An Efficient Genetic Algorithm Based Approach for the Minimum Graph Bisection Problem

Zhi-Qiang Chen, Rong-Long WANG and Kozo OKAZAKI

Faculty of Engineering, University of Fukui, Bunkyo 3-9-1,Fukui-shi, Japan 910-8507

Summary
The goal of the minimum graph bisection problem is to divide the vertices set V of the given undirected graph $G=(V,E)$ into two equal-size subsets V_1 and V_2 such that the number of edges connecting vertices in V_1 to vertices in V_2 is minimized. In this paper, we propose an efficient method based on genetic algorithm with conditional genetic operators. In the proposed approach, the genetic operators are performed conditionally instead of probably to make sure the algorithm has good global search and local search ability; furthermore, a selection method combining roulette selection with tournament selection is presented to reinforce the local search ability. The proposed approach is tested on a large number of instances and is compared with other optimization methods. The experimental results show that the proposed approach is superior to its competitors.

Key words:
Minimum graph bisection problem, Genetic algorithm, combinatorial optimization problems, NP-complete problem

1. Introduction

The graph bisection problem is an important problem in printed circuit board layout and communication networks [1] [2]. It was proved to be a NP-complete problem [3]. It is well known that there are no tractable algorithms to solve NP-complete problem, which motivates to find better algorithms that yield better approximate solutions. Several approximate algorithms have been proposed [4] [5] [6] for the minimum graph bisection problem. The benchmark algorithm for graph bisection problem is due to Kernighan and Lin [7]. By combining Kernighan-Lin algorithm [7] with the parallel hill climbing [8], and the seed-growth algorithm [9], Marks et al. proposed a new heuristic algorithm called PHC/SG+KL [10]. We have proposed an artificial neural network [11] based algorithm for the problem [12]. For solving such combinatorial optimal problems, the Genetic algorithm (GA) [13] also constitutes an important avenue. GA is an adaptive search technique based on the principles and mechanisms of natural selection from natural evolution. In GA, there is no rule of thumb to design the GA operators and select GA parameters [14]. Instead, trial-and-error has to be applied for every problem. Besides, due to the poor local search ability [15], the solution quality of GA is not very good. Some modified versions of genetic algorithm [16] [17] [18] have been proposed to improve the GA's performance on combinatorial optimization problems.
In this paper, we propose an efficient approach based on our previously proposed genetic algorithm with conditional genetic operators [18] for the minimum graph bisection problem. In the proposed approach, the genetic operators are performed conditionally instead of probably and roulette selection and tournament selection is combined to reinforce the search ability of the algorithm. We test proposed method by simulating a large number of randomly generated graphs. We also analyze the influence of the main parameters. We compare the simulation results with other kinds of optimization method including the PHC/SG+KL [10], the neural network based method [12]. The experimental results show that the proposed genetic algorithm approach works remarkably well and is superior to its competitors to solve the minimum graph bisection problem.

2. Problem Formulation

Given an undirected graph $G=(V,E)$, where V is the set of vertices and E is the set of edges. The edge from vertex i to vertex j is represented by e_{ij}. The minimum graph bisection problem is to find a partition of V into two non-empty, disjoin sets V_1 and V_2, such that $V_1 \cap V_2 = \phi$, $V_1 \cup V_2 = V$, $|V_1| = |V_2|$, and the number of the edges connecting vertices in V_1 to vertices in V_2 (The size of cut set, notated cut (V_1, V_2)) is minimized. For solving such problem, a typical genetic algorithm requires two things to be defined firstly: one is a genetic representation of the solution domain; another one is a fitness function to evaluate the solution domain. For a given graph with N vertices and M edges, as a graph bisection problem, its

Manuscript received June 5, 2008.
Manuscript revised June 20, 2008.

vertex set can be represented using vector $\vec{X} = (x_1, x_2, ..., x_N)$, where x_i $(i = 1, ... N)$ expresses $\# i$ vertex is partitioned into the subset V_1 or V_2. Thus, x_i has only two possible values and a solution of the graph bisection problem can be represented as a binary string. A value of 1 for x_i implies that $\# i$ vertex is partitioned into subset V_1, and a value of 0 denotes that it is partitioned into subset V_2. Using the above representation, the graph bisection problem can be mathematically transformed into the following optimization problem:

$$Minimize \sum_{ij} | x_i - x_j | \cdot e_{ij} \tag{1}$$

$$Subject(\frac{N}{2} - \sum_{i=1}^{N} x_i)^2 = 0 \tag{2}$$

Where N is the number of the vertexes, e_{ij} is 1 if there is an edge from vertex $\# i$ to vertex $\# j$ otherwise 0. Then the fitness function can be expressed as following:

$$F(x) = EdgeNum - A \sum_{ij} | x_i - x_j | . e_{ij} + B(\frac{N}{2} - \sum_{i=1}^{N} x_i)^2 \tag{3}$$

where $EdgeNum$ is the number of edge of the given graph, A and B is parameters.

3. Proposed genetic algorithm based approach

3.1 Genetic algorithm with conditional genetic operators

In our genetic algorithm approach, the crossover and mutation behavior is based on our previously proposed genetic algorithm with conditional genetic operators [18]. The detail is as depicted in Fig.1, where N_p denotes the population size and N_c denotes the current number of children generated. First Nc is set to zero, then (Nc - Np)/2 pairs of parent chromosomes are randomly selected, and the *difference-degree* d_i for every pair of parent chromosomes are calculated. The *difference-degree* d_i is defined as follows:

$$d_i = \frac{N_d}{N_g} \tag{4}$$

Where N_g is the size of chromosome, and N_d is the number of the different genes between two parent chromosomes. As an example, we consider the following two parent chromosomes for a graph with 6 vertices:

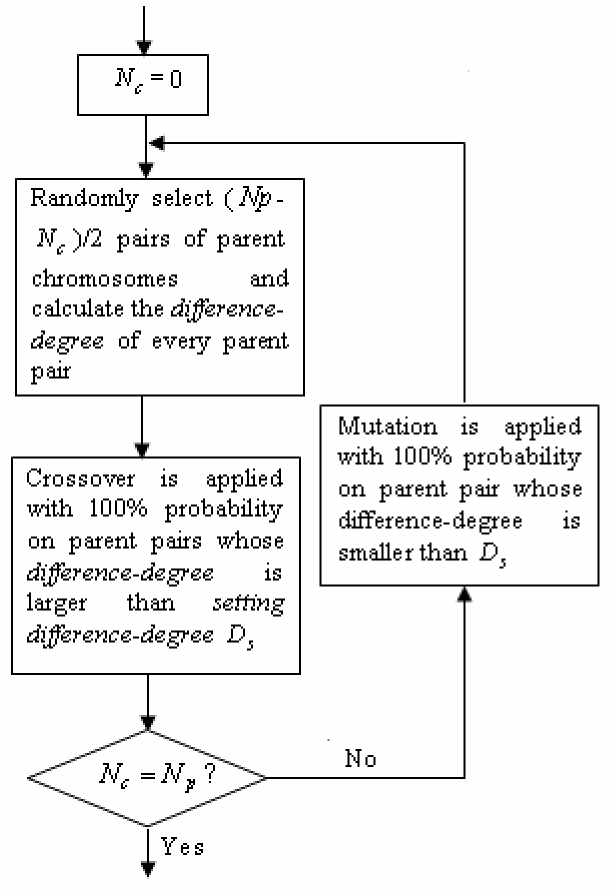
$N_c = 0$
Randomly select (Np - N_c)/2 pairs of parent chromosomes and calculate the *difference-degree* of every parent pair
Crossover is applied with 100% probability on parent pairs whose *difference-degree* is larger than *setting difference-degree* D_s
Mutation is applied with 100% probability on parent pair whose difference-degree is smaller than D_s
$N_c = N_p$?
No
Yes

Fig.1 Outline of crossover and mutation behavior of the proposed genetic algorithm

$$\vec{x} = (1,0,0,1,1,1)$$
$$\vec{y} = (0,1,01,0,1) \tag{5}$$

It is evident that $N_d = 3$ and thus the difference-degree (d_i) of the parent pair is 0.5.

To control the crossover and mutation behavior, a new parameter called *setting difference-degree* D_s is introduced. As shown in Fig.1, for every parent pair, if the *difference degree* (d_i for $\# i$ parent pair) is larger than the *setting difference-degree* D_s, then the crossover is applied with 100% probability on the parent pair to generate two children.

After crossover, we calculate the total number of children generated, and if the total number of children (N_c) is smaller than the population size N_p , the mutation is performed with 100% probability on chromosomes of parent pairs whose *difference-degree* are smaller than the *setting difference- degree* D_s . The mutation operators used in this work indirectly produce children to next generation.

The above procedure is performed in a loop until the total number of children is equal to the population size N_p. The *setting difference-degree* D_s is decreased in every generation and is defined as follows:

$$D_s(t+1) = \mu D_s(t) \qquad (6)$$

Where t denotes the *t#* generation and μ is a constant variation called cooling ratio, 0< μ <1. From the Eq.6, it can be seen that D_s decreases slowly to a near zero with the evolution of generation.

From the above crossover and mutation behavior, we can know that in the early stage of optimal process D_s has a relatively large value, thus many parent pairs have to undergo diverse mutation to produce the enough offspring by crossover. As a result, it makes sure the diversity of the solution and the efficient global search ability of the algorithm. On the other hand, in the late stage of optimal process, D_s become small and even equal approximately to zero, only few parent pairs perform mutation and most of parent pairs produce directly offspring by crossover. As a result, it makes sure the algorithm has good local search ability in the late stage of optimal process. Thus, the crossover and mutation behavior is controlled by *setting difference-degree* D_s dynamically instead of statically so that the ability of local and global search of the GA is balanced efficiently.

3.2 Selection method

For further reinforcing the local search ability of the proposed approach, we design a selection method which inherits the advantage of roulette selection and tournament selection. The proposed selection has two phases. The first phase is to create the candidate population whose individual is stochastically selected from the current generation using roulette selection, where the current generation is expressed as $P_{cur} = \{a_1, a_2, ...a_n\}$, and $a_i (i = 1,2,...,n)$ expresses the $\# i$ individual of the current generation, n is the size of population. The candidate population selected in the first phase is expressed as $P_{cd} = \{b_1, b_2,..., b_n\}$. And $b_i (i = 1,2,...,n)$ express the *#i* individual (chromosome) of the candidate population. The second phase decides if the individual (in the candidate population) with low fitness survives in the next generation or not. The detail of the second phase is as follows:

(1) Randomly pair the individual between P_{cur} and P_{cd}. We use $< a_i$, $b_i >$ to express the $\# i$ pair.

(2) Compare the fitness of individual for each pair. We use $f(a_i)$ to expresses the fitness of a_i .If $f(b_i) > f(a_i)$ then b_i survive in the next generation. Else then a_i replaces b_i and survives in the next generation by probability *P*. The probability *P* is defined by the following equation:

$$P = 1 - \mu^t \qquad (7)$$

Where t denotes the *t#* generation and μ is the same one as that in Eq.6. In other words, the individual (in the current generation) with better than its competitor survive in the next generation by probability *P*.

The proposed selection method inherits the advantage of roulette selection and tournament selection. In addition, it does not require global reordering. Selection pressure can be easily adjusted by changing probability *P*. With the evolution of population, *P* is gradually increasing and selection pressure also is gradually increasing with *P*. In the early stage, because *P* has rather smaller value, the diversity of the population isn't destroyed and as evolution of generation the local search ability of the proposed approach is reinforced.

3.3 Crossover and mutation method

In GA, besides the behavior of crossover and mutation, the methods of crossover and mutation are also very important for efficient search. In this section, we design the crossover and mutation methods. Our fundamental rule of designing crossover and mutation methods is that children generated are valid solutions.

One point-crossover or its extension (multipoint crossover) [19] is well-used and efficient methods for the problem with bit string encodings because of its simplification. However, for the graph bisection problem, the invalid solutions are also produced if performing the traditional one point-crossover or multipoint crossover. We now present a modified two-point crossover which can always produce valid solution. The detail is as follows:

(1).Randomly select two crossover point CP_1 and CP_2 $(CP_1 < CP_2)$.

(2).Use S_1 to express the number of the gene that value is 1 in parent 1, and S_2 in parent 2. Adjust the crossover point CP_1 and CP_1 according to the following procedure:

```
While ( $S_1 \neq S_2$ )
    (1) $CP_1 = CP_1 - 1$ or $CP_2 = CP_2 + 1$
    (2) Calculate $S_1$ and $S_2$
End
```

(3). Interchange the segment between CP_1 and CP_1 to generate two valid children.

Besides the crossover method, we also modify the simple-invert mutation [13] method to guarantee the feasibility and valid of the solution generated by mutation operator as follow:

(1).Randomly select two mutation point Mp_1 and

Table 1: Simulation results on the graph with 150 vertexes, 558 edges using different parameters.

Cooling ration μ	N_p = 50		N_p =80		N_p =100		N_p =150	
	best	Average.	best	Average	Best	Average	Best	Average
0.999	139	149	139	146.65	141	146.17	139	145.35
0.9991	141	149.31	139	146.39	139	146.25	139	145.26
0.9993	139	148.53	141	147.04	141	145.91	139	144.86
0.9995	142	148.29	139	145.53	139	145.86	139	144.34
0.9997	139	147.85	139	146.49	139	145.8	139	144.34
0.9999	139	148.02	139	145.87	139	145.3	139	144.59
0.99991	142	147.3	139	146.15	139	145.19	139	143.8
0.99993	141	147.53	139	145.37	139	145.25	139	144.6
0.99995	139	147.73	139	145.1	139	145.22	139	144.51

Mp_2 for a chromosome.

(2). Interchange the gene of # Mp_1 and # Mp_2 .

It is evident that the above crossover and mutation method can always produce valid child. Thus, the fitness of a solution can be evaluated according to the following equation simplified from Eq.3

$$F(x) = EdgeNum - \sum_{ij} | x_i - x_j | \cdot e_{ij} \qquad (8)$$

4. Algorithm

The following procedure describes the proposed approach for the bisection problem of *N*-vertex graph:

1: Set the parameter Np , D_s and μ , and generate Np valid initial chromosomes (solutions).

2: Produce Np child chromosomes according to following sub-procedure:

(1) Set N_c =0.

(2) Randomly select (Np - N_c)/2 pairs of parent chromosomes.

(3) Calculate the *difference-degree* d_i of every parent pair according to Eq.4

(4) Crossover is applied on parent pairs whose *difference-degree* d_i is larger than D_s .

(5) Calculate the total child chromosomes N_c , if $N_c = Np$ terminate this sub-procedure.

(6) Mutation is applied on parent pairs whose *difference-degree* d_i is smaller than D_s .

(7) Go to step (2)

3. Calculate the fitness of each chromosome according to Eq.8.

4: Decrease the *setting difference-degree* D_s using Eq.6

5: Terminate this procedure if D_s <0.01.

6: Create the chromosomes of next generation using our improved selection explained in the last section.

7: Go to step 2.

5. Simulation results and analysis

The proposed approach has three main parameters: population size Np , initial value of D_s and its cooling ratio μ . To analyze the influence of the main parameters, we simulated an undirected graph with 150 vertices 558 edges using different parameters. To measure the performance of the proposed approach, a total 18 graphs was simulated.

As mentioned, the cooling ratio μ is used to control the decreased speed of D_s in Eq.6 and the increased speed of P in Eq.7. In other word, it is used to balance the ability of global search and local search of the algorithm. If μ is too

Table 2 Simulation results on the graph with 150 vertexes, 558 edges using different population size

Population size N_p	Best value	Average value	Error%
10	144	154.64	11.25
20	142	150.94	8.59
30	142	150.12	8.0
40	141	148.89	7.12
50	141	148.01	6.48
60	141	146.86	5.65
70	139	145.68	4.81
80	139	145.65	4.78
90	139	145.39	4.60
100	139	145.5	4.68
110	139	144.94	4.27
120	139	144.77	4.15
130	139	144.62	4.04
140	139	144.31	3.82
150	139	144.04	3.62
160	139	144.19	3.73
170	139	143.66	3.35
180	139	144.14	3.70
190	139	144.13	3.60
200	139	144.13	3.60

large, D_s decreases too slowly and P increases too slowly, the algorithm would spend too long computation time. If μ is too small, the algorithm would converge quickly and lose the ability of global search for global optimal solution. To study the reasonable range of the value of the cooling ratio μ, we simulated an undirected graph with 150 vertexes and 558 edges. The experiments are classified into four groups by population size Np =50, Np =80, Np =100, Np =150. The value of the cooling ratio μ is 0.999 to 0.99995. The initial value of D_s is set to 0.4. 100 times simulations with different initial chromosomes have been performed for every parameters group. The best solution and average solution in 100 simulations are recorded. The simulation results are described in the Table 1. As table 1 shown, when the cooling ratio μ is smaller than 0.999, the optimization process is to converge quickly and no good

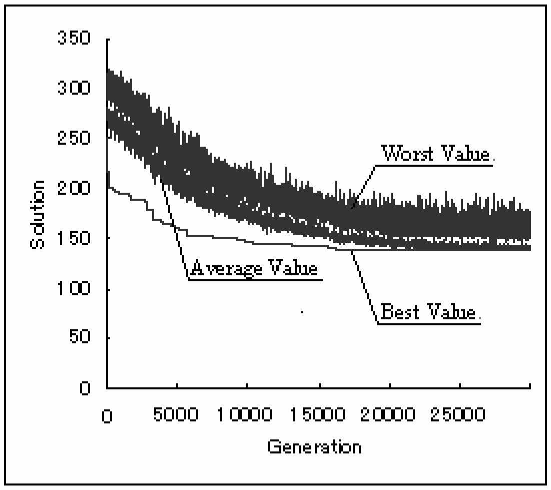

Fig.2. Generating procedure of solution using proposed approach.

solution was found. When the cooling ratio μ is or larger than 0.9999, the algorithm balances efficiently the ability of global search and local search and can find good solutions. We have also performed a large number of simulation to study the initial value of D_s, we found that the reasonable initial value of D_s is from 0.1 to 0.6. In the later
simulations, the cooling ratio μ and the initial value of D_s were fixed at 0.9999 and 0.4, respectively.

Population size Np is also an important parameter which affects the performance of genetic algorithm. To see the influence of Np in the proposed approach, we performed simulations using different population size. Note that the initial value of D_s and the cooling ratio μ are 0.4 and 0.9999 respectively. The experiment results are shown in the table 2. The " $Error\%$ " column corresponds to the average percentage error of solutions obtained with different runs as defined in the following equation.

$$Error\% = \frac{|\sum_i^N (Solution(i) - Best)|}{N} \times \frac{1}{Best} \times 100\% \quad (9)$$

Where *Solution(i)* is the solution found in the $\# i$ run, *Best* is the best solution of the problem. As the table 2 shown, when the population size Np is larger than 60, good solution can be found. In the later simulations, the populat-ion size Np was fixed at 100.

To verify the ability of global search and local converge of the proposed approach, we present the evolution procedure of the solution on the graph with 150 vertices 558 edges using the proposed approach. Fig 2 shows the variation in the best, worst and average solutions during the evolution procedure of the solution. From Fig 2, we can know the characteristic of the proposed approach. In the early stage, there is large difference between the best and worst solutions, which means that in this stage the proposed approach has good diversification. In the final stage, the difference between best and worst solutions become very small, which means that the proposed approach has good local search ability in this stage.

Table 3: The comparisons of simulation results produced by different algorithms

Graph			SGA	PHC/SG+KL[10]	Wang etc.[12]	Proposed algorithm
Vertex	Probability	Edges				
80	0.05	158	26	26	26	26
80	0.15	474	151	151	151	151
80	0.25	790	292	292	292	292
100	0.05	247	50	51	50	50
100	0.15	742	248	247	247	247
100	0.25	1235	473	473	473	473
150	0.05	558	141	139	139	139
150	0.15	1676	608	605	605	605
150	0.25	2790	1119	1113	1113	1113
200	0.05	995	278	274	274	273
200	0.15	2985	1135	1128	1128	1128
200	0.25	4975	2048	2044	2045	2044
250	0.05	1556	478	464	463	463
250	0.15	4668	1833	1823	1820	1820
250	0.25	7778	3307	3272	3271	3271
300	0.05	2242	724	711	712	711
300	0.15	6727	2712	2683	2681	2682
300	0.25	11212	4834	4795	4790	4790

In order to widely verify the proposed approach, we have also tested the algorithm with a large number of randomly generated graphs [20] defined in terms of two parameters, *n* and *p*. The parameter *n* specifies the number of the vertices in the graph; the parameter *p*, $0<p<1$, specifies the probability that any given pair of vertices constitutes an edge. To evaluate our results, we compared our results with the other kinds of optimization method including the PHC/SG+KL proposed by J.Marks.et al.[10], the improved neural network by Wang et al. [12], and the simple genetic algorithm(SGA, the crossover ratio is 0.7, the chromosome mutation ratio is 0.04, the crossover and mutation method is the same as the proposed ones in the paper). Information on the test graphs as well as all results is shown in Table 3. From Table 3 we can know that the proposed approach has more excellent ability than other existing algorithm for solving the minimum graph bisection problem.

6. Conclusions

We have proposed an efficient GA based approach for the graph bisection problem and showed its effectiveness by simulation experiments. In the proposed method, the genetic operators are performed conditionally instead of probably to reinforce the intensification and diversification in different stages. We have analyzed the approach by performing a large number of simulations. The simulation results showed that the proposed approach has good diversification and local search ability. We have also compared the proposed approach with the existing algorithms and found that the proposed approach works remarkable well and is superior to its competitors to solve the minim-um graph bisection problem.

References

[1] L. Tao and Y. C. Zhao, "Effective heuristic algorithm for VLSL circuit partition," IEE Proceedings-G, Vol. 140, No.2, pp.127-134, April, 1993.

[2] L. A. Sanchis, "Multiple-way network partitioning," IEEE Teans. Comput, Vol.38, No.1, pp.62-81, Jan, 1989.

[3] M. R. Garey, D. S. Johnson, and L. Stockmeyer. "Some simplified NP-complete graph problem," Theoretical Computer Science, Vol.1 pp.237-267, 1976.

[4] T. N. Nui, F. T. Leighton, S. Chaudhuri and M. Sipser, "Graph bisection algorithm with good average case behavior," Combinatorica, Vol.7, pp.171-191, 1987.

[5] E. R. Barnes, A. Vannelli and J. Q. Walker, " A new heuristic for partition the nodes of a graoh," SIAM Journal on Discrete Mathematics, Vol.3, pp.299-305, 1988.

[6] C. Tu and H. Cheng, "Spectral methods for graph bisection problem," Computers Ops Res. Vol.25, No.7, pp-519-530, 1998.

[7] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," The Bell System Technical Journal, Vol.49, No.2, pp.291-307, 1970.

[8] C. A. Tovey, "Hill climbing and multiple local optima," SIAM Journal on Algebraic and Discrete Methods, Vol.6 No.3, pp.384-393, 1985.

[9] W. E. Donath, "Logic partitioning," In B. Preas and M. Lorenzetti, Editors, Physical Design Automation of VLSI Systems, pp.65-86, 1988.

[10] J. Marks, W, Ruml, S. Shieber, and J.Ngo, "A Seed-growth Heuristic for Graph Bisection," proceedings of Algorithms and Experiments 98, Italy, R.Battiti and A. Bertossi(eds), pp.76-87, 1998.

[11] J. J. Hopfield, and D. W. Tank, "Neural computation of decisions in optimization problems," Biol. Cybernet. No.52, pp.141-152, 1985.

[12] R. L. Wang, Y. Yamanishi, and K. Okazaki, "A New Neuron Dynamics for Solving the Minimum Graph Bisection Problem," International Journal of Computer Science and Network Security, Vol.7 No.3, pp.55- 58, 2007.

[13] J. H. Holland, "Adaption in Natural and Artificial System," MIT press, Cambridge, MA, 1975.

[14] L. J. Schmitt and M. M. Amini, "Performance characteristics of alternative genetic algorithm approaches to the traveling salasman problem using path representation: an empirical study," European Journal of Operational Research, Vol.108, PP.551-570, 1998.

[15] L. G. Caldas and L. K. Norford, "A design optimization tool based on a genetic algorithm," Automation in Construction, Vol.11, pp.173-184, 2002.

[16] H. Satoh, I.Ono and S.Kobayashi, "A New Generation Alternation Model of Genetic Algorithms and Its Assessment," Journal of Japanese Society for Artificial Intelligence, Vol.22, pp.734-744. 1997.

[17] D. Whitley, R. Beverridge, C. Graves and K. Mathias, "New test functions and geometric matching," Journal of Heuristics, Vol.1, pp.77-104, 1995.

[18] R. L. Wang, "A genetic algorithm for subset sum problem," Neurocomputing, Vol.57, pp.463-468, 2004.

[19] E. Goldberg, "Genetic Algorithm," Eddison-Wesley Publishing Company, INC, pp106-120, 1989.

[20] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part 1, graph partitioning," Operations Research, vol.37, no.6, pp.865-892, 1989.

Zhi-Qiang Chen received a B.S. degree from WuHan University of Water-Conservancy and Electric Power, WuHan, China and an M.S. degree from ChongQing University, ChongQing China in 2001 and 2004 respectively. Since 2004, he has been a software engineer in Hanna Strategies Ltd. Shanghai, China. From 2008 he is working toward the Ph.D degree at Fukui University, Japan. His main research interests are genetic algorithm and optimization problems.

Rong-Long Wang received a B.S. degree from Hangzhe teacher's college, Zhjiang, 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 associate professor in Department of Electrical and Electronics Engineering. His current research interests include genetic algorithm, neural networks, and optimization problems.

Kozo Okazaki received his BS and MS degrees in Electrical Engineering from Hiroshima University, in 1967 and 1969, respectively. From 1969 to 1970 he was an assistant professor of Electrical Engineering in the Faculty of Engineering, Miyazaki University. Since 1991, he is a professor of Electical and Electronics Engineering in the Faculty of Engineering, Fukui University. His current research activities include in the field of image.