $U\_\max=20$, $U\_\min=-20$, and $T\_\max=500$.

(1) The initial values of the input $U_{ij}(t)$ for $i=1,\ldots,n$ and $j=1,\ldots,m$ are uniformly randomized between 0 and $U\_\min$. The initial values of the output $V_{ij}(t)$ for $i=1,\ldots,n$ and $j=1,\ldots,m$ are assigned to 0.

(2) Compute the change of the input $\Delta U_{ij}(t)$ based on the motion equation in (5) with the hill climbing heuristic in (7) and the omega function heuristic in (8):

if $(t \bmod T) < \omega$

$$\Delta U_{ij}(t) = -A\left(\sum_{k=1}^{m} V_{ik}(t) - 1\right) - B\left(\sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}(t)\right) V_{ij}(t)$$

$$+ Ch\left(\sum_{k=1}^{m} V_{ik}(t)\right) + Dh\left(\sum_{k=1}^{m} V_{ik}(t) + \sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}(t)\right) \tag{10}$$

or otherwise

$$\Delta U_{ij}(t) = -A\left(\sum_{k=1}^{m} V_{ik}(t) - 1\right) - B\left(\sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}(t)\right)$$

$$+ Ch\left(\sum_{k=1}^{m} V_{ik}(t)\right) + Dh\left(\sum_{k=1}^{m} V_{ik}(t) + \sum_{\substack{k=1 \\ k\neq i}}^{n} (1-d_{ik})V_{kj}(t)\right) \tag{11}$$

(3) Update the input $U_{ij}(t+1)$ based on the first-order Euler method:

$$U_{ij}(t+1) = U_{ij}(t) + \Delta U_{ij}(t) \tag{12}$$

(4) Use the input saturation heuristic in (9):

$$U_{ij}(t+1)=U\_\max \quad \text{if} \quad U_{ij}(t+1)>U\_\max$$

$$U_{ij}(t+1)=U\_\min \quad \text{if} \quad U_{ij}(t+1)<U\_\min \tag{13}$$

(5) Update the output $V_{ij}(t+1)$ based on the McCulloch-Pitts neuron model:

$$V_{ij}(t+1)=1 \quad \text{if} \quad U_{ij}(t+1)>0$$

$$0 \quad \text{otherwise} \tag{14}$$

(6) Check the termination condition if $(V_{ij}(t)=1)$ and

$$\sum_{\substack{k=1 \\ k\neq i}}^{n}(1-d_{ik})V_{kj}(t)=0 \quad \text{for} \quad i=1,\ldots,n \text{ and } \exists_{j\in}\{1,\ldots,m\}$$

or $t=T\_\max$, then terminate this procedure else increment t by 1 and go to step 2.

The state of nm processing elements for the n-vertex m-partition problem can be updated synchronously or asynchronously. In this paper the synchronous parallel system is simulated on a sequential machine. The synchronous parallel system can be performed on maximally nm processors. The following procedure/programme outlines how to simulate the synchronous parallel system using a sequential machine as if the programme runs on the parallel machine:

```
Program parallel-simulator-on-a-sequential-machine
. . . . .
initialization of U_ij and V_ij for i:=1 to n and for j:=1 to m;
. . . . .
{***Main Programme***}
while (a set of conflicts is not empty) to
begin
. . . . .
{***Updating all input values***}
   for i:=1 to n do
   for j:=1 to m do
      U_ij:=U_ij+ΔU_ij;
{***End of the first loop***}
. . . . .
{***Updating all output values***}
   for i:=1 to n do
   for j:=1 to m do
      If U_ij>0 then V_ij:=1 or otherwise V_ij:=0;
{***End of the second loop***}
. . . . .
end;
{***Main Programme end***}
```

In the first loop all input values U_{ij} are sequentially updated while all output values V_{ij} are fixed. Then in the second loop all output values V_{ij} are sequentially updated while all input values U_{ij} are fixed. It is equivalent to simultaneously updating the values of all inputs and outputs.

4. Simulation results and discussion

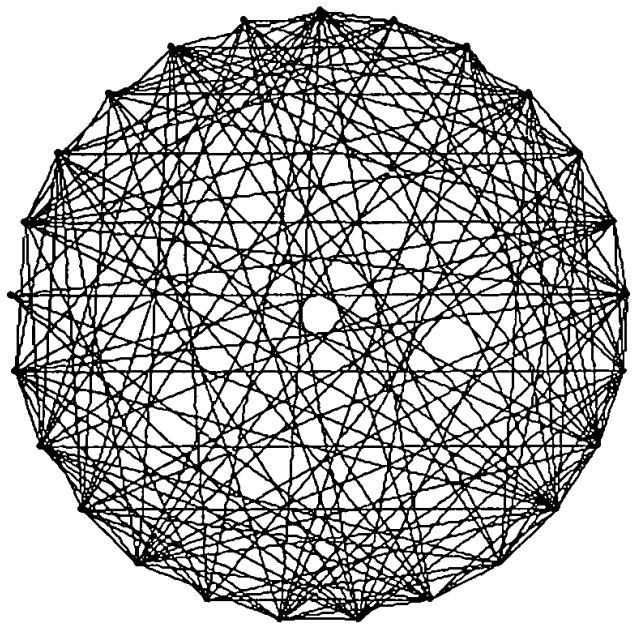
In order to verify the proposed algorithm the simulator has been developed on Macintosh SE/30 and IIfx. The programme was coded in Turbo Pascal. Ten clique vertex-partition problems in the table were examined where the number of vertices in the graphs was varied from eight to 300. Except the 8-vertex 17-edge graph problem, edges in the graphs were randomly generated. Figures 3 and 4 show the original graphs and solutions in the 25 vertex 162 edge graph problem and in the 50-vertex 631-edge graph problem respectively. In Fig. 3(*b*) 25 vertices are partititioned into six cliques which are indicated by *A*, *B*, ..., *F*. For each one of ten problems 100 simulation runs were performed from different initial values of $U_{ij}(t)$. The table also summarizes the simulation results where the average number of iteration steps to converge to the solution and the convergence frequency are shown. Figure 5 shows the relationship between the number of iteration steps to converge to the solution and the frequency in two problems. The simulation results show that the state of the neural network converged to the solution in nearly constant number of iteration steps. We conclude that with nm processors the proposed algorithm finds a solution for an n-vertex m-partition problem in nearly constant time.

5. Circuit design of the neural network

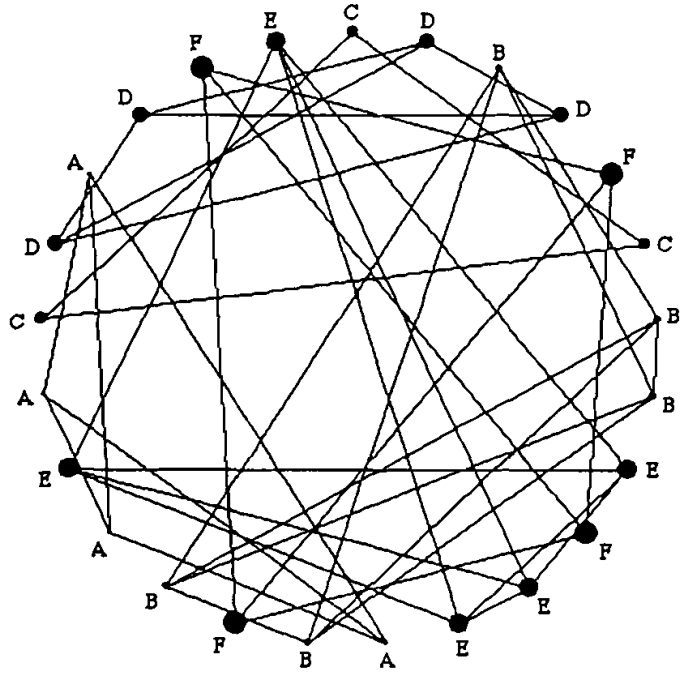
Figure 6(*a*) shows a clique vertex-partition problem with a 4-vertex 4-edge graph. The total of eight processing elements is required in this problem because four vertices be partitioned into two-cliques as shown in Fig. 6(*b*). Figure 7 outlines the analogue circuit diagram of the neural network model for solving the 4-vertex 2-partition problem. Figure 8(*a*) depicts the analogue circuit diagram of the 11th processing element in Fig. 7. The output of the first operational amplifier is equivalent to the right side of the motion equation with the hill climbing heuristics and the omega function heuristic:

Problem number	Number of vertices n	Number of edges	Number of cliques m	Average iteration steps to solutions	Convergence frequency to solutions
1	8	17	3	28·9	90%
2	25	162	6	189·0	9%
3	50	306	17	178·2	20%
4	50	631	10	270·3	10%
5	50	939	6	179·6	16%
6	100	2475	18	238·8	21%
7	150	5618	26	215·7	51%
8	200	9968	32	225·8	32%
9	250	15577	39	213·9	54%
10	300	22484	46	178·2	86%

Specifications of simulated problems and simulation results.

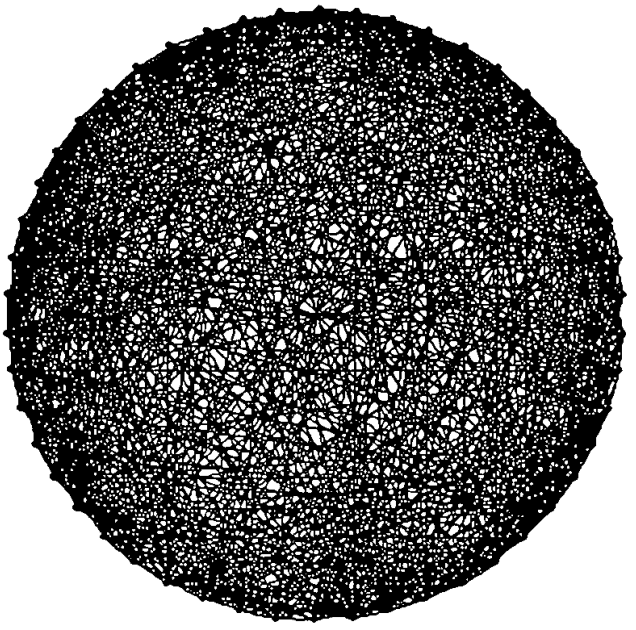


(*a*)

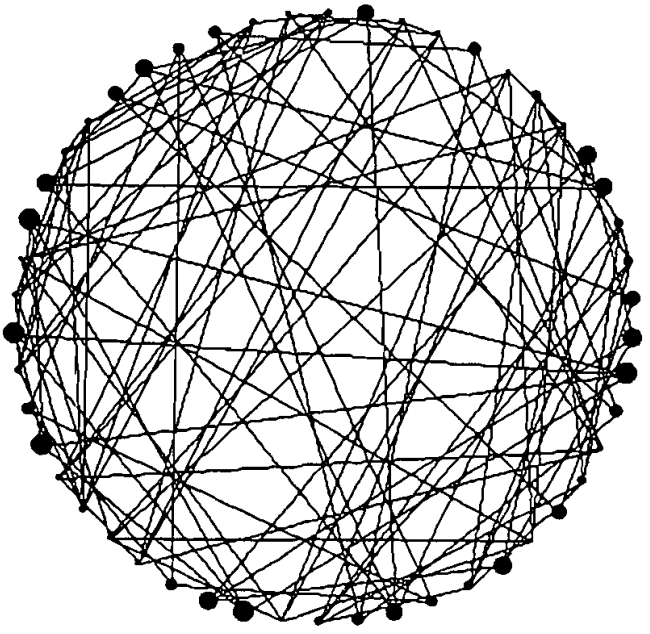


(*b*)

Figure 3. A 25-vertex 162-edge graph problem and the 6-clique solution: (*a*) 25-vertex 162-edge graph; (*b*) 6-clique solution.

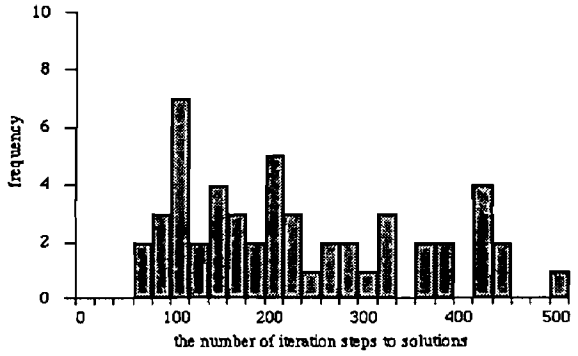


(*a*)

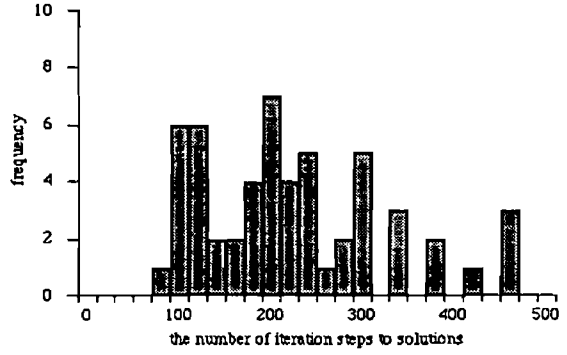


(*b*)

Figure 4. A 50-vertex 631-edge graph problem and the 10-clique solution: (*a*) 50-vertex-631 edge graph; (*b*) 10-clique solution.



(*a*)



(*b*)

Figure 5. Relationship between the number of iteration steps to converge to solutions and the frequency: (*a*) 150-vertex 5618-edge graph problem; (*b*) 250-vertex 15577-edge graph problem.

$$\frac{dU_{11}}{dt} = -R\left(\frac{V_{11}}{R_a} + \frac{V_{12}}{R_b} + \frac{V_{31}}{R_c} - \frac{V_x}{R_x} - \frac{V_y}{R_y} - \frac{V_z}{R_z}\right) \tag{15}$$

The third term V_{31}/R_c in (15) follows the omega function heuristic where the circuit diagram is described in Fig. 8(*b*). The clock input to the OR gate in Fig. 8(*b*) corresponds to the modulo function in (8). The fifth term V_y/R_y and the sixth term V_z/R_z in (15) follow the hill climbing heuristics where the circuit diagram are described in Fig. 8(*c*) and 8(*d*) respectively. The second operational amplifier in Fig. 8(*a*) integrates dU_{11}/dt into $-U_{11}$. The third operational amplifier reverses the sign into U_{11}. The last component outputs the binary voltage V_{11} according to U_{11} where the function $f(x)$ represents the binary function. The details of the circuit diagrams are discussed in Takefuchi and Lee (1991 a).

We simulated the proposed neural network circuit in order to verify the performance. Figure 9(*a*) and (*b*) show the transition of inputs U and outputs V from the initial state of the system to the solution state in the 4-vertex 2-partition problem. The solution state in this simulation corresponds to Fig. 6(*b*). Note that the time step is 0·05 s and the total number of iteration steps is 50. Figure 9(*a*) depicts that the hill climbing heuristics sometimes drastically increase the inputs U in order to escape from a local minimum, and that two inputs U_{11} and U_{12} are competing

with each other for the state of the system to converge to the feasible solution. The similar situations are also observed in Fig. 9(*b*).

6. Conclusion

This paper proposes the first parallel algorithm for clique vertex-partition problems in arbitrary graphs. The algorithm is based on the neural network model where it requires nm processing elements for the n-vertex m-partitition problem. A total of ten different problems was examined where the algorithm found a solution in nearly constant time with nm processors. The algorithm achieves the goal of the parallel computation in terms of the computation time and the solution quality. The circuit diagram of the neural network model is also proposed in this paper.

Appendix

Theorem

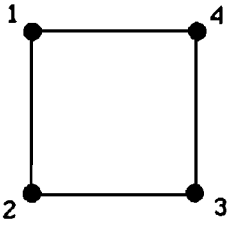
The system always satisfies $\Delta E/\Delta t \leqslant 0$ under two conditions such as $\Delta U_i/\Delta t = -\Delta E/\Delta V_i$ and $V_i = f(U_i)$ where E is the computational Liapunov energy function and $f(U_i)$ is the McCulloch-Pitts binary function:

$$f(U_i) = 1 \text{ if } U_i > 0$$

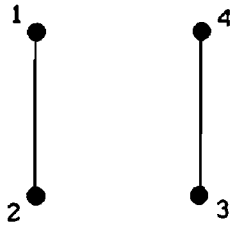
$$= 0 \text{ otherwise}$$

Proof

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



(*a*)



(*b*)

Figure 6. A 4-vertex 4-edge graph problem and the 2-clique solution: (*a*) 4-vertex 4-edge graph problem; (*b*) 2-clique solution.

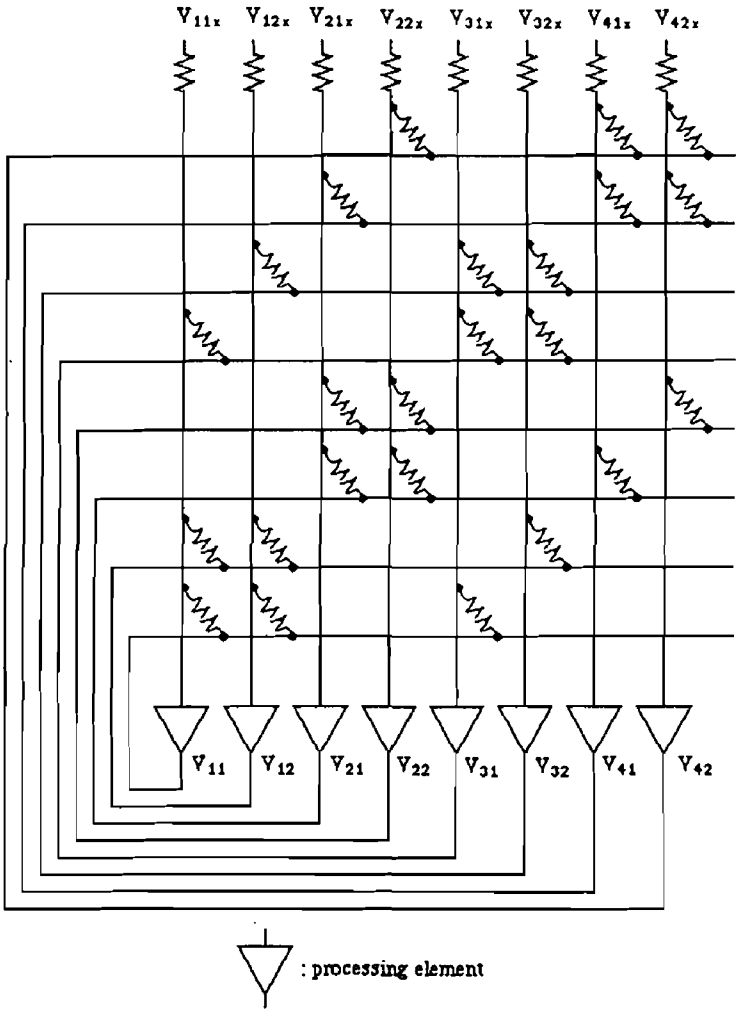


Figure 7. Circuit diagram of the neural network for the 4-vertex 4-edge graph problem in Fig. 6.

$$\frac{\Delta E}{\Delta t}=\sum_i \frac{\Delta V_i}{\Delta t}\frac{\Delta E}{\Delta V_i}=\sum_i \frac{\Delta V_i}{\Delta t}\left(-\frac{\Delta U_i}{\Delta t}\right) \text{ where } \frac{\Delta E}{\Delta V_i} \text{ is replaced by } \left(-\frac{\Delta U_i}{\Delta t}\right)$$

$$=-\sum_i \left(\frac{\Delta U_i}{\Delta t}\frac{\Delta V_i}{\Delta U_i}\right)\left(\frac{\Delta U_i}{\Delta t}\right)=-\sum_i \left(\frac{\Delta V_i}{\Delta U_i}\right)\left(\frac{\Delta U_i}{\Delta t}\right)^2$$

Let $\Delta U_i/\Delta t$ be $\{U_i(t+\Delta t)-U_i(t)\}/\Delta t$. Let $\Delta V_i/\Delta U_i$ be $\{V_i(t+\Delta t)-V_i(t)\}/\{U_i(t+\Delta t)-U_i(t)\}$.

It is necessary and sufficient to consider the following two regions:

Region 1: $U_i(t)>0$ and $V_i(t)=1$

Region 2: $U_i(t)\leqslant 0$ and $V_i(t)=0$

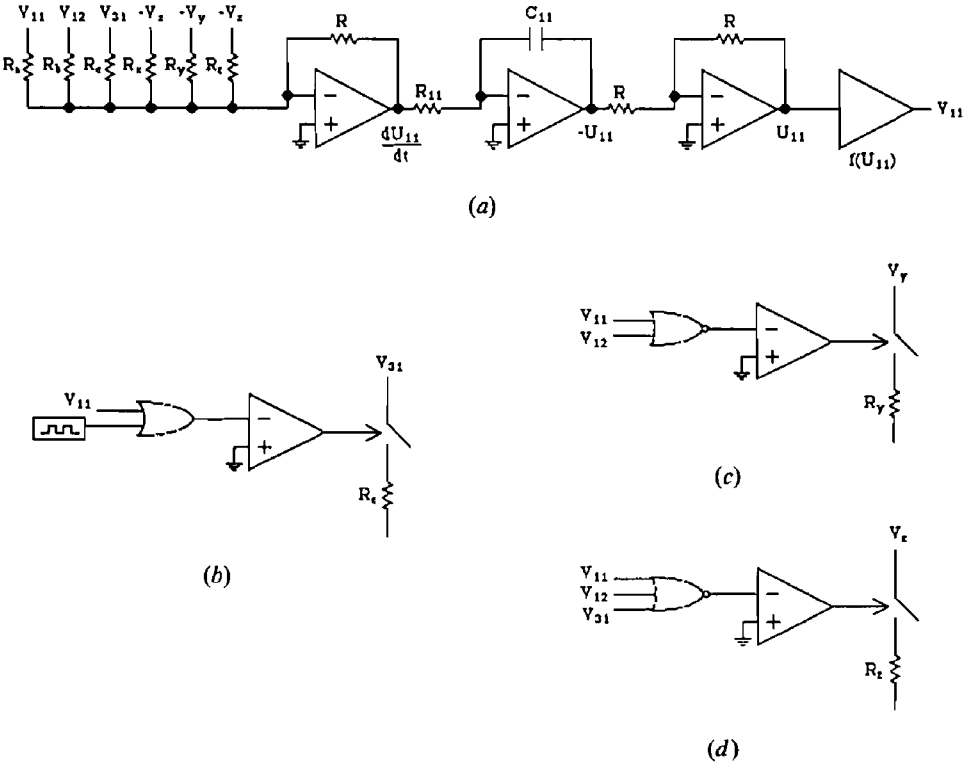


Figure 8. Circuit diagram of the 11th processing element in Fig. 7: (*a*) analogue circuit diagram of the 11th processing element; (*b*) circuit for the omega function heuristic (*B*-term); (*c*) circuit for the hill climbing heuristic (*C*-term); (*d*) circuit for the hill climbing heuristic (*D*-term).

In Region 1 we must consider four possible cases for $U_i(t+\Delta t)$:

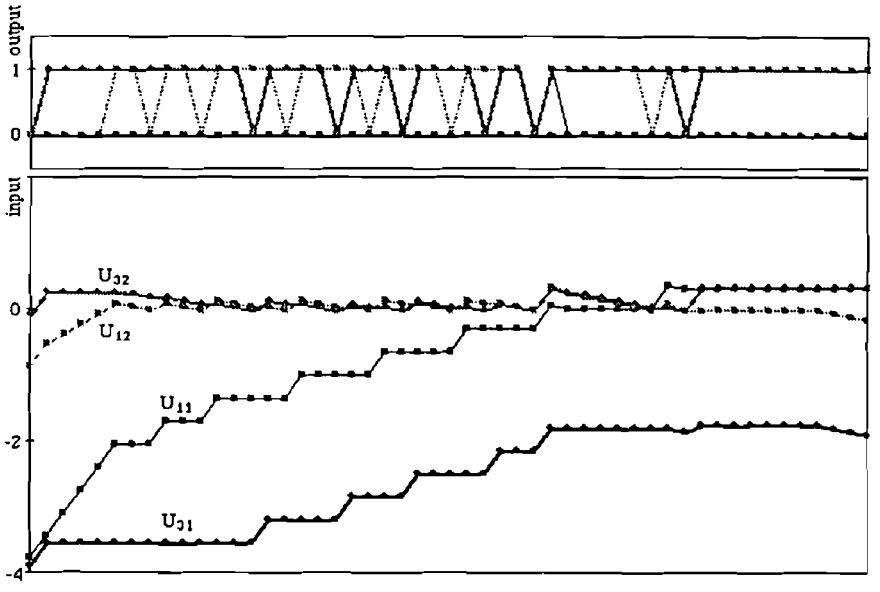
(*a*) $U_i(t+\Delta t) > U_i(t)$

(*b*) $0 < U_i(t+\Delta t) < U_i(t)$

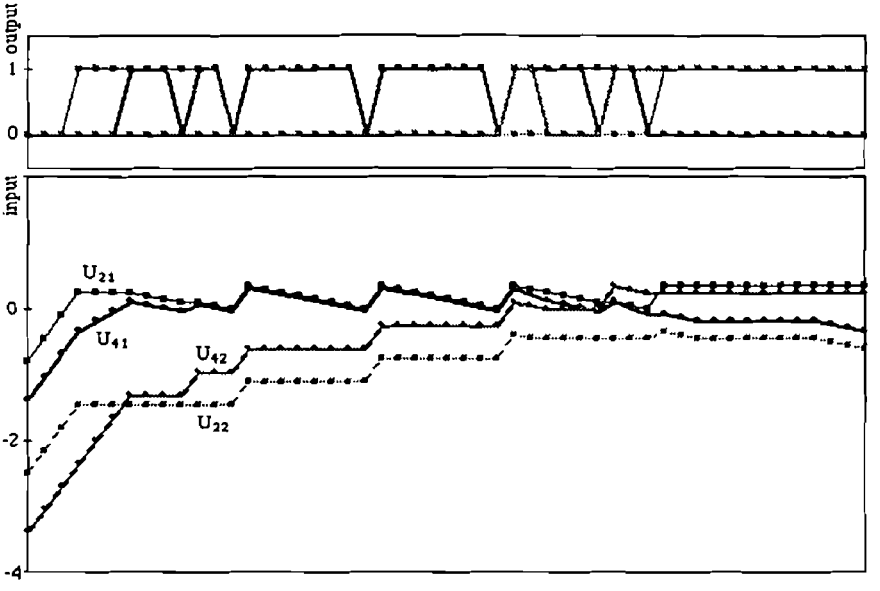
(*c*) $U_i(t+\Delta t) \leqslant 0 < U_i(t)$

(*d*) $U_i(t+\Delta t) = U_i(t)$

In (*a*) and (*b*), $V_i(t+\Delta t) = V_i(t) = 1 \Rightarrow \Delta V_i/\Delta U_i = 0$. Therefore $\Delta E/\Delta t = 0$.
In (*d*), $\Delta U_i/\Delta t = 0 \Rightarrow \Delta E/\Delta t = 0$.
In (*c*), $V_i(t+\Delta t) = 0 \Rightarrow \Delta V_i/\Delta U_i = (0-1)/(\text{negative number}) > 0$ and $\Delta U_i/\Delta t < 0$. Therefore $\Delta E/\Delta t < 0$.
It is concluded that $\Delta E/\Delta t \leqslant 0$ is always satisfied in Region 1.
Similarly, in Region 2, $\Delta E/\Delta t \leqslant 0$ is always satisfied. This completes the proof. □



(*a*)



(*b*)

Figure 9. Simulation result of the neural network circuit; (*a*) transition of the states of four processing elements; (*b*) transition of the states of four processing elements.

References

ERDÖS, P., FAUDREE, R., and ORDMAN, E. T., 1988, Clique partitions and clique coverings. *Discrete Mathematics*, **72**, 93-101.

FUNABIKI, N., and TAKEFUJI, Y., 1991 a, A parallel algorithm for spare allocation problems. *I.E.E.E. Transactions on Reliability*, **40**(3), 338-346; 1991 b, A parallel algorithm for channel routing problems. *I.E.E.E. Transactions on CAD/CAS*, to be published; 1991 c, A parallel algorithm for solving Hip games. *Neurocomputing*, **3**, 97-106; 1991 d, A parallel algorithm for traffic control problems in TDM hierarchical switching systems. *I.E.E.E. Transactions on Communications*, to be published; 1991 e, A parallel algorithm for traffic control problems in three-stage connecting networks. *Journal of Parallel and Distributed Computing*, to be published; 1991 f, A parallel algorithm for broadcast scheduling problems in packet radio networks. *I.E.E.E. Transactions on Communications*, to be published.

GAREY, M. R., and JOHNSON, D. S., 1979, *Computers and Intractability: a Guide to the Theory of NP-Completeness* (New York: W. H. Freeman).

GREGORY, D. A., MCGUINNESS, S., and WALLIS, W., 1986, Clique partitions of the cocktail party graph. *Discrete Mathematics*, **59**, 267-273.

GRÖTSCHEL, M., and WAKABAYASHI, Y., 1990, Facets of the clique partitioning polytope. *Mathematical Programming*, **47**, 367-387.

HOPFIELD, J. J., and TANK, D. W., 1985, Neural computation of decisions in optimization problems. *Biological Cybernetics*, **52**, 141-152; 1986, Computing with neural circuits: a model. *Science*, **233**, 625-633.

KUROKAWA, T., LEE, K. C., CHO, Y. B., and TAKEFUJI, Y., 1990, CMOS layout design of the hysteresis McCulloch-Pitts neuron. *Electronics Letters*, **26**, 2093-2095.

MCCULLOCH, W. S., and PITTS, W. H., 1943, A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115-133.

MCGUINNESS, S., and REES, R., 1990, On the number of distinct minimal clique partitions and clique covers of a line graph. *Discrete Mathematics*, **83**, 49-62.

PAIELLI, R. A., 1988, Simulation tests of the optimization method of Hopfield and Tank using neural networks. *NASA Technical Memorandum* 101047.

TAKEFUJI, Y., 1991, Neural network parallel algorithms. *Journal of Intelligent Manufacturing*, to be published; 1992, *Neural Network Parallel Computing* (Norwell, Mass: Kluwer Academic).

TAKEFUJI, Y., and LEE, K. C., 1989, A near-optimum parallell planarization algorithm. *Science*, **245**, 1221-1223; 1990 a, A parallel algorithm for tiling problems. *I.E.E.E. Transactions on Neural Networks*, **1**, 143-145; 1990 b, A super parallel sorting algorithm based on neural networks. *I.E.E.E. Transactions on Circuits and Systems*, **37**, 1425-1429; 1991 a, Artificial neural networks for four-coloring map problems and K-colorability problems. *I.E.E.E. Transactions on Circuits and Systems*, **38**(3), 326-333; 1991 b, An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics. *Biological Cybernetics*, **64**, 353-356.

TAKEFUJI, Y., LIN, C. W., and LEE, K. C., 1990 a, A parallel algorithm for estimating the secondary structure in Ribonucleic Acids. *Biological Cybernetics*, **63**, 337-340.

TAKEFUJI, Y., CHEN, L. L., LEE, K. C., and HUFFMAN, J., 1990 b, Parallel algorithms for finding a near-maximum independent set of a circle graph. *I.E.E.E. Transactions on Neural Networks*, **1**, 263-267.

TAKEFUJI, Y., TANAKA, T., and LEE, K. C., 1991, A parallel string search algorithm. *I.E.E.E. Transactions on Systems, Man and Cybernetics*, to be published.

TSENG, C. J., and SIEWIOREK, D. P., 1983, Facet: a procedure for the automated synthesis of digital systems. *Proceedings of the 20th ACM/I.E.E.E. Design Automation Conference*, pp. 490-496.

WILSON, G. V., and PAWLEY, G. S., 1988, On the stability of the traveling salesman problem algorithm of Hopfield and Tank. *Biological Cybernetics*, **58**, 63-70.