네트웍 구조의 다중 단말 신뢰도
(K-Terminal Network Reliability)

김 국*, 송기원**

ABSTRACT

시점이 1개 있고 연결되어야 할 단말이 다수개인 K-terminal 네트웍의 신뢰도 구조에서 신뢰도를 구하는 알고리즘을 제안하였다. 네트웍 구조의 신뢰도 계산은 일반으로 NP-hard 문제인데 여기서 새로운 해법을 제안한다. 두 가지 개념이 중요한 점인데 첫째는 분해법이고 두 번째는 재귀식 계산 방법이 가능한 점이다.

분해법을 할 때 키스톤 부품을 찾아내는 번거로운 절차 대신 시점으로 부터 전진방향(forward)으로 하나씩 구성품을 선택하여 분해한다. 이러한 방법은 어떠한 키스톤 부품을 선택해야 할지 기준을 생각할 필요가 없으므로 간단하며 알고리즘을 간단하게 만든다.

또한 이 방법에서는 분해에 의해서 두 개의 하위 문제가 생성되고 원 문제와 재귀관계를 수립할 수 있다. 이러한 재귀식 알고리즘은 컴퓨터 프로그램을 간단하게 만든다. 또한 하위 문제는 기억장치에 저장해 두고 차례로 계산에 사용한다.

1. 서론

신뢰도 구조의 분석은 직렬·병렬화 방법, 발생사건 나열법, 최소절단집합·연결집합 결합방법, 분해법(decomposition)과 같은 기법이 있다[Shooman,1968]. Hansler[1972]는 모든 링크가 양방향이고 노드가 확률로서 링크와 마찬가지로 고장 나는 통신 네트웍의 신뢰도 해법을 연구하였다. 네트웍 구조의 다중 단말 (K-terminal) 신뢰도는 시점으로부터 K개 단말 노드까지 경로가 작동할 확률이다. 통신과 인터넷이 발달한 요즘에 중요한 연구주제가 된다. 일반적으로 네트웍 신뢰도문제는 NP-Hard[Wood 1988]로 알려져 있다. Resende[1986]은 직병렬화 가능한 무방향성 네트웍 포트란 프로그램을 설명했다. Bell & Cameron[1986]은 방향성의 희박(sparse) 네트웍의 모든 단말연결 신뢰도를 설명하였다. Bailey & Kulkami [1986]은 단속시간의 마코브 연쇄로서 접근하였고, Ball[1986]은 네트웍 신뢰도의 계산복잡성에 대해 조감하였다. Park & Kim [1990]은 복잡한 네트웍 구조의 시점-종점 신뢰도를 재귀적으로 역진으로 분해법을 진행하는 당기고 가지치기(pull-prune) 해법과 이의 컴퓨터 프로그램을 설명하였다. Ahn, Kim & Ree[1996]은 경로집합, 절단집합을 짝으로 고려하여 확률의 덧셈법칙을 사용하는 해법을 제시하였는데 다른 방법보다 상대적으로 효과적이었다. Satyanarayana & Chang[1983]은 직병렬 단축과 함께 분해법을 사용하여 K-단말 신뢰도 계산의 복잡성 분석에 우월론을 적용하였다.

여기서는 K-단말 신뢰도를 구하는 해법을 제시한다. 기본 개념은 분해법을 재귀적으로 진행하는 것이다.

* 서울 성북구 정릉동, 서경대학교, kkim@skuniv.ac.kr
** 프라이즈텍 대표, keeweesong@hanmail.net

2. 문제 및 분해법

2.1 문제와 용어

문제의 네트웍을 그래프 G= {V, E}라고 하면 V= { 0, 1, 2, ⋯ }의 노드와 E = { (0,1), ⋯ }의 링크를 가진다. 시점노드의 번호는 0으로 한다. 연결되어야 할 단말 집합을 K라고 한다. 시점을 K에 포함할 수도 있으나 여기서는 제외하는 것으로 정의한다. 물론 K⊆V 이다. 링크의 신뢰도는 확률적으로 독립이고 행렬 P=[p_{ij}]는 신뢰도 구조를 표현하는 행렬이라 한다. 구하고자 하는 것은 K-단말 신뢰도 R(P, G(V, E), K)을 구하는 것이고 간단하게 R(P)라고 하자.

p_{ij} : (i,j) 링크의 단방향성 신뢰도
명목링크 : 신뢰도 1인 링크(항상 작동하는 구성품과 같음)
첫 링크 : 현 소스 노드로부터 나가는 링크
복귀 링크(루프) : 내부 노드로부터 소스 노드로 재방향 설정된 링크
직렬노드 : 두 직렬 링크 사이에 낀 노드
첫 노드 : 첫 링크의 도착 노드

2.2 분해법

분해법을 이용하면 다음과 같은 식으로 표현된다. (i,j) 링크를 피봇링크라 하고,

$$R(P) = R(P \mid p_{ij}=1) + (1-p_{ij})R(P \mid p_{ij}=0). \qquad (1)$$

여기서 (1)식 우변의 첫 번째 구조에서 피봇링크는 명목링크가 되고 두 번째 구조에서는 단절된 모습이 된다.

(1)식을 분해구조로 재귀 순환적으로 적용할 수 있다. 중요한 성질은, 명목링크로 연결된 노드들이 있을 때, 이들 간의 비명목 링크의 신뢰도는 문제의 신뢰도를 변화시키지 않는다는 점이다. 그 이유는 이미 "확실하게" 연결되어 있기 때문에 다른 링크의 작동 여부는 문제되지 않기 때문이다. 예를 들어 아래 그림에서 {0,1,2}가 확정적으로 연결되어 있기 때문에 (1,2)링크의 작동여부는 전혀 문제되지 않는다.

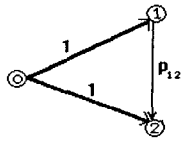

그림 1

2.3 해법

분할된 하위 구조는 P 가 재조정되고 새로운 "식별번호(identification number)"를 붙여서 잠시 유보해 놓는다. (P 는 뒤에서 보듯이 연결된 노드들로부터 연결 안된 노드들로의 링크 확률만 유지하면 된다. 즉 행렬 크기가 줄어드는 효과가 있다.)

식별번호 n 의 구조는 다음과 같은 정보집합을 갖게 된다.

P_n : 링크 확률 행렬

C_n : 연결집합. 즉 시점으로부터 명목링크들로 연결된 노드 집합

C'_n : C_n 의 여집합, 즉 연결되지 않은 노드 집합

K_n : K 중에서 연결되지 않은 나머지 단말들(remaining terminals)

r_n : 이 구조를 갖게 될 확률

초기의 구조는 $P_0 = P$, $K_0 = K$, $C_0 = \{0\}$, $r_0 = 1$ 이다.

그리고 하위구조 식별번호 집합을 N이라 하자. 최초의 N={ #0 }이다.

단계 0 : 현재의 문제식별번호 n=0라 하자. n을 N에서 제거한다.

단계 1 : 종료규칙 1: $K_n = \{\ \}$ 이면 종료한다. 그리고 r_n를 되돌려 준다.

단계 2 : $C_n \rightarrow C'_n$ 로의 링크 (z,y)를 찾는다. 선택기준은 "최소의 노드번호순"으로 한다. 즉, $p_{zy} > 0$인 $z \in C_n$, $y \in C'_n$.

종료규칙 2 : (z,y)가 없으면 종료한다. 그리고 0을 되돌려 준다.

단계 3 : (z,y)를 피봇링크로 삼아 분해법을 적용하여 다음과 같은 새로운 하위 문제 2개, 즉 n1, n2를 만든다. 생성된 문제는 문제집합 N에 추가한다. 여기서 n1은 N 중 가장 큰 번호+1, n2 = n1 + 1이다.

#n1. [$p_{zy} = 1$ 의 구조]

(z,y)가 명목링크가 되고 y는 연결집합의 요소로 옮겨진다.

$r_{n1} = r_n \cdot p_{zy}$,

P_{n1} : $p_{zy} = 1$, 다른 항은 그대로,

$C_{n1} = C_n + y$.

만일 y가 K_n 의 요소이면

$K_{n1} = K_n - y$,

아니면,

$K_{n1} = K_n$.

#n2. [$p_{zy} = 0$ 의 구조]

(z,y)는 단절되고 문제번호 n의 정보가 대부분 유지된다.

$r_{n2} = r_n \cdot (1 - p_{zy})$

P_{n2} : $p_{zy} = 0$, 다른 항은 그대로,

$C_{n2} = C_n$

$K_{n2} = K_n$.

단계 4 : N에서 문제 하나를 꺼내어 단계1로 가고, 과정을 재귀적으로 반복한다.

컴퓨터 프로그램의 경우 재귀함수 기능을 이용할 수 있으므로 이 해법은 매우 간단하게 표현된다.

2.4 예제

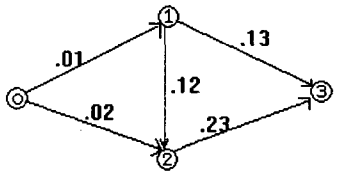

그림 2

문제번호 #0

C_0 = { 0 } C'_0 = { 1, 2, 3 } K_0 = { 2, 3 }, r_0 = 1.

$$P_0 = \begin{bmatrix} 1 & .01 & .02 & . \\ . & 1 & .12 & .13 \\ . & . & 1 & .23 \\ . & . & . & 1 \end{bmatrix}$$

※ 이후 P의 모습 대신 그래프로 보이기로 한다.

피봇요소로서 (0,1) 선택, 종료규칙은 해당되지 않으며 새로 생성된 문제에 의해 문제집합은 { #1, #2 }이 될 것이다.

#1, (0,1)의 연결

C_1 = { 0, 1 }

K_1 = { 2, 3 }

$r_1 = r_0 * p_{01} = 0.01$

P_1 중 $p_{01} = 1$

#2, (0,1)의 단절

C_2 = { 0 }

K_2 = { 1, 2, 3 }

$r_2 = r_0 * (1 - p_{01}) = 0.99$

P_2중 $p_{01} = 0$

문제 #1, #2의 그래프는 아래 그림과 같다.

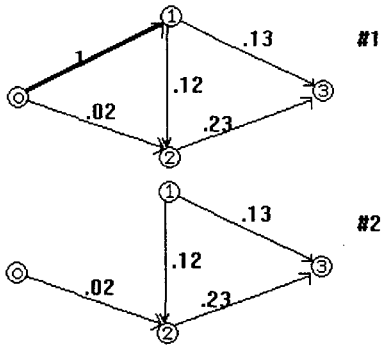

그림 3

이제 #1로부터 피봇요소 선택 기준에 의해 피봇요소로서 (0,2) 선택, 새로 #3, #4가 생성되어 문제집합은 { #2, #3, #4 }이다.

#3, (0,2)의 연결

C_3 = { 0, 1, 2 }

K_3 = { 3 }

$r_3 = r_1 * p_{02} = 0.01 * 0.02 = 0.0002$

P_3 중 $p_{02} = 1$

#4, (0,2)의 단절

C_4 = { 0 }

K_4 = { 2, 3 }

$r_4 = r_1 * (1 - p_{02}) = 0.01 * (1 - .02) = 0.0098$

P_4 중 $p_{02} = 0$

문제 #3, #4는 아래 그림과 같다.

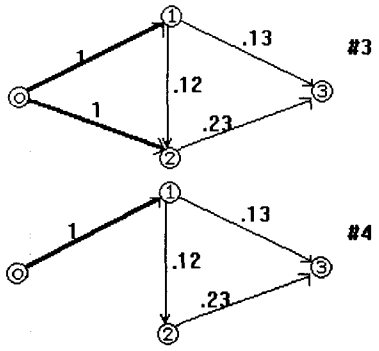

그림 4

이제 문제집합에서 #2를 꺼내어 다룬다. N={ #3, #4, #5, #6 } 이 될 것이다.

#5, (0,2)의 연결

C_5 = { 0, 2 }

K_5 = { 3 }

$r_5 = r_2 * p_{02} = 0.0198$

P_5 중 $p_{02} = 1$

#6, (0,2)의 단절

C_6 = { 0 }

K_6 = { 2, 3 }

$r_6 = r_2 * (1 - p_{02}) = 0.9702$

P_6 중 $p_{02} = 0$

※ 문제 #6은 다음번 재귀호출에서는 $C_6 \rightarrow C'_6$ 의 링크가 없으므로 종료하고(규칙 2) 0 을 되돌리게 된다.

문제 #5, #6의 그래프는 아래 그림과 같다.

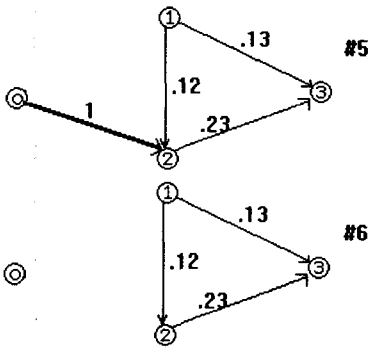

그림 5

이제 N={ #3, #4, #5, #6 }에서 #3를 꺼내어 다룬다. 종료규칙이 적용되지 않고 (1,3)링크가 선택되고 N={ #4, #5, #6, #7, #8} 이 된다.

#7, (1,3)의 연결

C_7 = { 0, 1, 2, 3 }

K_7 = { }

$r_7 = r_3 * p_{13} = 0.000026$

P_7 중 $p_{13} = 1$

※ 다음번 반복 때에는 이 문제는 종료규칙1에 따라 종료되고 r_7를 되돌리게 된다. (즉, K가 전부 연결된 것이다.)

#8, (1,3)의 단절

C_8 = { 0, 1, 2 }

K_8 = { 3 }

$r_8 = r_3 * (1 - p_{13}) = 0.000174$

P_8 중 $p_{13} = 0$

문제 #7, #8는 아래 그림과 같다.

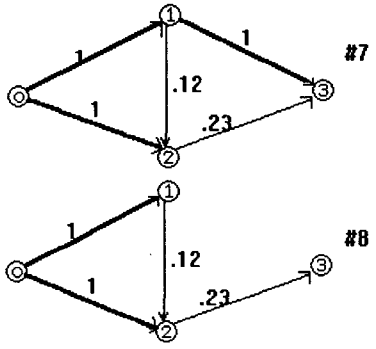

그림 6

이하는 생략한다. 이 과정을 표로 만들 수 있을 것이다.

3. 결론

이 해법은 이해하기가 쉽고, 실제 계산 프로그램을 간단하게 한다. 대부분의 프로그램 언어는 재귀 순환적인 함수 호출이 가능하므로 특히 간단하게 된다. 분해에 의해 생성되는 문제의 정보는 메모리에 입출력 시킬 수가 있고, 또는 하드디스크에 입출력 시킬 수도 있다. Park & Kim의 시점-종점 문제의 경우 효과적이었던 것처럼 K-단말 신뢰도에서 기존의 해법에 비해 효과적인 해법이라고 기대되며 실증을 하는 문제가 남았다.

여기서 설명한 해법은 이해하기 쉽게 설명한 것으로서 프로그램 상 효율성 등에 관해 몇 가지 언급할 것이 있다.

1) 문제식별번호는 컴퓨터를 사용할 때, 예제로 보인 것과 일치되지는 않는다.

2) 종료규칙은 N에서 문제를 꺼냈을 때 점검하는 것보다 앞서서 분해한 후 먼저 점검하는 것이 약간 효과적이다. 왜냐면 정보를 저장하는 절차를 줄이고 아울러 재귀함수를 1번 호출하는 것을 줄이기 때문이다.

3) 예제에서 보았듯이 링크가 연결된 경우 계산상 P를 고치지 않아도 된다. 왜냐면 C의 내부 링크는 문제되지 않으므로. 고쳐지는 정보는 C로 연결노드가 들어가고 r이 변경되면서 녹아든 셈이다. 계산량이 줄어드는 효과가 있다. 단절되는 경우는 물론 해당 링크확률을 0으로 바꿔줘야 한다. 요컨대 C→C′, C′→C′의 링크 확률만 유지하면 된다.

참고문헌

Ahn, H., Kim, K., and Ree, S., Path-cut pairwise approach to network reliability, 한국보전공학회지, 1.1, 1996, 9

Ball, M., IEEE Trs. Rel., 35, 1986, 230

Ball, M., and Cameron, E., Experiments with network reliability analysis algorithms, Modelling and Simulations: Proc. 17th Annual Pittsburgh Conf., (ed., W. Vogt and M. Mickle), 17, 1986, 1799

Chen, H., Shi, D., and Xu, W., Microelectron Reliab., 25, 1985, 35

Hansler, E., IEEE Trs. Commun., 20, 1972, 637

Kim, Y., Case, K., and Ghare, P., IEEE Trs. Rel., 21, 1972, 215

Misra, K., IEEE Trs. Rel., 19, 1970, 50

Park, K.S., and Kim, K., Pull and prune technique for complex structural reliability analysis, Int. J. Systems Sci., 21.11, 1990, 2081

Resende, M., IEEE Trs. Rel., 35, 1986, 24

Satyanarayana, A., and Chang, M.K., Network Reliability and the factoring theorem, Networks, 13, 1983, 107

Wood, R.K., Factoring algorithms for computing K-terminal network reliability, IEEE Trs. Rel., 35.3, 1988, 269

SESSION B

Session B3 : 신뢰성분석/Fault Tolerant System

좌장 : 박기진(아주대학교)

- 일반화 선형 모형을 이용한 냉음극 형광램프의 휘도 측정 시 온도 및 습도의 영향에 대한 연구
  윤양기 (한국전기전자시험연구원), 길영수 (한국외국어대학교)

- BGA 패키지에 사용된 유/무연 솔더의 신뢰성 평가
  이억섭, 허만재, 명노훈, 김동혁 (인하대학교)

- 고신뢰성 차세대 임베디드 컴퓨팅 시스템의 백업 최소화 방안
  박기진, 김광섭, 최석호 (아주대학교 산업정보시스템공학부)

- ESPI를 이용한 엔지니어링 플라스틱 열 변형 분석
  함선일, 최동준, 박상득 (삼성전자)