DOI: 10.3745/KIPSTB.2008.15-B.4.341

유전 알고리즘을 이용한 최소 무게 삼각화 문제 연구

한 근 희[†] · 김 찬 수[††]

요 약

Minimum Weight Triangulation (MWT) 는 최적화 문제로서 주어진 그래프에 대한 최소 무게 삼각화를 계산하는 문제이다. 본 문제는 많은 다른 그래프 문제들처럼 일반 그래프에 대하여 NP-hard 계열의 문제로 알려져 있으며 지금까지 simulated annealing 및 유전 알고리즘 등 heuristic algorithm 들이 제시되어 왔다. 본 논문에서는 MWT 문제에 대하여 GA-FF 라 불리우는 새로운 유전 알고리즘을 제시하며 또한 그 성능이 기존의 유전 알고리즘보다 더욱 효율적임을 보인다.

키워드 : 최소 무게 삼각화 문제, 유전 알고리즘, 코달 그래프

Solving Minimum Weight Triangulation Problem with Genetic Algorithm

Keunhee Han[†] · Chansoo Kim[††]

ABSTRACT

Minimum Weight Triangulation (MWT) problem is an optimization problem searching for the triangulation of a given graph with minimum weight. Like many other graph problems this problem is also known to be NP-hard for general graphs. Several heuristic algorithms have been proposed for this problem including simulated annealing and genetic algorithm. In this paper, we propose a new genetic algorithm called GA-FF and show that the performance of the proposed genetic algorithm outperforms the previous one.

Key Words : Minimum Weight Triangulation, Genetic Algorithms, Chordal Graphs

1. Introduction

A graph $G = (V, E)$ is called *chordal* if every cycle of length strictly greater than three contains a *chord*, that is, an edge joining two nonconsecutive vertices of the cycle. *Triangulation* of a graph is an embedding of an arbitrary graph G into a chordal graph by adding edges to G.

There are several versions of triangulation problems depend on the parameters of graph properties. For an arbitrary graph $G = (V, E)$, a set of edges F is called a *filled edges* if $G' = (V, E \cup F)$ is chordal and we denote G' the *filled graph*. F is a *minimal triangulation* if $G_0 = (V, E \cup F_0)$ is not chordal for any $F_0 \subset F$. The *minimum triangulation* problem is to find the triangulation of a graph with fewest filled edges. The *treewidth* problem is to find the triangulation of a graph with the size of largest clique minimized. Minimum triangulation has its applications in the field of sparse matrix computations, database management, knowledge based systems, and computer visions [1] while the treewidth problem has its applications in the field of artificial intelligence, database and VLSI design. These two problems have been proved to be NP-hard [2, 3].

A related application of triangulation is also emerged from the field of Bayesian Networks. In Bayesian networks, after the causal networks are transformed into moral graphs by linking all vertices (variables) with a common child, moral graphs, which is now a general graph, must be triangulated in order to facilitate the propagation of evidence. *Minimum Weight Triangulation* (MWT) is a triangulation of a graph G with *minimum weight* (defined later) and a well known main obstacle for constructing efficient Bayesian networks [4].

Since computing optimal MWT is NP-hard [5], any exact algorithm require an exponentially increasing number of steps as the problems become larger. Therefore, the authors in [6] applied Genetic Algorithm (GA) to MWT

† 종신회원 : 공주대학교 응용수학과 부교수
†† 정 회 원 : 공주대학교 응용수학과 조교수(교신저자)
논문접수 : 2008년 2월 14일
수 정 일 : 2008년 4월 1일
심사완료 : 2008년 4월 21일

and showed very interesting results on two test graphs called *Sparse* and *Dense* graphs. For the rest of this paper we call the genetic algorithm proposed in [6] as GA-MWT. In this paper, we develop a genetic algorithm, called GA *Fast Fill* (*GA-FF*), that can be applied to MWT problem and show that the results of GA-FF are more efficient than GA-MWT.

The rest of this paper is organized as follows. In section 2, we introduce the properties of minimum weight triangulation and prove that testing for chordality of $G - e$ can be done efficiently, where G is a chordal graph and e is an any edge of G. Section 3 and 4 contain the properties and the experimental results of GA-FF, respectively. Finally, section 5 contains the conclusions.

2. Minimum Weight Triangulation

2.1. Notations

For a graph $G = (V, E)$ with $|V| = n$, an ordering of V is a bijection $\alpha:\{1, 2, ..., n\} \leftrightarrow V$. For the rest of this paper $G(\alpha)$ denotes the ordered graph with some ordering $\alpha = \{v_1, v_2, ..., v_n\}$ on its vertex set. The *neighborhood* of a vertex v of the graph G, denoted $N(v)$, is the set consisting of all vertices which are adjacent to v. The *closed neighborhood* of a vertex v is defined as $N[v] = N(v) \cup \{v\}$. We say that vertex v is a *neighbor* of vertex w if v is adjacent to w in G. A vertex x of G is called *simplicial* if $N[x]$ induces a complete subgraph of G. A *perfect elimination ordering* (*peo*) of a graph G is an ordering of V with the property that for each i, j and l, if $i < j$, $i < l$, and v_i, $v_j \in N[v_i]$, then $v_l \in N[v_i]$. It is well known that a graph is chordal if and only if it admits a perfect elimination ordering [7]. The *deficiency* of vertex v in G is $D_G(v) = \{(u, x) | u, x \in N(v)$ and $(u, x) \notin E\}$. Note that if v is simplicial $D_G(v) = \phi$. A *clique* in a graph is a set of pairwise adjacent vertices and *maximal clique* of G is a clique and is not contained in any other clique of G.

2.2. Properties of MWT

The algorithm shown in (Fig. 1) [1] is a well known algorithm for the triangulation of graphs.

At each iteration i, since the algorithm forces vertex v_i to be a simplicial in G_i, clearly, the resulting graph $G^+(\alpha)$ is a chordal graph. Therefore, the input ordering α becomes a simplicial ordering of $G^+(\alpha)$.

Let $G = (V, E)$ be a graph and n_i ($< \infty$) denotes the number of states vertex v_i ($\in V$), then the *minimum weight triangulation* problem is to minimize the *weight* of $G(\alpha)$ computed as

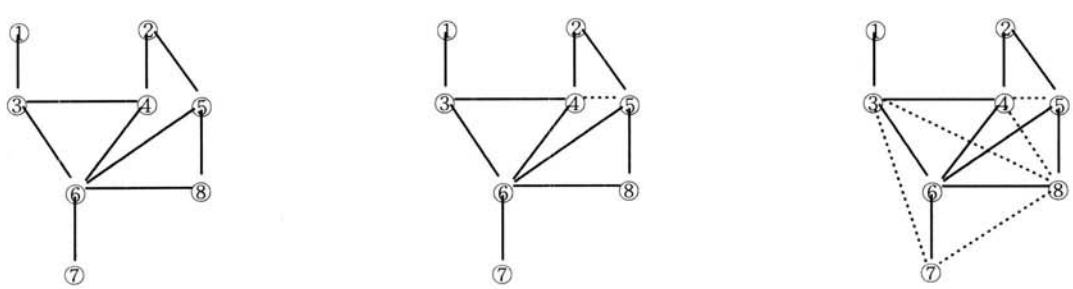
$$W(G(\alpha)) = \log_2 \sum_C \prod_{v_i \in C} n_i , \qquad (2.1)$$

where C is the maximal cliques of filled graph $G^+(\alpha)$ produced by the Elimination Game.

It is easy to see that for the different ordering of α's the algorithm produces different filled graphs; hence possibly different values of $W(G(\alpha))$. For example, if we apply Elimination algorithm with different orderings to the graph shown in (Fig. 2) (a), they produce different values of $W(G(\alpha))$ as shown in (Fig. 2) (b) and (c). Note that the

Algorithm: Elimination Game
Input: A graph $G = (V, E)$ and an ordering $\alpha = (v_1,..., v_n)$ of V.
Output: The filled graph $G^+(\alpha)$
1 $G_0 = G$;
2 for $i = 1$ to n do
3 Let $F^i = D_{G^{i-1}}(v_i)$;
4 Obtain G^i by adding the edges in F^i to G_{i-1} and removing v_i;
5 $G^+(\alpha) = (V, E \cup \bigcup_{i=1}^{n} F^i)$;

(Fig. 1) Elimination Game.



(a) (b) (c)

(Fig. 2) Applications of the algorithm Elimination Game. (a) a graph with 8 vertices. (b) $\alpha 1$ = (1, 2, 3, 4, 5, 8, 6, 7), W(G($\alpha 1$)) = 5.32, (c) $\alpha 2$ = (2, 5, 4, 6, 1, 3, 8, 7), W(G($\alpha 2$)) = 5.91. Dashed lines indicate the filled edges.

graph shown in (Fig. 2(a)) is one of the standard Bayesian networks developed in [8] with $n_i = 2$, $1 \le i \le 8$.

Note also that in (Fig 2.(b)) the maximal cliques are {{1, 3}, {2, 4, 5}, {3, 4, 6}, {4, 5, 6}, {5, 6, 8}, {6, 7}} while the maximal cliques of (Fig. 2(c)) are {{2, 4, 5}, {5, 4, 6, 8}, {4, 3, 6, 8}, {6, 7, 3, 8}, {1, 3}}.

The example in (Fig. 2(c)) clearly shows that the algorithm Elimination Game does not necessarily produces a minimal triangulation. Hence, let $G^+(\alpha) = (V, E \cup F)$ be a filled graph of a graph $G = (V, E)$ where F is the filled edges produced by Elimination Game. For any edge $e \in F$ if $G' = (V, E \cup (F - \{e\}))$ is chordal we say that edge e is *redundant*. For example, in (Fig. 2(c)) the filled edges (4, 8), (3, 8), (3, 7), and (7, 8) are redundant edges while (b) contains no redundant edges. If we remove these redundant edges from (Fig. 2(b)) then $W(G(\alpha_2))$ reduces to 5.32.

Let $G_1 = (V, E \cup F_1)$ and $G_2 = (V, E \cup F_2)$ be two triangulations of a graph $G = (V, E)$ where F_1 and F_2 are the filled edges of G_1 and G_2, respectively. In [9], the authors proved that if $F_1 \subset F_2$, then $W(G_1) \le W(G_2)$ and most often $W(G_1)$ is far less than $W(G_2)$. Therefore, the main idea of GA-FF is to remove as many redundant edges as possible after constructing triangulation of a graph G with the algorithm Elimination Game.

However, since not all the filled edges are redundant, removing redundant edges requires checking for chordality of a graph. *Lexicographic Breadth-First Search* (LexBFS) [7] and *Maximum Cardinality Search* (MCS) [10] are the two best known algorithms for recognizing chordal graphs and both run in time $O(|V| + |E|)$ for a given graph $G = (V, E)$. However, if a graph is dense then $|E| \in O(|V|^2)$, both algorithms are too expensive for genetic algorithms. To overcome these bottlenecks we need to closely examine the neighbors of the edge in question. Let C_n be the chordless cycle of length n (≥ 3).

Theorem 2.1 [7]. Let G be a chordal graph with edge (u, v). Then either $G - (u, v)$ is chordal or $G - (u, v)$ contains a C_4.

Corollary 2.1. Let $G = (V, E)$ be a chordal graph with edge (u, v). Then $G - (u, v)$ is a chordal graph if and only if u, v have no two common neighbors x and y such that $(x, y) \notin E$.

Proof: ($\leftarrow$) Let u, v have no common neighbors x and y such that $(x, y) \notin E$ and suppose that $G - (u, v)$ is not chordal. Then, by theorem 2.1, $G - (u, v)$ contains a C_4. Let $[u, s, v, t]$ be such C_4. However, this is a contradiction since $(s, v) \in E$. ($\rightarrow$) Let $G - (u, v)$ be chordal and suppose that u, v have two common neighbors s and t such that $(s, t) \notin E$. However, this is a contradiction to our assumption that $G - (u, v)$ is chordal since $[u, s, v, t]$ is a C_4 in $G - (u, v)$.

The *adjacency* (0,1)-*matrix* $M = M[i,j]$ of a graph G with n vertices is the $n \times n$ matrix in which $M[i,j] = 1$ if vertex i is adjacent with vertex j and $M[i,j] = 0$ otherwise. Based on Corollary 2.1, the procedure called *isChordal*() shown in (Fig. 3) can be used to check whether or not an edge can be removed while preserving the chordality of a chordal graph G.

Let $\Delta = \max |N(u) \cap N(v)|$ for all $u, v \in V$ of a graph $G = (V, E)$. Assuming that the graph is represented by an adjacency (0,1)-matrix M, constructing the set $S = N(u) \cap N(v)$ can be done in $O(|V|)$ and checking for the existence of nonadjacency among the vertices in S can be done in $O(\Delta^2)$. Therefore, Theorem 2.2 suggests an algorithm that can be used to check if an edge can be removed from a chordal graph while maintaining the chordality with running time in $O(|V| + \Delta^2)$. Theoretically, $\Delta \in O(|V|)$; however, in practice, Δ is much smaller than $|V|$.

```
procedure isChordal(M, (u, v))
1   Let S = N(u)∩N(v)
    // S is set of common neighbors of u and v
2   for i = 1 to |S| - 1
3      for j = i + 1 to |S|
4         if (M[S[i], S[j]] ≠ 1)
5            return false;
6      end for
7   end for
8   return true;
```

(Fig. 3) Procedure isChordal(M, (u, v)), where M is a (0-1) adjacency matrix of G and (u, v) is an edge of G.

3. Genetic algorithm for MWT

As noted in [6], in some sense, MWT is similar to the Travelling Salesman Problem (TSP). In TSP, we search for the optimum order of cities that yields the shortest tour of n cities. In contrast, in MWT, we search for the optimum order of vertices to eliminated that produces the minimum weight. TSP is one of the most widely researched problem in GA community.

Several representations and genetic operators have been developed for TSP with GA. *Path representation* [11] is a permutation of $[n]$, where $[n]$ denote the set of all possible natural numbers not greater than n. GA-FF uses path representation in order to represent the different ordering

α's of a graph G. For the genetic operators we adapt *Cycle* crossover (CX) and *Simple Inversion* mutation (SIM). The mechanisms of these two genetic operators are well known and can be found in [6, 11]. In CX every vertex of the offspring comes from one of the parents. For example, consider the following two parents p_1 and p_2:

$$p_1 = (1\ 2\ 3\ 4\ 5) \text{ and}$$
$$p_2 = (3\ 1\ 2\ 5\ 4).$$

The first vertex of the offspring o_1 takes the first vertex of p_1. Therefore, o_1 becomes (1 * * * *), where * represents "not yet decided". Since the vertex 3 of p_2 is just below the vertex 1 of p_2 we consider vertex 3 of p_2. The vertex 3 is in the third position of p_1; hence o_1 becomes (1 * 3 * *). In this way, the next vertex to be considered must be 2 and o_1 becomes (1 2 3 * *). With this rule the next vertex to be considered must be 1; however, vertex 1 is already on o_1. Therefore, the remaining vertices are filled from the p_2. The final list of o_1 is as follows:

$$o_1 = (1\ 2\ 3\ 5\ 4).$$

Simple inversion mutation (SIM) selects two cut points randomly and reverse the vertices between these two cut points. For example, let c = (1 2 3 4 5) be a chromosome and suppose that the second and fourth positions are selected as the cut points. Then result chromosome is c = (1 4 3 2 5).

We use formula (2.1) as our *fitness function*. For selection we use *roulette wheel* with slots sized according the fitness of each chromosome. *Eliticism* is a variation of simple selection of genetic algorithms. It enforces to preserve the best chromosome found so far in the iteration of the algorithm. Let $P(t)$ be the population at time t. In GA-FF, after selection, if $P(t)$ does not contain the best chromosome *best* of $P(t-1)$, then the worst chromosome of $P(t)$ is replaced by *best*, where $t > 0$.

We showed that, in the previous section, for different ordering α's of the vertices the Elimination Game yield different weights of the graph. Hence, for a given ordering α, GA-FF first execute the Elimination Game on $G(\alpha)$ and try to remove redundant edges from the filled graph $G^+(\alpha)$ using Corollary 2.1. (Fig. 4) and (Fig. 5) show the details of GA-FF and procedure *evaluate_P*(t).

If we replace the *evaluate_P*(t) by a usual evaluation procedure, i.e., does not remove any redundant edges, then GA-FF becomes the same as GA-MWT. Therefore, the major difference between GA-MWT and GA-FF lies on the evaluation of the chromosomes. After we apply

```
Algorithm GA-FF
1  t = 0;
2  initialize population P(t);
3  evaluate_P(t);
4  while not termination-condition do
5   t = t + 1;
6   select P(t);
7   crossover P(t);
8   mutate P(t);
9   evaluate_P(t);
                // see Fig. 5
10 end
```

(Fig. 4) Pseudo code of GA-FF.

```
procedure evaluate_P(t)
1  for each chromosome α of P(t)
2     apply Elimination Game with α to G
3     for each filled-edge e of ∪_{i=1}^{n} F^i
      // filled-edges are selected at random order
4        if isChordal(G+(α), e) then
5           delete edge e from G+(α)
6     end for
7  end for
```

(Fig. 5) Pseudo code of the procedure evaluate_P(t).

Elimination Game to each chromosome in line 2 of procedure *evaluate_P*(t), all the filled-edges produced by line 2 are checked whether or not they can be removed while preserving the chordality of $G^+(\alpha)$ in line 3 and 4 of the algorithm. If the procedure *isChordal*($G^+(\alpha)$, e) of line 4 confirms that e is a redundant edge then line 5 deletes e.

4. Experiments

Two test graphs called *Sparse* and *Dense* graph which contain 50 vertices each, and 100 and 359 edges, respectively, were used to measure the performance of GA-MWT. These two graphs were originally developed by Kjaerulff [9]. For both graphs the number of states were chosen at random between 2 and 5. See [6] for more details about the graphs and the number of states used for testing GA-MWT.

In [6], GA-MWT was executed with numerous number of genetic operators; partially-mapped (PMX), cycle (CX), order (OX1), order-based (OX2), position-based (POS), genetic edge recombination (ER), voting recombination (VR), alternating-position (AP) crossover, and displacement (DM), exchange (EM), insertion (ISM), simple-inversion (SIM), inversion (IVM), scramble (SM) mutation operators. However,

since, in [6], the best results were obtained from the CX for both test graphs we summarize only those results of applying CX in <Table 1 and 2>, respectively. Note that the termination condition used for GA-MWT is based on the definition of convergence of a population formulated by De Jong [12].

<Table 3 and 4> contain the results of applying GA-FF with cycle crossover (CX) and simple inversion mutation (SIM) on Sparse and Dense graphs, respectively. We do not include the results of other combinations of the genetic operators since their performance are very similar to the results of the combination of CX + SIM. We ran the algorithms with different size of population λ (10, 50 and 250) and mutation rate p_m (0.01, 0.05 and 0.08). For the termination condition of GA-FF we used fixed number of iterations; it is set to 10,000. Average values of the tables are the results of executing the algorithm 30 executions.

〈Table 1〉 Results obtained in [6] with Sparse graph, respectively: the best, average and worst evaluation found among the 10 executions of the algorithm, the average number of iterations of the algorithm before convergence.

	λ	DM	EM	ISM	SIM	IVM	SM
CX	10	22.62	22.63	22.63	22.61	22.66	22.64
		23.41	23.43	23.56	23.39	23.54	23.62
		26.04	25.47	25.98	26.64	25.33	28.11
		7,327	7,104	6,831	8,233	6,656	7,028
	50	22.61	22.61	22.61	22.61	22.61	22.61
		22.82	22.80	22.82	22.82	22.81	22.81
		23.42	23.30	24.19	24.08	24.29	24.08
		40,056	39,131	40,580	40,233	39,311	40,509
	250	22.61	22.61	22.61	22.61	22.61	22.61
		22.69	22.71	22.70	22.71	22.71	22.69
		22.95	23.28	22.85	23.55	23.28	22.88
		126,750	128,388	127,751	133,450	124,763	126,973

〈Table 2〉 Results obtained in [6] with Dense graph, respectively: the best, average and worst evaluation found among the 10 executions of the algorithm, the average number of iterations of the algorithm before convergence.

	λ	DM	EM	ISM	SIM	IVM	SM
CX	10	50.91	50.88	50.88	50.88	50.88	50.88
		52.69	52.54	52.27	52.59	52.67	52.60
		56.64	55.72	56.38	56.01	56.66	57.51
		10,318	10,287	100,68	10,751	10,193	11,011
	50	50.88	50.88	50.88	50.88	50.88	50.88
		51.88	51.77	51.66	51.73	51.93	51.73
		54.44	54.49	54.54	55.36	55.07	54.63
		24,038	25,499	25,597	25,788	24,771	24,873
	250	50.88	50.88	50.88	50.88	50.88	50.88
		51.09	51.14	51.09	51.06	51.08	51.07
		52.70	53.01	52.70	51.91	52.39	52.70
		82,688	89,069	86,897	82,104	81,019	84,865

〈Table 3〉 Results obtained with Sparse graph by applying GA-FF, respectively: the best, average and worst evaluation found from the 30 executions of the algorithm.

λ	pm		
	0.01	0.05	0.08
10	22.61	22.61	22.61
	22.67	22.66	22.67
	22.76	22.73	22.79
50	22.61	22.61	22.61
	22.66	22.66	22.66
	22.73	22.73	22.73
250	22.61	22.61	22.61
	22.64	22.64	22.65
	22.73	22.66	22.66

〈Table 4〉 Results obtained with Dense graph by applying GA-FF, respectively: the best, average and worst evaluation found from the 30 executions of the algorithm.

λ	pm		
	0.01	0.05	0.08
10	50.88	50.88	50.88
	51.07	51.01	51.04
	51.58	51.58	51.58
50	50.88	50.88	50.88
	50.88	50.89	50.88
	50.88	51.14	50.88
250	50.88	50.88	50.88
	50.88	50.88	50.88
	50.88	50.88	50.88

<Table 3 and 4> show that the best evaluations found by GA-FF are 22.61 and 50.88 for Sparse and Dense graphs, respectively. These two values were also found by GA-MWT as shown in <Table 1 and 2>. Even though it can not be verified theoretically, supported by our extensive experiments and the results of GA-MWT, we conjecture that the minimum weight of Sparse and Dense graphs are 22.61 and 50.88, respectively. Note that these two values were also obtained in [9] by Simulated Annealing.

However, <Table 3 and 4> show that GA-FF always found these conjectured optimum evaluations even when λ = 10. Furthermore, if we compare the values of the worst and average evaluations in <Table 1 and 2> and <Table 3 and 4>, it is evident that the performance of GA-FF is much more stable than the one of GA-MWT. For example, the last row of <Table 4> shows that GA-FF always find the conjectured optimum evaluation of 50.88 for all the 30 executions for Dense graph when λ = 250. <Table 3 and 4> also show that for the three different mutation

rate the results are very similar. It is hard to decide any optimal mutation rate. Therefore, we conclude that GA-FF is not sensitive to the mutation rate p_m.

5. Conclusions

In this paper we developed a genetic algorithm that can be applied to the minimum weight triangulation problems. Elimination game is an efficient tool for embedding arbitrary graph G into a chordal graph by adding additional edges to G. However, the resultant chordal graphs may contain redundant edges. By removing these redundant edges from the filled graphs we showed that the proposed genetic algorithm shows very stable results on two test graphs.

There are other ways of solving minimum weight triangulation problems, e.g., finding minimal triangulation using clique trees of graphs. Therefore, it will be quite interesting to compare the results of these different methods of computing minimum weight triangulation with genetic algorithms.

Reference

[1] Pinar Heggernes, "Minimal triangulations of graphs: A survey," Discrete Mathematics 306, pp.297-317, 2006.

[2] S. Arnborg, D.G. Corneil, A Proskurowsi, "Complexity of finding embeddings in a k-tree," SIAM J. Algebraic Discrete Methods 8, pp.277-284, 1987.

[3] M. Yannakakis, "Computing the minimum fill-in is NP-complete," SIAM J. Algebraic Discrete Methods 2, pp.77-79, 1981.

[4] Finn V. Jensen, "An introduction to Bayesian networks," UCL Press, 1996.

[5] Wen, W. X., "Optimal decomposition of belief networks," In Uncertainty in Artificial Intelligence 6 (P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, eds), North-Holland, Amsterdam, pp.209-224, 1990.

[6] Pedro Larrañaga, Cindy M. H. Kuijpers, Milel Poza and Roberto H. Murga, "Decomposing Bayesian networks: triangulation of the moral graphs with genetic algorithm," Statistics and Computing 7, pp.19-34, 1997.

[7] D.J. Rose, R.E. Tarjan and G.S. Lueker, "Algorithmic aspects of vertex elimination on graphs," SIAM J. Comput. 5, pp.266-283, 1976.

[8] Norsys Software Corp., http://www.norsys.com, 2006.

[9] Uffe Kjaerulff, "Triangulation of graphs - Algorithms giving small total state space," Research Report R-90-09, Department of Computer Science, Aalborg University, 1990.

[10] Tarjan, Robert Endre, "Maximum Cardinality Search and chordal graphs," Stanford Univ. Unpublished Lecture Notes CS 259.

[11] Zbigniew Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," Springer-Velag, Berlin, 1992.

[12] De Jong, K. A., An analysis of the behavior of a class of genetic adaptive systems, Ph.D. Dissertation, University of Michigan, 1975.

한 근 희

e-mail : kehan@kongju.ac.kr

1986년 건국대학교 물리학과 졸업(학사)

1992년 Univ. of Central Oklahoma 응용수학과 졸업(이학석사)

1996년 Univ. of Oklahoma 컴퓨터과학과 졸업(이학박사)

1996년~2000년 한국전자통신연구원

1999년~2000년 미국 NIST 객원연구원

2000년~현 재 공주대학교 응용수학과 부교수

관심분야 : 그래프 알고리즘, Genetic Algorithm

김 찬 수

e-mail : chanskim@kongju.ac.kr

1991년 부산대학교 전산통계학과 졸업

1997년 부산대학교 통계학과 졸업(이학박사)

2002년~현 재 공주대학교 응용수학과 조교수

관심분야 : 베이지안 네트워크, 유전 알고리즘