Paper

A Parallel Graph Planarization Algorithm Using Gradient Ascent Learning of Hopfield Network

Rong Long Wang* Student Member
Zheng Tang* Non-member
Qi Ping Cao** Non-member

This paper proposes a gradient ascent learning algorithm of the Hopfield neural networks for graph planarization. This learning algorithm which is designed to embed a graph on a plane, uses the Hopfield neural network to get a near-maximal planar subgraph, and increase the energy by modifying weights in a gradient ascent direction to help the network escape from the state of the near-maximal planar subgraph to the state of the maximal planar subgraph or better one. The proposed algorithm is applied to several benchmark graphs up to 150 vertices and 1064 edges. The performance of the proposed algorithm is compared with that of Takefuji/Lee's method. Simulation results show that the proposed algorithm is much better than Takefuji/Lee's method in terms of the solution quality for every tested graphs.

Keywords: graph planarization, Hopfield neural network, gradient ascent learning, NP-complete problem

1. Introduction

Given a graph $G = (V, E)$, the graph planarization is to find a largest subset F of E, such that $H = (V, F)$ is planar. The graph planarization problem is an important problem in automatic graph drawing, in designing printed circuit boards and routing very-large-scale integration (VLSI) circuits. Several graph planarization heuristics have been presented. In 1989 Jayakumar et al. proposed a $O(n^2)$ near-maximal planarity testing algorithm [1]. Kant[2] presented a corrected and more generalized version of Jayakumar's algorithm. Goldschmidt and Takvorian presented a two-phase graph planarization heuristic [3]. Based on two different ideas, Cai et al.[4] and Di Battista and Tamassia[5] described two $O(m \log n)$ algorithm for this problem. However the graph planarization problem is a well known NP-complete for general graphs [6]. No tractable algorithm is known for solving it. Furthermore, few parallel algorithms have been proposed to solve the planarization problem. A possible parallel algorithm for solving such optimization problems was introduced by Hopfield and Tank [7], which found a good solution to some optimization problems in a reasonable amount of time. Using the neural network techniques, Takefuji and Lee presented a parallel planarization algorithm for generating a near-maximal planar subgraph within $O(1)$ time [8][9]. Unfortunately, due to its inherent local minimum problems and sensitivity to parameter values [3][8], the rate to get the maximal planar subgraph is very low, and performance of the algorithm becomes poorer with large problem. This may be improved by some more sophisticated architecture, such as Boltzmann machine [10]. However, the Boltzmann machine is very slow because of the need for extensive averaging over stochastic variables [11].

In this paper we present a parallel planarization algorithm for graph planarization based on gradient ascent learning for Hopfield networks [12]. The learning algorithm has two phases, the Hopfield network updating phase and the gradient ascent learning phase. The first phase uses the Hopfield network to decrease the energy in state domain and find a near-maximal planar subgraph. The second phase intentionally increases the energy of the Hopfield network by modifying parameters in weight domain in a gradient ascent direction, thus making the network escape from the near-maximal planar subgraph (i.e., a local minimum) which the network once falls into in the phase one. The learning algorithm is applied to 19 graphs with up to 150 vertices and 1064 edges. Simulation results are compared with the ones found by Takefuji/Lee's algorithm [3][8]. It is found that the proposed algorithm works well on generating a maximal or a better planar subgraph than Takefuji/Lee's algorithm [8][9] for solving the graph planarization problem in the same vertex ordering within $O(1)$ time.

2. Hopfield Neural Network Updating Phase for Graph Planarization

A graph is said to be planar, or embeddable in the plane, if it can be drawn on a plane, such that no two edges crossing each other except for their end vertices. Given a nonplanar graph, the graph planarization problem is to find a spanning planar subgraph with a maximum number of edges. The problem is known to be NP-complete, and it is generally believed the computational power needed to solve it grows exponentially with the number of edges. Consider a simple undirected

* Faculty of Engineering, Toyama University, Toyama-shi, Japan 930-8555
** Tateyama Systems Institute, Toyama-shi, Japan 930-0001

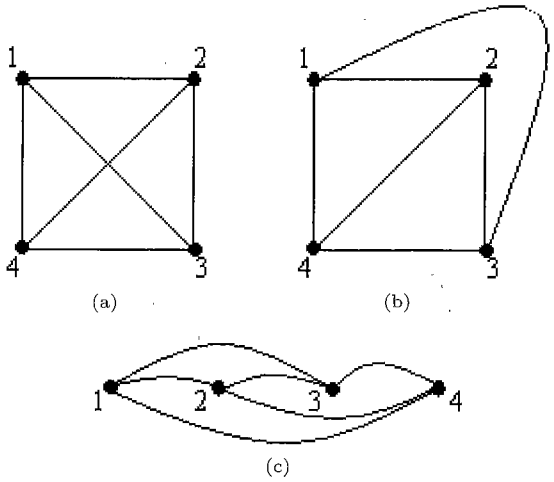


Fig. 1. (a) A graph with four vertices and six edges. (b) A planar graph (c) Possible planar graph based on the single-row routing representation.

graph composed of four vertices and six edges as shown in Fig.1(a). The graph is planar as long as two edges (1,3) and (2,4) do not cross each other. Figure 1(b) shows a planar graph. In the single-row routing representation used here, connection is established by either an upper edge or a lower edge. Two neurons (y_{ij} and y_{ji}, $j > i$) express the upper and lower line connection between the $i-th$ and $j-th$ vertices, respectively. The following states (y_{ij}=y_{ji}=0, $j > i$), (y_{ij}=0, y_{ji}=1, $j > i$) and ($y_{ij} = y_{ji} = 1$, $j > i$) express no connection, lower line connection, and double-line connection violation, respectively. Figure 1(c) shows a possible planar graph based on the single-row routing representation. The number of edges in a given graph determines the number of neurons required. Actually the system requires 2M neurons, where M is the number of edges in a given graph. For example the graph shown in Fig.1(c) can be represented by a 2×6 two-dimensional neural network array ($y_{12} = 1$, $y_{23} = 1$, $y_{34} = 1$, $y_{13} = 1$, $y_{42} = 1$, $y_{41} = 1$ and $y_{21} = 0$, $y_{32} = 0$, $y_{43} = 0$, $y_{31} = 0$, $y_{24} = 0$, $y_{14} = 0$). In general, the planarization problem for a graph with N vertices and M edges can be solved by an $N \times N$ Hopfield neural network. The double-line connection violation condition can be expressed by follow:

$$\sum_i \sum_{j \neq i} y_{ij} y_{ji} = 0 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (1)$$

The two-edge-crossing violation condition is:

$$\sum_{i,j} \sum_{kl} d_{ijkl} y_{ij} y_{kl} = 0 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (2)$$

where d_{ijkl} is 1 if $i < k < j < l$, $i > k > j > l$, $k < i < l < j$ or $k > i > l > j$, 0 otherwise.

Then the energy function for the graph planarization problem is given by:

$$e = \frac{1}{2} A \sum_i \sum_{j \neq i} y_{ij} y_{ji} + \frac{1}{2} A \sum_{ij} \sum_{kl} d_{ijkl} y_{ij} y_{kl} - \frac{1}{2} B \sum_{ij} g_{ij} y_{ij} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (3)$$

where g_{ij} is 1 if edge (i,j) exists in the given graph, 0 otherwise, and A, B are coefficients. The third term in Eq.(3) gives the number of embedded edges to be maximized.

Thus, the weights of the Hopfield network become:

$$w_{ijkl} = -A\delta_{il}\delta_{jk}(1 - \delta_{ij}) - Ad_{ijkl} \quad \cdots\cdots\cdots\cdots \quad (4)$$

And the thresholds become:

$$h_{ij} = \frac{1}{2} B g_{ij} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (5)$$

The Hopfield network energy function for the graph planarization problem can be rewrited as follow:

$$e = -\frac{1}{2} \sum_{i,j} \sum_{k,l} w_{ijkl} y_{ij} y_{kl} - \sum_{ij} h_{ij} y_{ij} + C \quad \cdots \quad (6)$$

where C is a constant.

Hopfield showed that the Hopfield network is guaranteed to converge with the energy taking on lower and lower values until the network reaches a steady state. It can be viewed as seeking a minimum in a mountainous terrain. Thus, in the first phase we can use the Hopfield network to find a local minimum or a global minimum of the energy function. However, it is usually difficulty for the network to find the global minimum because its inherent local minimum problem. Furthermore, there is no effective method to lead the network to reach the global minimum from a local minimum.

3. The Gradient Ascent Learning Phase for Graph Planarization

In the section 2, we have described the neural network representation of graph planarization. Using the Hopfield network updating we can find a local minimum or a global minimum of the energy function. However, it is usually difficulty for the network to find the global minimum because of its inherent local minimum problem. In this section we propose a learning algorithm which can help the network escape from a local minimum to the global minimum. In order to explain the learning method, we use a two-dimensional graph (Fig. 2) of energy function with a local minimum and a global minimum. The energy function value is reflected in the height of the graph. Each position on the energy terrain corresponds to a possible state of the network. For example, if the network is initialized onto the mountainous terrain A, the updating procedure of the Hopfield network makes the state of network move towards a minimum position and reach a steady state B (Fig. 2(a)).

Because the weights and the thresholds of the Hopfield network determine the energy terrain, we can change the weights and the thresholds to increase the energy at the point B so as to fill up the local minimum valley and

finally drive the point B out of the valley. Here, suppose that a vector $\mathbf{v}$ corresponds to the weights and the thresholds of the Hopfield network. Since for a parameter vector $\mathbf{v}$ the learning requires the parameter change to be in the positive gradient direction, we take:

$$\Delta \mathbf{v} = \varepsilon \nabla e(\mathbf{v}) \quad \cdots\cdots (7)$$

Where ε is a positive constant and ∇e is the gradient of energy function e with respect to the parameter vector $\mathbf{v}$ in the state B. Applying this learning rule (Eq. (7)) to the graph planarization problem, we can obtain

$$\Delta w_{ijkl} = p \frac{\partial e}{\partial w_{ijkl}} \quad \cdots\cdots (8)$$

$$\Delta h_{ij} = q \frac{\partial e}{\partial h_{ij}} \quad \cdots\cdots (9)$$

and

$$\frac{\partial e}{\partial w_{ijkl}} = -y_{ij} y_{kl} \quad \cdots\cdots (10)$$

$$\frac{\partial e}{\partial h_{ij}} = -y_{ij} \quad \cdots\cdots (11)$$

where, p and q are small positive constants and y_{ij}, y_{kl} correspond to the state of B.

Now we show that after we change the weights and the thresholds according to Eq.(10) and Eq.(11), point B will be on the slope of a valley. Suppose y_{Bij} represent the state of point B, y_{Pij} represent the state of any point P of energy terrain, then the change of energy in point P by the learning rule (Eq.(10) and Eq.(11)) will be:

$$\begin{aligned}
\Delta E_P &= \frac{1}{2} \sum_{i,j} \sum_{k,l} w_{ij,kl} y_{Pij} y_{Pkl} + \sum_{i,j} h_{ij} y_{Pij} \\
&\quad - \frac{1}{2} \sum_{i,j} \sum_{k,l} (w_{ij,kl} - p y_{Bij} y_{Bkl}) y_{Pij} y_{Pkl} \\
&\quad - \sum_{i,j} (h_{ij} - q y_{Bij}) y_{Pij} \\
&= \frac{p}{2} \sum_{i,j} \sum_{k,l} (y_{Bij} y_{Pkl})(y_{Bij} y_{Pkl}) \\
&\quad + q \sum_{i,j} (y_{Bij} y_{Pkl}) \quad \cdots\cdots (12)
\end{aligned}$$

Because point B is a minimum of energy function and the output of neuron in point B is at or near 0 or 1 [9], from Eq.(12), we can know easily that the increase of energy is largest when point P is at the same point as point B, and the larger the difference of state between point P and point B, the smaller energy increases in point P. Thus, we can see that the valley will be filled up in a most effective way. In general, point B may become a point on the slope of the valley. Thus, the learning (the second phase) makes the previous stable state B becomes a point on the slope of a valley (B'). After updating of the Hopfield network with the new weights and the new thresholds in the Hopfield network updating phase again, point B' goes down the slope of the valley and reaches a new stable state C (Fig.2(b)). Thus, the Hopfield network updating (Phase 1) and the gradient ascent learning (Phase 2) in turn may result

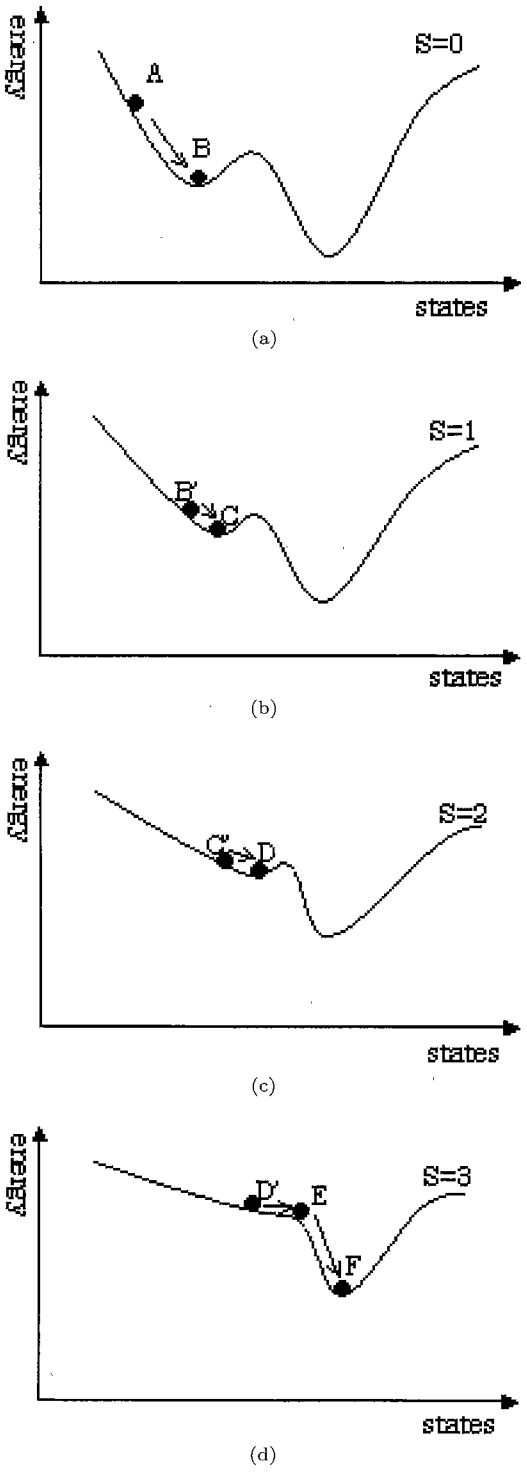


Fig. 2. The conceptual graph of the relation between energy and state transition in the learning process of the Hopfield network with two stable states.

in a movement out of a local minimum, and lead the network converge to a global minimum or a new local minimum (Fig.2(c) and (d)).

4. Algorithm

The following procedure describes the proposed algorithm. Note that there are two kinds of conditions for end of the learning. One has a very clear condition, for example, the N-queen problem in which the energy is zero if the solution is the optimal. Another one has not a clear condition, for example, the travelling salesman problem and the graph planarization problem in which the energy is not zero even the solution is the optimal. For the latter case, we have to set a maximum number of the learning (*learn_limit*) in advance. Learning stops if the maximum number of learning is performed. In general, we can determine the value of *learn_limit* according to the allowable computation time and the complexity of the problem. For planarization problem, we found that the network can always find good solutions within 10 learning times; therefore, we selected 30 as the maximum number of learning time in our simulations. If the *learn_limit* is the maximum number of learning times for the system termination condition, we have,

(1) Set $learn\_time = 0$ and set A, B and *learn_limit*.

(2) Randomize the initial values of y_{ij} for $i, j = 1 \cdots\cdots, N$ in the range of 0.0 to 1.0.

(3) The updating procedure is performed on the Hopfield network with original weights and thresholds until the network reaches a steady state (Phase 1).

(4) Use Eq.(8)-Eq.(11) to computer the new weights and the new thresholds (Phase 2).

(5) The updating procedure is taken on the Hopfield network with the new weights and thresholds until the network reaches a steady state.

(6)In order to avoid the shift of the state of the global minimum to a specific problem, the updating procedure on the Hopfield network may be re-performed with original weights and thresholds until the network reaches a new steady state.

(7) If the new steady state is better than the old one then the old state is replaced by the new state using the steady state obtained from step 6.

(8) Increment the *learn_time* by 1. If $learn\_time = learn\_limit$ then terminate this procedure, otherwise using the new steady state obtained from step 5, go to the step 4.

5. Simulation Results

The proposed algorithm was experimented on IBM NetVista A40 (Pentium III 733MHz) to a total of 19 benchmark graphs. These graphs provide a valuable source for testing the performance of algorithms for the planarization problem, because they arise from different areas of application. The Hopfield network updating phase used the weights and thresholds matrix defined in Eqs.(4) and (5) and the gradient ascent learning phase used the rules defined in Eqs.(8)-(11). Simulations refer to parameters set at or near $A = 1.0$ and $B = 1.0$. In the experiments *learn_limit* was set to 30. The initial values of neurons were randomized in the range of 0.0 to 1.0.

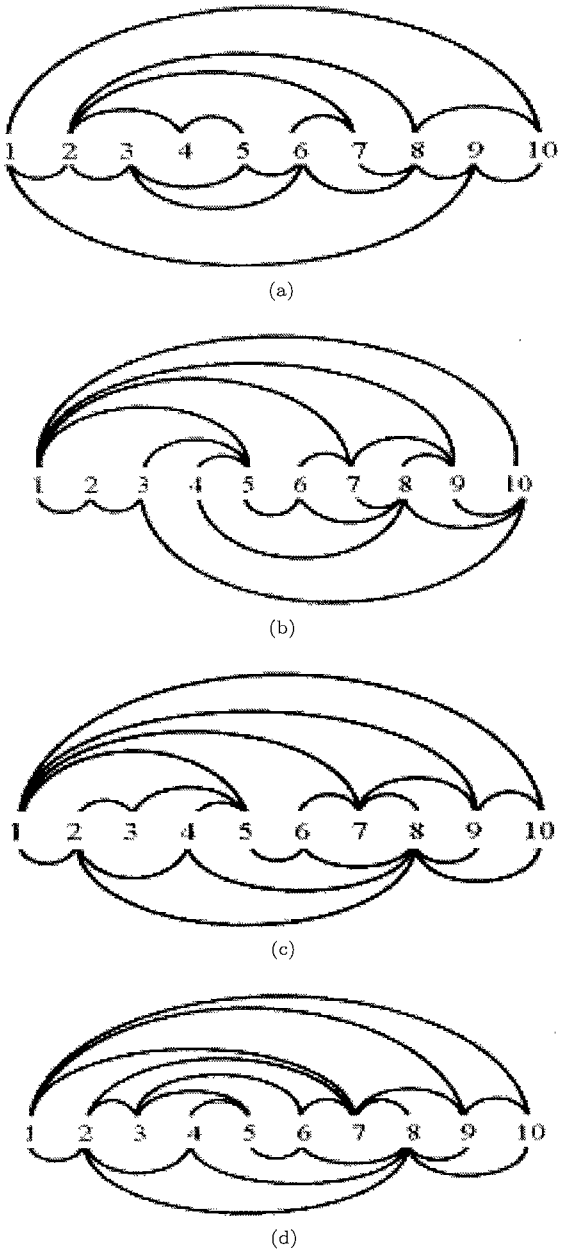


Fig. 3. (a) A maximal planar subgraph of $G1$ before learning. (b) After 2^{nd} learning. (c) After 3^{rd} learning. (d) Optimum maximal planar subgraph of $G1$ found by proposed algorithm

Fig. 3 shows the results of a simulation on the 10 vertices and 22 edges nonplanar graph($G1$) of Jayakumar et.al[1] which illustrate a typical progressive intermediate maximal planar subgraph during the Hopfield network updating phase (Phase 1) and the gradient ascent learning phase (Phase 2). Initially the Hopfield network converges to a time independent state (Fig.3(a)). It had 17 embedded edges. It is obviously not an optimum planar subgraph. After the 2^{nd} 3^{rd} and 8^{th} learning, the numbers of embedded edges increased to 18, 19 and 20, and generated the maximal planar subgraphs (b), (c) and finally (d), the optimum solution to the problem.

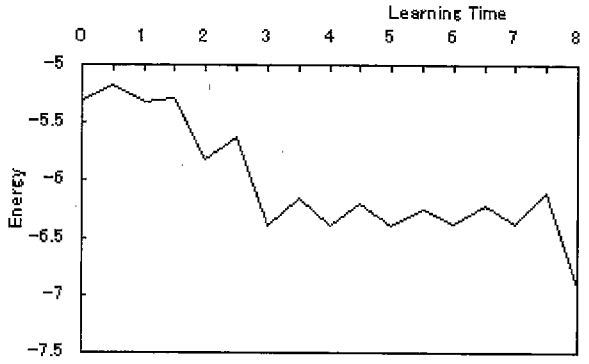


Fig. 4. The variation process of the energy during learning.

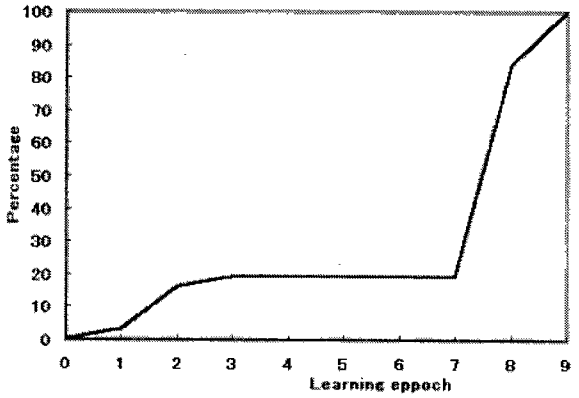


Fig. 5. The rate of the maximal planar subgraph from 100 different initial states using the proposed algorithm.

In order to gain some insight into the optimization process, the energy can be studied by plotting the energy as time. Figure 4 shows the variation of the energy during the Hopfield network updating and the gradient ascent learning. It can be seen from the figure that the 1^{st}, 4^{th}, 5^{th}, 6^{th} and 7^{th} learnings did not lead to a movement out of local minimum, and therefore there is no new planar subgraphs generated. The 2^{nd}, 3^{rd} and 8^{th} learning did yield a movement out of local minimum, thus resulting in new planar subgraphs (Fig.3(b), (c) and (d)).

To see how well the proposed algorithm was being made of a maximal planar subgraph, we generated 100 different initial states and performed updating and learning for $G1$ graph. Figure 5 shows the relation of percentage to find optimal solution of $G1$ and learning epochs with 100 different initial states using the proposed algorithm. The simulations found that the algorithm always converged to the optimal planar subgraph for every these 100 simulations within 10 learnings. Furthermore because of Hopfield network/Takefuji algorithm's sensitivity to parameters A and B, we chose several different parameters and performed our algorithm on these parameters. Fig. 6 shows the rate of the optimal planar subgraph from 100 different initinal states under different parameter values. As shown, our method worked well for every parameter set.

To evaluate our results, we compared Takefuji/Lee's

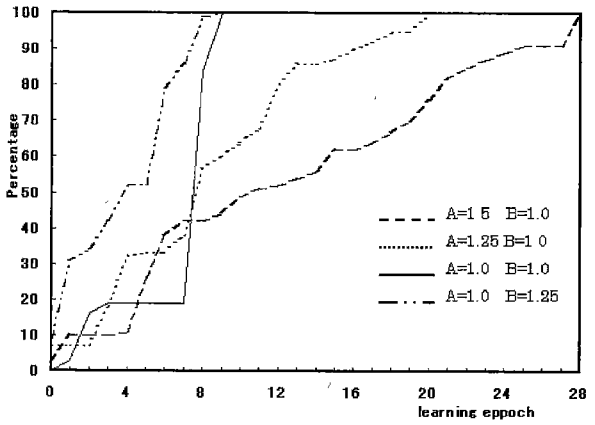


Fig. 6. The rate of the maximal planar subgraph from 100 different initial states under different parameter values.

results with our results because the Takefuji/Lee's algorithm is one of the most popular parallel algorithm and also because they claim superior performance to that of previously published algorithm [9]. Information on the test graphs as well as all results are shown in Table 1. The results that we recorded for each graph are the solutions in number of embedded edges, produced by Takefuji/Lee's algorithm (col 4), and by the proposed algorithm (col 5), the learning epochs (col 6) and the computation times (col 7) of the proposed algorithm to find the maximal planar subgraph. The learning epochs and the computation times are the average of 100 simulations.

We can see from Table 1 that under the same vertex ordering and $O(1)$ time, the proposed algorithm works well on generating a maximal or a better planar subgraph than Takefuji/Lee's algorithm [8][9] for solving the graph planarization problems. For the graphs ($G1$, $G2$ and $G4 - G7$) our algorithm and Takefuji/Lee's algorithm obtained the same planar subgraph. But our algorithm finds a hundred percent good solution, while Takefuji/Lee's algorithm could only find a very low percentage (about 20%) good solution to the problems. The results for graphs $G3$ and $G8 - G12$ show that our algorithm managed to obtain a triangulated graph in most case, which outperforms the Takefuji/Lee's algorithm. For example, for the graph of 10 vertices 24 edges ($G3$), the latest experiment using Takefuji/Lee's algorithm reported 21 edges. Our simulator using the proposed Hopfield network learning algorithm generated a new maximal planar subgraph with 22 edges, which is usually difficult and important for an algorithm or method to generate new, even a more edge for NP-complete problem [8]. Fig. 7 shows the maximal plannar subgraph of graph $G3$ generated by Takefuji/Lee's algorithm (a) and our algorithm (b). Furthermore, simulations were also carried out for several random graphs, for example $G13 - G19$ and showed that the proposed algorithm performed extremely well for these random graphs too. From the simulation result, we can summary that the proposed learning method is very efficient in the every

Table 1. Computational results.

Graph	No.vertices	No.edges	Takefuji/Lee	Proposed algorithm	Learning epochs	CPU times(s)
G1	10	22	20	20	8	0.30
G2	45	85	80	80	14	1.09
G3	10	24	21	22	6	0.24
G4	10	25	22	22	3	0.12
G5	10	26	22	22	3	0.16
G6	10	27	22	22	4	0.26
G7	10	34	23	23	5	0.40
G8	25	69	58	61	10	0.81
G9	25	70	59	61	15	1.15
G10	25	71	58	61	5	0.28
G11	25	72	60	61	7	0.30
G12	25	90	61	63	3	0.32
G13	50	367	70	82	11	82.17
G14	50	491	100	109	3	47.71
G15	50	582	101	115	4	107.43
G16	100	451	92	100	9	122.62
G17	100	742	116	126	4	246.98
G18	100	922	115	135	5	504.40
G19	150	1064	127	138	4	680.07

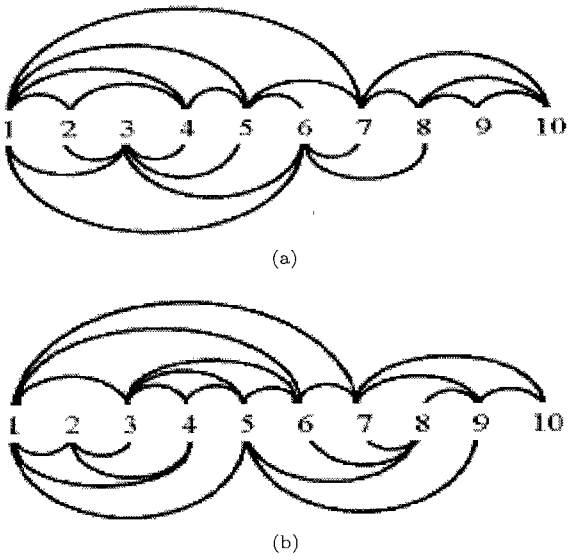


Fig. 7. (a) A maximal planar subgraph of G3 found by Takefuji/Lee's algorithm. (b) A new maximal planar subgraph of G3 found by proposed algorithm.

test kinds of graphs compared with Takefuji/Lee's algorithm. The reason why the proposed algorithm could generate better solution compared with that found by Takefuji/Lee's algorithm is that in the proposed algorithm, we used gradient ascent learning on the Hopfield network to help the network escape from the local minima as described in section 3. But in Takefuji/Lee's algorithm, there is no method to make the network escape from a local minimum to the global minimum.

6. Conclusion

We have proposed a Hopfield network learning algorithm for graph planarization and showed its effectiveness by simulation experiments. The learning algorithm which is designed to embed a graph on a plane, has two phases, the Hopfield network updating phase and the gradient ascent learning phase. In the first phase we implemented the Hopfield network for optimizing the energy function in state space. In the second phase we intentionally increased the energy of the Hopfield network by modifying parameters in weight domain in a gradient ascent direction, thus making the network escape from the near-maximal planar subgraph (i.e., a local minimum). As a parallel algorithm for the graph planarization, our algorithm not only generated an optimal or near-optimal planar subgraph from the nonplanar or planar graph, but also embedded the subgraph on a planar. The proposed algorithm was applied to many graphs up to 150 vertices and 1064 edges and was compared with Takefuji/Lee's method. The simulation results showed that the proposed algorithm was much better than Takefuji/Lee's method in terms of the solution quality for every tested graph.

Acknowledgment

This work was supported in part by the grant-in-aid for Science Research of the Ministry of Education, Science and Culture of Japan under Grant (C)(2)12680392.

(Manuscript received January 19, 2001, revised February 22, 2002)

References

(1) R. Jayakumar, K. Thulasiraman, and M. N. S. Swamy: "$O(n^2)$ algorithms for graph planarization", *IEEE Trans. Computer Aided Design*, Vol.8, No.3, pp.257-267 (1989)

(2) G. Kant: "An $O(n^2)$ maximal planarization algorithm based on PQ-tree", Technical Report RUU-CS-92-03, Dept.of Computer Science, Utrecht University, Utrecht, The Netherlands, (1992)

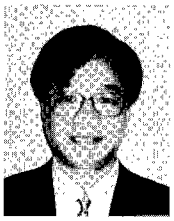
(3) O. Goldschmidt and A. Takvorian: "An efficient graph planarization two-phase heuristic", *Network*, Vol.24, No.2, pp.69-73 (1994)

(4) J. Cai, X. Han, and R. Tarjan: "An $O(m \log n)$ time algorithm for the maximal planar subgraph problem", Technical Report CS-TR-309-91, Dept. of Computer Science, Princeton University, Princeton (1991)

(5) G. Di Battista and R. Tamassia: "Incremental planarity testing", *Proceedings of the 30th Annual IEEE Symposium on Foundation on Computer Science*, Chapel Hill, North Carolina, pp.436-441 (1989)

(6) P. C. Liu and R. C. Geldmacher: "On the deletion of nonplanar edges of a graph", *Proceedings of the 10th South-East Conference on Combinatorics, Graph Theory, and Comput-*

ing, Boca Raton, FL, USA, pp.727-738 (1977)
(7) J.J.Hopfield and D.W.Tank: " 'Neural' computation of decisions in optimization problems", *Biol. Cybern.*, No.52, pp.141-152 (1985)
(8) Y. Takefuji and K. C. Lee: "A near-optimum parallel planarization algorithm", *Science*, Vol.245, No.4922, pp.1221-1223 (1989)
(9) Y. Takefuji, K. C. Lee, and Y. B. Cho: "Comments on $O(n^2)$ algorithm for graph planarization," *IEEE Trans. Comput. Aided Design*, Vol.10, No.12, pp.1582-1583 (1991)
(10) D. H. Ackley, G. E. Hinton, and T. J. Sejnowski: "A learning algorithm for Boltzman Machines," *Cognitive Science*, No.9, pp.147-169 (1985)
(11) J. Hertz, A. Krogh, and R. G. Palmer: Introduction to the Theory of Neural Computation, Addison Wesley Publishing Company (1991)
(12) R. L. Wang, Z. Tang, and Q. P. Cao: "A Near-Optimum Parallel Algorithm for Bipartite Subgraph Problem Using the Hopfield Neural Network Learning", *IEICE Trans. Fundamentals*, Vol.E85-A, No.2, pp.497-504 (2002)



Rong Long Wang (Student Member) received a B.S. degree from Hangzhe teacher's college, Zhejiang, China and an M.S. degree from Liaoning University, Liaoning, China in 1987 and 1990, respectively. Since 1990 he has been an Instructor in Benxi University, Liaoning, China. Now he is working toward the Ph.D degree at Toyama University, Japan. His main research interests are neural networks and optimization problems.

Zheng Tang (Non-member) received a B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tsinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an Instructor in the Institute of Microelectronics at Tsinhua University. From 1990 to 1999, he was an Associate Professor in the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a Professor in the Department of Intellectual Information Systems. His current research interests include intellectual information technology, neural networks, and optimizations.

Qi Ping Cao (Non-member) received a B.S. degree from Zhejiang University, Zhejiang, China and an M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China in 1983 and 1986, respectively. From 1987 to 1989, she was an Assistant Professor in the Institute of Microelectronics at Shanghai Jiaotong University, Shanghai, China. From 1989 to 1990, she was a Research Student in Nagoya University, Nagoya, Japan. From 1990 to 2000, she was a Senior Engineer in Sanwa Newtech Inc., Japan. In 2000, she joined Tateyama Systems Institute, Japan. Her current research interests include multiple-valued logic, neural networks, and optimizations.