Journal of the Korean Institute
of Industrial Engineers
Vol.20, No.3. September., 1994

팩토링 기법을 이용한 신뢰성 구조도의 신뢰도 계산 알고리즘

A Reliability Computational Algorithm for Reliability Block Diagram Using Factoring Method*

이 창훈**, 김 명규***, 이 상천**

Changhoon Lie**, Myungyu Kim***, Sangcheon Lee**

Abstract

In this study, two reliability computational algorithms which respectively utilize a factoring method are proposed for a system represented by reliability block diagram. First, vertex factoring algorithm is proposed. In this algorithm, a reliability block diagram is considered as a network graph with vertex reliabilities. Second algorithm is mainly concerned with conversion of a reliability block diagram into a network graph with edge reliabilities. In this algorithm, the independence of edges is preserved by eliminating replicated edges, and in computing the reliability of a converted network graph, existing edge factoring algorithm is applied. The efficiency of two algorithms are compared for example systems with respect to computing times. The results shows that the second algorithm is shown to be more efficient than the first algorithm.

Ⅰ. 서론

시스템의 신뢰도는 시스템 설계 및 운영 단계에서, 사용자가 보다 안정적으로 시스템을 사용할 수 있는 시스템의 임무 수행도로 정의 될수 있다. 시스템의 신뢰도 계산은 시스템 전체를 이루는 각각의 구성품 간의 관계를 파악하여 시스템의 신뢰도를 각 부품의

* 본 연구는 1993년도 서울대학교 공대 연구재단의 연구비 지원에 의하여 수행되었음.

** 서울대학교 산업공학과

*** 동양 정보 시스템(주)

신뢰도의 함수로 나타내는 과정을 의미한다.

그런데 기술의 발달과 더불어, 다루어야 할 시스템이 점차 복잡하고 거대해 지고 있으며, 따라서 이 복잡한 시스템의 신뢰도를 구하는 효율적인 방법의 개발이 요구되고 있다.

시스템 신뢰성 계산 과정은 크게 시스템의 모형화와 이모형에 따른 계산 알고리즘의 개발로 나눌수 있다. 먼저 지금까지 개발된 모형화 기법을 살펴보면 Fault Tree 기법, 네트워크 그래프, 신뢰성 구조도(Reliability Block Diagram)이다. Fault Tree기법은 시스템 자체의 고장 외에 외부적인 영향을 나타내는 사건들이 포함된 표현 기법으로써, 시스템의 구조적인 면 보다는 고장의 원인 결과의 관계를 나타내는 Event Diagram으로, 시스템을 표현하는 데 많은 비용과 시간이 필요하다는 단점을 가지고 있다. 네트워크 그래프는 통신 시스템과 같은 시스템의 구조자체가 네트워크 일때 주로 적용하며 vertex에 해당하는 부품이 완전하다는 가정 하에서 주로 사용하는 기법으로써 vertex의 고장이 중요한 영향을 줄 경우에는 적용에 한계가 있다. 이에 반하여 신뢰성 구조도는 시스템 자체의 논리적 구조를 그대로 표현하는 기법으로 시스템 내의 고장을 나타내는 부품들을 블럭들로 나타내고 그 블럭들 사이의 연결선은 완전하다고 가정하는 표현 기법으로, 회로도와 같은 구조도가 주어진 경우, 다른 기법들에 비해 상대적으로 단순하고 시스템으로부터 쉽게 이끌어 낼 수 있다는 장점이 있다. 모형에 따른 시스템 신뢰도 계산 알고리즘을 살펴보면, 개발된 신뢰도 계산 알고리즘들은 대부분 네트워크 중심이거나 혹은 Fault Tree중심의 알고리즘들이었다. 그리고 시스템을 보다 쉽게 이해할 수 있는 신뢰성 구조도에 대한 신뢰도 해법 연구는 아직 부족한 형편이다. 지금까지 개발된 신뢰도 계산알고리즘들은 대부분 네트워크 그래프의 신뢰도 계산 알고리즘으로, 최소 연결집합을 이용하여 시스템의 신뢰성을 구하는 Inclusion-Exclusion 기법, Sum of Disjoint Products 기법과, Edge 분해(Decompsition) 및 축소(Reduction) 기법을 이용하여 신뢰도를 구하는 팩토링(Factoring) 기법이 있으며, 이중 팩토링 기법이 더 효율적인 알고리즘으로 알려져있다.

따라서 본 연구에서는 보다 시스템의 구조도에 가까운 표현 방법인 신뢰성 구조도에서의 시스템 신뢰성을 구하는 효과적인 알고리즘을 제시하고자 한다. 이를 위해 먼저 신뢰성 구조도의 특성을 묘사할 수 있는 수학적 정의를 제시하고, 이 신뢰성 구조도의 신뢰도 계산을 위하여 구하는 두가지 접근 방법을 제시하고자 한다.

첫번째 방법은 신뢰성 구조도에 직접 팩토링 기법을 적용하여 시스템의 신뢰도를 구하는 알고리즘을 개발하는 것이며, 두번째 접근 방법으로는 신뢰성구조도를 edge중심의 네트워크로 변환한후, 기존의 Egde 팩토링 기법을 적용하여 시스템의 신뢰도를 구하는 방법을 제시하는 것이다.

Ⅱ. 신뢰성 구조도의 정의

일반적인 구조도는 그것이 가지는 단순함 때문에 여러 방면에서 많이 사용되어 왔다. 특히 전자, 전기 계통의 분야에서는 회로도의 단순한 표현으로써 사용하고 있다. 그러나 시스템의 표현에만 주안점을 두는 관계로

정확한 정의에 의해 구성된다고 하기 보다는 회로도의 형태를 그대로 단순화하여 구조도를 구성 한다. 일반적으로, 시스템의 신뢰도를 다루는 분야에서는 그와같은 구조도를 이루는 각 부품에 고장이 일어날 확률을 부여함으로써 시스템 전체의 신뢰도를 분석하는 연구를 수행하여 왔으며, 그 이름도 신뢰성 구조도라고 하여 일반적인 구조도와 구분하였다. 지금까지 만들어진 신뢰성 구조도에 대한 정의는 block은 시스템의 작동에 영향을 미치는 사건이나 부품을 의미하고 arc는 block사이의 관계를 의미한다는 것이다. 본장에서는 시스템 신뢰도 계산 알고리즘의 입력요소로서의 신뢰성 구조도의 표현방법을 수학적으로 명확히 정의 하고자 한다. 먼저 본 연구에서 사용된 용어를 정리하면 다음과 같다.

○ 용어 정리

· block ; 시스템의 고장에 영향을 주는 사건이나 부품

· arc : 신뢰성 구조도에서 block사이의 관계를 나타내는 선

· indegree (id_i) ; block i로 들어오는 arc의 수

· outdegree (od_i) ; block i에서 나가는 arc의 수

· vertex (V_i) ; 신뢰성 구조도 에서 network 로 전환할때 logical point를 표시한다.

· logical point ; 신뢰성 구조도에서 vertex의 indegree와 outdegree를 계산하여 2이상일 때와 2개의 vertex가 직렬일 때vertex 앞 또는 뒤에 첨가되는 완전한 vertex

· 가상 edge ; logical point사이를 연결할 때, 두개의 logical point 사이에 block(vertex)이 없을 경우 발생하는 edge

· 중복 edge (er) ; 서로의 작동이 완전히 종속적인 edge.

· 입력 집합 ; 하나의 vertex의 앞에 연결된 vertex들의 집합

· 출력 집합 ; 하나의 vertex의 뒤에 연결된 vertex들의 집합

위의 용어를 사용하여 본 연구에서 사용된 신뢰성 구조도의 정의는 다음과 같다.

○ 신뢰성 구조도의 정의

신뢰성 구조도는 source와 terminal을 갖고 순환로(circuit)가 없는 유방향 네트워크로 다음 성질들을 만족한다.

1. 표현을 위한 요소는 source, terminal, block, arc 등이다.
2. block은 시스템의 고장을 유발시키는 부품을 나타내며 작동 확률 p_i를 갖는다. 각 block의 작동상태는 이진(작동,고장)이며, 서로 독립이다.
3. arc는 block사이의 관계를 나타내며 완전하다(고장 확률=0).
4. 1개의 arc의 양끝에는 1개의 block, 또는 source 또는 terminal만이 올 수 있다.

위의 정의에 의해 bridge 구조를 신뢰성 구조도로 표현하면 그림 2.1과 같다.

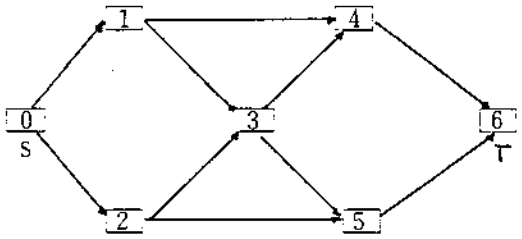


그림 2.1 bridge 구조의 신뢰성 구조도

Ⅲ. 신뢰도 계산 알고리즘

Ⅲ-1 Vertex 팩토링 알고리즘

기존의 팩토링 알고리즘은 무방향 네트워크 그래프를 대상으로, edge의 피봇팅(pivoting) 및 직렬, 병렬, degree 2 축소, polygon-to-chain 축소등의 축소 기법을 통해 신뢰도를 계산한다. 최근에는 vertex와 edge가 함께 불완전한 경우를 고려하여 함께 피봇하는 알고리즘에 대한 연구도 진행되었는데 이 방법도 무방향 네트워크에 대한 연구이다.

유방향 네트워크의 vertex에 피봇을 하는 방법은 피봇을 했을때, 분해되어 발생하는 2개의 작은 시스템의 형태를 유지시키는 방법이 문제가 되어 지금까지는 상대적으로 상당히 비효율적인 것으로 생각되었다. 왜냐하면 edge 피봇을 할 때, 피봇된 edge가 작동한다고 가정하면, 그 edge에 연결되어있는 두개의 vertex를 하나로 붙여서 시스템을 유지할 수 있고, edge가 고장이라고 가정하면, edge가 연결하고 있는 2개의 vertex를 분리하여 시스템을 유지할 수 있다. 반면에, vertex를 피봇할 때, 피봇된 vertex가 작동할 경우 그 vertex에 연결되어 있는 앞의 vertex와 뒤의 vertex를 확인하여 연결해야하고 이때 edge도 불완전하다면 매번 edge의 확률도 수정해야 한다. 또한 이때 발생하는 edge간의 종속성도 고려하여 문제를 풀어야한다. 또 만약 피봇된 vertex가 고장이라고 가정한다면 그 vertex에 연결된 모든 입 · 출력의 edge들을 제거하고 시스템의 구조를 수정하여야 한다. 이것은 전산화 과정에서 많은 기억용량과 계산량을 필요로 한다.

본 논문에서는 arc가 완전한 Vertex중심의 네트워크 시스템의 신뢰도를 구하려고 하므로 집합 개념을 사용하여 위의 문제점들을 어느정도 해결할 수 있다. 즉 모든 vertex에 대하여, 입력으로 들어오는 vertex들의 집합과 출력으로 나가는 vertex들의 집합을 보관하여 vertex들의 연결 관계를 이 집합들로서 확인하면서 피봇팅 기법을 적용하려는 것이다.

본 논문에서 제시하는 vertex 팩토링 알고리즘은 아래와 같이 정리될 수 있다.

Vertex 팩토링 알고리즘

* INPUT : directed graph G
  vertex reliability
* OUTPUT : P = R(G)

단계 1 신뢰성 구조도 인식.
  Case 1) 축소 가능 ⇒ 단계 2.
  Case 2) 축소 불가능 ⇒ 단계 3.

단계 2 series, parallel 축소
  Case 1) source에 terminal이 직접

연결 ⇒ STOP

단계 3 분해

1) vertex 제거

Case 1) graph 분리 ⇒ STOP

Case 2) source에 terminal이 직접 연결 ⇒ STOP

2) vertex 결합

Case 1) source에 terminal이 직접 연결 ⇒ STOP

Otherwise , 단계 2.

위의 알고리즘을 그림 2.1에 적용하면 다음과 같다.

단계1 : 입력 및 초기화

시스템을 입력한 후 시스템의 각 vertex는 다음과 같은 입력 집합과 출력 집합들을 갖게 된다. 또한 시스템 전체의 집합도 기억된다.

vertex 0 : in = { }, out = { 1, 2, 3 }
vertex 1 : in = { 0 }, out = { 3, 4 }
vertex 2 : in = { 0 }, out = { 3, 5 }
vertex 3 : in = { 0, 1, 2 }, out = { 4, 5 }
vertex 4 : in = { 1, 3 }, out = { 6 }
vertex 5 : in = { 2, 3 }, out = { 6 }
vertex 6 : in = { 4, 5 }, out = { }

vertex set = { 0, 1, 2, 3, 4, 5, 6 }

단계2, 3 : 분해 및 축소,

만약 vertex 3가 작동되면 vertex 3의 입력 집합의 원소인 0, 1, 2의 출력 집합이 vertex 3의 출력 집합을 포함하도록 수정하고 vertex 3은 그 집합들에서 제외된다. 또한 시스템 집합에서도 3을 제외시킨다. 결과적으로 아래와 같이 수정된 집합들이 생성된다.

vertex 0 : in = { }, out = { 1, 2, 4, 5 }
vertex 1 : in = { 0 }, out = { 4, 5 }
vertex 2 : in = { 0 }, out = { 4, 5 }
vertex 3 : in = { }, out = { }
vertex 4 : in = { 0, 1, 2 }, out = { 6 }
vertex 5 : in = { 0, 1, 2 }, out = { 6 }
vertex 6 : in = { 4, 5 }, out = { }

vertex set = { 0, 1, 2, 4, 5, 6 }

만약 vertex 3이 고장이면 vertex 3의 출력 집합의 원소 4, 5의 입력 집합에서 vertex 3을 제외하고 또 vertex 3이 출력 집합의 원소인 vertex 1, 2의 출력집합으로부터 3을 제외한다. 또한 시스템 전체의 집합으로부터도 3을 제외시킨다. 결과적으로 다음과 같은 결과가 나타난다.

vertex 0 : in = { }, out = { 1, 2, 4, 5 }
vertex 1 : in = { 0 }, out = { 4, 5 }
vertex 2 : in = { 0 }, out = { 4, 5 }
vertex 3 : in = { }, out = { }
vertex 4 : in = { 0, 1, 2 }, out = { 6 }
vertex 5 : in = { 0, 1, 2 }, out = { 6 }
vertex 6 : in = { 4, 5 }, out = { }

vertex set = { 0, 1, 2, 4, 5, 6 }

위의 과정을 그림으로 나타내면 그림 3.1

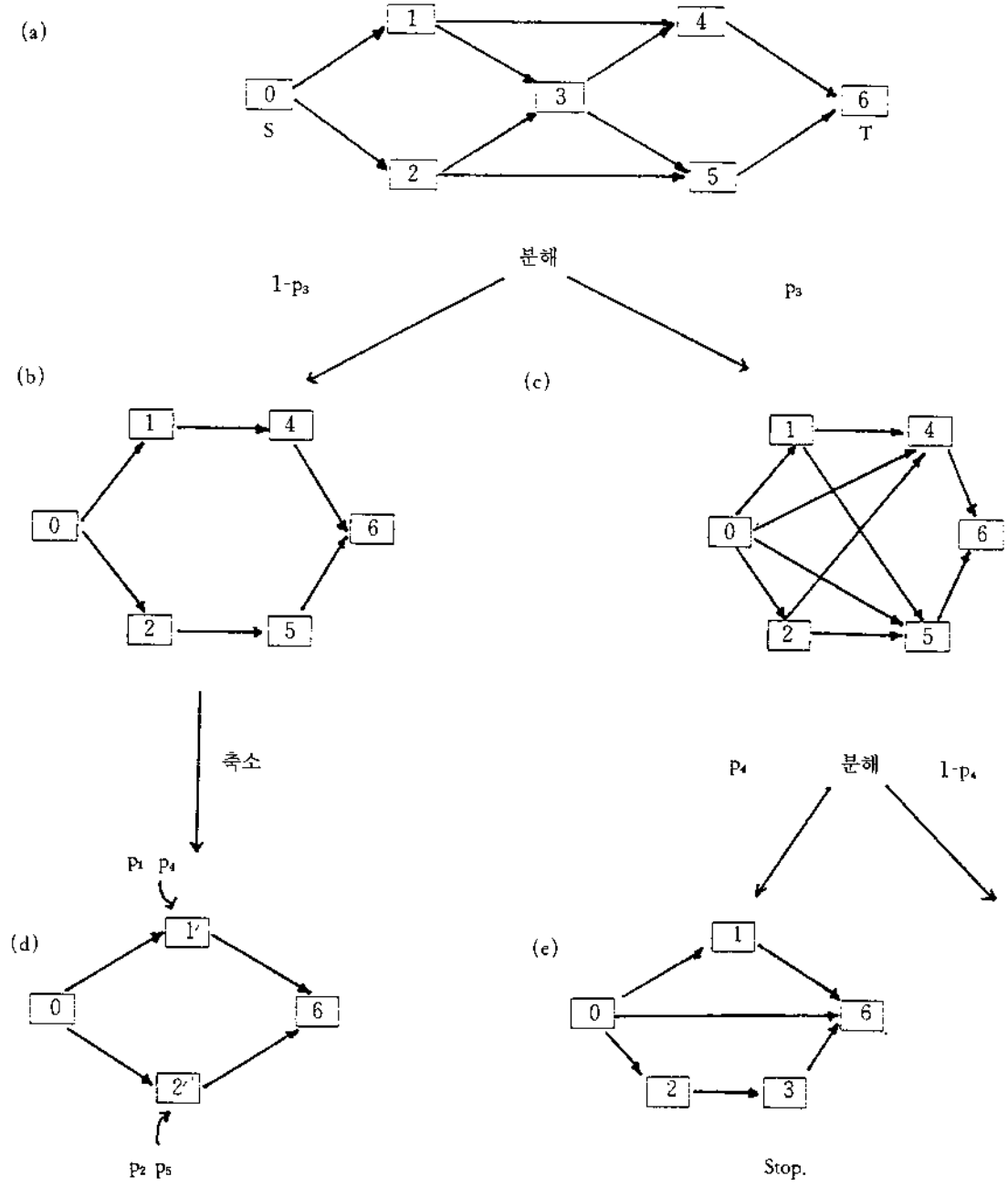


그림 3.1 vertex 피봇에 의한 신뢰도 계산 과정

과 같다.

물론 위의 과정만을 계속하면 단지 분해만을 반복하는 결과가 될 것이다. 그러나 그림 3.1의 (b)의 형태를 보면 vertex 1과 4가 직렬이고 2와 5가 역시 직렬 연결이므로 축소를 할 수 있다. 또 그 결과로 나온 그림 3.1의 (c)는 병렬 구조이므로 역시 축소가 가능하다. 이와같은 과정을 반복하면서 발생하는 매 분지의 끝에서 만약 시스템의 source가 terminal에 직접 연결되면 멈추게 된다. 또한 그래프의 source와 terminal이 완전히 분리되면 멈추게 된다.

Ⅲ-2 Edge 팩토링 알고리즘

edge중심의 네트워크(네트워크 그래프)에 대한 분석 방법 중에서 Edge 팩토링 방법은 경로에 의한 방법들보다 효율적이라고 알려져 있다. 그 이유는 첫째, 경로를 찾지않고 바로 시스템의 신뢰도를 구할 수 있고, 둘째, 분해와 신뢰성 보존 축소를 함께 사용하기 때문에 일반적인 경우 계산량이 위의 두가지 방법들 보다 적게되기 때문이다. 그런데 이 방법을 적용하기 위해서는 우선적으로 네트워크를 이루고 있는 모든 edge들이 확률적으로 독립이어야한다. 본 장에서는 신뢰성 구조도로 표현된 시스템을 네트워크로 변환하여 기존의 팩토링 방법을 적용하여 시스템의 신뢰도를 구할 수 있도록 하려고 한다. 그런데 신뢰성 구조도를 네트워크로 바꿀 때에는, 하나의 블록을 edge에 대응시키고 그 edge에는 부품의 신뢰도가 부여되도록 변환을 시키고, 그 edge들이 모이는 vertex에는 신뢰도 1을 부여하게 되는데, 이 과정에서 다음과 같은 문제가 발생한다. 즉 부품을 나타내는 블록이 edge로 바뀔 때 1 : 1 대응이 아닌 결과가 발생할 수도 있다는 것이다. 이경우 네트워크 그래프에서는 중복 edge가 발생하게 된다. 이렇게 되는 이유는 시스템의 신뢰도를 유지시키기 위해서 변환과정에서 그 시스템의 최소 성공 집합을 유지시키는 과정을 병행하기 때문이다. 이경우, 중복 edge는 하나의 edge의 고장이 동시에 나머지 하나의 고장도 의미하므로 그대로 팩토링 방법을 사용할 수 없기 때문에, 시스템 내의 edge간의 종속성을 제거하기 위해서는 multi-pivoting을 해야하고, 만약 중복된 것이 r개이면 적어도 2^r번의 계산과정이 첨가되어진다. 이와같은 edge간의 종속성의 문제를 제거하기 위해서, 본 장에서는 logical point라는 완전한 vertex를 첨가하고 그 수정된 시스템을 edge 중심의 네트워크로 만들어서 edge 피봇하는 알고리즘을 제시한다. 이 과정에서 필요한 개념이 logical point와 가상 edge이다. 이 개념을 사용하여 네트워크로 변환하는 알고리즘을 정리하면 다음과 같다.

변환 알고리즘

단계 1 : logical point 삽입
(source와 terminal도 logical point로 취급)

Case 1) block의 indegree $\geq 2 \Rightarrow$ block 앞에 logical point 삽입

Case 2) block의 outdegree $\geq 2 \Rightarrow$ block 뒤에 logical point 삽입

Case 3) series 연결 ⇒ 두 block 사이에 삽입

단계 2 : block을 edge로 변환

Case 1) 두 logical point 사이에 block이 있는 경우
⇒ block 신뢰도를 갖는 edge로 변환

Case 2) 두 logical point 사이에 block이 없는 경우
⇒ 신뢰도 1인 edge(가상 edge)로 변환

이 변환 알고리즘을 그림 3.2의 신뢰성 구조도에 적용해보자. 먼저 첫단계로 입력된 신뢰성 구조도의 각각의 block에 대한 indegree와 outdegree를 구하면 다음과 같다.

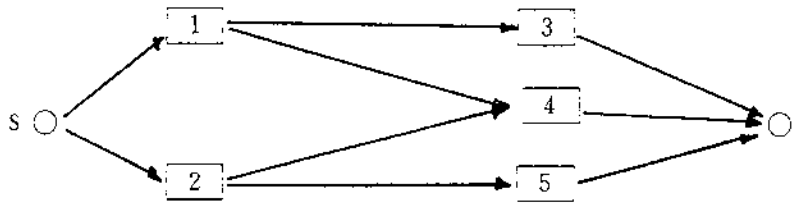
block 1 : indegree = 1 , outdegree = 2
block 2 : indegree = 1 , outdegree = 2
block 3 : indegree = 1 , outdegree = 1
block 4 : indegree = 2 , outdegree = 1
block 5 : indegree = 1 , outdegree = 1

이 중 degree가 2이상인 경우와 직렬인 경우 logical point를 삽입하는데 직렬 연결이 없으므로 degree 2인 경우에만 삽입하면 아래의 그림 3.2의 (b)와 같다. 다음 단계로 각 logical point 사이를 연결하면 그림 3.2의 (c)와 같이 표현된다. 이 때 block의 작동 확률이 edge에 부여되고 중간에 부품이 없는 경우의 edge는 확률 1을 갖게된다.

이와같은 과정에 의해서 구해진 네트워크 그래프를 Page와 Perry에 의해 구현된 Edge 피봇 팩토링 알고리즘을 이용하면 시스템 신뢰도를 구할 수 있다. Page와 Perry에 의해 만들어진 알고리즘은 terminal에 연결된 edge중에서 임의로 하나를 선택하여 피봇해 가면서 신뢰도를 구하도록 되어있으며 직렬, 병렬의 축소를 포함하고 있다.

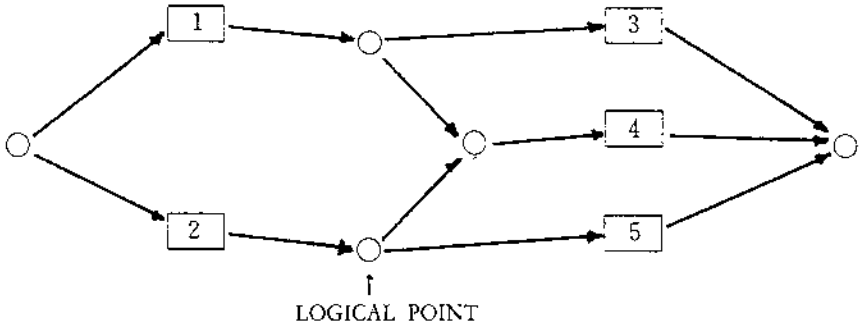
Ⅳ. 예제를 통한 알고리즘 비교

본 장에서 사용한 Vertex 팩토링방법은 신뢰성 구조도를 그대로 네트워크로 인식하여 구하는 방법이다. 이 방법은 또한 직, 병렬 축소를 사용하여 분해법에 비해 계산량을 줄일 수 있다. 그러나 이 방법의 시간 복잡도를 살펴보면 계산 시간이 지수적 증가를 보이게 된다. 또 edge 중심의 팩토링 방법도 마찬가지로 지수적 시간이 필요하게 된다. 그러므로 본 논문에서는 몇 가지 예제를 주어주고 그 계산 시간을 비교하는 방법을 사용하여 두 알고리즘간의 효율성 비교를 하였다. 물론 두번째 방법인 edge 팩토링 방법은 변환 알고리즘에 사용된 시간을 포함하여야 한다. 그러나 실제로 변환 알고리즘의 시간 복잡도를 따져보면 $O(N^3)$이 되며, 여기서 N은 vertex의 갯수를 의미한다. 그러므로 예제가 커지고 복잡해 질수록 이 변환 알고리즘에 의해 걸리는 시간의 비율은 상대적으로 적은 부분을 차지하게 된다. 시스템의 복잡도는 시스템에서의 edge 수와 arc수에 비례한다. 시스템의 복잡도를 증가 시키면서 위의 두알고리즘의 수행시간을 비교 하면 아래 그림4.1과 같다.



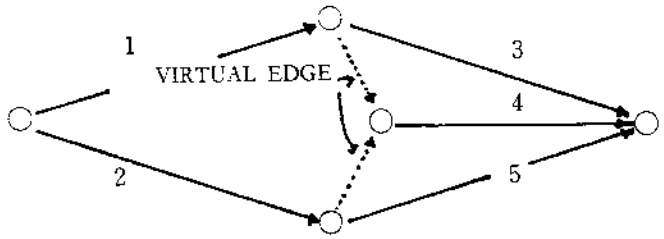
(a) 신뢰도 그래프 (block 4 : replicated edge)

단계 1)



(b)

단계 2)



(c) 변환된 네트워크 그래프

그림 3.2 신뢰성 구조도의 네트워크로의 변환과정

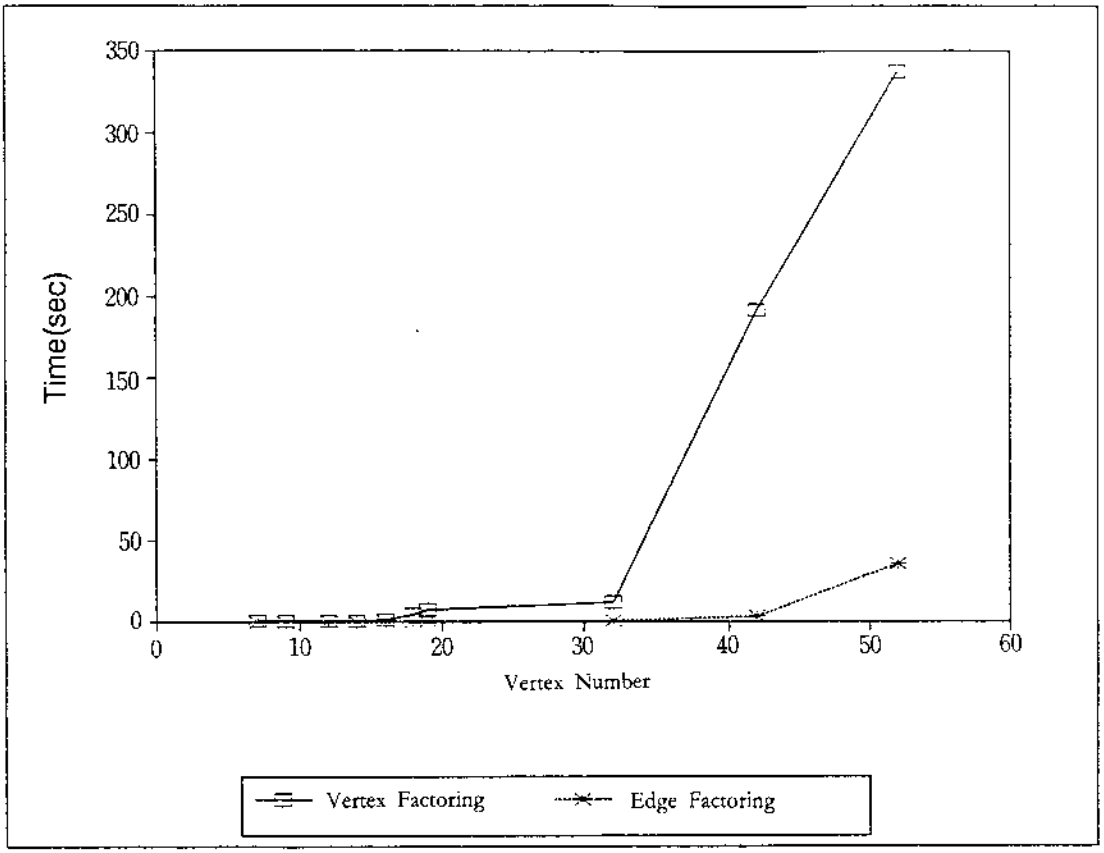


그림 4.1 신뢰성 구조도의 신뢰도 계산 알고리즘의 수행시간 비교
(P=0.5, vertex수/arc수=1.0)

V. 결론

본 연구에서는 시스템이 신뢰성 구조도의 형태로 주어졌을 때 시스템의 신뢰도를 구하는 해법을 제시하였다. 또한 알고리즘의 효율적 사용을 위한 시스템의 표현방법이 제시되었다. 기존의 표현들이 구조의 이해라는 면에만 치중했던 것이라면 본 연구의 표현에서는 계산을 위한 일관적인 입력 양식을 제공한다는 장점이 있다.

본 연구에서 제시된 두 개의 알고리즘은 모두 그 시간 복잡도가 지수적이기 때문에 수행시간을 비교하여 두 알고리즘 간의 효율성을 비교하였다. 비교 결과 네트워크로 변환하여 신뢰도를 구하는 방법이 더 적은 시간을 필요로 함을 알 수 있다. 이것은 결국, 두가지 방법이 모두 block의 수에 종속적인 수행시간을 가지지만, 두번째 알고리즘의 경우처럼 일단 시스템이 네트워크 구조로 바뀌고 나면 첫번째 알고리즘에 비해 축소가 보다 많이 일어날 수 있기 때문이다.

또한 시스템의 위상적 구조에 따라 신뢰도

를 계산하는 시간이 다름을 알 수 있다. 즉, 주어진 신뢰성 구조도가 복잡하여 네트워크로 변환하였을 때 가상 edge의 수가 많아지면, 그 만큼 시스템의 신뢰도를 계산하는데도 많은 시간이 소요됨을 알 수 있다. 마찬가지로 vertex 팩토링을 적용할 경우에도, 위상적 구조가 복잡할수록 보관해야 할 자료가 증가하므로 구조가 단순한 경우보다 계산 시간이 증가함을 알 수 있다. 물론 위에서 제시한 vertex 수 : edge 수의 비율이 시스템의 위상적 구조를 나타내는 명확한 자료를 제시하지는 못한다. 이것도 경우에 따라 달라지기 때문이다. 그러나 이러한 비율이 대략적인 복잡도를 나타내고 있음을 알 수 있다.

지금까지 일반적인 무방향 네트워크 구조의 시스템에 적용한 vertex 피봇팅은 vertex와 edge 두 가지가 모두 불완전하다는 가정 하에 수행된 것이고 이러한 경우, vertex에 대한 피봇팅은 edge에 대한 피봇팅 보다 비효율적이라고 알려져 있다. 그러나 본 논문에서 제안한 알고리즘에서는 vertex만이 불완전하다고 가정하고 팩토링 방법을 적용하였다. 따라서 이 경우에는 Vertex 팩토링 방법이 Edge 팩토링 방법보다 비효율적이라고 일반적으로 생각할 수 없다. 왜냐하면 본 논문에서 제안하는 Vertex 팩토링 알고리즘에서는 시스템의 형태를 기억하기 위한 기억용량 속에 edge에 대한 부분을 필요로 하지 않으므로 계산 과정에서 edge의 고장 확률을 고려하지 않게 되고, edge와 vertex 모두가 불완전한 경우에 비해서 필요한 계산량이 줄게 되기 때문이다.

앞으로 이 Vertex 팩토링 방법의 효율성을 높이기 위해서는 계산과정에서 발생하는 redundant문제를 해결하여야 하고, 보다 많은 축소 기법을 개발해야 할 것이다.

또한, 신뢰성 구조도의 개념을 보다 확대하여 2개 이상의 terminal을 가질 경우의 신뢰성 구조도에 대한 신뢰도를 팩토링 방법에 의해 구하는 연구가 앞으로 진행되어야 할 연구 과제이다. 이 밖에, 신뢰성 구조도에서 필요한 유방향성을 완화하여 vertex 중심의 무방향 네트워크에 대한 Vertex 팩토링 알고리즘을 찾는 문제가 제시될 수 있다. 이러한 연구는 실제로 vertex 중심의 무방향 네트워크에서 어느 특정의 두 지점 간을 연결할 신뢰도를 구하려 할 경우 유용하게 쓰일 수 있을 것이다.

참 고 문 헌

[1] K.K. Aggarwal, K.B. Misra, J.S. Gupta, "A Fast Algorithm for Reliability Evaluation", *IEEE Trans. Reliability*, vol R-24, 1975, Apr., pp 83-85.

[2] A. Agrawal, "Reliability Analysis of Rooted Directed Networks", Ph. D. Thesis, Department of Operations Research , 1982.

[3] M.O. Ball, "Network Reliability Analysis: Algorithms and Complexity", Ph. D. Thesis, Department of Operations Research , Cornell University, 1977.

[4] R.G. Bennetts, "Analysis of Reliability Block Diagrams by Boolean Techniques", *IEEE Trans. Reliability*, vol R-31, 1982, Jun., pp 159-165.

[5] T.B. Boffey, R.T.M. Waters, "Calculation

of System Reliability by Algebraic Manipulation of Probability Expressions", *IEEE Trans. Reliability,* vol R-28, 1979, Dec. pp 358-363.

[6] P.A. Jensen, M. Bellmore, "An Algorithm to Determine the Reliability of a Complex System", *IEEE Trans. Reliability,* vol R-18, 1969, Nov., pp 169-174.

[7] A. Satyanarayana and M.K. Chang, "Network Reliability and the Factoring Theorem", *NETWORKS,* vol 13, 1983, pp 107-120.

[8] A. Satyanarayana and A. Prabhakar, "New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Networks", *IEEE Trans. Reliability,* R-27, pp 82-100.

[9] A.Satyanarayana and R.K. Wood, "Polygon to Chain Reductions and Network Reliability", Technical Report # ORC 82-4, Operations Research Center, University of California, Berkeley, California, 1982.

[10] E.R. Woodcock, "The Calculation of Reliability Systems ; The Program NOTED", AH5B(S)R153. Authority Health and Safety Branch 1968; U.K.E.A.; 11 Charles Ⅱ Street; London SW1 England.

[11] 홍정식, " 종속인 구성 요소들로 이루어진 네트워크의 신뢰성 모형 및 분석, " 1989, 서울대학교 공학 박사 논문.