상태 발생확률 순서에 의한 네트워크 성능 분석 방법 연구

Hee Kyoung Lee*, Dong Ho Park* and Seung Min Lee*

Abstract

The traditional methods of evaluating the performance of a network by enumerating all possible states may quickly become computationally prohibitive, since the number of states grows exponentially as the number of components increases. In such cases, enumerating only the most probable states would provide a good approximation.

In this paper, we propose a method which efficiently generates upper and lower bounds for coherent performance measures utilizing the most probable states. Compared with Yang and Kubat's method, our procedure significantly reduces the complexity and memory requirement per iteration for computing the bounds and thereby, achieves the given degree of accuracy or the coverage within a shorter time.

1. 서론

현대생활에 나타나는 대부분의 복잡한 시스템들은 실제적으로 네트워크의 구조를 갖고 있다. 예를 들어, 수송망, 통신망, 전력 송신망 등은 네트워크의 형태로 구성되어 있다. 이러한 네트워크는 시작하거나 끝나는 지점 또는 중간에 머무는 지점을 표시하게 되는 노드(node)들과 연결 통로를 표시하는 링크(link)들로 구성되어 있다. V는 노드들의 집합, E는 링크들의 집합 이라 할때 네트워크는 G(V,E)의 그래프로 표현될 수 있다. 본 논문에서는 네트워크에서의 각 링크들이 서로 다른 용량을 갖고 있으며, flow network의 시작 노드와 마지막 노드가 정해져 있다고 가정한다.

* Department of Statistics, Hallym University

또한 네트워크의 성능을 측정하기 위하여 사용되는 척도는 coherence property를 만족하는 경우를 고려한다. 상태벡터는 고장난 링크들의 집합으로 표현한다. 네트워크의 링크 수가 커지면 상태공간의 크기가 지수적으로 증가하여 정확한 값을 계산하는 것은 불가능하므로 적절한 근사값을 구하여 사용하는 것이 보다 현실적인 경우가 대부분이다. Li & Silvester [1]는 발생확률이 높은 상태들(most probable states, MPS)을 선택하여 그 값을 구하는 방법을 제시하였고, Lam & Li [2], Shier [3], Gomes [5]등에 의해 개선되었다. 본 논문에서는 MPS를 이용하여 네트워크 성능척도의 상·하한값을 구하는 새로운 방법을 제시하고 Yang & Kubat [6]의 방법과 비교하여 효율성 면에서 보다 개선된 방법임을 보인다.

Assumptions

1. 각각의 링크들은 가동과 고장 두개의 상태를 갖는다.
2. 각각의 링크들의 고장은 서로 독립이다.

Notation

$p_j(\frac{1}{2} \le p_j \le 1)$	j번째 링크가 가동할 확률($j = 1, 2, \cdots, n$)
$q_j = 1 - p_j$	j번째 링크가 고장날 확률($j = 1, 2, \cdots, n$)
S_i	i번째 MPS에서 고장난 링크들의 집합($i = 1, \cdots, 2^n$)
S_i^P	S_i의 부모 MPS에서 고장난 링크들의 집합
$P(S_i)$	i번째 MPS의 확률
$P_L(S_i)$	S_i을 포함하는 상태들의 총 확률
$R(S_i)(R(S_i^P))$	S_i에서 흐르는 유량 (S_i^P에서 흐르는 유량)
$U(i)(L(i))$	i번째 MPS가 선택 되었을 경우 계산된 상한값(하한값)
$\bar{R}$	네트워크의 성능측도(에:평균최대유량)
$\alpha = R(S_1)$	최대값
$\beta = R(S_{2^n})$	최소값

Coherence property를 만족한다는 것은 가능한 상태 S_i와 S_j가 있을 때 $S_i \subset S_j$이면 유량에 대하여는 $R(S_i) \leq R(S_j)$이 성립됨을 의미한다.

2. Yang & Kubat의 방법

이진트리에서 노드 u의 가지에 노드 v'와 v''가 있다면, 노드 v'와 v''는 u의 자식노드라 하고 노드 u를 노드 v'와 v''의 부모노드라고 하며 노드 v'와 v''는 형제노드라고 한다. 자식노드를 갖지 않는 노드는 leaf라고 한다. 뿌리로부터 어느 주어진 leaf에 이르는 경로(path)는 오직 하나 밖에 존재하지 않으며, 가지는 링크의 상태에 따라 왼쪽은 가동 할때, 오른쪽은 고장 났을 때의 상태를 나타낸다.

네트워크의 표현 가능한 상태는 2^n개이고, 각 상태들의 확률은 $P(S) = \prod_{i=1}^{n} p_i^{x_i} \cdot q_i^{1-x_i}$이다. 예를 들어 5번째의 MPS로 링크1과 3이 고장인 상태가 선택된다면 $S_5 = \{1,3\}$이고, $S_5^P = \{1\}$이며, $P(S_5) = q_1 p_2 q_3 \prod_{i=4}^{n} p_i$이고, $P_L(S_5) = q_1 p_2 q_3$이다. $\bar{\mathbf{R}} = \sum_{\mathbf{i=1}}^{\mathbf{2^n}} \mathbf{R(S_i)} \cdot \mathbf{P(S_i)}$이며, n이 커지면 값을 계산하기 어려우므로 Li & Silvester [1]가 제안한 MPS를 이용해서 상·하한값을 구해 그 값을 추정하게 된다. 즉, 상한값은 $\mathbf{U(m)} = \sum_{\mathbf{i=1}}^{\mathbf{m}} \mathbf{R(S_i)} \cdot \mathbf{P(S_i)} + [\mathbf{1} - \sum_{\mathbf{i=1}}^{\mathbf{m}} \mathbf{P(S_i)}] \times \alpha$이고, 하한값은 $\mathbf{L(m)} = \sum_{\mathbf{i=1}}^{\mathbf{m}} \mathbf{R(S_i)} \cdot \mathbf{P(S_i)} + [\mathbf{1} - \sum_{\mathbf{i=1}}^{\mathbf{m}} \mathbf{P(S_i)}] \times \beta$이다. 여기서, MPS의 갯수를 정하는 기준으로는 상·하한간의 상대 오차의 크기($\Delta_{\bar{R}}$)인 $\Delta_{\bar{\mathbf{R}}} \simeq \frac{\mathbf{U-L}}{\mathbf{L}} \leq \varepsilon$을 사용한다.

알고리즘을 보면 MPS와 함께 상한값도 같이 계산하였는데, MPS는 링크의 상태에 따라 왼쪽가지는 가동, 오른쪽은 고장 났을 때를 나타내는데 각각은 weight를 가지며 l번째 있는 링크 u에서의 weight는 $\mathbf{w(u)} = \prod_{\mathbf{i=1}}^{\mathbf{l}} \mathbf{p_i^{x_i}} \cdot \mathbf{q_i^{1-x_i}}$으로 표현된다. 이 weight에서 다음에 나올 링크들이 모두 가동될때가 가장 큰 weight값이 되는데, 그 값은 $w(u) \cdot \prod_{i=l+1}^{n} p_i$이 된다. 각 링크가 나타날 때 마다 가장 큰 weight값을 비교하게 되는데, 왼쪽 링크의 가장 큰 weight는 $\mathbf{W_1(u')} = \mathbf{w(u')} \cdot \mathbf{p_{l+1}} \cdot \prod_{\mathbf{i=l+2}}^{\mathbf{n}} \mathbf{p_i}$이고, 오른쪽 링크는 $\mathbf{W_0(u')} = \mathbf{w(u')} \cdot \mathbf{q_{l+1}} \cdot \prod_{\mathbf{i=l+2}}^{\mathbf{n}} \mathbf{p_i}$이다. 따라서 $\mathbf{max}[\mathbf{W_1(u')}, \mathbf{W_0(u')}]$를 구해서 다음 링크를 선택하게 되며, leaf까지가 계산이 되면 하나의 상태인 MPS가 선택된다. 상한값은 왼쪽은 $\mathbf{U_1(u')} = \mathbf{w(u')} \cdot \mathbf{p_{l+1}} \cdot \alpha$, 오른쪽은

$\mathbf{U_0(u')} = \mathbf{w(u')} \cdot \mathbf{q_{l+1}} \cdot \alpha$으로 각 링크에서 weight와 함께 계산된다. MPS가 찾아지면 그 때의 최대유량을 구하고, 이것에 의해 update를 하게 되는데 MPS로 선택되어진 가장 큰 weight는 0으로 하고 $\mathbf{max[W_1(u'), W_0(u')]}$에 의해 update를 한다. 상한값은 α값이 아닌 구해진 유량으로 계산하면서 update를 하게 된다. Update가 다 되면 root에서 상한값을 계산하게 되는데, 상한값은 $\mathbf{U = U_1(root) + U_0(root)}$이고, 하한값은 $\mathbf{L(m) = \sum_{i=1}^{m} R(S_i)w(S_i) + [1 - \sum_{i=1}^{m} w(S_i)] \cdot \beta}$이다.

3. 알고리즘

알고리즘 구성상 유량을 구하는 알고리즘은 따로 서술하지 않고 함수에 의해 구해진다고 가정하고, MPS를 찾는 방법은 Yang & Kubat의 방법을 사용한다.

Algorithm

Step1 입력 : ε, p_j, $j = 1, 2, \cdots, n$

Step2 초기치 : $U(0) = \alpha$, $L(0) = 0$, $i = 0$, $\Delta_{\bar{R}} = 1$, root 생성

Step3 반복 :

1. 초기치 : $P_L(S_i) = w = 1$
2. $i = i + 1$
3. Yang & Kubat의 방법에 의해서 i번째 MPS인 S_i 찾기
4. 찾아진 MPS(S_i)의 유량 $R(S_i)$와 $P_L(S_i)$를 계산
5. Yang & Kubat의 방법에 의해 update하면서 부모유량 $R(S_i^P)$ 계산
   - 5-1. Update 과정에서 노드에 저장 되는 유량이 없으면 찾은 유량을 넣는다.
   - 5-2. 저장된 유량이 있으면 그 유량을 부모 유량으로 하고 더 이상 부모 유량 찾기 멈춘다.

6. $R(S_i) \neq R(S_i^P)$ 이면

$$U(i) = U(i-1) + P_L(S_i) \times (R(S_i) - R(S_i^P))$$

7. Lower bound 계산

$$L(i) = L(i-1) + w \times R(S_i)$$

8. 상대오차의 값 계산

$$\Delta_{\bar{R}} = \frac{U(i)-L(i)}{L(i)}$$

9. $\Delta_{\bar{R}} \geq \varepsilon$까지 [Step3]을 반복한다.

Lemma 1 선택되어진 상태에서의 상한값을 계산할 때 선택된 상태의 부모상태의 유량을 이용한다.

$$U(i) = U(i-1) + P_L(S_i) \times (R(S_i) - R(S_i^P))$$

<**증명**> 노드 u의 자식 노드가 m개 있으며, 각각 $u_1, u_2, \cdots, u_m$으로 표시하자. 노드 u에서의 총 확률은 $P_L(u)$이고 상태의 확률은 $P(u)$이며, 유량은 $R(u)$이다. 자식 노드의 확률은 각각 $P_L(u_1), \cdots, P_L(u_m)$이고 유량은 $R(u_1), \cdots, R(u_m)$이다. 여기서 $R(u) \geq R(u_1), \cdots, R(u) \geq R(u_m)$이며, $P_L(u) = P_L(u_1) + \cdots + P_L(u_m) + P(u)$이다. $i-1$번째까지의 상한값이 $U(i-1)$일때 i번째에서 u_1이 선택 되었다면 u는 먼저 선택 되어졌으므로 $U(i-1)$안에 있다. 따라서 RU가 노드 u가 선택된 경우를 뺀 나머지 값이라면,

$$\begin{aligned}
U(i-1) &= RU + P_L(u) \cdot R(u) \\
&= RU + (P_L(u_1) + \cdots + P_L(u_m) + P(u))R(u) \\
&= RU + P_L(u_1)R(u) + (P_L(u_2) + \cdots + P_L(u_m) + P(u))R(u) \text{: } R(u) \geq R(u_1) \\
&\geq RU + P_L(u_1)R(u_1) + (P_L(u_2) + \cdots + P_L(u_m) + P(u))R(u) = \mathbf{U(i)} \\
U(i) &= RU + P_L(u_1)R(u_1) + (P_L(u_2) + \cdots + P_L(u_m) + P(u))R(u)
\end{aligned}$$

$$
\begin{aligned}
&= RU + P_L(u_1)R(u_1) + (P_L(u) - P_L(u_1))R(u) \\
&= RU + P_L(u_1)R(u_1) + P_L(u)R(u) - P_L(u_1)R(u) \\
&= U(i-1) + P_L(u_1)(R(u_1) - R(u))
\end{aligned}
$$

이다.

Example 1. 5개의 링크로 구성된 그림1의 bridge network에서 각각의 링크가 가동할 확률은 $p_1 = 0.5$, $p_2 = 0.6$, $p_3 = 0.7$, $p_4 = 0.8$, $p_5 = 0.9$이고, 유량은 $c_1 = 2$, $c_2 = 6$, $c_3 = 4$, $c_4 = 5$, $c_5 = 3$이다. 전체 상태 공간은 $2^5 = 32$이고, $\bar{R} = 3.4952$, $\varepsilon \geq 0.05$이다.

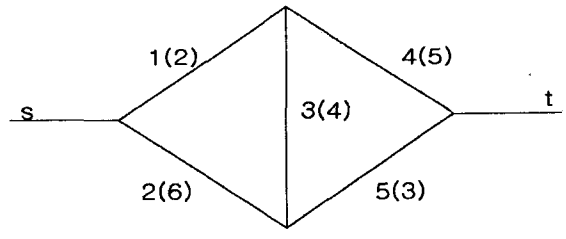

그림 1: bridge network

그림2는 Yang & Kubat의 방법을 이용한 것인데 MPS가 8일때까지 계산을 했을때 실선인 가지를 모두 계산한 것이고 표1은 제안된 방법에 의하여 같은 8개를 계산하는 과정을 설명하고 있는데, 계산과정을 보면 Yang & Kubat의 방법보다 제안된 방법이 매우 간단함을 알 수 있다. 그리고 Yang & Kubat의 MPS 구하는 방법을 같이 사용해서 상·하한값을 계산 하였을 때, 찾아지는 MPS의 갯수는 16개이고 coverage는 91.2%이며 그 때의 상한값은 3.604로 Yang & Kubat의 방법과 같은 결과를 얻으나 걸리는 시간은 Yang & Kubat의 방법은 0.000105이고 제안된 방법은 0.000093으로 많은 시간이 단축됨을 볼 수 있다. (사용된 컴퓨터:팬티엄 III-450 Dual CPU, RAM 128M, wowlinux7.0, gcc 2.96)

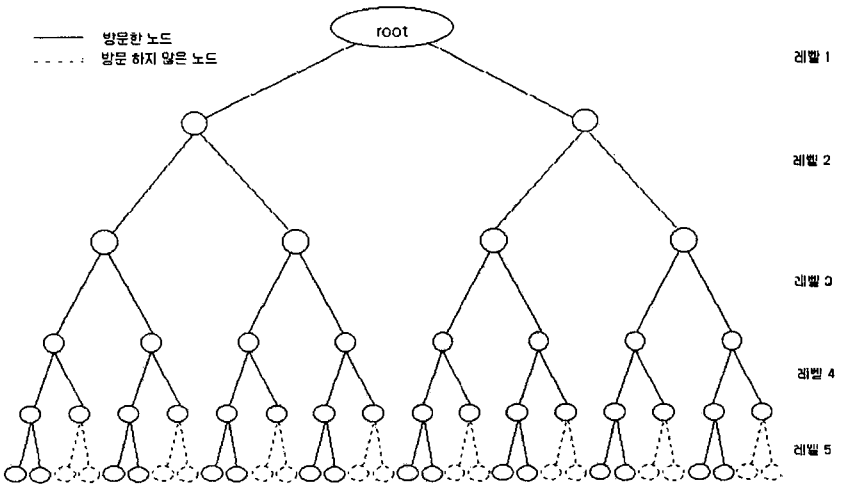

그림 2: Yang & Kubat의 방법

표 1: bridge network의 계산식

Iteration(i)	State(S_i)	Flow($R(S_i)$)	$P_L(S_i)$	계 산 식
1	ϕ	8		8
2	1	6	0.5	$8+(0.5)\times(6-8)=7$
3	2	2	0.2	$7+(0.2)\times(2-8)=5.8$
4	1,2	0	0.2	$5.8+(0.2)\times(0-6)=4.6$
5	3	5	0.09	$4.6+(0.09)\times(5-8)=4.33$
6	1,3	3	0.09	$4.33+(0.09)\times(3-6)=4.06$
7	2,3	2	0.06	$4.06+(0.06)\times(2-2)=4.06$
8	1,2,3	0	0.06	$4.06+(0.06)\times(0-0)=4.06$

4. 결론

본 논문에서는 주어진 네트워크에서 각 링크들의 가동확률과 링크 용량이 주어졌을 때, coherence property를 갖고 있는 네트워크 성능측도의 상·하한값을 구하는 효율적인 알고리즘을 제시하였다. 제안된 방법은 Yang & Kubat의 방법에 비해서 주어진 coverage나 오차한계내 근사치를 구하는데 있어, 메모리와 시간이 절약됨을 알 수 있다. 또한 본 논문에서의 상·하한값 계산 방법은 MPS를 찾는 알고리즘과는 독립적으로 운용될 수 있으므로, 보다 효율적으로 MPS를 찾는 방법과 결합 운용될 경우 그 효율은 더욱 증대 될 것이다.

참고문헌

[1] V.O.K. Li and J.A. Silvester, *"Performance analysis networks with unreliable components"*, IEEE Transactions on Communications, vol. COM-32, pp 1105-1110, 1984.

[2] Y.F. Lam and V.O.K. Li, *"An improved algorithm for performance analysis of networks with unreliable components"*, IEEE Transactions on Communications, vol. COM-34, pp 496-497, 1986.

[3] D.R. Shier, *"A new algorithm for performance analysis of communication systems"*, IEEE Transactions on Communications, vol. COM-36, pp 516-519, 1988.

[4] E.J. Valvo, D.R. Shier and R.E. Jamison, *"Generating the most probable states of a communication system"*, Proc. IEEE INFOCOM '87, San Francisco, pp 1128-1136, 1987.

[5] T.M.S. Gomes and J.M.F. Craveirinha, *"Algorithm for sequential generation of states in failure-prone communication network"*, IEEE Proc Commun. vol. 145, pp 73-79, 1998.

[6] C.L. Yang and P. Kubat, *"An algorithm for network reliability bounds"*, Operations Research Society of America Journal on computing, vol. 2, No. 4, fall 1990.