An Efficient Neural Algorithm for Two-layer Planarization Problem in Graph Drawing

Xinshun Xu* Student Member

Zheng Tang* Non-member

Ronglong Wang** Member

Jiahai Wang* Non-Member

Guangpu Xia* Non-member

A neural algorithm for the two-layer planarization problem using a gradient ascent learning of the Hopfield network is presented. This algorithm which is designed to embed a two-layer graph on a plane, uses the Hopfield network to get a near-maximal two-layer planar subgraph, and increases the energy by modifying the weights and the thresholds in gradient ascent direction to help the network escape from the state of near-maximal two-layer planar subgraph to the state of the maximal two-layer planar subgraph. The experimental results show that the proposed algorithm generates much better solutions than traditional Hopfield network and simulated annealing.

Keywords : Two-layer planarization problem, Maximal planar subgraph, Hopfield neural network, Gradient ascent learning, NP-complete problem

1. Introduction

A bipartite graph is a graph $G=(A, B, E)$ with vertex sets A and B , and an edge set E connecting vertices in A with vertices in B . There are no edges between two vertices in the same vertex set. A bipartite graph is a two-layer graph if it is drawn in such a way that all the vertices in A appear on a line (we call it the upper line), the vertices in B appear on the other line which is parallel with the upper line (we call it the lower line), and the edges are drawn as straight lines. A two-layer graph is two-layer planar graph if its edges have no crossing except at a common endpoint⁽¹⁾. For a given two-layer graph $G=(A, B, E)$, the two-layer planarization problem is to extract a subgraph $G'=(A, B, F)$, $F \subseteq E$, of maximum edges ($|F|$ is maximum), which contains no crossing⁽¹⁾. The similar problem is two-layer straightline crossing minimization problem which is to change the permutation of nodes so that the crossings of edges are minimal. It is well known that these problems are NP-complete⁽²⁾⁽³⁾. These problems have many applications in circuit and VLSI design, network design and analysis, DNA mapping and visualization⁽³⁾⁽⁵⁾, and so on. They have been considered in many literatures so far. Mutzel et al.⁽¹⁾⁽⁶⁾ proposed algorithms based on their polyhedral studies and used branch-and-cut algorithm to solve these problems. But when the nodes are larger than 30, it can not solve the planarization problems in reasonable running time⁽⁷⁾. An exact branch-and-bound algorithm for the graph planarization problem was introduced by Foulds and Robinson⁽⁸⁾, but was also limited to small dense graphs. Eades and Whitesides showed that the two-layer planarization problem could be solved by transforming it to a long increasing subsequence problem if two layers are

fixed⁽⁹⁾. Michael Junger et al.⁽¹⁰⁾ described an exact and heuristic algorithm for two-layer straightline crossing minimization. The first neural network representation for solving such optimization problems was introduced by Hopfield and Tank⁽¹¹⁾. Unfortunately, due to its inherent local minimum problems⁽¹²⁾, the rates to get the optimal solution are very low. This may be improved by some more sophisticated architecture, such as the Boltzmann machine⁽¹³⁾. However it is very slow because it needs extensive averaging over stochastic variables.

In this paper we present a neural algorithm for the two-layer planarization problem based on the Hopfield network learning. The neural 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 to find a near-maximal two-layer planar subgraph. The second phase intentionally increases energy of the Hopfield network by modifying parameters in weight and threshold domain in gradient ascent direction, thus making the network escape from the near-maximal two-layer planar subgraph (i.e., a local minimum) which the network once falls into in phase one. The proposed algorithm is applied to 22 randomly generated graphs with up to 200 vertices and 1800 edges. We found that it worked much better on generating a maximal two-layer planar subgraph than the traditional Hopfield network and Simulated Annealing. Furthermore, the proposed algorithm is much faster than Simulated Annealing.

2. Formulation

For a given two-layer graph $G=(A, B, E)$, A and B are vertex sets and E is edge set. We let $|A|=n$, $|B|=m$. Binary variables (y_{ij} , $i=0,1, \dots, n-1$, $j=0,1, \dots, m-1$) express the connection between the vertices of the upper line and the vertices of the lower line. The variable has only two values (1 or 0) which express the connection exists or not. For example, $y_{ij}=1$ indicates that there is a line

* Faculty of Engineering, Toyama University
Toyama-shi, 930-8555, Japan.

** Faculty of Engineering, Fukui University
Fukui-shi, 910-8507, Japan

Fig. 1. The crossing condition between two edges for the two-layer planarization problem.

connection between the i^{th} vertex of the upper line and the j^{th} vertex of the lower line, $y_{ij}=0$ indicates no line. So the number of vertices of the given graph determines the number of variables. For example, if there are n vertices in the upper line, and m vertices in the lower line, then $n \times m$ variables are required. Thus the maximum number of embedded edges condition can be expressed by:

$$e_1 = \sum_{ij} g_{ij} y_{ij} \quad (1)$$

where $g_{ij}=1$ if there is an edge between the i^{th} vertex of the upper line and the j^{th} vertex of the lower line in the given graph, 0 otherwise.

$$e_2 = \sum_{ij} \sum_{kl} d_{ijkl} y_{ij} y_{kl} \quad (2)$$

where d_{ijkl} expresses if two edges (y_{ij} and y_{kl}) crossed, $d_{ijkl}=1$ if $i < k$ and $l < j$ or $k < i$ and $j < l$ when $g_{ij}=1$ and $g_{kl}=1$, 0 otherwise. Fig.1 shows the crossing conditions between two edges (y_{ij} and y_{kl}) for the two-layer planarization problem.

The objective of the problem is to maximize e_1 with e_2 zero. Thus, the two-layer planarization problem can be described as to decrease the following energy function:

$$E = \frac{1}{2} A \sum_{ij} \sum_{kl} d_{ijkl} y_{ij} y_{kl} - \frac{1}{2} B \sum_{ij} g_{ij} y_{ij} \quad (3)$$

where A, B are coefficients. Comparing with the standard energy function of the Hopfield network, we can deduce the weights and the thresholds of the Hopfield network as:

$$w_{ijkl} = -A d_{ijkl} \quad (4)$$

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

3. Gradient Ascent Learning

Hopfield showed that the Hopfield network is guaranteed to converge with energy taking on lower and lower values until it reaches a stable state⁽¹¹⁾. It can be viewed as seeking a minimum in a mountainous terrain. But it is difficult for the network to find the global minimum for its inherent local minimum problem. Although many variations of the Hopfield network also have been proposed in order to improve it, each of those solutions doesn't work well when processing large problems. For example, the Boltzmann machine will be very slow because it needs extensive averaging over stochastic variables⁽¹⁴⁾.

In order to help the network generate a maximal two-layer

planar subgraph, we propose a learning algorithm that can help the network escape from a local minimum. Because the energy of Hopfield network is determined by its weights and its thresholds, the learning can be performed by changing the weights and the thresholds of the network in the gradient ascent direction so that energy at the local minimum increases. Since for a parameter vector W , the learning requires the parameter change to be in the positive gradient direction, we take:

$$\Delta W = \varepsilon \nabla E(W) \quad (6)$$

where ε is a positive constant. Applying this learning rule to the two-layer planarization problem, we can obtain:

$$\Delta w_{ijkl} = p \frac{\partial E}{\partial w_{ijkl}} \quad (7)$$

$$\Delta h_{ij} = q \frac{\partial E}{\partial h_{ij}} \quad (8)$$

where, p and q are small positive constants.

From the energy function we can get:

$$\frac{\partial E}{\partial w_{ijkl}} = -y_{ij} y_{kl} \quad (9)$$

$$\frac{\partial E}{\partial h_{ij}} = -y_{ij} \quad (10)$$

Then:

$$\Delta w_{ijkl} = -p y_{ij} y_{kl} \quad (11)$$

$$\Delta h_{ij} = -q y_{ij} \quad (12)$$

where, y_{ij} and y_{kl} correspond to the state of a local minimum.

In order to explain the learning method, we use a two-dimensional graph (Fig.2) of the 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)). Then we 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.

Now we show that after we change the weights and the thresholds according to Eq.(11) and Eq.(12), point B will be on the slope of a valley. Suppose $y_{bij}(i, j=1, 2, \dots, N)$ represent the state of point B, $y_{pij}(i, j=1, 2, \dots, N)$ represent the state of any point P of energy terrain, then the change of energy in point P by the learning rule (Eq.(11) and Eq.(12)) can be calculated by Eq. (13).

Because point B is a minimum of the energy function and the output of neuron in point B is at or near 0 or 1⁽¹⁵⁾, from Eq.(13), we can know easily that the change of energy is the 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 changes 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.

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.

$$\begin{aligned}
 \Delta E_p &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N w_{ijkl} y_{Pij} y_{Pkl} + \sum_{i=1}^N \sum_{j=1}^N h_{ij} y_{Pij} \\
 &\quad - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N (w_{ijkl} - p y_{Pij} y_{Pkl}) y_{Pij} y_{Pkl} \\
 &\quad - \sum_{i=1}^N \sum_{j=1}^N (h_{ij} - q y_{bij}) y_{Pij} \\
 &= \frac{p}{2} \sum_{i=1}^N \sum_{k=1}^N (y_{bij} y_{Pij}) (y_{bkl} y_{Pkl}) + q \sum_{i=1}^N \sum_{j=1}^N (y_{bij} y_{Pij})
 \end{aligned}
 \tag{13}$$

Thus, the learning 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 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 termination 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 does not have a clear condition, for example, the traveling salesman problem and the planarization problem in which the energy is not zero even the solution is optimal. For the later case, we have to set a maximum number of learnings (*learning_limit*) in advance. Learning stops if the maximum number of learning is performed. If the *learning_limit* is supposed to be the maximum number of learning times for the system termination condition, we have the following algorithm:

1. Set *learn_time*=0, and set A, B and *learn_limit*.
2. Randomize the initial values of y_{ij} ($i=0, 1, \dots, n-1; j=0, 1, \dots, m-1$) in the range of 0.0 and 1.0.
3. Perform the updating procedure on the Hopfield network with original weights and thresholds until the network reaches a steady state (Phase 1).
4. Record the stable state and set the best solution as this stable state.
5. Use Eq.(11) and Eq.(12) to compute the new weights and the new thresholds (Phase 2).
6. The updating procedure is taken on the Hopfield network with the new weights and the new thresholds until the network reaches a new steady state.
7. Because the weights and the thresholds of the Hopfield network determine the energy terrain, once the weights and the thresholds are changed, the energy terrain may be changed and the position of global minimum in energy terrain may also be changed. In order to check if the local minimum is filled up and avoid the possible false solution to a specific problem due to the shift of the state of the global minimum, the updating procedure on the Hopfield network may be re-performed with original weights and thresholds until the network reaches a new stable state. If this stable state is different to recorded state then set the weights and the thresholds to the original ones and state of network to this stable state, and replace the recorded stable state to this stable state, otherwise, the weight, the thresholds and the state of network are set to those at the end of step 6.
8. If the solution obtained from step 7 is better than the best solution, then the best solution is replaced by the solution obtained from step 7.
9. Increment the *learn_time* by 1. If *learn_time*=*learn_limit* then terminate this procedure, otherwise go to the step 5.

5. Experimental Results

The proposed algorithm was experimented on PentiumIII 800MHz (256MB) on a total of 22 randomly generated two-layer graphs up to 200 vertices and 1800 edges. We set $A=1.0$ and $B=1.0$, $p=0.001$ and $q=0.01$. The *learn_limit* was set to 40. The initial values of neurons were randomized in the range of 0.0 to 1.0.

Fig.3 shows the results of a simulation on a given graph with 20 vertices and 30 edges which illustrates a typical progressive

process for generating maximal two-layer planar subgraph during Hopfield network updating and the gradient ascent learning. The Hopfield network started from an randomly initialized graph (Fig.3.(a)) and converged to a two-layer planar subgraph (Fig.3.(b)). After the 3rd and 7th learning, the network found a two-layer planar subgraph having 10 edges (Fig.3.(c)), and finally a two-layer planar subgraph having 12 edges (Fig.3.(d)).

In order to gain some insight into the optimization process of Fig.3, we plotted the variation of energy during the Hopfield network updating phase and the gradient ascent learning phase on Fig.4. From Fig.4, we can see that the Hopfield network updating decreased energy from the initial point A to point B, and generated a two-layer planar subgraph (Fig.3.(b)) which had 8 edges. After the 3rd learning, energy increased from 0.035893 (point D) to 0.532975 (point D') and then dropped down to -1.33034 (point E), thus resulting in a new two-layer planar subgraph (Fig.3.(c))

Fig. 3. The graph shows the results in different learning times when processing a given graph with 20 vertices and 30 edges.

which had 10 edges. After the 7th learning, energy increased from 0.51624 (point H) to 0.94334 (point H') and then dropped down to -1.31965 (point I), the maximal two-layer planar subgraph (Fig.3.(d)) was obtained. Because we have changed the weights and the thresholds in the algorithm, the energy of global minimum may be larger than local minima. Therefore, the energy of point I (a global minimum) is larger than the energy of point B and point E (two local minima).

Furthermore, this process can be described using a conceptual graph of the energy function (Fig.5). Started in state A, the first Hopfield network updating (phase I) moved in a general "downhill" direction of the energy function, reached a steady state (point B of Fig.5.(a)), and stopped changing. Then the first gradient ascent learning (phase II) drove state B to become a point on the slope of a valley (point B') by changing weights and thresholds in gradient ascent direction. After the second Hopfield network updating with the new weights and thresholds, point B' went down the slope of the valley and reached a new steady state C (Fig.5.(b)). Then the second learning was performed, and drove point C to point C' (Fig.5.(c)). The network reached the third steady state D (Fig.5.(c)). It is worthy to be noted that the first and second learning did not produce a new two-layer planar subgraph while the third learning found a better solution. We could see that the learning helped the network escape from the first local minimum and find the second local minimum (Fig.5.(d)). In the same way, the 4th, 5th, 6th and 7th learnings filled up the second local minimum and helped the network find the global minimum (Fig.5.(h)). Although we visualized this learning process in two dimensions, they are actually as many dimensions as there are processing units. Furthermore, although we plotted the energy function with two local minima and one global minimum, some "local minima" might have been smoothed during the learnings.

As mentioned above, our algorithm provided an effective method of solving the local minimum problem. Focusing on the local minimum problem of optimization techniques, some other optimization techniques have been proposed. Simulated Annealing (SA)⁽¹⁶⁾ is a widely used meta-heuristic. It could be described as a randomized scheme, which reduces the risk of getting trapped in local minima by allowing moves to inferior solution. Although Simulated Annealing is a powerful method for solving local

Fig. 4. The variation process of energy as time changes when processing the graph of Fig.3(a)

Fig. 5. The conceptual graphs in different learning times for the graph of Fig.3. They show how the learning help the network escape from local minima and find the global minimum.

minimum problems, it always requires more iterations than exhaustive search to find a good solution⁽¹⁷⁾. To assess the effectiveness of the proposed method, Simulated Annealing was executed for comparison. In our simulations, we used the scheme proposed by Connolly⁽¹⁸⁾⁽¹⁹⁾ to adapt simulated annealing to the two-layer planarization problem, because Connolly's Simulated Annealing algorithms was built for arbitrary 0-1 problems and very good solutions were reported in their work for quite a wide range of combinational problems. It is well known as GPSIMAN. In the two-layer planarization problem, we took the neighbors of a solution $s = \{y_{01}, y_{02}, \dots, y_{(n-1)/(m-1)}\}$ to be all those solutions that variables were chosen and altered at random. Other annealing parameters were selected to be those reported in⁽¹⁸⁾. The comparison of computation times and solutions in number of embedded edges produced by the traditional Hopfield network,

Simulated Annealing and our learning method were shown in Table 1. In order to verify that our algorithm can get the maximal two-layer planar subgraph, we used exhaustive search to find the maximal two-layer planar subgraph. But we only gave the results of G1, G2, G4 in Table 1, because we could not get the solutions of other graphs within five days. From Table 1, we can see that our algorithm did produce the maximal two-layer planar subgraph for G1, G2 and G4 in a very short time.

In Table 1, the graphs we used were randomly generated with different edge-density parameter (0.15, 0.3, 0.5). The vertices in Table.1 are the total vertices of both lines. The results that we recorded for each graph are the solutions in number of embedded edges, produced by the traditional Hopfield network, Simulated Annealing and the proposed algorithm to find the maximal two-layer planar subgraph. The learning epochs and the

Table 1. Simulation results for the two-layer planarization problem using traditional Hopfield network , simulated annealing algorithm and the proposed algorithm.

Graph				Hopfield		SA		Proposed Algorithm			Exhaustive	
No.	Nodes	Density	Edges	Max	Time(s)	Max	Time(s)	Max	Epochs	Time(s)	Max.	Time(s)
G1	20	0.15	15	5	0.03	5	8.29	5	1	0.04	5	1320
G2	20	0.3	30	8	0.031	12	38.75	12	7	1.392	12	40988
G3	20	0.5	50	10	0.01	16	37.62	16	5	0.32	-	-
G4	30	0.15	33	12	0.14	12	53.13	12	1	0.42	12	595646
G5	30	0.3	67	12	0.07	18	61.44	18	14	1.37	-	-
G6	30	0.5	112	17	0.13	23	63.57	23	12	1.19	-	-
G7	40	0.15	60	14	0.62	18	172.79	18	8	5.62	-	-
G8	40	0.3	120	16	1.12	23	133.32	23	2	3.12	-	-
G9	40	0.5	200	21	1.10	32	82.18	33	30	21.42	-	-
G10	60	0.15	135	16	4.69	25	229.54	25	12	64.33	-	-
G11	60	0.3	270	21	7.87	39	335.88	39	22	113.39	-	-
G12	60	0.5	450	28	1.34	43	389.38	46	3	23.26	-	-
G13	80	0.15	240	12	2.64	31	1443.04	31	2	43.75	-	-
G14	80	0.3	480	20	8.81	43	1183.95	43	15	305.34	-	-
G15	80	0.5	800	38	22.79	53	1136.65	56	34	824.18	-	-
G16	100	0.15	375	22	22.6	43	1443.78	43	30	946.16	-	-
G17	100	0.3	750	38	36.11	54	2103.19	61	12	732.51	-	-
G18	100	0.5	1250	59	82.19	74	2678.45	74	9	624.44	-	-
G19	120	0.15	540	23	59.15	48	4784.72	48	12	942.82	-	-
G20	120	0.3	1080	43	85.89	64	7128.81	67	5	611.33	-	-
G21	120	0.5	1800	68	96.03	89	8008.42	89	7	794.57	-	-
G22	200	0.15	1500	32	267.14	-	-	113	6	1257.79	-	-

computation times are the average of 10 simulations. We can see that the proposed algorithm worked much better than the traditional Hopfield network. Although SA could get the results as good as our algorithm for most graphs, it spent much longer time, and it could not get the feasible solution for G22 within 5 hours. So we can summarize that our learning method outperformed to both the traditional Hopfield network and Simulated Annealing.

6. Conclusions

We have proposed an efficient neural algorithm for the two-layer planarization problem and showed its effectiveness by simulation experiments. The neural algorithm, which is designed to embed a two-layer 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 domain. In the second phase we intentionally increased energy of the Hopfield network by modifying parameters in gradient ascent direction, thus making the network escape from local minima. The proposed algorithm has been applied to many graphs up to 200 vertices and 1800 edges. The simulation results showed that the proposed algorithm was much better than the traditional Hopfield network and Simulated Annealing on solving two-layer planarization problem.

(Manuscript received Nov. 11, 2002, revised Sep. 28, 2004)

References

- (1) P. Mutzel and R. Weiskircher : "Two-layer planarization in graph drawing", ISAAC: 9th International Symposium on Algorithms and Computation, pp.69-78 (1998)
- (2) F. Comellas : "Using genetic algorithm for planarization problems", Computational and Applied Mathematics I, Eds. C. Brezinski and U. Kulish, Elsevier Science Publishers B.V. (North Holland), pp.93-100 (1992)
- (3) M.R. Garey and D.S. Johnson : "Crossing number is NP-complete", SIAM Journal of Algebraic and Discrete Methods, Vol.4, pp.312-316 (1983)
- (4) T. Lengauer : Combinatorial Algorithms for Integrated Circuit Layout, B.G. Teubner (1990)
- (5) T. Nishizeki and N. Chiba : Planar Graphs: Theory and Algorithms, North Holland (1998)
- (6) M. Junger, E. K. Lee, P. Mutzel, and T. Odenthal : "A polyhedral approach to the multi-layer crossing minimization problem", Lecture Notes in Computer Science, Vol.1353, pp.13-24 (1997)
- (7) M.G.C. Resende and C.C. Ribeiro : "Graph planarizaion", Encyclopedia of Optimization, Kluwer Academic Press (2001)
- (8) L. R. Foulds and R. W. Robinson : "A strategy for solving the plant layout problem", Operational Research Quarterly, Vol.27, pp.845-855 (1976)
- (9) P. Eades and S. Whitesides : "Drawing graphs in two layers", *Theoretical Computer Science*, Vol.131, pp.361-374 (1994)
- (10) M. Junger and P. Mutzel : "Exact and heuristic algorithms for 2-layer straightline crossing minimization", Lecture Notes in Computer Science, Vol.1027 (1996)
- (11) J. J. Hopfield and D. W. Tank : "Neural computation of decisions in optimization problems", Biological Cybernetics, Vol.52, pp.141-152 (1985)
- (12) Y. Takefuji and K. C. Lee : "A near-optimum parallel planarization algorithm", *Science*, Vol.245, No.4922, pp.141-152 (1989)
- (13) D. H. Ackley, G. E. Hinton, and T. J. Sejnowski : "A learning algorithm for Boltzman Machines", *Cognitive Science*, No.9, pp.147-169 (1985)
- (14) J. Hertz, A. Krogh, and R.G. Palmer : Introduce to the Theory of Neural

Computation. Reading, MA: Addison Wesley (1991)

- (15) J. J. Hopfield : "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the National Academy of Sciences*, Vol.81, pp.3088-3092 (1984)
- (16) S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi : "Optimisation by simulated annealing", *Science*, Vol.220, pp.671-680 (1983)
- (17) P. J. M. van Laarhoven and E. H. L. Aarts : *Simulated Annealing: Theory and Applications*. Kluwer, Dordrecht (1988)
- (18) D. Connolly : "General purpose simulated annealing", *Journal of the Operational Research Society*, Vol.43, pp.495-505 (1992)
- (19) D. Abramson and M. Randall : "A simulated annealing code for general integer linear programs", *Annals of Operations Research*, Vol.86, pp.3-21 (1999)

Xinshun Xu

(Student Member) received a B.S. degree from Shandong Normal University, Shandong, China and an M.S. degree from Shandong University, Shandong, China in 1998 and 2002, respectively. From 1998 to 2000, he was an Engineer in Shandong Provincial Education Department, Shandong, China. Now he is working toward the Ph.D. degree at Toyama University, Toyama, Japan.

His main research interests are neural networks, pattern recognition, image processing, 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, Tsinghua 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.

Ronglong Wang

(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. He received his D.E. degree from Toyama University, Toyama, Japan in 2003. From 1990 to 1998, he was an Instructor in Benxi University, Liaoning, China. In 2003, he joined University of Fukui, Fukui Japan, where he is currently an Instructor in Department of Electrical and Electronics Engineering. His current research interests include genetic algorithm, neural networks, and optimization problems.

Jiahai Wang

(Non-member) received a B.S. degree from Gannan Teachers College, Jiangxi, China and an M.S. degree from Shandong University, Shandong, China in 1999 and 2001, respectively. Now he is working toward the Ph.D degree at Toyama University, Toyama, Japan. His main research interests are neural networks and optimization problems.

Guangpu Xia

(Non-member) received a B.S. degree from Metallurgical Management's college, Beijing, China in 1994. From 1994 to 2000, she was an Accountant in Beijing Labour and Social Security Bureau, Beijing, China. She received an M.S. degree from Toyama University, Toyama, Japan in 2003. From 2003 she is working toward the Ph.D degree at Toyama University, Japan. Her main research

interests are neural networks and optimization problems.