확률적 네트워크의 신뢰도 평가를 위한 분산 감소기법의 응용*

An Application of Variance Reduction Technique for Stochastic Network Reliability Evaluation

하경재**, 김원경***
Kyung-Jae Ha, Won Kyung Kim

Abstract

The reliability evaluation of the large scale network becomes very complicate according to the growing size of network. Moreover if the reliability is not constant but follows probability distribution function, it is almost impossible to compute them in theory. This paper studies the network evaluation methods in order to overcome such difficulties. For this an efficient path set algorithm which seeks the path set connecting the start and terminal nodes efficiently is developed. Also, various variance reduction techniques are applied to compute the system reliability to enhance the simulation performance. As a numerical example, a large scale network is given. The comparisons of the path set algorithms and the variance reduction techniques are discussed.

Key Word : Network, Path set, Reliability, Simulation, Variance reduction technique

* 본 연구는 2000년도 경남대학교 학술연구 조성비의 지원에 의한 것임
** 경남대학교 정보통신공학부
*** 경남대학교 벤처창업학부

1. 서론

지금까지 네트워크 구조의 신뢰도를 평가하고자 할 때 여러 이론적인 방법이 연구되어 왔다. 이러한 이론적인 방법들은 네트워크의 구조가 복잡해지고 그 크기가 커지게 되면 연산량이 기하 급수적으로 증가하게 되어 실제로 그 값을 구하기가 매우 어려워진다. 더군다나 네트워크 구조의 부품신뢰도가 상수 값이 아닌 확률분포를 갖는 확률변수이라면 특수한 경우를 제외하고는 이론적으로 신뢰도를 평가하는 것이 거의 불가능해진다. 이것을 해결하는 유일한 대안은 시뮬레이션으로 신뢰도를 구하는 것이다. 본 연구는 네트워크로 구성된 coherent system의 신뢰도를 구하는 문제이므로, 먼저 네트워크를 그래프로 표현한 후 여러 가지 신뢰도 계산기법을 위한 네트워크 그래프의 경로집합(path set)을 구하는 알고리즘, 각 부품의 고장확률이 확률분포함수로 주어 졌을 때의 신뢰도 값을 확률 통계적인 접근방법 및 다양한 시뮬레이션을 통해 신뢰도를 추정하는 방법을 연구한다. 특히 방향 네트워크의 경로집합을 효과적으로 찾는 방법과 이를 이용하여 신뢰도를 구하기 위한 시뮬레이션의 효율을 높이기 위한 분산감소기법(Variance Reduction Technique)을 적용하고 이들 기법들을 대규모 네트워크 예를 통해 비교연구 하고자 한다.

2. 네트워크 경로집합의 탐색

네트워크 구조의 신뢰도를 이론적으로 구하는 기법은 과거부터 많이 연구되어 왔다[1,2,6,7, 10,12]. 그 가운데에서도 네트워크의 경로집합을 이용하여 신뢰도를 계산하는 방법들이 시도되었다. 경로집합을 구하기 위해서는 네트워크의 연결행렬을 멱승하여 성공행렬을 구하는 방법[5], 연결행렬의 소행렬식을 이용하는 방법[8,11] 등이 개발되었다. 예로써, n개의 node로 이루어진 시스템의 네트워크 구조를 표시하기 위해 다음과 같은 연결행렬(Connection Matrix) $[C_{ij}]_{n\times n}$와 몇 가지 변수들을 정의하자.

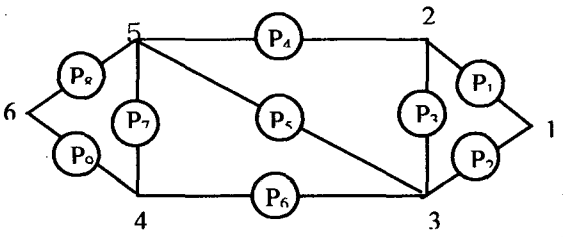
P_k = 부품 k

i = 노드 번호

C_{ij} = 노드 i에서 노드 j로 연결된 링크 또는 부품의 번호

행렬 C는 (i,j) 요소값으로 표시한다. 즉 $P_k = C_{ij}$이다.

네트워크의 구조는 다음의 <그림 1>과 같다고 하자.



<그림 1> 간단한 네트워크의 예

이 경우 위 네트워크의 연결행렬 C는 다음과 같게 된다.

$$[C] = \begin{bmatrix} 1 & P_1 & P_2 & 0 & 0 & 0 \\ P_1 & 1 & P_3 & 0 & P_4 & 0 \\ P_2 & P_3 & 1 & P_6 & P_5 & 0 \\ 0 & 0 & 0 & 1 & P_7 & P_9 \\ 0 & 0 & P_5 & P_7 & 1 & P_8 \\ 0 & 0 & 0 & P_9 & P_8 & 1 \end{bmatrix}$$

이 연결행렬을 이용하여 (n-1)회의 멱 행렬곱을 계산한 후 성공행렬을 구하게 되면 경로집합들을 구할 수 있고 이를 시스템의 신뢰도를 구하는 데에 이용할 수 있다. 그러나 성공행렬은 네트워크의 구조가 커지면 엄청난 횟수의 계산과 메모리를 요구하게 되므로 비효율적이다. 연결행렬의 소 행렬식을 이용하는 방법도 성공행렬보다는 계산 량이 감소하기는 하나, 경로와는 관계없이 불필요한 폐쇄 경로와 중복노

드가 포함된 경로가 같이 구하여지므로 이를 제거하는 방법을 강구하여야 되고 실제 이를 컴퓨터 프로그램 상에서 구현하는 데에 있어서 메모리가 낭비되고 계산시간이 길어지게 된다. 그러므로 지금부터는 이러한 방법보다 더욱 쉽게 경로집합을 쉽게 찾는 재귀적(recursive) 탐색알고리듬에 대해 논하고자 한다.

이 알고리듬은 단순 명료하여 실제 컴퓨터 로직 구현이 용이하며 특히 대형 문제의 경우에 기존의 다른 어떠한 방법보다도 경로집합을 찾는데 있어서 빠른 시간 내에 적은 메모리로도 구할 수 있다. 다음은 이를 위한 경로집합 탐색 알고리듬 Pathset과 여기서 사용되는 변수들의 정의이다.

r = 경로집합 내의 항 수
q = 경로항 내에서의 노드 수
n_f = 네트워크 상의 노드 번호 f
$L_{n_f,i}$ = 노드 f 에서 분기되어 나가는 i번째 노드의 번호
$L_{n_f,0}$ = 노드 f 에서 분기되는 노드 수
s = 시작 노드(source node)의 번호
t = 종점 노드(termination node)의 번호
E_{rq} = r 번째 경로의 q 번째 노드의 번호
E_{r0} = r 번째 경로의 노드 수

Recursive Subroutine Pathset(n_f, q)

if $n_f = s$ then $E_{1,1} = n_f$
(만일 노드 f가 시작 노드이면 첫 번째 경로의 첫째 노드는 시작 노드이다)
if $n_f = t$ then $r = r+1$ and return
(만일 노드 f가 종점 노드이면 경로의 수 r을 하나 증가시키고 호출함수로 되돌아간다)
For i=1 to $L_{n_f,0}$
(노드 f에서 분기되는 모든 노드에 대해서)
$n_k = L_{n_f,i}$
(노드 k 는 노드 f에서 분기되는 i 번째 노드이다)
For j=1 to q
(경로항 내의 모든 노드에 대해서)
if $E_{r,j} = n_k$ then next i
(만일 추가되는 새 노드 k 가 기존의 경로 r에 이미 있으면 다음 인덱스 i를 실행)
End for
$r_0 = r$
(현재의 경로번호 r을 r_0에 저장)
q_0=q
(현재 경로의 노드 수 q를 q_0에 저장)
$q = q_0 + 1$
(다음 노드 수 q는 현재의 노드 수에 1을 증가시킨 값)
$E_{r,q} = k$
(노드 k를 경로 r의 q번째 요소로 등록)
$E_{r,0} = q$
(경로 r의 현재 노드 수 q를 저장)
Call Pathset(n_k ,q)
(노드 k와 현재까지의 노드 수가 q라는 정보를 재귀함수로 넘겨 호출한다)
$q = q_0$
(노드 수를 재귀함수 호출전의 노드 수로 복귀시킨다)
For j=1 to q
(경로항 내의 모든 노드에 대해서)
$E_{r,j} = E_{r_0,j}$
(현재 경로 r의 노드정보는 재귀함수 호출전의 경로 r_0와 같다)
End for
End for
Return

위의 재귀루틴을 처음에 $r=1$, $n_f=s$, $q=1$로 하여 호출하면 네트워크내의 총 r 개의 경로에 대한 경로집합 $\{E_{i,q},\ i=1,\cdots,r,\ q=1,\cdots,E_{i0}\}$가 구해진다. k 개 부품들의 신뢰도에 따라 각 부품이 정상 또는 고장인 경우에 이 시스템의 정상 또는 고장여부를 판단하는 재귀적 알고리듬 Success와 이를 위한 변수들의 정의는 다음과 같다.

h_i = 노드 i가 방문되었을 경우는 1, 아니면 0

w_k = 부품 k가 정상이면 1, 아니면 0

R = 시스템이 정상이면 1, 아니면 0

s = 시작 노드

t = 종점 노드

$C_{n_i n_j}$ = 노드 i 에서 노드 j로 연결된 링크 또는 부품의 번호

Recursive Subroutine Success(n_f)

if $R=1$ then Return (이미 시스템이 정상이면 수행할 필요 없이 호출함수로 되돌아간다)

if $n_f=s$ then $h_i=0$,for i=1,...,n (시작노드로부터의 호출이면 노드 방문이 없다고 초기화)

For j=1 to $L_{n_f,0}$

(노드 f로부터 분기되는 모든 노드에 대해서)

$n_t=L_{n_f,j}$

(노드 t는 노드 f에서 분기된 j번째 노드)

if $h_{n_t}=1$ then next j

(노드 t가 이미 방문되었다면 다음 인덱스 j를 실행)

$k=C_{n_f,n_t}$

(k는 노드 f에서 노드 t로 연결된 부품의 번호)

if $w_k=0$ then next j

(만일 부품 k가 고장이면 다음 인덱스 j를 실행)

if $n_t=t$ then $R=1$ and Return

(만일 노드 t가 종점노드이면 시스템은 정상이고 호출함수로 되돌아간다)

$h_{n_f}=1$

(노드 f가 방문되었음을 기록한다)

Call Success(n_t)

(노드 t로 시작하는 함수를 재귀적으로 호출한다)

if $R=1$ then Return

(이미 시스템이 정상이면 나머지 인덱스 j를 실행하지 말고 호출함수로 되돌아간다)

End for

Return

3. 네트워크 신뢰도 시뮬레이션

연결행렬의 소행렬식과 성공행렬을 이용하는 이론적 기법들은 부품의 신뢰도가 상수가 아닌 확률분포함수를 따르게 되면 그나마 더 이상 적용이 어려워지고 단순히 특정 시점에서의 부품신뢰도의 평균을 상수로 가정하여 시스템 신뢰도를 구하게 된다. 네트워크의 신뢰도를 구할 때 경로집합을 구하지 않고 전체 사상을 전개하여 구하는 방법의 하나로 사상 공간법(Event Space Method)[7]이 있다. 이 방법은 주어진 시스템에서 발생 가능한 모든 경우를 나열하여 사상공간의 목록을 작성한 후 목록의 i 번째 사상의 시스템이 고장인 경우와 정상인 경우를 Success 루틴을 이용하여 판단한다. 이때 정상인 경우의 확률총계가 전체 시스템의 신뢰도가 된다. 즉 R_i 를 i 번째 조합의 시스템 신뢰도라 하면 전체 시스템의 신뢰도 R_s는 다음과 같다.

$$R_s=\sum_{i=1}^{2^k} R_i$$

이 방법은 목록 작성이 간편하다는 장점이 있으나 시스템의 전체 부품수가 k개이라면 목록의 총 사건 수는 2^k개가 되므로 k가 클 경우 사실상 이용이 어려워지는 단점이 있다. 예를 들어 부품수가 32개 이상이라면 총 사건수가 4,294,967,296개 이상이 된다. 이는 PC에서 처리하기에는 너무 오랜 시간이 소요되므로 사실상 불가능하다. 그러나 부품 수 k가 크지 않다면 각 조합이 서로 독립적이므로 각 부품신뢰도 분포함수의 평균치를 알 경우 정확한 신뢰도를 이론적으로 구할 수 있다는 것이 장점이다.

여기서는 네트워크 규모가 커질 때 문제를 해결하는 방법으로 앞에서 개발한 경로탐색 알고리듬을 이용하여 경로집합을 찾은 후 시스템의 신뢰도를 시뮬레이션으로 구하는 다음과 같은 방법에 대해 살펴보자.

1) 성공경로법(Success Path Method)
2) 확률적경로법(Random Path Method)
3) 경로집합법(Pathset Method)

확률적 경로법(Random Path Method)

이 방법은 각 부품의 고장 또는 정상여부를 재귀적 루틴 success의 호출 전에 미리 정하여 놓는 것이 아니라 루틴 내에서 다음 노드로 이동할 때 부품의 신뢰도 p_i에 따라 w_k가 1인지 또는 0인지를 확률적으로 결정하여 노드 k로의 이동여부를 결정하는 방법이다. 이러한 작업을 m회 반복하여 매 번의 루틴 호출 시에 시스템이 정상일 경우 성공 횟수를 하나씩 증가시킨다. 그러면 총 m회의 반복에서 성공한 횟수의 비율이 전체 시스템의 신뢰도가 된다. 즉,

$$R_s = \frac{1}{m}\sum_{i=1}^{m} I(\text{시스템 정상})$$

여기서 함수 $I()$는 i 번째 반복 수행 시에 시스템이 정상이면 1이고 아니면 0가 되는 Indicator 함수이다. 이 방법은 부품수가 아무리 커져도(즉 큰 k에 대해서도) 시스템 신뢰도 R_s를 구할 수 있다는 것이 장점이다. 이때 시뮬레이션 반복횟수 m이 클수록 정확한 신뢰도가 구해진다.

성공경로법(Success Path Method)

각 부품의 정상 또는 고장여부를 부품의 신뢰도 p_i에 따라 확률적으로 미리 정하여 놓은 후 Success 루틴을 호출한다. 이러한 작업을 m회 반복하여 매 Success 루틴 호출 시에 시스템이 정상일 경우 성공 횟수를 하나씩 증가시킨다. 그러면 총 m회의 반복에서 성공한 횟수의 비율이 전체 시스템의 신뢰도가 된다. 즉,

$$R_s = \frac{1}{m}\sum_{i=1}^{m} I(\text{시스템 정상})$$

여기서 함수 $I()$는 i 번째 반복 수행 시에 시스템이 정상이면 1이고 아니면 0가 되는 Indicator 함수이다. 이 방법은 부품수가 아무리 커져도(즉 큰 k에 대해서도) 시스템 신뢰도 R_s를 구할 수 있다는 것이 장점이다. 이때 시뮬레이션 반복횟수 m이 클수록 정확한 신뢰도가 구해지게 된다. 시스템 신뢰도 R_s를 구하는 알고리듬은 다음과 같고 신뢰도의 추정치는 $R_s = x/m$ 로 계산된다.

```
x=0 (성공 횟수의 초기치)
For i=1 to m (m회의 반복에 대하여)
        Find {w_j=0 or 1, j=1,…,k}
        (각 부품의 고장 여부를 정한다)
        R=0
        (호출 전에는 시스템이 고장이라고 가정)
        Call Success(s)
        (시작 노드 s로부터 호출)
        if R=1 then  x=x+1
```

(시스템이 정상이면 성공 횟수를 1 증가)
End for

경로집합법(Pathset Method)

이 방법은 시스템의 정상여부를 재귀루틴 Success를 이용하는 것이 아니라 Pathset 루틴을 사용하여 구한 경로집합을 이용하여 판단한다. 경로집합에서 구한 경로들 중 어느 하나라도 성공이 이루어지면 시스템은 정상이다. 이러한 작업을 m회 반복하여 시스템이 정상인 횟수의 비율을 구하면 전체 시스템의 신뢰도가 된다. 즉,

$$R_s = \frac{1}{m}\sum_{l=1}^{m} S_l$$

여기서 l 번째 반복 수행 시에 시스템이 정상이면 S_l 가 1이고 아니면 0이다. 이 방법도 부품수가 아무리 많아져도 시스템 신뢰도 R_s를 구할 수 있다는 것이 장점이다. 이때 시뮬레이션 반복횟수 m이 클수록 정확한 신뢰도가 구해지게 된다.

이 방법으로 구한 신뢰도의 결과 값은 성공경로법과 같으며 성공여부를 판단하는 함수 S_l을 구하는 알고리듬은 다음과 같다. 여기서 변수 E_{ij}는 노드 i에서 노드 j로 연결된 부품의 번호이고 E_{i0}는 이때 연결된 부품의 총 수로 정의하자.

function S_l
For i=1 to r (r 개의 모든 경로에 대해서)
  For j=1 to E_{i0}
    (경로내의 모든 부품에 대해서)
    $k = E_{ij}$
    (k는 경로 i의 j 번째 부품)
    if $w_k = 0$ then next i
      (부품 k가 고장이면 다음 인덱스 i에 대해 실행)
  End for
  S_l = 1 and Return
    (경로내의 모든 부품이 정상이므로 시스템은 정상이고 호출루틴으로 되돌아간다)
End for
S_l = 0
  (어느 하나의 경로도 성공이 아니므로 시스템은 고장이다)
Return

4. 분산감소기법의 적용

앞에서 논의된 방법으로 시스템의 신뢰도를 구하게 되면 여러 번의 반복실험을 통하여 얻어진 실험치이므로 분산값을 갖게 된다. 이 절에서는 신뢰도 추정시 시뮬레이션의 효율성을 기하기 위해 추정된 모수의 분산을 줄이기 위한 다음과 같은 분산감소기법들[3,4,9]을 적용한다.

1) Antithetic Variate (AV)
2) Control Variate (CV)
3) Antithetic Variate와 Control Variate의 동시적용 (AV+CV)

Antithetic Variate (AV)

AV는 단일 시스템의 시뮬레이션에 적용할 수 있는 기법으로 각 시뮬레이션 수행간에 음의 상관을 야기시키는 방법이다. 이 방법의 기본 개념은 시스템 수행의 쌍(pair)을 형성함으로써 처음 수행에서 작은 관측치는 두 번째 수행에서 큰 관측치를 상쇄하게 되어 두 관측치 들이 음으로 상관되게 하는 것이다. 관측결과 m 개의 시스템 수행인 신뢰도 쌍 $(R_1^{(1)}, R_1^{(2)}), \cdots, (R_m^{(1)}, R_m^{(2)})$을 형성

함으로써 각 쌍은 다른 쌍과는 독립이 되도록 한다. 여기서 $R_j^{(1)}$과 $R_j^{(2)}$는 음의 상관을 가지며 앞에서 구한 여러 알고리듬을 사용하였을 경우 j 번째 반복에서의 시스템 신뢰도이다. 이제

$$R_j=(R_1^{(1)}+R_1^{(2)})/2$$

라하고, $\mu=E(R_j^{(l)})=E(R_j)$의 점 추정치를

$$\overline{R}(n)=\sum_{j=1}^{n}R_j/n$$

이라 하면

$$Var[\overline{R}(n)]=[Var(R_j^{(1)})+Var(R_j^{(2)})+2Cov(R_j^{(1)},R_j^{(2)})]/4n$$

이다. 공분산항이 음이라면 $Var[\overline{R}(n)]$ 은 감소한다. 이 방법은 앞에서 언급한 모든 시스템 신뢰도를 구하는 기법에 쉽게 적용할 수 있다. 시스템 신뢰도 $R_i^{(1)}$와 $R_i^{(2)}$를 추정할 때 부품의 신뢰도 p_i 가 시간에 종속적인 분포함수와 관계될 경우에는 역변환 방법(Inverse Transformation Method)을 사용하여 확률난수(Random Variate)를 생성시켜야 한다. 이때 $R_i^{(1)}$에는 $U(0,1)$에서 추출한 일양난수 U를 적용하고 $R_i^{(2)}$에는 일양난수 (1-U)를 적용한다.

Control Variate Estimator (CV)

시스템의 신뢰도 R의 추정치 $\hat{R}$을 구하고자할 때 부품의 신뢰도 추정치 $\hat{p}_i$는 R과 음이던 양이던 상관이 있는 시뮬레이션에 포함된 확률변수이다. $p_i=E[\hat{p}]$의 값은 미리 알고 있으므로 $\hat{p}_i$과 p_i의 편차인 $(\hat{p}_i-p_i)$를 사용하여 제어된 추정치는 다음과 같이 정의된다.

$$R_c=\hat{R}-a[\hat{p}_i-p_i]$$

여기서 a는 $\hat{p}_i$와 $\hat{R}$ 사이의 상관관계와 같은 부호를 갖는 상수이며, R_c을 조절하는 데에 있어서 $(\hat{p}-p_i)$ 비율로서 사용된다. 그러면 R_c는 $\hat{R}$ 보다 더 작은 분산을 갖는 R의 불편 추정치가 된다. 즉,

$$Var(R_c)=Var(\hat{R})+a^2Var(\hat{p}_i)-2aCov(\hat{R},\hat{p}_i)$$

에서 $a^2Var(\hat{p}_i)<2aCov(\hat{R},\hat{p}_i)$이면 R_c는 $\hat{R}$ 보다 분산이 더 작게 된다. 일반적으로는 k개 부품의 CV 추정치에 대해 각 기대치 $p_1,p_2,\cdots,p_k$ 를 알고 있으므로 제어된 추정치는

$$R_c=\hat{R}-\sum_{i=1}^{m}a_i[\hat{p}_i-p_i]$$

로 정의된다. 이제 변수 V_i를 i 번째 시뮬레이션 반복에서 부품 i가 정상이면 1 아니면 0이라 하면, $\hat{R}$ 과 양의 상관 관계를 갖는 확률변수 $\hat{p}_i$ 는 i 번째 부품의 신뢰도 추정치인

$$\hat{p}_l=\frac{1}{m}\sum_{i=1}^{m}V_i$$

로 구할 수 있다.

AV와 CV의 동시적용

앞에서 제안된 AV와 CV를 함께 적용하면 분산을 더욱 감소시킬 수 있다. 즉, AV의 기본개념

인 음 상관을 유도한 뒤 그 결과에 다시 CV를 적용하는 방법이다. AV를 적용하여 얻어진 각 부품의 신뢰도를 조절변수로 두고 시스템의 신뢰도를 출력확률변수로 하는 CV기법을 적용한다. 얻어진 $(R_1^{(1)}, R_1^{(2)}), (R_2^{(1)}, R_2^{(2)}), \cdots, (R_n^{(1)}, R_n^{(2)})$에 대해 $\widehat{R_i}=(R_i^{(1)}+R_i^{(2)})/2$와 같이 부품 i의 신뢰도를 계산한다. 계산된 각 부품의 신뢰도 $\widehat{p_1}, \widehat{p_2}, \cdots, \widehat{p_k}$ 를 CV로 하여 다중 CV기법을 적용하기 위해서는 출력확률변수 $\widehat{R_i}$를 조절하는데 필요한 a_i가 필요하다. 따라서 m회의 반복수행을 하여 아래의 결과를 얻은 후에 이로부터 중회귀분석을 실시하여 회귀계수를 추정한다.

$$\begin{pmatrix}\widehat{R_1}\\ \widehat{R_2}\\ \\ \widehat{R_m}\end{pmatrix}=\begin{pmatrix}1 & \widehat{p_{11}} & \widehat{p_{12}} & \cdots & \widehat{p_{1k}}\\ 1 & \widehat{p_{21}} & \widehat{p_{22}} & \cdots & \widehat{p_{2k}}\\ & & & & \\ 1 & \widehat{p_{m1}} & \widehat{p_{m2}} & \cdots & \widehat{p_{mk}}\end{pmatrix}\begin{pmatrix}a_0\\ a_1\\ \\ a_k\end{pmatrix}$$

추정된 a_i를 사용하여 조절된 추정치 R_c는 다음과 같이 구한다.

$$R_c=\hat{R}-\sum_{i=1}^{m}a_i\,[\hat{p}_i-p_i]$$

이제 추정치를 구하는 방법을 l회 반복하여 $Var[\overline{R_c}(l)]$을 구한다.

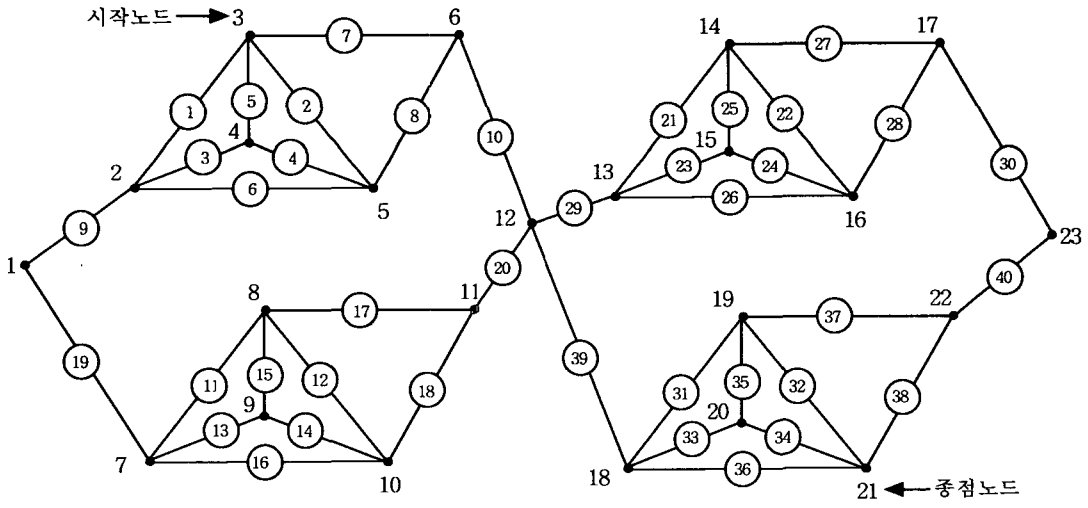
다음절에서는 일정한 횟수의 시뮬레이션을 수행한 후에 분산감소기법을 적용하지 않고 crude method로 구한 신뢰도와 분산감소기법을 적용한 신뢰도와의 점 추정치와 분산을 구하여 서로 비교하기로 한다.

5. 응용 예

응용 예로서 <그림 2>와 같이 노드수가 23개이고 부품수가 총 40개인 네트워크를 살펴본다.

그림에서 동그라미 안의 숫자로 표시된 것이 부품번호이고 꼭지점의 숫자는 노드번호이다.

각 40개의 부품마다 적용된 분포함수와 모수는 다음의 <표 1>과 같다.



<그림 2> 복잡한 네트워크의 예

<표 1> 네트워크에 사용된 부품의 확률분포

부품	분포	모수
1, 2, 6, 7, 11,12,16,17, 21,22,26,27, 31,32,36,37	Exponential	$\mu=10$ (평균)
3, 8, 13,18 23,28, 33,38	Normal	$\mu=8$ (평균) $\sigma=2$ (편차)
4, 9, 14,19, 24,29, 34,39	Weibull	$\beta=2$ (Shape) $\eta=7$ (Scale) $\gamma=0$ (Location)
5, 10, 15,20, 25,30, 35,40	Uniform	$a=5$ (최소) $b=9$ (최대)

<그림 2>의 네트워크를 표시하는 연결행렬은 <표 2>와 같다. 행렬요소 중에서 -1은 프로그램 로직 구현상의 편의를 위한 대각요소임을 표시하고, 0은 연결되지 않았음을 나타내며 그 외의 숫자는 행과 열의 노드번호에 따라 연결된 네트워크상의 부품번호를 나타낸다.

경로탐색 알고리듬의 성능을 알아보기 위해 위의 네크워크에서 시작 노드는 임의로 3번을 선택하고 종점 노드는 21로 설정하였다. 그 결과 경로집합 탐색알고리듬이 찾아낸 총 경로 개수는 5092개 였고, 이를 위한 연산시간은 600MHz CPU의 펜티엄 3 컴퓨터에서 포트란으로 프로그래밍하여 수행한 결과 0.05초가 소요되어 탐색알고리듬의 속도가 상당히 빠름을 알 수 있다.

<표 2> 그림 2의 네트워크의 연결행렬

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	-1	9	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	9	-1	1	3	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	-1	5	2	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	3	5	-1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	6	2	4	-1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	7	0	8	-1	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
7	19	0	0	0	0	0	-1	11	13	16	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	11	-1	15	12	17	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	13	15	-1	14	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	16	12	14	-1	18	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	17	0	18	-1	20	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	10	0	0	0	0	20	-1	29	0	0	0	0	39	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	29	-1	21	23	26	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	21	-1	25	22	27	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	23	25	-1	24	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	26	22	24	-1	28	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	28	-1	0	0	0	0	0	30
18	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	-1	31	33	36	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	-1	35	32	37	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	35	-1	34	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	32	34	-1	38	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	38	-1	40
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	40	-1

분산감소기법을 적용하지 않은 Crude 방법과, 분산감소기법의 하나인 Antithetic 방법으로 신뢰도를 추정할 경우의 시뮬레이션 반복 수는 5000번으로 하였다. Crude 방법과 동일한 총 반복횟수로 공정한 비교를 하기 위하여 Control Variate를 써서 추정할 경우에는 회귀분석 계수를 추정하기 위한 반복 수는 500회, 그리고 각 회귀점에서의 반복 수를 100회로 하여 전체 반복 수는 5000번으로 동일하게 설정하였다. 각 부품의 신뢰도가 시간에 종속적인 분포함수로 주어지므로 시간의 범위는 3에서부터 7까지로 하여 시스템의 신뢰도가 10% 이상이 되는 범위로만 한정하였다.

사상공간법은 시뮬레이션 구현시에 부품의 개수가 32개를 넘었기 때문에 PC에서의 연산시간이 매우 크므로 이 예제에는 적용하지 않았다. 경로집합법은 성공경로법과 시뮬레이션 구현방법은 다르나 수행된 결과가 같기 때문에 여기서는 싣지 않았으나 수행시간은 훨씬 오래 걸렸다. 따라서 본 논문에서는 확률경로법과 성공경로법 두 알고리듬만을 적용한 결과만을 살펴보기로 한다.

시뮬레이션은 분산감소기법을 적용하지 않은 Crude 기법과 다음과 같은 여러 가지 상황에 분산감소기법을 적용한 결과들을 비교하여 살펴보기로 한다:

- Antithetic Variate 기법(AV)
- Control Variate 기법(CV)
- 단계적 회귀(Stepwise Regression)로 선택한 변수의 Control Variate 기법(ST)
- Control Variate에 Antithetic를 적용한 기법 (CV+AV)
- Stepwise Regression 에 Antithetic Variate를 적용한 기법(ST+AV)

Control Variate를 적용할 때 Control 선택기준의 조합이 총 2^{40} 가지가 되므로 어느 Control을 선택하느냐가 문제인데 여기서는 두 가지 방법으로 선택하였다. 하나는 모든 부품을 Control로 선택하는 방법(CV)이고 다른 하나는 단계적 회귀(stepwise regression)를 적용하여 선택된 변수에만 Control Variate를 적용하는 방법(ST)이다. 본 예제에서 CV는 분산감소를 최대로 하지만 그만큼 연산시간이 많이 요구된다. Control 수를 줄이면서 분산감소량을 만족할 만큼 얻기 위한 ST는 대신 단계별 회귀를 실시해야 하는 연산시간이 더 필요하게 된다. 단계적 회귀를 사용하면 40개의 전체 부품가운데서 일정한 기준을 만족하면서 시스템 신뢰도에 우선적으로 영향을 끼치는 부품들을 선정해 준다. 여기서 적용한 기준은 변수의 선택과 탈락 시에 부분 F-검정의 기각 또는 채택 판정에 사용되기 위한 기준으로 유의수준을 각각 $\alpha=\beta=0.2$로 설정하였다.

다음의 <표 3>은 단계적 회귀를 확률경로법에 적용하였을 때 선택된 Control(부품)의 목록이다. 대체로 시간이 지남에 따라 더 많은 부품이 선택되어 시스템의 신뢰도에 영향을 주고

<표 3> 단계적 회귀를 확률경로법에 적용하였을 때 선택된 부품

Time	부품 번호
3	2 7 11 16 21 24 26 29 34 38 39
4	1 2 4 6 7 16 22 26 27 28 29 32 36 37 38 39
5	2 4 6 7 8 13 18 21 22 23 24 26 27 28 29 32 33 36 37 38 39
6	2 4 5 6 7 8 9 10 12 14 19 21 23 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
7	1 2 3 4 5 6 7 8 10 11 12 16 18 19 21 22 24 25 26 27 28 29 30 31 32 33 34 36 37 38 39 40

있음을 알 수 있다.

<표 4>는 확률적 경로법에 의한 시스템 신뢰도(%)의 평균값을 나타낸 표이다. 시간이 지남에 따라서 신뢰도가 하락하고 있으며 모든 방법들의 신뢰도 추정치의 평균치가 대체로 비슷함을 알 수 있다. 신뢰도 추정치의 분산은 <표 5>에 나타나 있다. 각 분산감소기법간에 어느 정도의 분산이 감소되었는지 비교하기 위하여 Crude 일 경우와의 상대적인 감소량을 퍼센트로 표시한 것이 <표 6>이다. 즉, <표 6>에서의 분산감소량은

$$\text{분산 감소량}(\%) = \frac{(Crude\text{일 경우의 분산} - \text{분산감소 기법적용시의 분산})}{Crude\text{일 경우의 분산}} * 100$$

<표 6>에서 보면, Antithetic Variate 기법만을 적용한 경우에는 Crude 일 경우보다 오히려 분산이 증가한 경우도 있었으나, Antithetic Variate 기법에 Control Variate를 함께 적용한 경우(AV+CV)에는 분산이 대체로 60% ~ 98% 정도로 현저하게 감소하고 있음을 알 수 있다. CV와 ST는 비슷한 정도의 분산감소량을 보여주었으며, 특히 AV도 함께 적용한 경우도 분산의 감소량은 비슷하였으나 CV 쪽이 약간 더 낮은 감소량을 보여주고 있다. 이는 ST 기법이 분산감소에 유의한 영향을 미친다고 판단되는 부품들만을 고른 것에 반하여 CV는 모든 부품을 다 선택한 것에 기인한 것이지만, 때로는 ST가 CV보다 낮은 경우도 있었다. 결국 분산감소를 이루기 위해 ST를 사용하는 노력보다는 모든 부품에 CV를 적용하는 편이 시간과 노력이 적게 든다고 본다.

<표 4> 확률적 경로법에 의한 시스템 신뢰도의 평균(%)

Time	Crude	AV	CV	ST	CV+AV	ST+AV
3	97.06	97.1	97.07	97.07	97.07	97.07
4	91.38	91.37	91.37	91.37	91.39	91.39
5	80.35	80.34	80.35	80.35	80.35	80.35
6	45.75	45.81	45.73	45.73	45.71	45.73
7	16.73	16.79	16.66	16.67	16.66	16.68

<표 5> 확률적 경로법에 의한 시스템 신뢰도의 분산

Time	Crude	AV	CV	ST	CV+AV	ST+AV
3	0.0096	0.0109	0.0004	0.0004	0.0007	0.0006
4	0.0327	0.0266	0.0020	0.0017	0.0008	0.0009
5	0.0434	0.0210	0.0028	0.0023	0.0023	0.0026
6	0.0265	0.0599	0.0081	0.0084	0.0081	0.0100
7	0.0238	0.0262	0.0033	0.0042	0.0019	0.0022

<표 6> 확률적 경로법에 의한 시스템 신뢰도의 분산감소량(%)

Time	Crude	AV	CV	ST	CV+AV	ST+AV
3	0	-13.29	95.63	95.52	93.02	93.34
4	0	18.59	93.82	94.67	97.43	97.18
5	0	51.68	93.47	94.70	94.81	93.92
6	0	-125.92	69.54	68.19	69.31	62.38
7	0	-10.16	85.95	82.28	91.90	90.79

<표 7> 성공경로법에 의한 시스템 신뢰도의 평균값

Time	Crude	AV	CV	ST	CV+AV	ST+AV
3	96.92	96.95	96.92	96.92	96.96	96.96
4	90.61	90.63	90.56	90.58	90.62	90.62
5	78.13	78.19	78.10	78.10	78.17	78.18
6	38.04	38.00	38.07	38.07	38.06	38.04
7	13.00	12.95	12.98	13.00	12.98	12.96

<표 8> 성공경로법에 의한 시스템 신뢰도의 분산값

Time	Crude	AV	CV	ST	CV+AV	ST+AV
3	0.0040	0.0083	0.0046	0.0036	0.0051	0.0045
4	0.0151	0.0142	0.0175	0.0162	0.0287	0.0228
5	0.0381	0.0241	0.0127	0.0133	0.0558	0.0464
6	0.0557	0.0231	0.0244	0.0244	0.0096	0.0159
7	0.0274	0.0233	0.0151	0.0208	0.0198	0.0170

<표 9> 성공경로법에 의한
시스템 신뢰도의 분산감소량(%)

Time	Crude	AV	CV	ST	CV+AV	ST+AV
3	0	-108.51	-15.70	10.16	-28.95	-13.09
4	0	6.13	-15.94	-7.69	-90.29	-51.00
5	0	36.67	66.72	64.97	-46.56	-21.82
6	0	58.55	56.25	56.20	82.78	71.45
7	0	15.05	44.92	24.05	27.75	38.15

마찬가지로 성공경로법을 적용하여 신뢰도 추정치의 평균, 분산, 그리고 분산감소량을 구한 것이 각각 <표 7>, <표 8>, <표 9>에 나타나 있다.

우선 평균을 비교하여보면 각 분산감소기법들 간에는 거의 비슷한 값을 구해 주고 있다. 분산 감소량의 경우, AV와 CV가 분산이 증가하는 경우도 있었으나 대체로는 감소시키고 있다. 그러나 CV와 ST를 AV와 함께 적용한 경우에는 오히려 분산이 증가하는 경우도 많이 있다는 것을 알 수 있다. 즉, CV나 ST에 AV를 함께 적용한다고 하여도 반드시 분산이 감소되리라는 보장은 없다는 것을 알 수가 있다.

이제는 성공경로법과 확률경로법을 비교하여 보자. 우선 평균의 경우, <표 4>의 확률경로법과 <표 7>의 성공경로법과는 일정한 차이가 있음을 보여주고 있다. 이는 신뢰도를 추정하는 로직간의 상이점에서 기인한 것으로 보이며, 부품 개수가 적어 이론적인 평균치를 구할 수 있는 사상공간법 경우의 예를 보면 성공경로법이 좀더 이론 치에 가까운 불편(unbiased) 추정치임을 알 수 있었다.

분산을 살펴보면, Crude와 AV의 경우에는 성공경로법이 확률경로법보다 대체로 분산값 자체가 작았으나 그 외 나머지 기법의 경우에는 분산이 커졌음을 알 수 있다. 총 연산시간을 비교하여 보면, 확률경로법은 320.17초가 소요되었고 성공경로법은
566.99초가 소요되어 확률경로법이 연산시간 면에서는 빠름을 알 수가 있다.

6. 결론

대규모 컴퓨터의 네트워크, 네트워크 라우팅 장비의 설계 문제, 또는 군사 및 과학의 여러 분야에서 사용되고 있는 복잡한 구조의 네트워크 기기나 장비들은 높은 신뢰도를 요구한다. 이러한 경우에는 제품의 설계 시나 사용 시에 네트워크의 경로집합과 신뢰도를 효율적으로 계산할 수 있어야 한다. 현재까지 네트워크의 경로 탐색과 관련하여 무 방향 네트워크의 경로집합이나 절단집합을 구하는 것이 다수 연구가 되어 왔으며, 이러한 경로집합을 구하는 하나의 방법으로 연결 행렬의 멱 행렬이나 소 행렬식을 이용하는 방법이 개발되었다. 그러나 실제 컴퓨터 상에서 구현하기에는 메모리의 문제와 경로와 무관한 폐쇄경로를 제거시켜야 하는 등의 불편한 점이 있다.

본 연구에서는 이를 해결하기 위한 경로탐색 알고리듬을 빠른 시간 내에 구해 주는 재귀적 알고리듬을 개발하였다. 또한 부품의 신뢰도를 상수가 아닌 분포함수를 따르는 확률변수로 보아야 하는 경우가 현실적이다. 그러나 아직까지도 이러한 부품 신뢰도의 분포함수가 주어지는 경우에 대한 연구는 이론적으로 너무나 복잡하고 불가능하여 거의 이루어지지 않고 있는 실정이다. 이러한 네트워크 시스템 신뢰도의 계산을 시뮬레이션으로 해결하고자 하였으며, 시뮬레이션의 퍼포먼스를 올리기 위하여 몇 가지의 분산감소 기법을 적용하였다. 적용한 결과 상당한 정도의 분산을 감소시킬 수 있었다. 분산감소 기법 중에서 Antithetic Variate 기법은 로직 구현이 쉽고 간단하여 추가 노력이 크게 요구되지 않는 장점이 있으며, Control Variate와 단계별 회귀분석의 응용은 회귀분석을 추가로 실시해야 하는 노력과 시간이 요구되나 그 효과는 상당히 큰 것으로 판명되었다. 특히 Antithetic Variate와 다른 기법들을 병행하여 적용하면 분산의 감소가 더욱 효과적임을 알았다.

본 논문의 결과를 응용하여 일단 네트워크의 경로 또는 절단집합을 구하게 되면 최단경로문제, 대규모 네트워크의 신뢰도 평가, 인터넷 라우팅 설계, 네트워크 대기행렬 분석 등 여러 분야에 본 연구의 결과를 활용할 수 있을 것으로 기대된다. 이 분야는 본 논문의 향후 연구분야로 남겨 놓는다.

참고문헌

[1] AboElFotoh, H. A. and Colbourn, C. J., "Efficient Algorithms for Computing The Reliability of Permutation and Interval Graphs", Networks, Vol. 20(1990), pp. 883-898,

[2] Aggarawal, K. K., Reliability Engineering, Kluwer Academic Publishers, 1983.

[3] Banks, Jerry and Carson, John S. II, and Nelson, Bary L., Discrete-Event System Simulation, Prentice Hall International, 1996, pp.321-350.

[4] Brateley, P. and Fox, B. L. and Schlage, L. E., A guide to Simulation, Spring -Verlag, New York, 1983.

[5] Deo, N., Graph Theory with Applications to Engineering and Computer Science, Prentice Hall Inc. 1974.

[6] Elperin, T. and Gertsbakh, I. and Lomonosov. M., "Estimation of Network Reliability Using Graph Evolution Models", IEEE Tranactions On Reliability, Vol. 40, No. 5(1991), pp. 572-581.

[7] Elsayed A. Elsayed, Reliability Engineering, Addison Wesley Longman, Inc., 1996, pp.69-111.

[8] Hartless, G. and Leemis, L., "Computational Algebra Applications in Reliability Theory", IEEE Transactions on Reliability, Vol. 45, No.3(1996), pp. 393 -399.

[9] Law, Averill M. and Kelton, W. David, Simulation Modeling & Analysis, McGraw Hill, 1991.

[10] Locks, Mitchell O., "recent Developments in Computing of System Reliability", IEEE Transactions On Reliability, vol. 34(1985), pp. 425-435.

[11] Rai, S. and Aggarawal, K. K.,"An Efficient Methods for Reliability Evaluation of a General Networks", IEEE Transactions on Reliability, Vol. R-27(1978), pp. 206-211.

[12] Soh, S. and Rai, S., "Experimental Results, On Processing of Path/Cut Terms in Sum of Disjoint Products Technique", IEEE Transactions On Reliability, Vol. 42(1993), No. 1, pp.24-33.

저자소개

김원경
1977년 서울대학교 산업공학과 학사
1979년 서울대학교 대학원 산업공학과 석사
1989년 미국 Ohio 주립대학교에서 M.S.
1993년 미국 University of Houston에서 Ph.D.
현재 경남대학교 벤처창업학부 교수
관심분야: 시뮬레이션, 신뢰도 공학, 응용통계

하경재
1980년 성균관대학교 전기공학과 학사
1982년 성균관대학교 대학원 전기공학과 석사
1989년 성균관대학교 대학원 전기공학과 박사
1997년 미국 Wayne 주립대학 visiting scholar
현재 경남대학교 정보통신공학부 교수
관심분야: 지능시스템, 소프트컴퓨팅, 신뢰도 공학