論文 11-1-4

Domination 이론을 이용한 acyclic digraph의 빠른 신뢰도 계산을 위한 연구

A study of fast Reliability evaluation for acyclic digraph using domination theory

이 광 원*
Kwang-Won Rhie

ABSTRACT

The aim of this paper is to develop more fast algorithm for evaluation of the reliability of networks and system. It is illustrated with examples.

This paper derived the algorithm to calculate the acyclic directed graph G(deals with the problem of the s-t graph). The language PASCAL was used to implement the algorithm. Three Examples are calculated and the calculation time is shorter than the time by program in 「21」.

1. 요 약

현대 산업사회에서 네트워크나 시스템의 신뢰도는 전기, 전자, 통신, 화공, 핵공학 등에서 광범위하게 사용되고 있다. 통신의 경우 순간의 고장도 사회에 직접적으로 미치는 파급 효과가 크므로, 예측되는 부품의 고장 등으로 인한 네트워크의 사용 가능 여부, 확률 등이 관심의 대상이 되고 있다. 이러한 문제들은 종종 신뢰도 공학의 고전적인 문제인 source to terminal problem으로 표현될 수 있으며, 이들의 문제해결을 위하여 그래프 이론과 domination 이론이 점점 중요한 비중을 차지하고 있다.

예를 들면 13개의 minimal path로 구성된 어떤 네트워크(그래프)를 관찰할 때, 신뢰도 계산을 위하여는 이들 13개 m-path의 모든 조합($2^{13}-1=8191$개)을 관찰하여야 하나, 「1」에서 발표된 예제는 domination 이론을 기초로 한 topologic식을 사용하면 정확성의 상실없이 123개 항으로 감소시킬 수 있음을 보여주었다. A. Satyanarayana와 A. Prabhaker등은 「1-19」에서 그래프로 표현되는 시스템이나 네트워크들의 정확한 신뢰도 계산을 위

* 호서대학교 산업안전공학과

하여 m-path를 사용한 domination 이론을 연구하고, 몇 가지 알고리즘을 제시하였다.

하지만 어떤 네트워크를 관찰할 때 "왜 정상인가?" 보다는 "왜 고장인가?"를 관찰하여야 할 경우가 더 많으며, 이런 경우 m-path보다는 m-cutset을 사용한 신뢰도분석이 더 요구된다. 「20」에서는 m-cutset을 근거로 한 네트워크(그래프)의 domination을 연구하였으나, 「1」의 m-path를 기초로 한 경우처럼 간단한 topologic식이 성립될 수 없음을 밝혔다.

본 논문에서는 「20」에서 발표된 몇 가지 결과를 이용하여 acyclic directed 그래프(네트워크) G의 신뢰도 계산을 위한 새로운 식(12)를 유도하였다. 이 식은 최대 항수가 2^n(n은 내부절점수)개이며 동시에 m-cutset도 구할 수가 있고, 이때 소요되는 계산 시간 t는 $t \leq \Omega \sum_{i=0}^{n} {}_nC_i(n-i+1)$($\Omega$는 항 1개를 계산하는데 소요되는 평균시간)이다.

위의 계산 시간은 다른 문헌「1, 8, 10, 11, 13, 15, 21」과 비교하여 볼 때 이론적으로 가장 빠르다.

6장에서는 식(12)를 프로그램 언어 파스칼로 실현화 시킨 프로그램 "RAFRC"를 제시하였고, 몇 가지 예를 다른 문헌에 나타난 프로그램「6, 21」과 같이 실행시킨 결과 RAFRC가 가장 빠르게 계산을 실행하였다.

2. 문제의 제기 및 논문의 목적

2.1 시스템의 정의

어떤 시스템이나 네트워크에서 시스템의 성분들의 신뢰도 혹은 고장도가 주어지고 전체 시스템의 신뢰도 검사가 자주 요구된다. 이때 시스템의 구조는 그래프나 logic tree로 주어지게 된다.

신뢰도 검사는 이들 그래프의 두 절점(vertex) 혹은 여러 개의 절점이 연결되어 있을 확률을 계산하며, logic tree인 경우 Top 사상이 일어날 확률을 계산하여 실현화 시킨다. 이때 확률 계산의 복잡도는 주어진 시스템의 크기와 구조에 의해 결정된다.

본 논문에서는 절점의 집합 V와 방향이 주어진 선(혹은 가지, directed edge)들의 집합 E로 이루어지고 cycle이 포함되지 않은 선형 acyclic digraph G=(V, E)만을 관찰하기로 한다. 여기에서 선형(linear) 그래프는 그래프 상에 한 선과 실제 시스템의 부품이 1대 1로 대응하는(한 부품은 한 개의 선으로 표현되는) 그래프를 뜻한다.

E에 속하는 모든 선 e_i는 확률 p_i를 갖는 정상 상태와 확률 $1-p_i(=q_i)$를 갖는 고장 상태만을 갖게 되며, 선의 형태로 나타나는 부품들은 고장시 서로 독립적이라 가정한다. 절점들은 선들(부품들)의 연결을 나타내 주며 완벽하다고(고장날 수 있는 확률이 0이라고) 가정한다.

신뢰도 검사의 목적에 따라 그래프의 source절점 V_s(혹은 s)와 terminal절점들 V_K(만약 절점이 하나인 경우 V_t 또는 t)가 결정되며 V_K의 수 $|V_K|=k$에 따라

1) source to terminal(s-t) problem ($|V_K|=1$)
2) source to all terminal(SAT) problem ($|V_K|=n-1$)
3) source to K-terminal(SKT) problem ($1 \leq |V_K| \leq n-1$)

으로 구분되며, 본 논문에서는 s-t problem만을 관찰하기로 한다.

2.2 시스템의 신뢰도

그래프 이론상으로 s-t신뢰도 R(G)는 source절점 s와 terminal절점 t가 연결되는 확률로 정의된다.

어떤 그래프 G에서 minimal path(m-path) P_i는 절점 s와 t를 연결시켜 주는 최소한의 선의 집합으로 정의되며 minimal cutset(m-cutset) C_i는 절점 s와 t를 잇는 모든 m-path를 절단시켜주는 최소한의 선의 집합으로 정의된다.

결국 어떤 시스템 G의 신뢰도 R(G)는 그래프 G가 최소한 하나의 작동되는 m-path를 갖을 확률로, 고장도 Q(G)(=1−R(G))는 최소한 하나의 m-cutset에 포함되는 모든 부품이 동시에 고장나는 확률로 쓸 수가 있다.

G의 모든 m-cutset의 family를 $C(G)=\{C_1, C_2, \cdots, C_m\}$이라 하고 고장사건 A_i를 C_i에 속하는 모든 선(이에 대응하는 부품)이 고장일 사건이라 하면 Q(G)는

$$Q(G)=q\left(\bigcup_{i=1}^{m} A_i\right) \quad \cdots\cdots (1)$$

로 쓸 수 있다. 이 식을 풀기 위한 일반적인 방법

은 다음과 같이 3부류로 나눌 수 있다.

1) Inclusion-Exclusion

$$Q(G)=\sum_{i=1}^{m} q(A_i)-\sum_{i=1}^{m}\sum_{j<i} q(A_i\cap A_j)$$
$$+\cdots+(-1)^{m-1}q(A_1\cap A_2\cap A_3\cap\cdots\cap A_m) \quad \cdots\cdots(2)$$

2) Sum of disjoint products

$$Q(G)=q(A_1)+q(\bar{A}_1\cdot A_2)+\cdots$$
$$+q(\bar{A}_1\cdot\bar{A}_2\cdot\bar{A}_3\cdots\bar{A}_{m-1}\cdot A_m) \quad \cdots\cdots(3)$$

3) pivotal decomposition(factoring)

$$Q(G)=q(e)\cdot Q(G\mid \bar{e})+(1-q(e))\cdot Q(G\mid e) \quad \cdots\cdots(4)$$

여기에서 사건 $A_i\cap A_j\cdots\cap A_1$는 해당되는 m-cutset들 C_i, C_j, …, C_1에 포함되는 모든 선이 고장일 사건을 뜻하며, $Q(G\mid\bar{e})$는 선 e가 고장시 그래프 G의 고장도를, $Q(G\mid e)$는 선 e가 정상시 그래프 G의 고장도를 표시한다.

식(2)의 항수는 m-cutset의 수가 m개인 경우 2^m-1개이며, 관찰하고자 하는 시스템이 복잡하거나 클 경우 m역시 폭등하므로 실제로는 식(2)의 제1항이나 3항까지만의 계산으로 만족할 수밖에 없었다.

하지만 식(2)에서 생성되는 2^m-1개의 항 중에서 같은 선부분집합이면서 부호만 다른 항이 많이 생성되며 소거되지 않는 항들은 상대적으로 소수에 불과하다.

[예제 1] 다리구조(bridge structure)를 갖는 그래프 G에서의 s-t problem

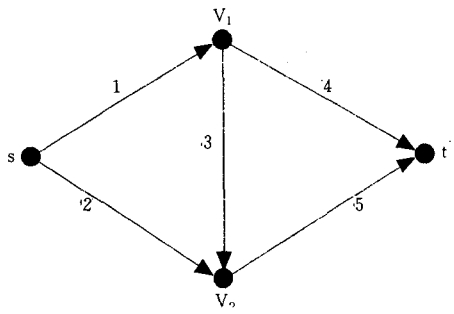


Fig. 1 A graph with bridge structure

Fig. 1과 같이 다리구조 G는 선 1,2,3,4,5와 절점 4개로 구성된 acyclic digraph이다. 이 그래프 G

Table 1 The number of m-cutset and term of eq.(2)

	n=2	n=5	n=10	n=100
최대가능 m-cutset수 ($m<=2^n$)	4	32	1024	1.267E30
식(2)의 제1항($_mC_1$)	4	32	1024	1.267E30
식(2)의 제2항($_mC_2$)	6	496	523776	8.026E59
식(2)의 제3항($_mC_3$)	4	4960	1.784E08	3.389E89
…				
식(2)의 총수(2^m-1)	15	4.29E09	>1.0E100	>1.0E100

의 m-cutset family C(G)는

$$C(G)=\{\{1,2\},\ \{1,5\},\ \{2,3,4\},\ \{4,5\}\}$$

이며, 식(2)의 Inclusion-exclusion 방법에 의한 G의 고장도는

$$Q(G)=q\left(\bigcup_{i=1}^{4}A_i\right)$$
$$=\sum_{i=1}^{4}q(A_i)+\sum_{i=1}^{4}\sum_{j<i}q(A_i\cap A_j)+$$
$$+\cdots+(-1)^{4-1}q(A_1\cap A_2\cap A_3\cap A_4)$$
$$=q(A_1)+q(A_2)+q(A_3)+q(A_4)$$
$$-q(A_1\cap A_2)-q(A_1\cap A_3)-q(A_1\cap A_4)$$
$$-q(A_2\cap A_3)-q(A_2\cap A_4)-q(A_3\cap A_4)$$
$$+q(A_1\cap A_2\cap A_3)+q(A_1\cap A_2\cap A_4)$$
$$+q(A_1\cap A_3\cap A_4)+q(A_2\cap A_3\cap A_4)$$
$$-q(A_1\cap A_2\cap A_3\cap A_4)$$
$$=q(1,2)+q(1,5)+q(2,3,4)+q(4,5)$$
$$-q(1,2,5)-q(1,2,3,4)-q(1,2,4,5)$$
$$-q(1,2,3,4,5)-q(1,4,5)-q(2,3,4,5)$$
$$+q(1,2,3,4,5)+q(1,2,4,5)$$
$$+q(1,2,3,4,5)+q(1,2,3,4,5)$$
$$-q(1,2,3,4,5)$$

이중에서 $\{1,2,4,5\}$는 2번, $\{1,2,3,4,5\}$는 5번 나타나며, 이중 6개항은 서로 상쇄되어진다. 본 논문의 알고리즘에 의한 결과는

$$Q(G)=q(1,2)(1-q(3,4))(1-q(5))$$
$$+q(2,3,4)(1-q(5))+q(4,5)$$
$$+q(1,5)(1-q(4))$$

로 계산되며 당연히 이식을 모두 풀어 쓰면 Inclusion-Exclusion에 의해 얻어지는 항과 같다.

위의 [예제 1]에서 보듯이 식(2)에 의한 방법은 15개의 항이 생성되며 6개의 항이 소거되어질 수 있고, 본 논문의 알고리즘에 의하여는 4개항(총 계산항은 8개)이 생성된다. 「1」에서는 m-path를

기초로 한 식(2)의 소거되지 않는 항들 만을 생성하는 알고리즘을 소개하였고, [예제 1]의 다리구조인 경우 3개의 m-path($\{1,4\}$, $\{1,3,5\}$, $\{2,5\}$)가 존재하며, 이로부터 7개의 소거되지 않는 항($\{1,4\}$, $\{1,3,5\}$, $\{2,5\}$, $\{1,3,4,5\}$, $\{1,2,4,5\}$, $\{1,2,3,5\}$, $\{1,2,3,4,5\}$)들을 관찰하여야 한다.

본 논문에서는 m-cutset을 미리 찾을 필요도 없으며, 정확성을 잃지 않고 소거되어지는 대부분의 항들을 제외시킨 후 나머지 항들을 몇 개의 항으로 위의 [예제 1]과 같이 "묶어서" 빠르게 신뢰도를 계산할 수 있는 알고리즘과 프로그램을 제시하고자 한다.

3. 수학적 배경

3.1 신뢰도 공학의 기초 개념

본 장에서는 전 장에서 언급되지 않은 신뢰도 공학과 그래프 이론의 중요한 개념과 수치들을 정의, 설명한다[22].

3.1.1 p-선, p-그래프, k-선과 t-선

s-t problem을 관찰할 때 m-path에 포함되는 선을 p-선, 이들로만 구성된 그래프를 p-그래프라 한다[1]. $|V_k|=k$(SKT problem) 또는 $|V_k|=n-1$(SAT problem)인 경우 「5,6」에서는 p-선을 k-선과 t-선이라고 부른다. k-선 또는 t-선을 포함하지 않은 그래프들을 k-그래프 또는 t-그래프라 부른다. m-path와 m-cutset의 정의에서 보듯이 m-path에 속하지 않는 선($\bar{p}$-선, 또는 not-p-선)들은 m-cutset에도 속하지 않게 되며, 결국 p-그래프들은 모든 선이 최소한 한번은 m-cutset에 속하는 그래프라고도 할 수 있다.

식(2)에서 보듯이 m-cutset에 속하지 않는 선($\bar{p}$-선)들은 신뢰도 계산에 영향을 주지 못하므로 우리가 앞으로 관찰하는 그래프 G는 p-선들로만 구성된 p-그래프에 한한다.

3.1.2 Indegree와 Outdegree, Incut과 Outcut

어떤 p-그래프에서 임의의 절점 V_i를 관찰할 때 들어오는(도착되어지는) 모든 선의 집합을 그 절점의 Incut이라 하고 Incut(i)(또는 I_i)라 표시하며, V_i에서 나가는(출발하는) 모든 선의 집합을 그 절점의 Outcut이라 하고 Outcut(i)(또는 O_i)로 표시한다.

indegree(i) 또는 outdegree(i)는 절점 V_i에 도착되는 선 또는 출발하는 선의 수를 나타내 주며 간단히 Ind(i) 또는 Outd(i)로 표시한다.

정의에 의해 source절점 s에서 Ind(s)=0이며 terminal절점에서 Outd(t)=0이 된다.

[예제 2] Fig. 2에서 Incut과 Outcut, Indegree와 Outdegree는 다음과 같다.

	Incut	Outcut	Indegree	Outdegree
s	{ }	{1,2}	0	2
V_1	{1}	{3,4}	1	2
V_2	{2,3}	{5}	2	1
t	{4,5}	{ }	2	0

3.1.3 Domination과 Formation

G의 어떤 임의의 부분그래프(subgraph)를 G_a라 하고, $E=\{e_1, e_2, \cdots, e_b\}$를 G의 선집합, E_a ($\subseteq E$)는 G_a의 선집합, $M(G)=\{M_1, M_2, \cdots, M_n\}$을 E의 원소를 기초로 한 임의의 집합 family ($M_i \subseteq E$)라 하면 모든 M(G)의 subfamilyset $\underline{M}_i=\{M_i, M_j, \cdots, M_l\}$들이

$$\bigcup_{j=i}^{l} M_j = E_a$$

인 경우 G_a의 formation이라 한다. 만약 $|\underline{M}_i|$ (= $\underline{M}_i$에 포함된 M_i수)가 짝수이면 짝수 formation, 그렇지 않으면 홀수 formation이라 하고 N_e와 N_o를 짝수와 홀수 formation의 수를 나타낸다고 하면, M(G)를 기초로 한 부분그래프 G_a의 domination d(G_a, M(G))는

$$d(G_a, M(G)) = N_o - N_e \quad \cdots\cdots (5)$$

로 정의된다.

[예제 3]

M(G)가 Fig. 1의 m-cutset family C(G)인 경우를 관찰하자. $C_1=\{1,2\}$, $C_2=\{1,5\}$, $C_3=\{2,3,4\}$, $C_4=\{4,5\}$라 하면 선의 집합 $\{1,2,3,4,5\}$를 갖는 그래프의 formation은 5개가 존재한다. 즉 $C_2 \cup C_3$, $C_1 \cup C_2 \cup C_3$, $C_1 \cup C_3 \cup C_4$, $C_2 \cup C_3 \cup C_4$, $C_1 \cup C_2 \cup C_3 \cup C_4$가 존재하며 3개의 홀수 formation과, 2개의 짝수 formation을 갖으므로 C(G)를 기초로 한 $\{1,2,3,4,5\}$를 갖는 그래

프(즉 G)의 domination은
$d(G, C(G))=n_o-n_e=3-2=1$이 된다.

(2)식에서 항 $A_i \cap A_j \cdots \cap A_l$는 이들이 포함하고 있는 선들 만을 포함하는 G의 어떤 부분그래프 G_a에 대응하게 되며 결국 식(2)는

$$Q(G)=\sum_{\forall G_a} d(G_a, C(G))\cdot q(G_a)$$

여기서 $q(G_a)$: G_a의 모든 선이 고장일 확률
G_a : 식(2)에 나타나는 항들과 대응되는 부분그래프
로 나타낼 수 있다.

3.1.4 Topologic Number

acyclic digraph G에서는 topological하게 번호를 부여할 수 있다[23]. 즉 어떤 임의의 선 e의 출발점을 i, 도착점을 j라 할 경우 항상 절점 i의 topologic number가 j의 topologic number보다 빠르도록 모든 절점에 대하여 번호를 부여할 수 있다. source절점은 통상 0을 terminal절점은 n−1번을 부여하며, 내부절점은 1에서 n−2까지의 수를 갖게 된다. 여기에서 n은 절점의 수이다.

3.2 Domination 이론을 사용한 신뢰도 계산 방법 연구의 최근 동향

어떤 (cyclic)그래프 G와 임의의 부분그래프 G_a를 관찰할 때, A. Satyanarayana는 「1」에서 이들의 domination이 m-path를 기초로한 경우

$$d(G_a, P(G)) = \begin{cases} 0, & \text{만약 } G_a\text{가 cyclic−or } \bar{p}\text{-부분그래프} \\ (-1)^{b-n+1}, & \text{만약 } G_a\text{가 acyclic p-부분그래프} \end{cases} \quad \cdots\cdots (7)$$

가 됨을 발표하였다. 여기서 b와 n은 G_a 부분그래프의 선수와 절점수이다.

이 결과를 토대로 m-path를 기초로 한 식(2)의 소거되지 않는 모든 항들이 관찰하는 그래프 G의 acyclic p-부분그래프와 1 대 1로 각각 대응하고, 이 항들의 부호는 $(-1)^{b-n+1}$로써 그에 대응하는 acyclic p-부분그래프의 선수와 절점수에 의하여 결정된다고 발표하였다.

Satyanarayana와 동료들은 s-t problem을 위한 모든 acyclic p-부분그래프을 찾는 알고리즘을 「2, 3, 4」에서 SAT-problem을 위한 모든 acyclic t-부분그래프을 찾는 알고리즘을 「5, 8」에서 그리고 SKT-problem을 위한 모든 acyclic k-부분그래프를 찾는 알고리즘을 「7」에서 보여 주었으며 이들 모두가 결국은 Inclusion-exclusion식의 소거되지 않는 항들을 찾는 것이다.

그들이 사용한 주요한 개념은

1) 만약 어떤 acyclic-not-trivial-t-그래프 G에는 항상 최소한 하나의 선 e가 존재하며, 이의 제거 후 G-e는 항상 다시 acyclic t-부분그래프가 되며 G와 G-e의 domination은 서로 다른 부호를 갖게 됨을 보이고,
2) cycle을 포함한 그래프에서는 formation의 family를 분할(Partition)하고 이들의 짝수와 홀수 formation의 수가 같음을 보여 domination이 0임을 증명하였다.
3) 「5」에서 Satyanarayana는 SKT problem을 위하여 K(G)를 기초로 한 어떤 p-그래프의 domination이 P(G)를 기초로 한 것과 같음을 보여 주었다. 여기서 P(G)는 G의 spannened tree의 family이다.

그 후 m-cutset을 사용한 domination연구「20」에서 C(G)를 기초로한 어떤 그래프 G의 부분그래프 G_a의 domination결정이 m-path를 사용한 경우처럼 그 부분그래프의 절점수 n과 선수 b로 표현될 수 없는 경우(부분그래프가 bipartite 구조를 갖을 때)가 있음을 밝혔다. 그러나 위 논문에서는 어떤 acyclic p-그래프 G를 관찰할 때 다음의 사실을 증명하였다.

- domination이 0이 아닐 수 있는 모든 부분그래프의 수(즉, 식(2)에서 소거되지 않는 항의 수)는 최대 3^n이 되며(n은 내부절점수),
- 이들 부분그래프들은 관찰하는 그래프 G에서 Incut(A)와 Outcut(B)에 포함되는 선들을 제거함으로써 구해지며(단 A, B는 내부절점의 부분집합이고 서로 공유하는 절점이 없다. 즉, $A \subseteq N$, $B \subseteq \{N-A\}$이며 N은 모든 내부절점의 집합($=\{V\text{-}s\text{-}t\}$)을 의미한다.),
- 모든 가능한 A와 B의 조합에 의하여 생성되는 부분그래프의 family는 $\{G-I_A-O_B \mid A \subseteq N, B \subseteq \{N-A\}\}$로 쓸 수 있으며, 관찰하는 그래프 G의 고장도 Q(G)는 식(6)에서

$$Q(G)=\sum_{\forall A \subseteq N} \sum_{\forall B \subseteq (N-A)} (-1)^{n-|A|-|B|}\cdot q(G-I_A-O_B) \quad \cdots\cdots (8)$$

로 써지게 된다. 여기서 $q(G-I_A-O_B)$는 $G-I_A-O_B$에 포함되는 모든 선들이 동시에 고장날 확률이다.

식(8)는 우리가 관찰하여야 하는 항의 최대수가 3^n임을 말해 준다. 이는 n개의 내부절점을 갖는 그래프가 최대로 갖을 수 있는 m-cutset의 수가 2^n임을 감안할 때 식(2)에서 나타날 수 있는 최대 항수가 2^{2^n}이므로 신뢰도 계산에 있어서 상당히 유리하다고 할 수 있다.

· 만약 어떤 I_A가 Outcut O_b를 포함하는 경우(단 $b\in B$)

$$\sum_{\forall B\subseteq |N-A|} (-1)^{n-|A|-|B|}\cdot q(G-I_A-O_B)$$
$$=\sum_{\forall B'\subseteq |N-A-b|} (-1)^{n-|A|-|B|}\cdot q(G-I_A-O_{B'})+\sum_{\forall B'\subseteq |N-A-b|} (-1)^{n-|A|-|B|-1}\cdot q(G-I_A-O_{B'}-O_b)$$

로 분리하여 쓸 수 있으며, 우변의 두 부분그래프 $G-I_A-O_{B'}$와 $G-I_A-O_{B'}-O_b$는 같은 선의 집합이고 부호가 서로 다르므로 우변의 두항들은 서로 상쇄된다. 결국 식(8)는 다시

$$Q(G)=\sum_{\forall A\subseteq N, I_A\notin O_b}\ \sum_{\forall B\subseteq |N-A|} (-1)^{n-|A|-|B|}\cdot q(G-I_A-O_B) \quad \cdots\cdots\cdots (9)$$

로 쓸 수 있다.

4. 신뢰도 계산을 위한 빠른 알고리즘

4.1 신뢰도 계산을 위한 새로운 식의 유도

이 장에서는 식(9)을 기초로 새로운 식을 유도한다.

어떤 임의의 절점집합 $A(A\subseteq N,\ O_b\notin I_A,$ 단 $b\in B)$에 대하여 부분그래프의 family $\{G-I_A-O_B \mid B\subseteq\{N-A\}\}$를 생각하여 보자. Incut과 Outcut의 정의에 의하여 그래프 G의 선들은 $O_{s,N}$이라 표현할 수 있고, 부분그래프들의 family $\{G-I_A-O_B \mid B\subseteq\{N-A\}\}$는

$$\{G-I_A-O_B \mid B\subseteq\{N-A\}\}$$
$$=\{O_{s,N}-I_A-O_B \mid B\subseteq\{N-A\}\}$$
$$=\{O_{s,N}-I_A-(O_{|N-A|}-O_{|N-A-B|}) \mid B\subseteq\{N-A\}\}$$
$$=\{(O_{s,N}-O_{|N-A|}-I_A)\cup(O_{|N-A-B|}-I_A) \mid B\subseteq\{N-A\}\}$$
$$=\{(O_{s,A}-I_A)\cup(O_{|N-A-B|}-I_A) \mid B\subseteq\{N-A\}\}$$

로 표현된다.

여기서 $\{B \mid B\subseteq\{N-A\}\}$이나 $\{N-A-B \mid B\subseteq\{N-A\}\}$는 같은 family이므로 $\{G-I_A-O_B \mid B\subseteq\{N-A\}\}$는 $\{(O_{s,A}-I_A)\cup(O_B-I_A) \mid B\subseteq\{N-A\}\}$로 $n-|A|-|B|$는 $|B|$로 바꾸어 쓸 수 있다. 이로부터 식(9)은

$$Q(G)=\sum_{\forall A\subseteq N, I_A\notin O_b}\ \sum_{\forall B\subseteq |N-A|} (-1)^{|B|}\cdot$$
$$q((O_{S,A}-I_A)\cup(O_B-I_A))$$
$$=\sum_{\forall A\subseteq N, I_A\notin O_b} q(O_{S,A}-I_A)$$
$$\sum_{\forall B\subseteq |N-A|} (-1)^{|B|}\cdot q(O_B-I_A) \quad \cdots\cdots\cdots (10)$$

[예제 4] Fig. 1에서 Q(G)는 식(10)에 의하여

$$Q(G)=\sum_{\forall A\subseteq N, V_1, V_2, I_A\notin O_b} q(O_{s,A}-I_A)$$
$$\sum_{\forall B\subseteq |N-A|} (-1)^{|B|}\cdot q(O_B-I_A)$$
$$=q(O_s)\cdot\{1-q(O_1)-q(O_2)+q(O_{1,2})\} \leftarrow A=\{\ \}$$
$$+q(O_{s,1}-I_1)\cdot\{1-q(O_2-I_2)\} \leftarrow A=\{1\}$$
$$+q(O_{s,2}-I_2)\cdot\{1-q(O_1-I_1)\} \leftarrow A=\{2\}$$
$$+q(O_{s,1,2}-I_{1,2}) \leftarrow A=\{1,2\}$$
$$=q(1,2)\cdot\{1-q(3,4)-q(5)+q(3,4,5)\}$$
$$+q(2,3,4)\cdot\{1-q(5)\}$$
$$+q(1,5)\cdot\{1-q(4)\}$$
$$+q(4,5)$$

i와 j가 절점의 부분집합 B의 임의의 원소라 할 때(단 $i\neq j$) 선의 집합 O_i-I_A와 O_j-I_A는 항상 쌍으로 공유하는 선이 서로 소이다. 결국 식(7)에서 다시

$$\sum_{\forall B\subseteq |N-A|} (-1)^{|B|}\cdot q(O_B-I_A)$$
$$=\sum_{\forall B\subseteq |N-A|}\ \prod_{b\in B} (-1)\cdot q(O_b-I_A)$$

로 써지며, 인수분해를 하면 윗 식은 다시

$$\sum_{\forall B\subseteq |N-A|}\ \prod_{\forall b\in B} (-1)\cdot q(O_b-I_A)$$
$$=\prod_{\forall b\in |N-A|} (1-q(O_b-I_A))$$

가 되므로 식(10)은

$$Q(G)=\sum_{\forall A\subseteq N, I_A\notin O_b} q(O_{S,A}-I_A)$$
$$\prod_{\forall b\in |N-A|} (1-q(O_b-I_A)) \quad \cdots\cdots\cdots (11)$$

가 된다. 위 식은 내부절점 N의 부분집합 $A\subseteq N$중 I_A가 Outcut $O_b(b\in\{N-A\})$를 포함하지 않는

모든 A에 대하여 각각 $|N-A+1|$번의 곱셈이 요구된다. S(G)를 이러한 A의 family라 정의하면 식(11)은 다시

$$Q(G)=\sum_{\forall A\in S(G)} q(O_{s,A}-I_A) \prod_{\forall b\in |N-A|}(1-q(O_b-I_A)) \quad \cdots\cdots (12)$$

로 쓸 수 있으며, 식(12)의 계산 시간은 $\sum_{i=0}^{n} {}_nC_i(n-i+1)$에 비례하게 된다. 여기에서 S(G)에 포함되는 모든 A에 대하여 선집합들의 family $\{O_{s,A}-I_A \mid A\in S(G)\}$는 m-cutset을 모두 포함한다. 우리는 이런 $O_{s,A}-I_A$를 semi-cutset이라 한다.

다음 장에서는 재귀적인 procedure를 사용하여 위의 식을 충족시키는 알고리즘을 소개한다.

[예제 5]

[예제 4]에서 식(12)에 의하면 다음과 같이 쓸 수 있다.

$$\begin{aligned}Q(G)=&q(12)(1-q(3,4))(1-q(5))\\&+q(2,3,4)(1-q(5))\\&+q(1,5)(1-q(4))\\&+q(4,5)\end{aligned}$$

우선 S(G)를 구하기 위한 효율적인 알고리즘을 찾기 위해서는 다음의 정리가 필요하다.

정리 :

모든 내부절점의 구조를 감안하여 번호(topologic number)를 주는 경우 내부 절점 집합 N의 어떤 부분집합 A와 A에 속한 모든 절점의 topologic number보다 큰 topologic number를 갖는 절점 a가 있다 하자. 이 경우 A가 S(G)에 속하지 않는다면 $A\cup a$ 역시 S(G)에 속하지 않는다.

증명 :

S(G)의 정의에 따라 어떤 절점의 부분집합 A가 S(G)에 속하지 않는다면 I_A는 어떤 Outcut O_b ($b\notin A$)를 반드시 포함한다. 이때 절점 b에서 나가는 선(Outcut O_b)은 모두 A에 속하는 절점들로 도착되므로 b의 topologic number보다 큰 번호를 갖는 절점이 A에 반드시 존재하게 된다. 결국 $b\neq a$ 이고 $I_{A\cup a}$ 역시 $O_b(b\notin \{A\cup a\})$를 포함하게 되므로 $I_{A\cup a}$는 S(G)에 속할 수 없다.

모든 절점을 topological하게 일련번호를 주고, $A=\{\ \}$인 root에서 시작하여 topologic number 순으로 모든 필요한 절점의 부분집합 A($A\in S(G)$)들을 rooted-directed-tree형태로 재귀적인 procedure를 사용하여 구한다(Fig. 2 참조).

이 경우 만약 어떤 절점 A가 S(G)에 포함되지 않는 경우 A다음에 나오는 모든 부분집합들은 위의 정리에 의하여 S(G)에 포함되지 않는다.

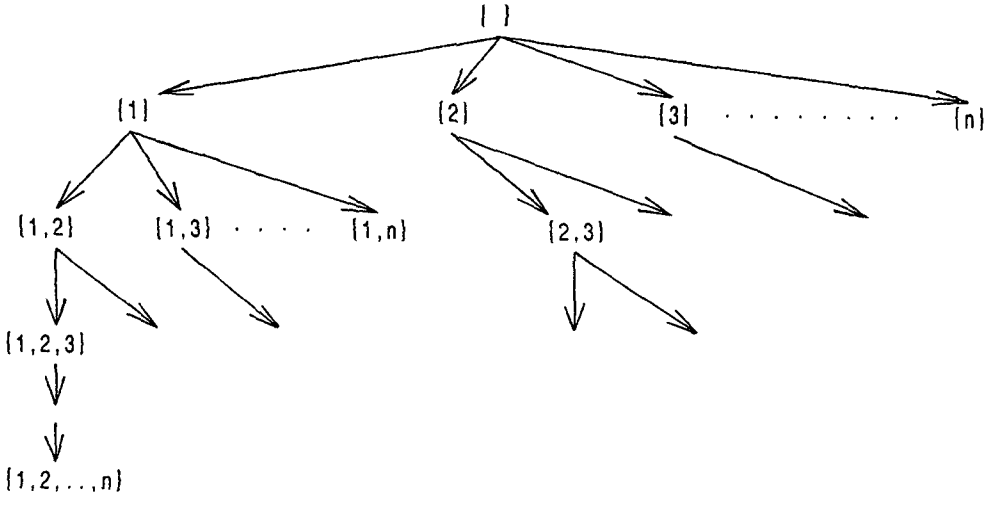


Fig. 2 A rooted-directed tree for get all S(G)

4.2 알고리즘

주프로그램

begin

〈단계 1〉 입력데이타를 읽는다

〈단계 2〉 초기값 부여 A=$\{\ \}$

〈단계 3〉 $Q=q(O_s)\cdot \prod_{\forall b\in N}(1-q(O_b))$를 구한다.

모든 $a\in N$에 대하여 topologic number 순으로 RECO(a)를 부른다.

end

procedure RECO(A)

〈단계 4〉 만약 A가 S(G)의 원소이면($A\in S(G)$) 단계 5로 간다.

그렇지 않으면 단계 6으로 간다.

〈단계 5〉 $Q=Q+q(O_{s,A}-I_A)\sum_{\forall b\in N-A}(1-q(O_b-I_A))$를 구한다.

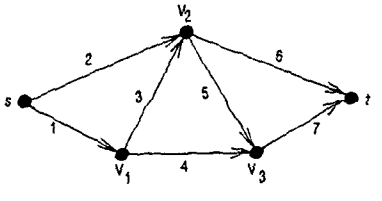
모든 $a\in\{N-A\}$에 대하여 RECO(A∪a)를 부른다.

〈단계 6〉 return

4.3 알고리즘의 예

실제 신뢰도 계산이 위 4-2장의 알고리즘에 의해서 어떻게 진행되는지 예를 통해서 살펴보기로 한다.

source절점과 terminal절점이 각각 하나이고, 내부 절점이 3개, 부품(선)의 수가 7개이고 각 부품의 신뢰도가 0.9인 s-t 그래프 G를 관찰하자.



5 7
1 2
1
3 4
2 3
5 6
4 5
7
6 7

Fig. 3 A acyclic graph G with 5 nodes and 7 edge and it's input data

위 입력 자료에서 첫 줄은 그래프의 모든 절점수 |V| 와 선(부품)수 |E| 를 나타낸 것이고 둘째 줄은 source절점의 outcut을 나타내며, 다음은 topologic number의 크기 순으로 V_1의 incut, V_1의 outcut, V_2의 incut, V_2의 outcut, V_3의 incut, V_3의 outcut을 차례로 나타낸 것이고, 끝 줄은 terminal 절점의 incut을 나타낸 것이다.

1. 입력 자료(Input.txt)를 읽는다.(단계 1)
2. 초기값부여 A=$\{\ \}$(단계 2)
3. A=$\{\ \}$일 때 Q값을 계산한다.(단계 3)
   $Q_1=q(1,2)(1-q(3,4))(1-q(5,6))(1-q(7))=0.00882$
   모든 내부절점 $\{V_1,V_2,V_3\}$에 대하여 topologic number순으로 절점 a와 함께 RECO(a)를 부른다.(단계 4)
   $a=V_1$
4. A=$\{V_1\}$(단계 4)
   I_1(=선1)은 outcut을 포함하지 않으므로 단계 5로 간다.
5. Q값 계산(단계 5)
   $Q_2=0.00882+q(2,3,4)(1-q(5,6))(1-q(7))$
   $=0.00882+0.00089$
   $=0.00971$
   모든 내부절점 $\{V_2,V_3\}$에 대하여 topologic number순으로 절점 a와 함께 RECO(V_1∪a)를 호출한다.
   $a=V_2$
6. A=$\{V_1,V_2\}$(단계 4)
   $I_{1,2}$(=선 1,2,3)는 outcut을 포함하지 않으므로 단계 5로 간다.
7. Q값 계산(단계 5)
   $Q_3=0.00971+q(4,5,6)(1-q(7))$
   $=0.00971+0.0009$
   $=0.01061$
   모든 내부절점 $\{V_3\}$에 대하여 topologic number순으로 절점 a와 함께 RECO($\{V_1,V_2\}$∪a)를 호출한다.
   $a=V_3$
8. A=$\{V_1,V_2,V_3\}$(단계 4)
   $I_{1,2,3}$(=선 1,2,3,4,5)는 output을 포함하지 않으므로 단계 5로 간다.
9. Q값 계산 (단계 5)
   $Q_4=0.01061+q(6,7)$
   $=0.01061+0.01$

=0.02061

모든 내부 절점이 출연하여 {N−A}=.{ }

이므로 5에서 두 번째 절점 V_3에 대하여 a=V_3와 함께 RECO(V_1Ua)를 호출한다.

a=V_3

10. A={V_1, V_3} (단계 4)

$I_{1,3}$(=선 1,4,5)는 outcut을 포함하지 않으므로 단계 5로 간다.

위와 같은 순서로 알고리즘은 진행되며 간단히 도식화하면 Fig. 4와 같다. 이때 화살표 옆의 번호는 진행 순서를 나타내 준다.

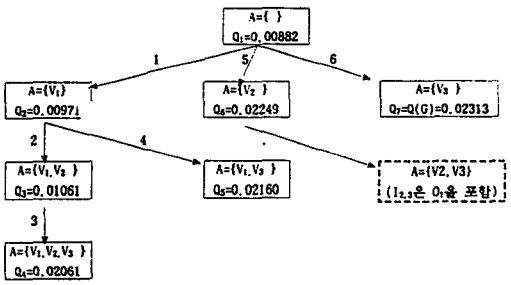


Fig. 4 A rooted directed tree for evaluation of Fig. 3

부록에서는 4-2장에서 설명한 알고리즘을 파스칼 언어를 사용한 프로그램 RAFRC를 제시한다. data입력으로는 "Input.txt"를 출력file은 "Output.txt"를 사용하였다. 프로그램의 처음에 나오는 상수 정의 부분에서 최대 선수(MaxLink)와 최대 degree(MaxD), 최대 절점수(MaxVertex)와 각 부품의 고장도가 모두 같은 경우 고장도의 값(URofC)을 정의할 수 있다.(현재 값은 각 20,5,16,0.1) Input file은 처음 줄에 절점수와 선수를, 둘째 줄에는 source절점의 Outcut을, 다음 두 줄에는 topologic number 1을 갖는 절점의 Incut과 Outcut을 그리고 마지막에는 terminal 절점의 Incut을 순차적으로 기입하며, 부품의 고장도가 동일하지 않는 경우에는 다음 줄에 선의 번호순으로 한 줄씩 고장도를 기입한다. 다음은 Fig. 3의 신뢰도 계산을 위하여 실행시킨 프로그램 RAFRC의 결과를 보여준다.

OUTPUT

given InCut, OutCut

```
Vertex Number=5        Link Number=7
                       OutCut[S]=1 2
InCut[1]=1             OutCut[1]=3 4
InCut[2]=2 3           OutCut[2]=5 6
InCut[3]=4 5           OutCut[3]=7
InCut[T]=6 7
Q(G)=q(1,2)*(1-q(3,4))*・(1-q(5,6))*
     (1-q(7))+q(2,3,4)*(1-q(5,6))*
     (1-q(7))+q(4,5,6)*(1-q(7))
     +q(6,7)+q(2,3,7)*(1-q(6))
     +q(1,5,6)*(1-q(4))*(1-q(7))
     +q(1,2,7)*(1-q(3))*(1-q(6))
Unreliability=0.023132
Reliability=0.976868
The number of semi-Cutset=7
Excute time is 0Hour 0Min. 0Sec. 5Hund.
```

5. 비교 및 결과

Y.B. Yoo와 Narsingh Deo는 s-t problem의 알고리즘을 비교하는 논문[21]에서 수정된 DOTSON의 것이 가장 빠르다고 발표하였다. 본 장에서는 domination이론을 사용한 프로그램 TRAP4[6]와 수정된 Dotson의 프로그램을 "RAFRC"프로그램과 비교하여 본다. TRAP4와 수정된 Dotson의 프로그램은 포트란77로 RAFRC는 터보 파스칼로 컴파일하였으며, 컴퓨터는 PC 486 DX66/2를 사용하였다. 다음 표는 각 프로그램의 특성과 적용 범위 등을 설명하고, 몇 가지의 간단한 예(CASE1,

Table 2 Comparision of three programs

(unit : sec)

	RAFRC	DOTSON	TRAP4
적용범위	acyclic directed 그래프 (s-t problem)	directed 그래프 (s-t problem)	(un)directed 그래프 (SKT problem)
신뢰도 계산	정확한 신뢰도 값을 구할 수 있다.	근사치도 계산	정확한 신뢰도 값을 계산한다.
시스템 함수	Cutset을 기초로 한 시스템 함수 출력	시스템 함수가 출력되지 않는다.	path를 기초로한 시스템 함수를 출력
기 타	semi-cutset을 구할 수 있다.		m-path를 구한다.
CASE1 실행시간	0.05	0.10	0.31
CASE2 실행시간	0.06	0.11	0.38
CASE3 실행시간	0.49	1.28	2.37

CASE2, CASE3)를 통하여 각 프로그램의 성능을 비교하여 보았다.

어떤 acyclic directed 그래프 또는 네트워크에서 하나의 출발점과 하나의 도착점이 주어진 경우 m-cutset의 family C(G)를 기초로한 domination 관계식을 사용하여 새로운 식을 유도하였고 이를 기초로 신뢰도 계산을 위한 알고리즘과 파스칼언어를 사용한 프로그램을 제시하였다. 문헌에 나온 다른 알고리즘과의 비교에서 본 논문이 제시한 알고리즘이 빠른것으로 판명되었다.

앞으로의 과제는 SKT problem 또는 SAT problem, 그리고 cycle을 포함한 경우와 방향이 없는 선(undirected edge)이 존재하는 경우, 더 나아가서는 비선형 그래프(예 : Fault tree) 등에 대한 신뢰도 분석의 빠른 알고리즘 연구라 하겠다.

CASE1 : Fig. 1 참조

OUTPUT

given InCut, OutCut

Vertex Number=4 Link Number=5

OutCut[S]=1 2

InCut[1]=1 OutCut[1]=3 4

InCut[2]=2 3 OutCut[2]=5

InCut[T]=4 5

$$Q(G)=q(1,2)*(1-q(3,4))*(1-q(5))+q(2,3,4)*(1-q(5))+q(4,5)+q(1,5)*(1-q(4))$$

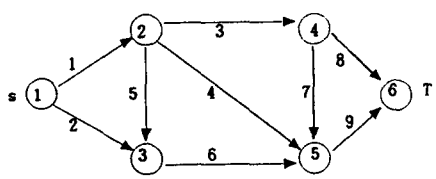
Unreliability=0.028810

Reliability=0.971190

The number of semi-Cutset=4

Excute time is 0Hour 0Minute. 0Sec. 5Hund.

CASE 2




```
6 9
1 2
  1
3 4 5
2 5
  6
  3
7 8
4 6 7
  9
8 9
```

Fig. 5 A acyclic graph with 6 nodes and 9 edge and it's input data

OUTPUT

given InCut, OutCut

Vertex Number=6 Link Number=9

OutCut[S]=1 2

InCut[1]=1 OutCut[1]=3 4 5

InCut[2]=2 5 OutCut[2]=6

InCut[3]=3 OutCut[3]=7 8

InCut[4]=4 6 7 OutCut[4]=8

InCut[T]=8 9

$$Q(G)=q(1,2)*(1-q(3,4,5))*(1-q(6))*(1-q(7,8))*(1-q(8))+q(2,3,4,5)*(1-q(6))*(1-q(7,8))*(1-q(8))+q(3,4,6)*(1-q(7,8))*(1-q(8))+q(4,6,7,8)*(1-q(8))+q(8)+q(3,8)*(1-q(8))+q(2,4,5,7,8)*(1-q(6))*(1-q(8))+q(1,6)*(1-q(3,4))*(1-q(7,8))*(1-q(8))+q(1,6,7,8)*(1-q(4))*(1-q(8))+q(1,8)*(1-q(3))*(1-q(8))+q(1,2,7,8)*(1-q(5,4))*(1-q(6))*(1-q(8))$$

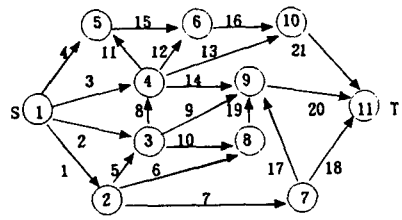
Unreliability=0.135162

Reliability=0.864838

The number of semi-Cutset=11

Excute time is 0Hour 0Minute. 0Sec. 6Hund.

CASE 3




```
11 21
1 2 3 4
    1
  5 6 7
  2 5
  8 9 10
    3 8
11 12 13 14
  4 11
    15
12 15
    16
     7
17 18
  6 10
    19
9 14 17 19
    20
13 16
    21
18 20 21
```

Fig. 6 A acyclic graph with 11 nodes and 21 edge and it's input data

OUTPUT

given InCut, OutCut

Vertex Number=11	Link Number=21
	OutCut[S]=1 2 3 4
InCut[1]=1	OutCut[1]=5 6 7
InCut[2]=2 5	OutCut[2]=8 9 10
InCut[3]=3 8	OutCut[3]=11 12 13 14
InCut[4]=4 11	OutCut[4]=15
InCut[5]=12 15	OutCut[5]=16
InCut[6]=7	OutCut[6]=17 18
InCut[7]=6 10	OutCut[7]=19
InCut[8]=9 14 17 19	OutCut[8]=20
InCut[9]=13 16	OutCut[9]=21
InCut[T]=18 20 21	

이 논문은 1994년도 학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

참 고 문 헌

1) A. Satyanarayana and A. Prabhaker, New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex networks, *IEEE Trans. Reliability*, Vol. R-27, pp. 82~100, June, 1978.
2) A. Satyanarayana, A. Prabhaker, Comments on New Topological Formular and Rapid Algorithm for reliability analysis of Complex networks, *IEEE Trans. Reliability*, Vol. R-28, p. 264, August, Oct., 1979.
3) R. R. Willie, A Theorem Concerning Cyclic Directed Graphs with Applications to network Reliability, *networks*, Vol. 10, pp. 71~78, 1980.
4) A. Satyanarayana, J. N. Hagstrom, Combinatorial Properties of Directed Graphs Useful in network Reliability, *networks*, Vol. 11, pp. 357~366, 1981.
5) A. Satyanarayana, A unified Formula for Analysis of Some network Reliability Problems, *IEEE Trans. Reliability*, Vol. R-31, No. 1, pp. 23~32, April, 1982.
6) A. Satyanarayana, J. N. Hagstrom, A New Algorithm for the Reliability Analysis of Multi-Terminal networks, *IEEE Trans. Reliability*, Vol. R-30, pp. 325~334, Oct., 1982.
7) A. Satyanarayana and M. k. Chang, network Reliability and the Factoring Theorem, *networks*, Vol. 13, pp. 107~120, 1983.
8) J. A. Buzacott, A Recursive Algorithm for Directed-Graph Reliability, *networks*, Vol. 13, pp. 241~246, 1983.
9) A. Agrawal and R. E. Barlow, A Survey of network Reliability and domination Theory, Operations Research, Vol. 32, No. 3, pp. 478~491, May~June, 1984.
10) A. Agrawal and A. Satyanarayana, An O( ) Time Algorithm for Computing the Reliability of a Class of Directed networks, *Operations Research*, Vol. 32, No. 3, pp. 493~515, May~June, 1984.
11) J. S. Provan and M. O. Ball, Computing network Reliability in Time Polynomial in the Number of Cuts, Operations Research, Vol. 32, No. 3, pp. 516~526, May~June, 1984.
12) A. Agrawal and A. Satyanarayana, network Reliability Analysis Using 2-Connected Di-Graph Reductions, *networks*, Vol. 15, pp. 239~256, 1985.
13) R. k. Wood, A Factoring Algorithm Using Polygon-to-Chain Reductions for Computing k-Terminal network Reliability, *networks*, Vol. 15, pp. 173~190, 1985.
14) T. Politof and A. Satyanarayana, network Reliability and Inner-Four-Cycle-Free Graphs, *Mathematics of Operations Research*, Vol. 11, No. 3, pp. 484~505, 1986.
15) T. Politof and A. Satyanarayana, Efficient Algorithms for Reliability Analysis of Planar networks-A Survey, *IEEE Trans. Reliability*, Vol. R-35, No. 3, pp. 252~259, August, 1986.
16) R. k. Wood, Factoring Algorithms for Com-

puting k-Terminal network Reliability, *IEEE Trans. Reliability*, Vol. R-35, No. 3, pp. 269~278, August, 1986.

17) J. A. Buzacott, Node Partition Formula for Directed Graph Reliability, *networks*, Vol. 17, pp. 227~240, 1987.

18) Suresh Rai and Arun Kumar, Recursive Technique For Computing system Reliability, *IEEE Trans. Reliability*, Vol. R-36, No. 1, pp. 38~44, April, 1987.

19) T. Politof, On a Property of Cyclic Covers of p-Graphs, *networks*, Vol. 18, pp. 51~53, 1988.

20) Kwang-won Rhie, Zur Domination und Zuverlaessigkeit linearer Graphen aufgrund ihrer Minimalschnitte, *Dissertation*, Technische Universitaet in Berlin, July. 1994.

21) Y. B. Yoo and Narsingh Deo, A Comparison of Algorithms for Terminal-Pair Reliability, *IEEE Trans, Reliability*, Vol. 37, No. 2, pp. 210~215, June, 1988.

22) F. Harary, *Graph Theory*, Reading, Mass. : Addison-Wesley, 1969.

23) M. N. S. Swamy, k. Thulasiraman, Graphs, networks and Algorithms, Chapter 5, Willey-Intersceince, 1981.