신뢰성응용연구 신뢰성기술
제14권, 제2호. pp.103-107, 2014

모호수 연산을 적용한 네트워크 신뢰도*

김 국†

서경대학교 산업공학과

Reliability Approach to Network Reliability Using Arithmetic of Fuzzy Numbers

Kuk Kim

Department of Industrial Engineering, Seokyeong University

An algorithm to get network reliability, where each link has probability of fuzzy number, is proposed. Decomposition method and fuzzy numbers arithmetic are applied to the algorithm. Pivot link is chosen one by one from start node recursively at time of decomposition, and arithmetic of fuzzy complementary numbers is included at the same time. No criteria of pivot link selection and the recursive calculation make the algorithm simple.

Keywords: Network Reliability, Decomposition Method, Fuzzy Numbers Arithmetic, Recursive Algorithm, Fuzzy Complementary Number

1. 서 론

모호집합이론(Fuzzy Set Theory)은 집합의 원소에 내재하는 모호성을 정형화하는 이론이다. 기본적으로 원소가 집합의 정의에 속하는 정도를 소속함수(membership function)로 표현한다(Zadeh, 1965). 모호수(fuzzy number)는 숫자에 대해 모호성이 반영된 것으로 한 예로서 삼각 모호수와 같은 것이 있다(Dubois and Prade, 1980). 삼각 모호수는 (0.1, 0.2, 0.3)과 같이 세 개의 수로 표시된다.

한편 네트워크 구조의 신뢰도는 시점으로부터 단말까지 경로가 작동할 확률을 구하는 것이다. 일반적으로 네트웍 신뢰도문제는 NP-Hard로 알려져 있다(Wood, 1988). 신뢰도 구조의 분석은 직렬・병렬화 방법, 발생사건 나열법, 최소절단집합・연결집합 결합방법, 분해법과 같은 기법이 있는데, 네트워크 신뢰도 문제에 이를 적용할 수 있다. 그런데 Park and Kim(1990)]은 네트워크 구조의 시점-종점 신뢰도를 재귀적으로 역진으로 분해법을 진행하는 당기고 가지치기(pull-prune) 해법과 이의 컴퓨터 프로그램을 설명하였다. 안해일 외(1996)은 경로집합, 절단집합을 짝으로 고려하여 확률의 덧셈법칙을 사용하는 해법을 제시하였는데 상대적으로 효과적이었다.

Satyanarayana and Chang(1983)은 직병렬 단축과 함께 분해법을 사용하는 것이 복잡한 K-단말 신뢰도 계산에서 우월하다고 하였다. 김국・송기원(2005)은 K-단말 신뢰도를 구하는 해법으로 분해법을 재귀적으로 진행하였다. Ball and Cameron (1986)은 여러 네트워크 신뢰도의 알고리듬, 모델링, 시뮬레이션을 실험하였다.

김국(2011)은 네트워크의 각 링크의 신뢰도가 모호수의 확률일 경우, 네트워크 신뢰도를 구하기 위해 재귀적 해법의 기초를 보여주었지만 불완전하였고, 예제에서 링크 확률이 적은 값을 사용하였기 때문에 문제점이 드러나지 않았다.

이 논문은 김국(2011)의 논문을 개선하여, 모호수의 네트워크 신뢰도를 구하는 데 모호수 연산과 모호 보수 개념을 사용하여 재귀적 전진 방향의 분해법을 적용하는 해법을 제시한다.

2. 모호수를 사용한 네트워크 신뢰도 문제 및 분해법

2.1 문제와 용어

문제의 네트워크를 그래프 G = {V, E}라고 하면 V = {0, 1,

* 본 연구는 2012학년도 서경대학교 교내연구비 지원에 의하여 이루어졌음.
† 교신저자 kimkuk99@daum.net
2014년 2월 11일 접수; 2014년 4월 15일 수정본 접수; 2014년 5월 2일 게재 확정.

2, ⋯, t}의 노드와 E = {(0, 1), ⋯}의 링크를 가진다. 시점노드의 번호는 0으로 하고, 종점을 t라고 한다. 링크의 신뢰도는 확률적으로 독립이고, 각 링크의 확률을 삼각 모호수(fuzzy number) $\tilde{p}_{ij}=\left(p_{ij}^{(1)},\ p_{ij}^{(2)},\ p_{ij}^{(3)}\right)$라고 하면 네트워크 신뢰도 행렬은 각 원소마다 3개의 값을 가지는 식 (1)과 같은 모호 행렬 $\tilde{P}$로 표현된다. 구하고자 하는 네트워크 신뢰도를 $R(\tilde{P};\ G(V,\ E))$라고 하면, 이 값은 모호수가 될 것이다.

$$\begin{aligned}\tilde{P}&=\left[\tilde{p}_{ij}\right]\\&=\left\{\left(p_{ij}^{(1)},\ p_{ij}^{(2)},\ p_{ij}^{(3)}\right)\right\},\ 1\leqq i\leqq t,\ 1\leqq j\leqq t.\end{aligned}\tag{1}$$

2.2 분해법의 적용

김국(2011)의 분해법과 모호집합을 이용하면, (i, j) 링크를 피봇링크라 할 때, 식 (2)로 표현된다.

$$R(\tilde{P})=\tilde{p}_{ij}R(\tilde{P}\mid p_{ij}=1)+(1-\tilde{p}_{ij})R(\tilde{P}\mid p_{ij}=0).\tag{2}$$

여기서 식 (2) 우변의 첫 번째 구조에서 피봇링크는 명목링크가 되고, 두 번째 구조에서는 단절된 모습이 된다. 모호수 (1, 1, 1)는 1로, (0, 0, 0)은 0과 같다.

Dubois and Prade(1980)의 모호수 연산을 적용하면,

$$1-\tilde{p}_{ij}=\left(1-p_{ij}^{(3)},\ 1-p_{ij}^{(2)},\ 1-p_{ij}^{(1)}\right).\tag{3}$$

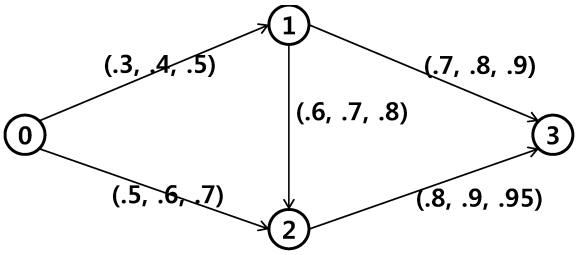
그리하여 식 (2)의 첫 번째 구조는 $\tilde{p}_{ij}=\left(p_{ij}^{(1)},\ p_{ij}^{(2)},\ p_{ij}^{(3)}\right)$를 곱하고, 두 번째 구조는 식 (3)을 곱하는 것이 김국(2011)의 해법이었다.

식 (2)의 분해구조는 프로그래밍 해법에 있어서 각 하위구조에 대한 정보를 입력으로 하여 신뢰도 R을 호출하는 재귀순환을 적용할 수 있다. 중요한 성질은, 명목링크(확률 1인 링크)로 연결된 노드들에 있어서, 이들 간의 링크의 작동 여부는 문제되지 않는다는 점이다.

2.3 과거 연구의 오류

김국(2011)의 해법은 실제 적용하면 불완전하다. 해당 논문은 각 링크 확률을 작게 사용하여서 문제가 나타나지 않았다. <그림 1>은 이 문제점을 보여 주기 위해 확률을 크게 한 것이다. 여기서 초기치는 $C=\{0\}, C'=\{1,\ 2,\ 3\},\ \tilde{r}=(1,\ 1,\ 1)$와 같다. 이들 기호의 정의는 제 2.4절에 보일 것이며, $\tilde{r}$은 신뢰도가 모호수로 표시된 것이다.

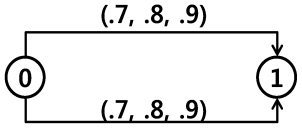
이 그림의 김국(2011)의 해법 결과는 (.37172, .70736, 1.20725)와 같다. 그러면 1.20725와 같은 값이 나와 확률이 1 이하라는 원칙에 맞지 않다. 곱하는 과정에 모호수의 폭이 넓어지고, 최종 합산 시 넓어진 폭의 누적 효과 때문에 폭이 더욱 커지는 것을 용이하게 볼 수 있다.



<그림 1> An Example of Fuzzy Network Reliability

기존의 모호이론과 신뢰성 이론에 따르면 네트워크 신뢰도가 1을 초과하는 것은 이론적으로 발생이 불가능하다는 논의가 있었다. 그런데 <그림 2>와 같은 간단한 네트워크 구조의 예를 보자. 이는 곧 병렬구조와 같고 $\tilde{p}_1+\tilde{p}_2-\tilde{p}_1\times\tilde{p}_2$이다. Dubois의 모호수 연산이론에 의하면, $-\tilde{p}_1\times\tilde{p}_2=(-.81,\ -.64,\ -.49)$로 정의되고, 따라서 체계신뢰도는 식 (4)와 같다. 이것은 신뢰도가 1을 넘는 오류가 나온다.

$$\begin{aligned}\tilde{p}_1+\tilde{p}_2-\tilde{p}_1\times\tilde{p}_2&=(.7,\ .8,\ .9)\\&\quad+(.7,\ .8,\ .9)+(-.81,\ -.64,\ -.49)\\&=(.59,\ .96,\ 1.31).\end{aligned}\tag{4}$$



<그림 2> A Simple Case

2.4 새로운 해법

신뢰도와 불신뢰도는 합해서 1이어야 한다. $\tilde{p}=\left(p^{(1)},\ p^{(2)},\ p^{(3)}\right)$가 주어질 때 과거처럼 $1-\tilde{p}=\left(1-p^{(3)},\ 1-p^{(2)},\ 1-p^{(1)}\right)$로 정의하면 $\tilde{p}+(1-\tilde{p})$은 일반으로 1이 되지 않는다. 그러므로 여기서 모호 보수(fuzzy complimentary number)를 식 (5)와 같이 정의한다. 이 이론의 근거는 새로운 것으로서, 좌표의 방향을 반대로 함으로서, 두 수가 자유롭지 않고 보수의 관계성이 있음을 나타내는 것이다. 따라서 이 경우 신뢰도와 불신뢰도를 합해서 1로 나온다.

$$1-\tilde{p}=\left(1-p^{(1)},\ 1-p^{(2)},\ 1-p^{(3)}\right),\tag{5}$$

그리하여 새로운 해법에서는 식 (2)의 첫 번째 구조는 피봇링크 확률 $\tilde{p}=\left(p^{(1)},\ p^{(2)},\ p^{(3)}\right)$을 곱하고, 두 번째 구조는 피봇링크 확률의 모호보수 식 (5)를 곱한다.

그러면 분해법에 의해 식 (2)에서 분할된 하위 구조는 2개가 생성되어 각각 $\tilde{P}$가 재조정되어, $\tilde{P}\mid p_{ij}=1$과 $\tilde{P}\mid p_{ij}=0$이 된다. 각 $\tilde{P}$는 '연결된 노드들'로부터 '연결 안된 노드들'로의 링크 확률만 유지하면 되므로, 실제 사용되는 행렬의 크기는 줄어드는 효과가 있다.

하위문제는 다음과 같은 정보집합을 갖게 된다.

$\tilde{P}$: 모호 확률 행렬

C : 연결집합, 즉 시점으로부터 명목링크들로 연결된 노드 집합

C' : C의 여집합($= V - C$), 즉 시점으로부터 연결되지 않은 노드 집합

$\tilde{r} = (r^{(1)},\ r^{(2)},\ r^{(3)})$: 현재의 구조를 갖게 되는 모호 확률

단계 0 : 초기값, 최초의 $\tilde{P}$, $C = \{0\}$, $C' = \{1,\ 2,\ 3, \cdots,\ t\}$, $\tilde{r} = (1,\ 1,\ 1)$.

단계 1 : 종료규칙 1. $t \in C$이면 종료하고, $\tilde{r}$을 되돌려 준다. 아니면 단계 2로 간다.

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

종료규칙 2. $(z,\ y)$가 없으면 종료하고, 0을 되돌려 준다. 아니면 단계 3으로 간다.

단계 3 : $(z,\ y)$를 피봇링크로 삼아 분해법을 적용하여 새로운 하위 문제 2개를 만든다. 하위문제 #1과 #2의 구조 정보는 다음과 같다.

#1 : [$p_{zy} = 1$의 구조]. $(z,\ y)$가 명목링크가 되고 노드 y는 연결집합 C로 옮겨진다.

$$C_{new} = C + y,\ \ C'_{new} = C' - y,$$
$$\tilde{r}_{new} = \tilde{r} \cdot \tilde{p}_{zy}$$
$$= \left(r^{(1)},\ r^{(2)},\ r^{(3)}\right) \cdot \left(p_{zy}^{(1)},\ p_{zy}^{(2)},\ p_{zy}^{(3)}\right)$$
$$= \left(r^{(1)} \cdot p_{zy}^{(1)},\ r^{(2)} \cdot p_{zy}^{(2)},\ r^{(3)} \cdot p_{zy}^{(3)}\right),$$
$$\tilde{P}_{new} = \left\{\tilde{P} | \tilde{p}_{zy} = (1,\ 1,\ 1)\right\}.$$

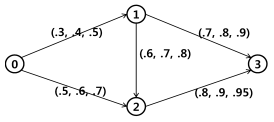
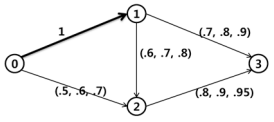
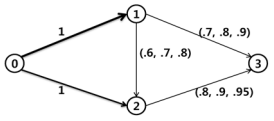
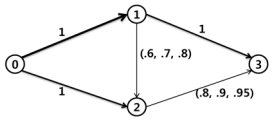
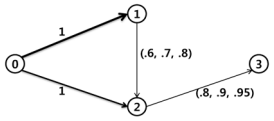
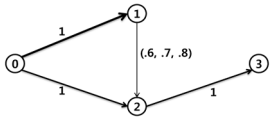
#2 : [$p_{zy} = 0$의 구조]. $(z,\ y)$는 단절된다.

$$C, C' \text{ (분해전과 같음)},$$
$$\tilde{r}_{new} = \tilde{r} \cdot (1 - \tilde{p}_{zy})$$
$$= \left(r^{(1)},\ r^{(2)},\ r^{(3)}\right)\left(1 - p_{zy}^{(1)},\ 1 - p_{zy}^{(2)},\ 1 - p_{zy}^{(3)}\right)$$
$$= \left(r^{(1)} \cdot (1 - p_{zy}^{(1)}),\ r^{(2)} \cdot (1 - p_{zy}^{(2)}),\ r^{(3)} \cdot (1 - p_{zy}^{(3)})\right),$$
$$\tilde{P}_{new} = \left\{\tilde{P} | \tilde{p}_{zy} = (0,\ 0,\ 0)\right\}.$$

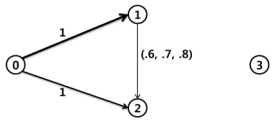
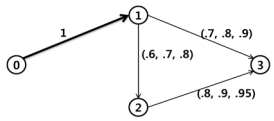
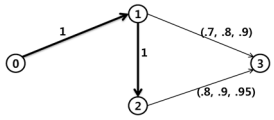
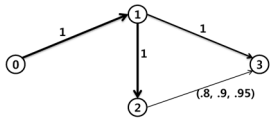
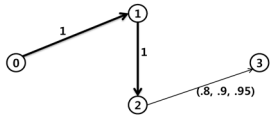
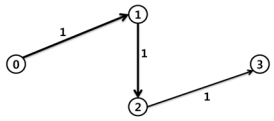
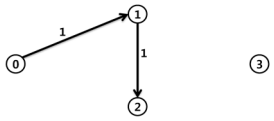
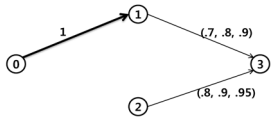
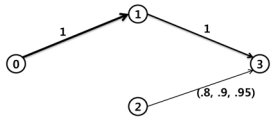
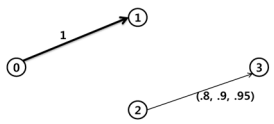
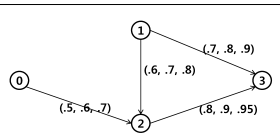
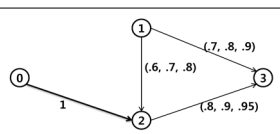
이 과정을 재귀적으로 반복한다. 이 해법은 분해된 두 개의 하위구조를 불러오는 재귀 호출의 개념이므로, 컴퓨터 프로그램에서 R을 재귀함수로서 호출하면 간단하게 구현할 수 있다.

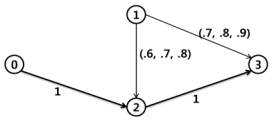
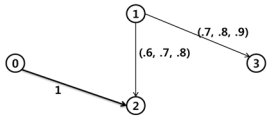
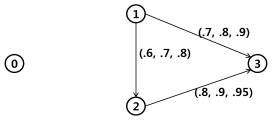
$\tilde{P}$를 고치는 것은 과정 중 $C \rightarrow C'$의 링크를 찾는 것과 관련된다. 그런데 피봇링크가 연결되는 경우는 C 내의 링크는 $C \rightarrow C'$와 무관하기 때문에 실제 프로그래밍에서는 $p_{yz} = (1,\ 1,\ 1)$로 고치지 않아도 된다(프로그램이 간단해지고 계산량도 줄어든다). 단절되는 경우는 반드시 $p_{yz} = (0,\ 0,\ 0)$으로 바꿔줘야 한다.

〈표 1〉 Algorithm Illustration of the Example

Problem ID Number	Pivot Link	Current $\tilde{r}$	C, C', and $\tilde{P}$	Stopping Rule
#0		(1, 1, 1)		
#1	(0, 1) success	(.3, .4, .5)		
#1-1	(0, 2) success	(.3, .4, .5)×(.5, .6, .7) = (.15, .24, .35)		
#1-1-1	(1, 3) success	(.15, .24, .35)×(.7, .8, .9) = (.105, .192, .315)		by rule 1, return $\tilde{r}$
#1-1-2	(1, 3) fail	(.15, .24, .35)×(.3, .2, .1) = (.045, .048, .035)		
#1-1-2-1	(2, 3) success	(.045, .048, .035)×(.8, .9, .95) = (.036, .0432, .03325)		by rule 1, return $\tilde{r}$

〈표 1〉 (계속)

Problem ID Number	Pivot Link	Current $\tilde{r}$	C, C', and $\tilde{P}$	Stopping Rule
#1-1-2-2	(2, 3) fail	0		by rule 2, return 0
#1-2	(0, 2) fail	(.3, .4, .5)×(.5, .4, .3) = (.15, .16, .15)		
#1-2-1	(1, 2) success	(.15, .16, .15)×(.6, .7, .8) = (.09, .112, .12)		
#1-2-1-1	(1, 3) success	(.09, .112, .12)×(.7, .8, .9) = (.063, .0896, .108)		by rule 1, return $\tilde{r}$
#1-2-1-2	(1, 3) fail	(.09, .112, .12)×(.3, .2, .1) = (.027, .0224, .012)		
#1-2-1-2-1	(2, 3) success	(.027, .0224, .012)×(.8, .9, .95) = (.0216, .02016, .0114)		by rule 1, return $\tilde{r}$
#1-2-1-2-2	(2, 3) fail	0		by rule 2, return 0
#1-2-2	(1, 2) fail	(.09, .16, .25)×(.4, .3, .2) = (.036, .048, .05)		
#1-2-2-1	(1, 3) success	(.036, .048, .05)×(.7, .8, .9) = (.0252, .0384, .045)		by rule 1, return $\tilde{r}$
#1-2-2-2	(1, 3) fail	0		by rule 2, return 0
#2	(0, 1) fail	(.7, .6, .5)		
#2-1	(0, 2) success	(.7, .6, .5)×(.5, .6, .7) =(.35, .36, .35)		

Problem ID Number	Pivot Link	Current $\tilde{r}$	C, C', and $\tilde{P}$	Stopping Rule
#2-1-1	(2, 3) success	(.35, .36, .35)×(.8, .9, .95) =(.28, .324, .3325)		by rule 1, return $\tilde{r}$
#2-1-2	(2, 3) fail	0		by rule 2, return 0
#2-2	(0, 2) fail	0		by rule 2, return 0
Solution		(.5308, .70736, .84515) = recursive sum of $\tilde{r}$'s of stopping rule 1		

이 해법을 <그림 1>의 예제에 적용하고 이를 <표 1>로 보인다. C, C', $\tilde{P}$는 그림으로 보인다. 결과는 (.5308, .70736, .84515)로서 훨씬 타당한 값을 보여 준다.

Park and Kim(1990)은 재귀적 해법이 프로그래밍의 편리함을 제공하고, 또 시점-종점의 순차적 적용은 이미 연결되었거나, 단절된 링크에 대해 계산이 불필요하므로 효율적이라고 하였고 PC의 계산 시간으로 이를 보여 주었다.

3. 결 론

이 논문은 각 링크의 확률이 모호수로 주어지는 네트워크 신뢰도를 계산하는 해법을 과거 연구로부터 개선하여 제시한 것이다. 단순한 모호수 연산은 불완전하고 모호 보수를 정의하여 사용하는 것이 타당하다. 시점-종점 문제에 분해법을 재귀적으로 진행하는 Pull-prune 기법과, 모호 보수 연산을 적용하였다. 이 논문은 실제로 컴퓨터 계산을 위한 해법을 보여준다. 재귀함수 호출의 알고리즘은 이해하기가 쉽고, 실제 프로그램을 간단하게 한다. 네트워크 신뢰도 문제는 NP-hard로서 문제의 크기가 커지면 계산량이 급증하는 데, 피봇링크를 선택하는 규칙이 간단하고, 중간에 불필요한 하부 문제의 계산을 생략하므로 나열법이나 최소절단집합·연결집합 방법에 비해 더 효과적이다.

종료규칙에 있어서, 하위 문제를 꺼냈을 때 점검하는 것보다, 분해한 후 먼저 점검하는 것이 효과적이다. 왜냐면 정보를 저장하는 절차를 줄이고 재귀함수 호출을 1회 줄이기 때문이다. n개의 노드와 m개의 링크로 구성된 일반적인 네트워크에 대해, 제안한 알고리듬의 개선 효율성은 향후 연구를 진행할 계획이다.

참고문헌

[1] 김 국 (2011), 모호집합을 적용한 네트워크 신뢰도, 2011년 한국신뢰성학회 춘계학술대회, 한국신뢰성학회, 대구대학교, pp. 311-318.

[2] 김 국・송기원(2005), 네트웍 구조의 다중 단말 신뢰도, 2005년 한국신뢰성학회학술발표대회, 한국신뢰성학회, 전자부품연구원, pp. 271-278.

[3] 안해일・김 국・이상복 (1996), Path-cut pairwise approach to network reliability, 한국보전공학회지, 제1권, 제1호, pp. 9-23.

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

[5] Dubois, D. and Prade, H. (1980), Fuzzy Sets and Systems, Academy Press.

[6] Mahapatra, G. S. and Roy, T. K. (2012), Reliability evaluation of complex system with fuzzy reliability of components using interval nonlinear programming, *Electronic Journal of Applied Statistical Analysis*, Vol. 5, No. 2, pp. 151-163.

[7] Park, K-S. and Kim, K. (1990), Pull and prune technique for complex structural reliability analysis, *International Journal of Systems Science*, Vol. 21, No. 11, pp. 2081-2089.

[8] Satyanarayana, A. and Chang, M. K. (1983), Network Reliability and the factoring theorem, *Networks*, Vol. 13, p. 107.

[9] Wood, R. K. (1988), Factoring algorithms for computing K-terminal network reliability, *IEEE Transactions on Reliability*, Vol. 35, No. 3, p. 269.

[10] Zadeh, L. (1965), Fuzzy sets, *Information and Control*, Vol. 8, pp. 338-353.