〈研究論文〉

Journal of the Korean Institute of Industrial Engineers
Vol. 12, No. 1, June 1986

시간제약하의 네트워크 신뢰성 계산에 대한 알고리즘

(An Algorithm for Computing the Source-to-Terminal Reliability in the Network with Delay)

洪 渟 植*
李 昶 勳*

Abstract

In this paper, we are modeling the problem of the reliability evaluation in the network with delay. The triconnected decomposition and factoring algorithm for the network reliability, known as the most efficient algorithm, does not work in this constrained problem. So, we propose some ideas that reduce the above constrained problem to the general network reliability problem. We also present an algorithm for the reliability evaluation in the network with delay based on these ideas.

1. 서 론

네트워크신뢰성(network reliability)의 계산은 컴퓨터네트워크나 통신네트워크 그리고 수송네트워크의 분석에 있어 매우 중요하다. 이러한 네트워크신뢰성의 척도로는 source-to-terminal reliability, all-terminal reliability, expected number of connected pairs 등이 있다 [2].

여기서 source-to-terminal reliability는 주어진 네트워크에서 source와 terminal이 연결될 확률을 의미한다. 그런데 특별히 통신네트워크를 고려해 보면 source와 terminal이 교신을 하고자 할때 시간제약으로 인하여 특정 path는 전혀 사용을 않는 경우가 발생하게 된다. 이와 같은 문제를 Park과 Tanaka [5]는 delay가 있는 네트워크의 신뢰성측정이라는 문제로 모형화 하였다. 즉 주어진 네트워크의 vertex와 edge에 delay가 추가된 경우에 특정 vertex가 주어진 delay의 상한선내에서 나머지 모든 vertex와 교신될 확률을 구하는 알고리즘을 제시하였다. 그들의 알고리즘은 Boolean Algebra 기법을 이용함으로써 모든 vertex에서의 delay가 같고 또한 vertex 신뢰성을 1로 두는 무리한 가정을 문제의 모형화에 도입하였다.

*서울大学校 工科大学 産業工学科

본 논문에서는 이와 같이 delay가 있는 네트워크에서 source와 terminal이 delay의 주어진 상한선을 벗어나지 않는 범위안에서 교신할 확률을 구하는 효율적인 알고리즘을 제시하고자 한다. 이러한 문제의 모형화에 있어 본 논문은 위의 두 가정을 제거하고 있다. 이런 경우 delay가 있는 네트워크신뢰성 계산의 문제를 일반 네트워크신뢰성 계산의 문제로 변환하는 과정을 제시하고자 한다.

본 논문은 실례로 통신네트워크를 들어 설명해 나가고 있으나 수학적으로는 vertex와 edge에 신뢰성이 추가된 확률적 graph에 delay가 또한 추가된 네트워크의 신뢰성 측정으로 표현될 수 있으며, 이러한 척도는 통신네트워크 외에도 여러 네트워크의 성능 분석에 있어 유용한 도구가 될 수 있을 것이다. 또한 일반적인 네트워크 신뢰성 계산에 대한 위의 알고리즘을 delay가 추가된 네트워크 신뢰성 계산에 있어 수정·보완하여 사용하는 과정은 이들 알고리즘의 또다른 이론적 측면을 드러내 줄 것이다.

2. 문제의 모형화

기 호

V : vertex 들의 집합 $\{v_1, \cdots, v_n\}$

E : edge 들의 집합 $\{e_{ij} | v_i$와 v_j를 연결하는 edge$\}$

G(V, E) : V와 E로 이루어진 graph, 간단히 G로도 표기함.

s : 특정 vertex인 source

t : 특정 vertex인 terminal

p_i (d_i) : vertex v_i의 신뢰성(delay)

p_{ij} (d_{ij}) : edge e_{ij}의 신뢰성(delay)

Ai : s와 t를 연결하는 source-to-terminal path, 간단히 path로 표기함.

$L(A_i)$: path A_i를 구성하는 vertex와 edge 들의 delay의 총합, path A_i의 길이(length)로 정의함.

D : delay의 상한선

$N(G, p_i, p_{ij}, d_i, d_{ij}, s, t, D)$: graph(G)와 신뢰성(p_i, p_{ij})와 delay(d_i, d_{ij})와 delay 상한선(D) 그리고 특정 vertex (s, t)로 이루어진 네트워크.

Ge : graph G에서 edge e에 인접한 두 vertex를 융합한 graph

G-e : graph G에서 edge e를 제거한 graph.

∐ : $p_1 ∐ p_2$는 $1-(1-p_1)(1-p_2)$를 나타내는 기호.

가 정

(1) vertex와 edge의 상태는 작동과 작동중지의 2가지 상태이다.

(2) vertex와 edge들은 서로 독립이다.

위의 2가지 가정하에서 delay가 있는 네트워크 신뢰성을 구하고자 할때, 기존 네트워크 신뢰성 문제에 제약으로 작용하는 변수가 vertex와 edge의 delay인 d_i와 d_{ij}이다. 그런데 여기서 다음과 같은 간단한 변형에 의해 vertex의 delay를 edge의 delay로 변환할 수 있다. 즉,

$$d_{ij}' = d_i + d_j + d_{ij} \cdots\cdots\cdots \quad (1)$$

이와 같은 변형이 가능한 것은 우리가 신뢰성 계산에 있어서 사용하는 path의 경우에 이 path가 edge e_{ij}를 사용할 경우 반드시 vertex v_i와 v_j를 사용하므로 d_i와 d_j와 d_{ij}는 항상 함께 path의 길이에 사용되기 때문이다. 위의 간단한 변형에 의해서 우리는 다음과 같은 네트워크를 다루게 된다. 즉,

$N = (G, p_i, p_{ij}, d_i, s, t, D)$

이러한 네크워크에서 우리는 path A_i의 길이 $L(A_i)$가 D보다 작은 path들 만을 사용하여 주어진 특정 vertex s와 t가 서로 연결될 확률을 구하고자 한다. 이것을 수식으로 표현하면

$$R_{st}^{D}(G) = \Pr\left(\bigcup_{A_i \in K_1} A_i\right) \cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (2)$$

여기서 R^{D}_{st} (G) : 주어진 네트워크에서 특정 vertex s와 t가 delay의 상한선을 위반하지 않고 연결될 확률.

K_1 : delay의 상한선을 위반하지 않는 path 들의 집합 즉 $\{A_i | L(A_i) \leq D\}$

K_2 : delay의 상한선을 위반한 path들의 집합 즉 $\{A_i | L(A_i) > D\}$

이제 R^{D}_{st} (G)를 효율적으로 구하는 알고리즘을 살표보자.

3. 알고리즘

일반적인 source-to-terminal reliability를 구하는 기법은 크게

(i) inclusion-exclusion method와

(ii) pivotal decomposition method와

(iii) Boolean algebra method가 있다[1].

그런데 최근 Satyanarayana와 Chang [7]과 Wood [8], 그리고 Hagstrom [4]의 새로운 reliability-reduction 기법과 domination theory의 발견에 의해 방법(ii)는 가장 효율적인 알고리즘을 제공해 주고 있다. 이 알고리즘은 factoring 알고리즘등으로 불리우며 이의 요체를 수식으로 표현하면,

$$R_{st}(G) = p_e \cdot R_{st}(G \mid e) + (1 - p_e) R_{st}(G \mid \bar{e}) \quad \cdots\cdots (3)$$

여기서 R_{st} (G | e) : edge가 작동하는 경우의 source-to-terminal reliability.

R_{st} (G | $\bar{e}$) : edge e가 작동 중지인 경우의 source-to-terminal reliability.

그런데 식(3)을 계속 적용하면 2^n개의 항이 나오게 되어 가장 원시적인 경우가 된다. 그러나 factoring 알고리즘에서는 여러가지 reliability-preserving reduction을 $R_{st}(G|e)$와 $R_{st}(G|\bar{e})$의 계산에 사용하게 되어 우리가 다루는 항의 수가 줄어든다. 그러나 원래의 graph에서 어느 하나의 path라도 사용불가능한 경우에는 $R_{st}(G|e)$와 $R_{st}(G|\bar{e})$가 대상으로 하는 graph인 Ge와 G-e를 얻을 수 없게 된다. (즉 G에서의 사용불가능한 path를 Ge와 G-e에서 유지할 수 없다). 여기서 다루는 delay가 있는 네트워크가 바로 이 경우로 factoring 알고리즘을 직접 사용할 수가 없기 때문에 일반 네트워크 계산문제로 변환하는 과정을 고려해 보고자 한다. 즉, K_2에 속하는 path의 수가 적을 경우 K_2에 속하는 path A_i에 대해서 이 path의 edge 중 K1에 요소로 사용되지 않는 edge가 있다고 하자. 이때 이 edge들을 제거하면 그 결과로 subgraph에서는 원래의 path A_i를 사용치 않는 것이 된다. A_i에 대응되는 이러한 edge들의 집합을 C_i라 할때, 이러한 문제는 다음의 2경우로 나뉘어 진다.

경우(1) : K_2에 속한 모든 A_i에 대해 C_i가 공집합이 아님.

경우(2) : K_2에 속한 일부의 A_i에 대해 C_i가 공집합.

경우(1)일때, 원래의 graph G로부터 모든 C_i에 속한 edge들을 제거한 graph 을 G′라 하면 R^{D}_{st} (G)는 바로 $R_{st}(G')$와 동등해지고 우리는 일반적인 네크워크 신뢰성에 대한 알고리즘을 사용하게 된다. 그러므로 가장 효율적인 알고리즘인 graph decomposition과 factoring 알고리즘을 사용할 수 있다. 경우(2)일때는 우선 모든 C_i에 속한 edge들을 제거함으로써 G보다 간단한 graph G″를 얻게 된다. 이 경우는 G″에 아직 사용을 할 수 없는 path가 존재하므로 일반적인 네트워크 신뢰성 알고리즘을 수정없이 사용할 수 없다. 여기서 인위적으로 그러한 path의 joint probability를 0으로 두면,

$$R^{D}_{st}(G) = R^{D}_{st}(G'') = \Pr\Big(\bigcup_{A_i' \in K_1} A_i'\Big) = \Pr\Big(\bigcup_{A_i' \in K_1 \cup K_2} A_i'\Big)$$

여기서 A_i' : graph G″에서의 path

$\Pr(A_i') = 0, \quad i \in K_2$

그러면 모든 path를 사용하게 되므로 일반적인 source-to-terminal reliability 문제로 축소되었는데 그에 따라 2절에서의 가정(2)가 성립하지 않게 된다. Satyanarayana [6]의 TRAP 알고리즘은 edge가 서로 종속인 경우에도 적용되므로 TRAP 알고리즘을 써서 이 경우를 해결 할 수가 있다.

이러한 아이디어에 따라 알고리즘을 작성하면 다음과 같다.

Input : $N(G,\ p_i, p_{i,j}, d_{i,j}, D,\ s,\ t)$

Output : $R_{st}^{D}(G)$

Method

Step 1 : Classification of the paths

If $L(A_i) \le a$, then $A_i \in K_1$

If $L(A_i) > a$, then $A_i \in K_2$

Step 2 : Construction of C_i

If C_i is nonempty for all $i \in K_2$, go to step 3. Otherwise go to step 4.

Step 3 : Factoring algorithm

Delete all edges of C_i from the graph $G(V, E)$ for all $i \in K_2$

Apply the factoring algorithm to the subgraph $G'(V', E')$

where, $E' = E \setminus \bigcup_{A_i \in K_2} \{C_i\}$

Step 4 : TRAP algorithm

Delete all edge of C_i from the graph $G(V, E)$ for all $i \in K_2$

Set $Pr(A_i) = 0$, for $A_i \in K_2$ and C_i empty.

Apply the TRAP algorithm to the subgraph $G'(V', E')$ with the above joint probability.

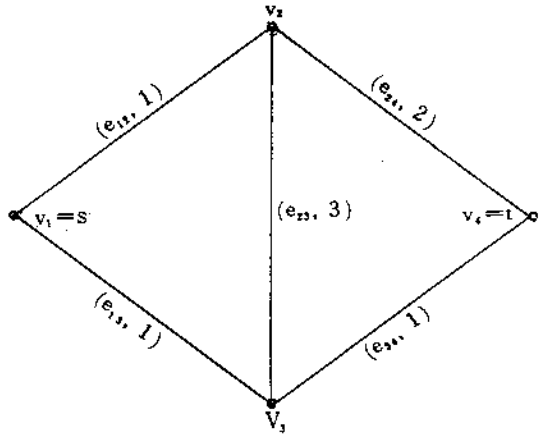

그림 1. 네트워크 $N = (G,\ p_i,\ p_{i,j},\ d_{i,j},\ s,\ t,\ D)$

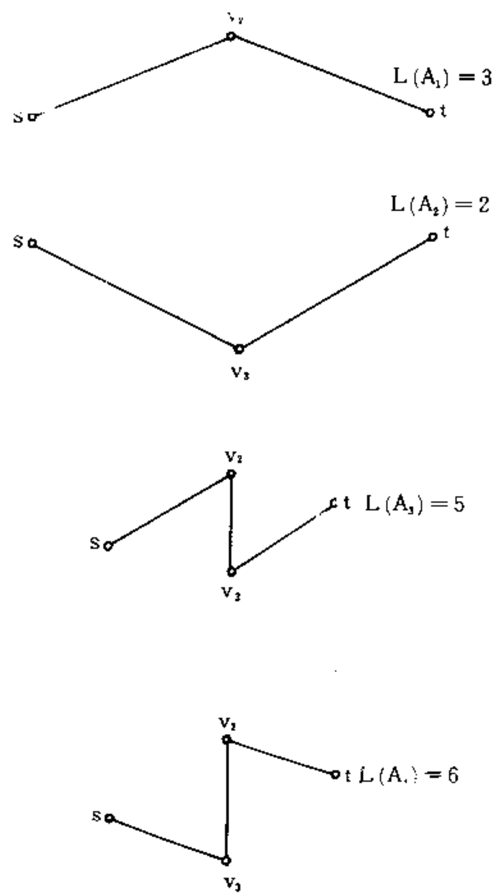

그림 2. 그림 1의 graph의 path와 길이

4. 실 례

그림1과 같은 간단한 네트워크를 고려해 보

자. 여기서 $s = v_1$ 이고 $t = v_4$ 이며 $D = 5$이고 d_{ij}는 $(e_{ij}, \cdot)$로 주어져 있다. 이 graph에서 path와 그것의 길이는 그림2와 같다.

그러므로 step1을 적용하면

$K_1 = \{A_1,\ A_2\}$이고, $K_2 = \{A_3,\ A_4\}$이다. 따라서 step2를 적용하면

$C_3 = \{e_{23}\}$, $C_4 = \{e_{23}\}$임을 알 수 있다. 이 경우 A_3와 A_4에 대해 각각 A_1과 A_2에 없는 edge인 e_{23}를 갖고 있으므로 step 3으로 가게 된다. step3에서 e_{23}을 graph G로부터 제거하면 그림3의 간단한 graph이 된다. 이 경우 factoring algorithm에서 곧바로 reliability-preserving reduction을 쓰면,

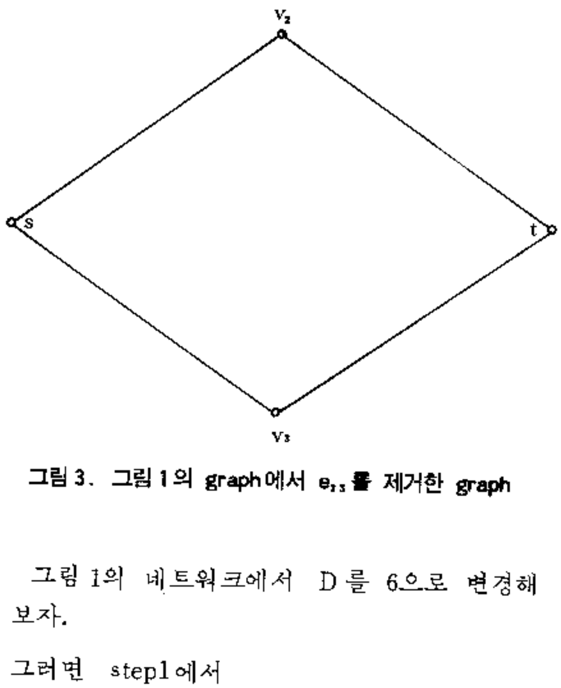

그림 3. 그림 1의 graph에서 e_{23}를 제거한 graph

그림 1의 네트워크에서 D를 6으로 변경해 보자.

그러면 step1에서

$K_1 = \{A_1,\ A_2,\ A_3\}$이고 $K_2 = \{A_4\}$가 된다. step2를 적용하면 A_4에 있는 edge 모두가 K_1에서 요소로 사용되므로 $C_4 = \phi$인 것을 알 수 있다. 따라서 이 경우는 step4로 가게 된다. TRAP 알고리즘을 사용하여 source-to-terminal reliability를 구하고자 할때 graph를 p-directed-graph [6]으로 변경시켜서 구하므로 그림1의 p-digraph을 보면 그림4와 같다.

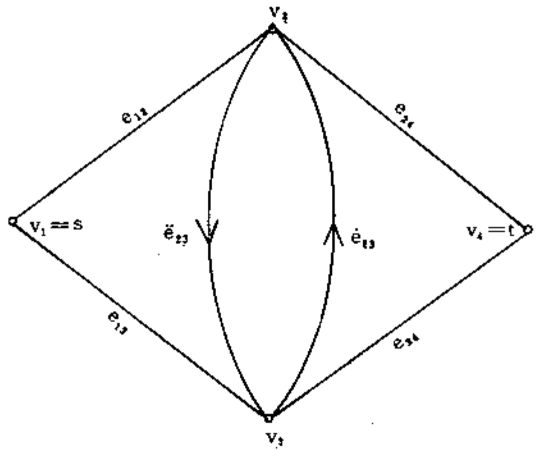

그림 4. 그림 1의 P-digraph

이 경우 인위적인 joint probability는 $P_r\{A_4\} = P_r\{e_{13},\ e_{23},\ e_{24}\} = 0$이므로 TRAP 알고리즘의 output 중 $\{e_{13}, e_{23}, e_{24}\}$를 제거한 output은 다음 표1과 같다. 따라서,

$$
\begin{aligned}
R^{6}_{st}(G) &= R_{st}(G'') \\
&= p_{12}p_{24}p_1p_2p_4 + p_{13}p_{34}p_1p_3p_4 + \\
&\quad p_{12}p_{23}p_{34}p_1p_2p_4 - p_{12}p_{13}p_{23}p_{34}p_1p_3p_4 \\
&\quad - p_{12}p_{23}p_{24}p_{34}p_1p_2p_3p_4 - p_{12}p_{13}p_{24} \\
&\quad p_{34}p_1p_2p_3p_4 + p_{12}p_{13}p_{23}p_{24}p_{34}p_1p_2p_3p_4
\end{aligned}
$$

표 1. 그림 4에 대한 TRAP output

i	$G_{a,i}$	$P_r\{G_a, i\}$
1	$e_{12}e_{24}$	$+p_{12}p_{24}p_1p_2p_4$
2	$e_{13}e_{34}$	$+p_{13}p_{34}p_1p_3p_4$
3	$e_{12}\ddot{e}_{23}e_{34}$	$+p_{12}p_{23}p_{34}p_1p_3p_4$
4	$e_{12}e_{13}\ddot{e}_{23}e_{34}$	$-p_{12}p_{13}p_{23}p_{34}p_1p_3p_4$
5	$e_{12}\ddot{e}_{23}e_{34}e_{24}$	$-p_{12}p_{23}p_{34}p_{24}p_1p_2p_3p_4$
6	$e_{12}e_{13}e_{24}e_{34}$	$-p_{12}p_{13}p_{24}p_{34}p_1p_2p_3p_4$
7	$e_{12}e_{13}\ddot{e}_{23}e_{24}e_{34}$	$+p_{12}p_{13}p_{23}p_{24}p_{34}p_1p_2p_3p_4$

5. 결론 및 토의

네트워크신뢰성 계산문제는 NP-hard 문제임이 입증되었다[3]. 그리고 우리가 다루는 대부분의 네트워크는 그 크기가 방대하여 이처럼 방대한 네트워크의 신뢰성을 효율적으로 계산하기 위한 연구가 진행되고 있다. 그 결과 최근에 사용되는 접근방법으로는 우선 네트워크를 보다 크기가 작고 connectivity 가 큰 subnetwork 로 나누어서 각각의 subnetwork 의 신뢰성을 구하여 전체 네트워크의 신뢰성을 구하게 된다. 그리고 본론에서 언급한 바 이러한 subnetwork 의 신뢰성 계산에는 factoring algorithm 이 가장 효율적이다. 그런데 네트워크신뢰성의 척도중 가장 간단한 source-to-terminal reliability 에 있어 s-t path 중 하나라도 사용을 할 수 없는 경우에는 위의 네트워크 decomposition 과정과 factoring algorithm 을 그대로 사용할 수 없게 된다. 따라서 본 논문에서는 이와 같은 경우에 사용할 수 있는 알고리즘을 제시하였다. 이 방법은 특히 delay 를 위반한 path 가 적을수록 그리고 graph 의 edge 밀도가 적을수록 유용할 것으로 생각된다.

References

1. Agrawal, A. and R. E. Barlow (1982), "A Survey of Network Reliability" *ORC 83-5*, Operations Research Center, Univ. of California, Berkeley.
2. Ball, M. O. (1979), "Computing Network Reliability" *Operations Research* ,Vol.27,pp.832-838.
3. Ball, M. O. (1980), "Complexity of Network Reliability Computations", *Networks*, Vol. 10, pp. 153-165.
4. Hagstrom, J. N. (1983), "Using the Decomposition-Tree of a Network in Reliability Computations", *IEEE Trans. on Reliability*, Vol. R-32, pp. 71-78.
5. Park, Y. J. and H. Tanaka (1979), "Reliability Evaluation of a Network with Delay", *IEEE Trans. on Reliability*, Vol. R-28, pp. 320-324.
6. Satyanarayana, A. and A. Probhakar (1978), "New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Networks", *IEEE Trans. on Reliability*, Vol. R-27, pp. 82-100.
7. Satyanarayana, A. and M. K. Chang (1983), "Network Reliability and the Factoring Theorem", *Networks*, Vol. 13, pp. 107-120.
8. Wood, R. K. (1982), "Polygon-to-Chain Reductions and Extensions for Reliability Evaluations of Undirected Network", *ORC 82-12* , Ph.D. Thesis., Univ. of California, Berkeley.