IEICE TRANS. INF. & SYST., VOL. E80-D, NO. 9 SEPTEMBER 1997

PAPER Special Issue on Architectures, Algorithms and Networks for Massively Parallel Computing

Neural Computing for the m-Way Graph Partitioning Problem

Takayuki SAITO†, *Student Member and* Yoshiyasu TAKEFUJI††, *Nonmember*

SUMMARY The graph partitioning problem is a famous combinatorial problem and has many applications including VLSI circuit design, task allocation in distributed computer systems and so on. In this paper, a novel neural network for the m-way graph partitioning problem is proposed where the maximum neuron model is used. The undirected graph with weighted nodes and weighted edges is partitioned into several subsets. The objective of partitioning is to minimize the sum of weights on cut edges with keeping the size of each subset balanced. The proposed algorithm was compared with the genetic algorithm. The experimental result shows that the proposed neural network is better or comparable with the other existing methods for solving the m-way graph partitioning problem in terms of the computation time and the solution quality.

key words: *neural network, graph partitioning, heuristic algorithm, and combinatorial optimization*

1. Introduction

The graph partitioning problem (GPP) is to divide nodes into subsets such that the sum of weights on cut edges are minimized with keeping the size of each subset balanced. GPP has many applications in VLSI circuit design [8], task allocation in distributed computer systems [9], network partitioning [10] and so on. In VLSI circuit design, a complex circuit is decomposed into circuits. The goal is to minimize the total number of interconnections among subcircuits. When a circuit is too large to fit on a single chip, it must be decomposed into several chips. The techniques for solving GPP are used for allocating elements of the circuit to several chips. In task allocation in distributed computing systems, application software program is partitioned into modules and they are allocated to several processors where the goal is to minimize interprocessor communication and their load balancing. Proper allocation of modules is important to use resources efficiently.

GPP is one of the most famous combinatorial problems [11]. GPP has proven to be an NP-Complete problem. Due to its computational intractability to obtain the optimal solution, many heuristic approaches have been proposed to find suboptimal solutions: Kernighan-Lin [6] algorithm, Simulated Annealing [13], Tabu Search [15], [16], Genetic Algorithm [17], [18].

Kernighan-Lin (KL) algorithm is the well-known and common heuristic algorithm for solving GPP. Kernighan and Lin proposed a two-way graph partitioning algorithm. They start with arbitrary partitions and apply pairwise swapping of nodes between partitions until no further improvement is possible. KL algorithm is basically the two-way partitioning algorithm. They extended their algorithm to the m-way graph portioning. It is reported that the performance of KL algorithm degrades when weights of all nodes are not equal [7]. It is also reported that their algorithm is inferior to other methods when it is applied to the m-way graph partitioning [7].

Simulated Annealing (SA) is a stochastic optimization method based on the analogy of physical annealing [12]. Johnson et al. proposed SA approach to GPP [13]. They showed that SA was superior to KL algorithm in terms of the solution quality. SA usually produces good solutions, but it is sometimes time consuming.

Tabu Search (TS) is the meta-level heuristic for solving optimization problems [14]. The characteristic of the TS is to use short-term memory called Tabu List to escape from the local minimum. It starts with an initial solution and iteratively improves the solution. If the improved solution is not in the Tabu List, the solution is adopted. Tabu List is used to prevent the system from generating the same solutions repeatedly in a certain period. Fujisawa et al. applied TS to the two-way graph partitioning problem and showed that TS is superior to SA in terms of both the quality of the solution and the computation time.

Genetic Algorithm (GA) is the stochastic search algorithm based on the natural evolution model. In this algorithm, a number of local optimum solutions are prepared as chromosomes. Repeating crossover and mutation, chromosomes which represent better optimal solutions will survive. Pirkul et al. [17] proposed the GA based approach to the two-way graph partitioning problem. Ahmad et al. [18] proposed the problem-space genetic algorithm (PSGA) based approach to the m-way partitioning problem. PSGA is the integration of the simple and fast heuristic algorithm and GA. In this algorithm, chromosomes have the information for the simple heuristic algorithm. Chromosomes are evaluated by this heuristic algorithm and better solutions

Manuscript received January 20, 1997.

†The author is with the Graduate School of Media and Governance, Keio University, Fujisawa-shi, 252 Japan.

††The author is with the Faculty of Environmental Information, Keio University, Fujisawa-shi, 252 Japan.

are selected by GA. Ahmad et al. showed that PSGA is superior to SA and some other algorithms for the m-way graph partitioning problem.

Neural network is also one of the superior heuristic methods for solving combinatorial problems. Neural network has been used to solve graph problems and circuit design problems [2], [3]. Bout et al. applied Mean Field Annealing (MFA) [20] which combines the characteristics of the simulated annealing and the Hopfield neural network [1] to the m-way graph partitioning problem where the weighted graph is not used.

In this paper, a novel neural network for the m-way graph partitioning problem is proposed where the maximum neuron model [5] is used as an input/output function. Here, more than three-way graph partitioning is considered. The proposed method was compared with PSGA. The reasons to compare the proposed method with PSGA is that (1) PSGA gives better solutions than SA in terms of both the solution quality and the computation time and is compared with other methods, and (2) PSGA has been experimented with the undirected graph with weighted node and weighed edges for the m-way graph portioning where m is more than three. The experimental result shows that the proposed method can generate better or comparable solutions with PSGA in terms of the solution quality and the computation time.

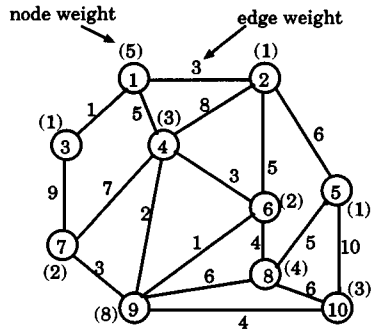


Fig. 1 10 nodes graph.

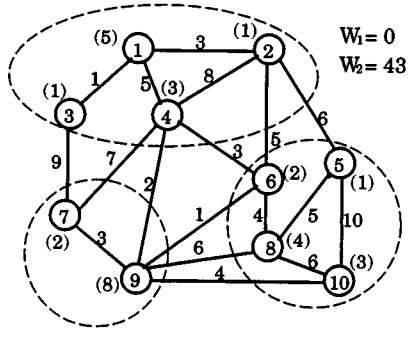


Fig. 2 Partitioned graph.

2. Problem Formulation

Let $G = (V, E)$ be a finite node-weighted, edge-weighted, undirected graph where V, $|V| = n$, is the set of nodes, and E is the set of edges, $E \subseteq V \times V$. The edge from node v to node u is represented by $e_{v,u} \in E \ \forall u, v | u, v \in V$. $w(v) \in Z^+ \ \forall v | v \in V$ defines weights on nodes where Z^+ is the set of positive integers. $l(e_{v,u}) \in Z^+$ defines weights on edges whose endpoints are v and u. Figure 1 shows an example of the undirected 10 nodes graph with weighted nodes and weighted edges.

The m-way graph partitioning problem is to find a partition of V into m subsets $V_1, V_2, \ldots, V_m$ such that $V_1 \cup V_2 \cup \ldots \cup V_m = V$ and $V_i \cap V_j = \phi$ for $i \neq j$, where m is the number of subsets.

Let $S(i)$ be the size of the subset defined as $\sum_{v \in V_i} w(v)$ and $C_{i,j}$ be the sum of weights on cut edges between two different subsets defined as $\sum_{u \in V_i, v \in V_j} l(e_{u,v})$ for $i \neq j$. The cut edge is an edge which has its endpoints in different two subsets. The objective of GPP is to minimize the sum of weights on cut edges among subsets W_2,

$$W_2 = \sum_{1 \leq i < j \leq m} C_{i,j} \tag{1}$$

with minimizing the imbalance W_1 of the size among subsets,

$$W_1 = \sum_{1 \leq i < j \leq m} |S(i) - S(j)| \tag{2}$$

Figure 2 shows one of the solutions partitioned into 3 subsets. The size of each subset is balanced completely.

In [18], the total cost is defined. This cost reflects the goal of the optimization defined by the following equation:

$$\text{Cost} = \lambda_1 W_1 + \lambda_2 W_2 \tag{3}$$

λ_1, λ_2 are weight parameters. In this paper, $\lambda_1 = 1$ and $\lambda_2 = 1$. The solution quality of the proposed method is evaluated with this total cost to compare with the solution by PSGA method.

3. Neuron Model and Neural Representation

Figure 3 shows a neural representation for GPP. $m \times n$ neurons are used where n is the number of nodes and m is the number of subsets respectively. This neural representation is for 10 nodes graph of Fig. 1. Each black square shows that the output of i, jth neuron is one, $V_{i,j} = 1$. This means that jth node is in the ith subset. The white square shows that the output of the i, jth neuron is zero, $V_{i,j} = 0$. This means that the jth node is not in the ith subset. The maximum neuron model is adopted as a input/output function where one and only

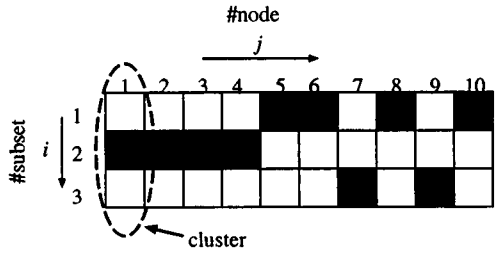


Fig. 3 Neural representation.

one output of the neuron with the maximum input in a cluster is one. In this problem, the jth cluster represents the group of neurons in the jth column. For example, the oval in Fig. 3 shows the first cluster. The cluster is called the maximum neuron. The input/output function of the jth maximum neuron is given:

$$V_{i,j} = \begin{cases} 1, & U_{i,j} = \max(U_{x,j}) \text{ for } x = 1, \ldots, m \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$U_{\max}$ and $U_{\min}$ are the upper limit of the input U and the lower limit of the input U respectively.

The interconnection between the ith neuron and other neurons are determined by the motion equation. The motion equation of the ith neuron is generally given by:

$$\frac{dU_i}{dt} = -\frac{\partial E(V_1, V_2, \ldots, V_n)}{\partial V_i} \tag{5}$$

where E is the computational energy function. E can be obtained by:

$$E = \int dE = -\int \frac{dU_i}{dt} dV_i \tag{6}$$

The goal of the neural network for solving optimization problems is to minimize the computational energy function E. Whatever E is given, the motion equation forces it to decrease. The convergence theorem/proof of the neural network are given in [2]. It is usually easier to obtain the motion equation than the energy function. The motion equation can be obtained by considering the necessary and sufficient constraints and/or the cost function from the given problem. The way to build a motion equation is important for solving optimization problems in neural computing.

4. Motion Equation

The motion equation for GPP is given by:

$$\begin{aligned} \frac{dU_{i,j}}{dt} = & -A\left(\sum_{k=1}^{n} w_k V_{i,k} - \frac{\sum_{k=1}^{n} w_k}{m}\right) \\ & -B\left(\sum_{p=1, p\neq i}^{m} \sum_{q=1, q\neq j}^{n} d_{j,q} V_{p,q}\right) \\ & +C\left(\sum_{k=1}^{n} d_{j,k} V_{i,k}\right) \\ & +Dh\left(\sum_{k=1}^{n} V_{i,k}\right) \end{aligned} \tag{7}$$

A, B and C are constant parameters. The first term is for balancing the size of each subset. w_x is the weight on xth node. The ideal size is the mean size, $\frac{\sum_{k=1}^{n} w_k}{m}$, then the size of each subset is balanced completely. If the size of the subset in the jth subset is larger than the mean size, then the output of the i,jth neuron is suppressed to be zero and if the size is smaller than the mean size, then the output of i,jth neuron is forced to be one.

In the second term, the sum of weights on cut edges between the jth node in the ith subset and other nodes in other subsets is computed. $d_{x,y}$ is the symmetric weight matrix of the edge. For example, the weight matrix $d_{x,y}$ for Fig. 1 is given by:

$$d_{x,y} = \begin{pmatrix} 0 & 3 & 1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 8 & 6 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 & 0 & 3 & 7 & 0 & 2 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 10 \\ 0 & 5 & 0 & 3 & 0 & 0 & 0 & 4 & 1 & 0 \\ 0 & 0 & 9 & 7 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 5 & 4 & 0 & 0 & 6 & 6 \\ 0 & 0 & 0 & 2 & 0 & 1 & 3 & 6 & 0 & 4 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 6 & 4 & 0 \end{pmatrix}$$

The output of the i,jth neuron is suppressed to be zero in proportion to the sum of weights on cut edges.

In the third term, the sum of weights on edges which are not cut, in other words, whose endpoints from the jth node are in the same subset, is computed. The output of the i,jth neuron is forced to be one in proportion to the sum of weights on edges which are not cut.

The fourth term forces the output of the i,jth neuron to be one when all outputs of the neurons in the ith subset are zeros because at least one node must be in each subset. The function $h(X)$ in the fourth term is given by:

$$h(X) = \begin{cases} 1, & \text{if } X = 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

This term is called "Hill-Climbing Term" [2]. To make this term more effective, D is given by:

$$D = \begin{cases} c, & \text{if } t \text{ modulo } 10 < a \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

where a and c are constants and t represents the number of iteration steps.

Data of the weights on both nodes and edges is normalized in the range $(0, 1]$.

In the motion equation, the size of subsets and the sum of weights on cut edges are computed locally. To compute the total cost of the partitioning, W_1 and W_2 are computed by the following equations:

$$W_1 = \sum_{1 \leqq x < y \leqq m} \left| \sum_{k=1}^{n} w_k V_{x,k} - \sum_{k=1}^{n} w_k V_{y,k} \right| \tag{10}$$

$$W_2 = \sum_{x=1}^{n} \sum_{y=x+1}^{n} \sum_{z=1}^{m} \sum_{k=1,k\neq z}^{m} d_{x,y} V_{z,x} V_{k,y} \tag{11}$$

5. Algorithm

The following procedure describes the proposed algorithm. Note that *t_limit* is the maximum number of iteration steps for the system termination condition and *target_cost* is the target total cost set by a user as an expected total cost.

1. Set $t = 0$ and set $A, B, C, t\_limit, target\_cost, U_{\max}$, $U_{\min}$, a and c.
2. The initial values of $U_{i,j}(t)$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$ are randomized. The range for $U_{i,j}(t)$ is from 0 to 10.
3. For $j = 1, \ldots, n$,
   a. Evaluate $V_{i,j}(t)$ for $i = 1, \ldots, m$, using Eq. (4).
4. Compute the total cost, using Eqs. (10), (11). If the total cost is less than *target_cost*, then terminate this procedure.
5. Increment t by 1.
6. For $j = 1, \ldots, n$,
   a. Compute Eq. (7) to obtain $\Delta U_{i,j}(t)$ for $i = 1, \ldots, m$:
   $$\Delta U_{i,j}(t) = \frac{dU_{i,j}}{dt} \tag{12}$$
   b. Update $U_{i,j}(t+1)$ for $i = 1, \ldots, m$, based on the first-order Euler method:
   $U_{i,j}(t+1) = U_{i,j}(t) + \Delta U_{i,j}(t)$
   c. Evaluate $V_{i,j}(t+1)$ for $i = 1, \ldots, m$, using Eq. (4).
7. Compute the total cost, using Eqs. (10), (11). If

Table 1 Experimental result.

Node	Degree	Edge	m	Neural Network				PSGA			
				W_1	W_2	*Cost*	CPU time (s)	W_1	W_2	*Cost*	CPU time (s)
50	4	98	4	3	163	166	0.02	3	263	266	4.05
			6	17	230	247	0.88	25	326	351	3.58
			8	27	293	320	0.22	19	363	382	3.45
50	8	198	6	5	620	625	0.33	5	790	795	2.15
			8	41	685	726	0.01	19	827	846	2.20
			10	61	747	808	1.87	33	878	911	4.75
50	12	312	4	7	887	894	1.02	19	1044	1063	3.10
			8	65	1169	1234	1.60	7	1359	1366	5.13
			12	159	1318	1477	3.10	11	1490	1501	5.07
100	8	398	6	82	1928	2010	0.45	18	2934	2952	7.02
			8	134	2209	2343	3.58	26	3139	3165	8.55
			10	236	2287	2523	1.28	70	3228	3298	4.13
			15	406	2735	3141	4.97	146	3477	3623	8.20
100	12	519	6	60	3285	3345	2.10	30	4470	4500	3.85
			8	96	3653	3749	0.05	118	4767	4885	8.63
			10	124	3947	4071	3.42	82	5024	5106	11.55
			15	444	4445	4889	0.63	146	5317	5463	12.37
100	16	819	6	18	5024	5042	0.80	50	6477	6527	5.33
			8	138	5458	5596	3.58	54	6915	6969	2.77
			10	166	5813	5979	3.62	58	7219	7277	5.00
			15	398	6411	6809	2.00	100	7620	7720	5.17
150	8	594	6	106	2742	2848	3.57	58	4478	4536	15.16
			8	120	3197	3317	1.70	80	4841	4921	15.80
			10	324	3376	3700	6.26	98	4955	5053	13.13
			15	440	3885	4325	4.78	108	5270	5378	15.58
150	12	894	6	126	4976	5102	4.00	22	7146	7168	16.23
			8	78	5511	5589	4.18	82	7594	7676	16.28
			10	232	5916	6148	1.03	122	7906	8028	12.65
			15	434	6600	7034	11.90	134	8327	8461	13.81
150	16	1193	6	74	6922	6996	4.58	88	9483	9571	12.63
			8	72	7624	7696	3.78	78	10096	10174	17.43
			10	230	8087	8317	2.47	48	10541	10589	13.38
			15	526	8913	9439	13.51	246	10977	11223	7.88

the total cost is less than *target_cost*, then terminate this procedure.

8. If $t = t\_limit$ then terminate this procedure else increment t by 1 and go to the step 6.

The maximum neurons have been also studied by Takenaka et al. [19].

6. Experimental Results

The proposed algorithm is implemented in C Language on DEC Alpha Station $200^{4/166}$ workstation.

The proposed algorithm was experimented with randomly generated graphs [13] with weights on nodes and edges. Edges of the random graph are generated with the probability p $(0 < p < 1)$. The expected average degree of the node is given by $p(n-1)$. In the experiment, 50, 100, 150 nodes graphs with different degrees are used. Integer numbers are given randomly on both nodes and edges as weights. The weights of the range 1–10 were given for 50 nodes graphs and the weights of the range 1–20 were given for 100, 150 nodes graphs. These are almost same conditions as ones in [18].

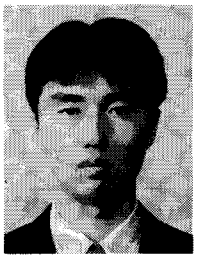
Table 1 depicts the result of the experiment of both the proposed algorithm and PSGA. A, B, C in Eq. (7), *t_limit* and *target_cost* are set to 2.0, 1.0, 1.0, 1000 and 0 respectively. a and c in Eq. (9) are set to 4 and 4 respectively. The same parameters in [18] are used for PSGA: the number of the population is 200 and the number of the generation is 100. Table 1 shows the minimum total cost obtained in the experiments. These results show that the proposed algorithm can generate better solutions in terms of the total cost and the computation time. The proposed algorithm can minimize W_2 compared with PSGA. On the other hand, PSGA can minimize W_1 compared with the proposed algorithm. However, W_1 of the solution by the proposed algorithm is almost equal or smaller than that of the solution by PSGA when the number of subsets is small.

7. Conclusion

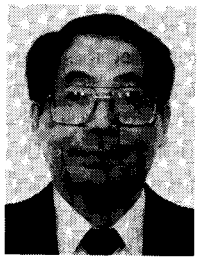
We have proposed a new neural network algorithm for solving the m-way graph partitioning problem where the maximum neuron model is used as a input/output function. The experimental result shows that the proposed neural network can generate better or at least comparable solutions in terms of both the total cost and the computation time compared with PSGA. The proposed method can generate better solutions, especially when minimizing the sum of weights on cut edges is more important than balancing the size of each subset or when the number of subsets is not so large. For future work, the imbalance cost W_1 has to be improved compared with PSGA.

References

[1] J.J. Hopfield and D.W. Tank, "'Neural" computation of decisions in optimization problems," Biological Cybernetics, vol.52, pp.141–152, 1985.
[2] Y. Takefuji, "Neural Network Parallel Computing," Kluwer Academic Publishers, 1992.
[3] Y. Takefuji and J. Wang, "Neural Computing for Optimization and Combinatorics," World Scientific Publishing, 1996.
[4] Y. Takefuji and K.C. Lee, "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics," Biol. Cybern, vol.64, pp.353–356, 1991.
[5] Y. Takefuji, K. Lee, and H. Aiso, "An artificial maximum neural network: A winner-take-all neuron model forcing the state of the system in a solution domain," Biological Cybernetics, vol.67, pp.243–251, 1992.
[6] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," The Bell System Technical Journal, pp.291–307, Feb. 1970.
[7] C-H Lee, C-I Park, and M Kim, "Efficient algorithm for graph-partitioning problem using a problem transformation method," Computer-aided Design, vol.21, no.10, pp.611–618, Dec. 1989.
[8] L. Tao and Y.C. Zhao, "Effective heuristic algorithms for VLSI circuit partition," IEE Proceedings-G, vol.140, no.2, pp.127–134, April 1993.
[9] A.K. Sarje and G. Sagar, "Heuristic model for task allocation in distributed computer systems," IEE Proceedings-E, vol.138, no.5, pp.313–318, Sept. 1991.
[10] L.A. Sanchis, "Multiple-way network partitioning," IEEE Trans. Comput., vol.38, no.1, pp.62–81, Jan. 1989.
[11] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H. Freeman, New York, 1979.
[12] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by simulated annealing," Science, vol.220, pp.671–680, May 1983.
[13] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning," Operations Research, vol.37, no.6, pp.865–892, Nov.-Dec. 1989.
[14] F. Glover, "Tabu search: A tutorial," Interfaces, vol.20, no.4, pp.74–94, July-Aug. 1990.
[15] K. Fujisawa, M. Kubo, and S. Morito, "An application of tabu search to the graph partitioning problem," T.IEE Japan, vol.114-C, no.4, pp.430–437, 1994.
[16] E. Rolland, H. Prikul, and F. Glover, "Tabu search for graph partitioning," Annals of Operations Research, vol.63, pp.209–232, 1996.
[17] H. Pirkul and E. Rolland, "New heuristic solution procedures for the uniform graph partitioning problem: Extensions and evaluation," Computers Ops. Res., vol.21, no.8, pp.895–907, 1994.
[18] I. Ahmad and M.K. Dhodhi, "On the m-way graph partitioning problem," The Computer Journal, vol.38, no.3, pp.237–244, 1995.
[19] Y. Takenaka, N. Funabiki, and S. Nishikawa, "Maximum neural network algorithm for N-queen problems," Transactions of Information Processing Society of Japan, vol.37, no.10, pp.1781–1788, Oct. 1996.
[20] D.E. Van Den Bout and T.K. Miller, III, "Graph partitioning using annealed neural networks," IEEE Trans. Neural Networks, vol.1, no.2, pp.192–203, June 1990.



Takayuki Saito is a Ph.D. candidate of Graduate School of Media and Governance, Keio University. He received the MS degree from Keio University in 1997. His research focusses on neural network computing for combinatiorial optimization problems. He is a member of Information Processing Society of Japan and The Institute of Electronics, Information and Communication Engineers.



Yoshiyasu Takefuji is on the tenured faculty of Electrical Engineering at Case Western Reserve University since 1988 and also on the faculty of Keio University since 1992. His research focusses on neural computing in solving real-world problems. He is interested in VLSI applications and silicon architecture. He received the National Science Foundation/Research Initiation Award in 1989 and received the distinct service award from IEEE Trans. on Neural Networks in 1992 and has been an NSF advisory panelist. He has received the Information Processing Society of Japan's best paper award in 1980, the TEPCO research award in 1993/1994/1995 respectively, the Kanagawa Academy of Science and Technology research award in 1993/1994/1995 respectively, and the Takayanagi research award in 1995. He was an Editor of the Journal of Neural Network Computing, associate editor of IEEE Trans. on Neural Networks, Neural/parallel/scientific computations, and Neurocomputing, and an guest editor of Journal Analog Integrated Circuits and Signal Processing in the special issue on analog VLSI neural networks and also guest editor of Neurocomputing in the special issue on neural network optimization. He is currently an associate editor of the international journal of multimedia tools and applications and also a guest editor of the special issue on multimedia projects at various governmental agencies.