다중상태 유닛들의 망 신뢰도 근사 계산을 위한 알고리즘

오 대 호* · 염 준 근**

* 한림정보산업대학 컴퓨터응용과, ** 동국대학교 통계학과

An Algorithm For Approximating The Reliability of Network with MultiState Units

Dae Ho Oh* · Joon Keun Yum**

* Department of Computer Application, Hallym College of Information and Industry,

** Department of Statistics, Dongguk University

Keywords: Most probable states, Multistate unit, Network reliability

Abstract

A practical algorithm of generating most probable states in decreasing order of probability, given the probability of each unit's state, is suggested for approximating reliability(performability) evaluation of a network with multistate(multimode) units. Method of approximating network reliability for a given measure with most probable states is illustrated with a numerical example. The proposed method in this paper is compared with the previous method regarding memory requirement. Our method has some advantages for computation and achieves improvement with regard to memory requirement for a certain condition judging from the computation experiment.

1. 서 론

통신망, 다중 프로세서의 상호연결망, 교통망 혹은 전력 망 등과 같은 복잡한 시스템들은 주로 망(network) 구조를 갖는다. 망 신뢰도는 기존의 혹은 제안된 망 시스템의 유효성(effectiveness)을 평가하는데 있어 중요한 요소로 간주된다. 특정 망 신뢰성(혹은 성능)측도가 선택되고 각 상태 확률이 주어지면 망 신뢰도는 각 상태에 대한 신뢰성의 기대값으로 계산된다. 그러나 정확한 망 신뢰도의 계산은 일반적으로 매우 복잡하며, 주로 수학적으로 다루기 힘든 경우가 대부분이다. 특히, 망 구성 장치들의 수가 증가할수록 망의 상태공간(state space)의 크기가 지수적(exponentially)으로 증가하기 때문에 모든 망의 상태를 나열함으로써 망 신뢰도를 평가하는 것은 망의 구성 장치들이 적은 것

에 대해서 유용하며, 구성 장치들이 많은 망에 대해서는 많은 계산시간이 요구된다. 따라서 정확한 망 신뢰도를 계산하는 것보다 그 근사값을 도출하는 방법들이 요구된다. 망 구성요소들의 신뢰도는 일반적으로 매우 고 신뢰도를 갖으며, 따라서 적은 수의 망 상태들만으로도 전체 상태 확률 중에서 그 비율은 상당히 크다. 망의 작동 환경에서 발생 확률이 낮은 상태들을 계산에서 제외하고, 그 중에서 발생 가능성이 큰 혹은 신뢰도 값의 기여 정도가 큰 망 상태 들 만을 통해 그 상·하한계나 근사값등을 계산하는 것은 매우 타당한 근사 해법이 될 수 있다. 통상 망 유닛들이 고 신뢰도를 갖을 때 상태 공간의 10% 이내의 상태들만으로도 95%이상의 상태공간 포함정도(coverage)를 만족한다고 알려져 있다[1]. 이러한 관점에서, Li와 Silvester(1984)는 망 유닛들이 이진 상태일 때 망 신뢰도의 근사 값을 계산하기 위하여 발생 가능성이 큰 상태(most probable state, 이하 MPS)의 개념을 도입하였으며 이들 상태들을 생성하는 알고리즘을 제시하고, 특정 망 신뢰도 측도에 대해서 MPS에 의해 상·하한계를 계산 함으로써 망 신뢰도(성능 신뢰도)의 근사값으로서 유용한 방법임을 보였다. 이 방법은 Lam과 Li(1986)에 의해서 실질적인 방법으로 개선되었으며,Shier(1988)에 의하여 더욱 효율적인 방법으로 개선되었다. 이진 상태의 유닛들에 관한 연구들이 많지만, 망 유닛들은 고장이거나 작동인 이진 상태와 같이 작동에서 완전 고장 상태로 되기보다는 작동 상태에서 그 성능이 여러 단계로 점차 감소되는 경우 즉, 유닛들의 품질 혹은 효율성 면에서 여러 다른 수준(level)으로 작동하는 경우가 많다. Chiou 와 Li(1986)는 이러한 다중 상태의 망 구성요소로 구성된 망의 신뢰도 평가에 있어서, Li 와 Silvester(1984)의 이진 상태의 알고리즘을 수정하여, MPS의 생성 알고리즘을 처음 제안하고 망 성능 측도로는 메시지 지연에 대하여 망 신뢰도에 대한 상·하한계를 예시하였다. 이 방법은 MPS의 개수 m이 정해지고 생성된 후, m을 다시 수정하기가 어렵고 많은 정열 과정을 필요로 하는 것으로 실질적이지 못한 단점을 갖고 있다. Gaebler와 Chen(1987)은 Chiou 와 Li의 이러한 단점을 보완하기 위해 실질적인 알고리즘을 제안하였다. 이 방법에서 최근에 얻어진 MPS로부터 후보가 될 가능성이 높은 상태들만을 생성하고 자료구조인 힙(heap)에 저장하여 힙으로부터 가장 확률이 큰 후보를 얻는 방법을 택하고 있다. 본 논문에서는 다중상태의 유닛들로 구성된 망 에 대해서 MPS를 그 상태확률의 크기에 따라 내림차순으로 생성하는 방법을 제안하고 수치 예제를 통해 알고리즘의 자원 효율성 측면 중 메모리 소요량에 관하여 비교하며 장단점을 예시 하고자 한다. 2절에서는 일반적인 다중 상태 망 신뢰도 모형과 기존의 방법을 언급하고, 3절에서는 본 논문에서 제안하는 알고리즘을 기술하며, 4절에서는 수치 예제를 통해 그 효율성을 예시하기로 한다.

기 호

Ω	망의 상태 공간
M_i	유닛 i 의 상태 수
X_i	임의의 망의 상태; $(x_1, \cdots, x_n)$

단, $x_i=0,1,\cdots,M_{i-1}$

$P(X_i)$ 상태 X_i 의 확률

$R(X_i)$ 망의 상태가 X_i 일 때의 신뢰도 혹은, 성능의 측도 값

$\overline{R}$ 망 신뢰도; $\sum_{X_i\in\Omega} P(X_i)R(X_i)$

S_i X_i 들의 상태 확률의 내림차순으로 순서화된 상태

p_{ij} $P\{x_i=j\}$, $j=0,\cdots,M_{i-1}$, $i=1,\cdots,N$

$\overline{R}_U$ 망 신뢰도의 상 한계

$\overline{R}_L$ 망 신뢰도의 하 한계

α $\max_i R(X_i)$

β $\min_i R(X_i)$

2 다중 상태 모형

가 정

1. 망은 1부터 N으로 번호가 부여된 N 개의 유닛들로 구성된다
2. 유닛은 망의 노드 혹은 링크이다.
3. 유닛 i 는 M_i 상태중 하나의 상태가 될 수 있다; $0,1,\cdots,M_{i-1}$
4. 유닛 i 는 상태 j 에서 주어진 확률 $p_{ij}>0$, $i=1,\cdots,N$, $j=0,\cdots,M_{i-1}$, $p_{i0}\geq p_{i1}\geq\cdots\geq p_{iM_{i-1}}$, $\sum_{j=1}^{M_i} p_{ij}=1$, 로 작동하며 서로 독립이다.

망 시스템들은 그 유닛들이 이진 상태로서, 작동이거나 고장인 상태가 되기보다는 점차적으로 그 성능의 수준이 여러 단계로 감소하는 형태로 작동하는 경우가 많다. 또한, 품질이나 효율성의 단계가 아니라 여러 작동 유형, 예를 들어 통신 매체가 음성이거나 혹은 문자의 전송일수 있는 것과 같이, 여러 작동 모드를 갖는 경우도 있다. 이러한 다중 상태 혹은 작동 모드를 갖는 유닛들로 구성된 망의 신뢰도를 평가함에 있어서 망의 상태공간의 크기는 유닛들의 상태 수와 유닛의 수에 따라서 기하급수적으로 증가한다. 따라서 망의 상태를 통해 그 신뢰도를 평가하고자 할 때 어떤 종료 기준(stopping rule)을 만족할 만한 정도의 확률이 큰 상태들만을 고려하여 망 신뢰도의 근사 값이나 그 상·하 한계를 계산하고자 하는 것이다. 우리는 망 구성요소들의 상태별 작동 확률이 주어지고 망에 대한 특정 신뢰도 혹은 성능(성능신뢰도, performability)에 대한 측도가 주어졌을 때 상태 확률의 내림차순(decreasing order)으로 망 상태 $S_1,S_2,\cdots,$ 을 특정한 종료기준을 만족할 때 까지 하나씩 생성하는 것이 목적이다. 임의의 노드에서 다른 하 나(또는 그 이상)의 노드로의 연결이 가능한 경로가 있는가와 같은 연결성의 측도는 다중 상태 보다는 이진상태에 더욱 합당한 측도 이므로 여기서는 최대 유량(maximum flow)을 망 측도로 하여 예시하기로 한다. M_i 다중 상태를 갖는 N 유닛들의 각 상태, $j\in\{0,1,\cdots,M_{i-1}\}$ 은 그 확률 크기순서로 순서화 되어있다 하자. 망의 상태는 주로 $(0,0,\cdots,0)$, $(0,2,\cdots,0)$등과 같이 표현되나, 우리는 각 망의 상태를 유닛들의 가장 확률이 큰 상태를 제외한 나머지 상태들만을 다시 레이블링(labeling)한 정수의

집합 S_i 로 표현하기로 한다. 따라서 가장 확률이 큰 상태는 $S_1=\phi$, 그 다음으로 확률이 큰 상태, $S_2=\{1\}$ 등으로 표현하며 S_1에 대응되는 상태 벡터는 $(0,0,\cdots,0)$ 이다. Chiou 와 Li(1986)는 각 유닛의 상태를 그 확률의 크기 순서로, 다음과 같이 순서화 하였다; $p_{ij} > p_{ik}$, $j < k$, $i=1,\cdots,N$, $j=1,\cdots,M_{i-1}$. 따라서 상태 0의 확률이 유닛 내 에서 가장 크며 그 다음이 상태 1 등의 순서이다. 각 유닛의 상태 0을 제외한 모든 유닛들을 확률 p_{ij} 의 크기순서로 총 $Q=\sum_{i=1}^{N}(M_i-1)$개로 순서화 하고 이를 $q_i\in\{1,2,\cdots,Q\}$라 할 때 각 q_i는 유닛 i의 j 상태, (i,j)에 대응된다. 종료 기준에 따라 얻고자 하는 MPS의 개수 m 이 정해지면 가장 확률이 큰 상태의 집합, $A=\{(0,\cdots,0)\}$로부터 $q_1=1$ 에 대응되는 유닛을 해당 상태로 변환시켜 집합 B를 얻고 예로써, q_1은 (2,1)에 대응된다면 $B=\{(0,1,\cdots,0)\}$을 얻고, A와 병합을 한 후 정열(sort)하여 $A=\{(0,\cdots,0),(0,1,\cdots,0)\}$를 생성하고 m 개의 MPS를 얻을 때까지 다시 $q_2,\cdots,q_Q$ 와 이를 반복하며 상태벡터를 확률의 크기에 따라 매 단계마다 정열하여 상태의 집합 A를 얻어 나가되 이중 정열된 m 개의 상태를 취하는 것이다. 따라서 이 알고리즘은 MPS의 개수 m이 주어지고 연산이 수행된 후 MPS의 수를 수정하고자 한다면 알고리즘의 모든 단계의 연산을 처음부터 다시 시작해야 하며, 또한 매 단계마다 정열하고 그 상태벡터의 집합을 저장해야 하므로 메모리의 소요량이 증가하며 실행시간 역시 많이 소요되는 단점을 갖고 있다. Gabler와 Chen(1987)은 Chiou와 Li와는 달리 알고리즘 반복과정에서 한번에 하나의 MPS만을 생성하고 생성된 MPS로부터 그 다음으로 MPS가 될 수 있는 후보들만을 생성하여 힙(heap) 구조에 저장하는 방법을 취한다. 또한 힙의 기본연산(operation)중에서 추출(extraction)연산으로부터 힙에 저장된 그 다음 순서의 MPS를 쉽게 선택하여 매 단계마다 하나씩 얻는 알고리즘을 제안하였다. 힙은 저장된 자료 중 최대 혹은 최소의 값을 선택하는(selection) 문제에서 효율적인 성능을 갖고 있다.[6] 여기서는 이진 힙(binary heap)을 이용한다. 이 방법은 각 유닛들의 상태를 유닛 내의 확률크기에 따라 순서화 한 것은 Chiou와 Li방법과 유사하나 모든 p_{ij}에 대해 레이블링 한 것이 아니라, 각 유닛별 상태 1과 상태 0의 확률비율, p_{i1}/p_{i0} 에 의해서 유닛 번호를 재 순서화(reordering)하였다. 즉, 유닛별로 상태 1중에서 가장 확률이 큰 유닛 번호를 1, 그 다음을 2 등이다. 따라서 망의 상태는 $S_1=(0,0,\cdots,0)$, $S_2=(1,0,\cdots,0)$등으로 표현된다. $S(i)$를 S의 i 번째 유닛의 상태라 하자. 이 방법은 크게 3 단계로 이루어져 있다. 첫째, MPS의 후보 상태들을 생성하고, 둘째, 전 단계까지 얻어진 후보 상태들과 함께 힙에서 정열되고, 셋째, 힙으로부터 최대의 확률 값을 갖는 후보를 추출하여 현 단계의 MPS를 취하는 것이다. 힙의 각 노드(node)들은 망의 상태 표현과 그 확률로 구성된다. 후보를 생성하는 방법은, 현 단계에서 얻은 MPS로부터 $h=\max(i|S(i)\neq 0)$ 와 $j=S(h)$를 얻고 만일

$j<M_h-1$ 이면 h 번째 유닛을 $j+1$로 대체하고, $h<N$ 이면 $h+1$ 번째 유닛을 1로 하며, $h<N$ 이고 $j=1$ 이면 h번째는 0, $h+1$번째는 1로 한다. 이러한 생성 기준을 통해서 현 단계의 MPS와 확률의 크기 면에서 가장 근접하리라 보이는 후보들을 최대 3가지를 생성하는 탐색적 규칙을 이용한다. 이때, 3단계 과정 중 첫 번째 단계에서 후보 상태들을 어떻게 효율적으로 생성할 것인가의 문제가 알고리즘의 효율성에 많은 영향을 미친다. 본 논문에서는 Gabler와 Chen과 같이 둘째, 셋째단계는 같은 방법을 취하나, 후보들을 생성하는 방법에서 다른 규칙을 취하며 상태 벡터의 형태가 아니라 Chiou와 Li와 유사한 레이블링된 집합의 형태로 망의 상태를 표현하여 MPS를 생성하는 알고리즘을 제안하고 Gabler와 Chen의 방법과 수치예제를 통해 비교, 검토하고자 한다. 알고리즘의 시간 복잡도(time complexity)는 힙의 크기에 의존적이며 그 외는 실행 효율성에 크게 영향을 미치지 않으며, 최악 분석의 표현에서는 같기 때문에 여기서는 힙의 크기 측면에서만 검토 하고자 한다. m 개의 MPS를 얻는다면, Li와 Silvester[1]가 제안한 망 신뢰도의 상 한계와 하 한계를 다음과 같이 구할 수 있다.

$$\overline{R}_U=\sum_{k=1}^{m}R(S_k)P(S_k)+(1-\sum_{k=1}^{m}P(S_k))\alpha \quad (2.1)$$

$$\overline{R}_L=\sum_{k=1}^{m}R(S_k)P(S_k)+(1-\sum_{k=1}^{m}P(S_k))\beta \quad (2.2)$$

3. 알고리즘

다중 상태 모형에서 확률의 내림차순으로 처음 m 개의 MPS를 얻는 알고리즘의 기술을 위해 아래의 용어를 이용하기로 하며 2절의 가정을 적용하기로 한다.

기 호

H 힙

Q $\sum_{i=1}^{N}(M_i-1)$; 각 유닛 i 의 상태 0을 제외한 모든 상태의 수

r_h p_{ij}/p_{i0}, $i=1,\cdots,N$, $j=1,\cdots,M_{i-1}$, $h=1,\cdots,Q$

e_i $r_i\geq r_j$ 에 대해 $e_i<e_j$ 을 만족하는 순열, $e_i=i$, $i\in\{1,\cdots,Q\}$

(i,j) 유닛 i의 상태 j

S_i 집합 $\{e_1,\cdots,e_q\}$

X_i S_i 에 대응되는 망의 원(original) 상태 벡터

$S_i-\{e_j\}$ S_i 로 부터 e_j를 제외한 집합

$S_i+\{e_j\}$ S_i 와 e_j을 부가한 집합

$e_L(S_i)$ 집합 S_i의 마지막 원소

$n\{e_i\}$ e_{i+1}

SR_i 제안된 방법에서 반복 i 에서 H의 근노드의 상태 집합

모든 N 유닛들이 $M_i, i=1,\cdots,N$ 상태를 갖으며 각각 $0,1,\cdots,M_{i-1}$일 때 각 유닛의 상태 0을 제외한 모든 상태들을 p_{ij}/p_{i0}, $i=1,\cdots,N$, $j=1,\cdots,M_{i-1}$ 의 크기 순으로

레이블링하고 이를 $e_1, \cdots, e_Q$, 라 하자. 즉, $e_i < e_j$, $r_i \geq r_j$, $i, j = 1, \cdots, Q$이며 e_i 는 유닛 i의 상태 j, (i, j)에 대응된 것이다. 그리고 임의의 망 상태를 e_i 의 집합, S_i 로 표현하기로 한다. 예로서, $e_1 = 1$은 (2,1)과 대응되고, $e_2 = 2$는 (1,2)와 대응된다면, $S_i = \{1,2\}$은 망의 상태 $(2,1,0,\cdots,0)$을 의미한다. 따라서 확률이 큰 망의 상태들은 $S_1 = \phi$, $S_2 = \{1\}$등의 순서로 표현이 된다. 이때 상태확률은 다음과 같이 표현된다.

$$P(S_h) = \left(\prod_{(i,j)\notin S_h} p_{ij}\right)\left(\prod_{(i,j)\in S_h} p_{ij}\right)$$

$$= \left(\prod_{i=1}^{N} p_{i0}\right)\left(\prod_{h\in S_h} r_h\right) \qquad (3.1)$$

이러한 레이블링 방법으로부터, Gabler와 Chen과 같이 모든 후보들을 힙에 저장할 때 상태벡터 형태로 하여 저장하지 않으므로 메모리 양이 적게 요구되며 또한, 후보들을 생성할 때 상태벡터의 리스트(list)를 검색하지 않는 계산적 이득을 얻을 수 있다. 제안된 알고리즘은 다음 <그림 3.1> 과 같다. 현 단계의 MPS, S_i의 마지막 원소가 Q 가 아니면 매 단계마다 최대 2가지 CS_1, CS_2 의 후보 상태를 도출하게 된다. INSERT함수는 생성된 후보 상태 CS_1와 그 확률을 계산하여 힙에 추가하는 함수이며, EXTRACT함수는 힙의 근 노드(root node)를 추출하여 가장 확률이 큰 후보 상태를 힙으로부터 얻는 함수이다. 만일 현 반복 단계에서 $\{1,2\}$를 얻는다면 알고리즘으로부터 $\{1,3\}$과 $\{1,2,3\}$의 후보 상태들을 얻는다.

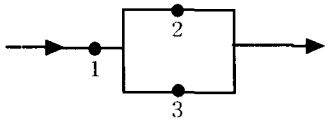
```
Algorithm
Initialize  i=1, S1={e1}, H=φ,
            flag ← 0
Repeat
begin
 If  eL(Si) ≠ Q  Then
 begin
   NL = n{eL(Si)}
   Repeat
     CS1 = Si − {eL(Si)} + NL
     If a element of CS1 has not common
        unit Then
     begin
       flag ← 1; break;
     end
     NL = n{NL}
   Until  NL > Q
   If flag = 1 then
   begin
     compute P(CS1);
     INSERT( CS1, P(CS1), H);
     flag ← 0
   end
   Repeat
     CS2 = Si + NL
     If  a element of  CS2  has not common
        unit Then
     begin
        flag ← 1; exit;
     end
     NL = n{NL}
   Until  NL > Q
   If flag = 1  Then
   begin
     compute P(CS2);
     INSERT( CS2, P(CS2), H);
   end
 end
   Si+1 ← EXTRACT(H);
   i←i+1
end
Until  enough MPS have been chosen
 /* end algorithm */
```

<그림 3.1> MPS 생성 알고리즘

이때 레이블 1 은 유닛 1의 2 상태, (1,2)에 대응되고, 레이블 2는 (2,1), 3은 (3,1)에 각각 대응된다면, 상태 $(2,1,0,\cdots,0)$으로부터, $(2,0,1,\cdots,0)$과 $(2,1,1,\cdots,0)$을 후보로서 각각 얻고 이들의 확률을 식 (3.1)로 부터 각각 계산하고 확률 값과 상태의 정보를 힙에 저장하게 된다. 그리고 현 단계까지 힙에 저장된 상태들 중 확률이 가장 큰 것을 힙으로부터 추출하여 다음 단계의 MPS를 얻게되며 종료 기준을 만족할 때까지 이 과정을 반복함으로써 내림차순으로 망의 상태를 차례로 생성할 수 있다

4. 수치예제

본 절에서는 다음 <그림 4.1>의 예제 망에 관해, 성능 측도로는 망의 최대 유량으로 취하고 망 유닛들을 링크(link)만을 고려하기로 하며 제안된 알고리즘을 적용하여 MPS의 생성방법과 망 신뢰도의 상, 하 한계의 계산 결과를 예시하고 Gabler와 Chen의 방법과 비교, 검토하고자 한다. 이때 유닛별 그 용량(capacity)은 여러 수준으로 작동하며 이때 망의 각 유닛별 상태와 용량, 그때의 확률은 <표 4.1>과 같이 주어져 있다.



<그림 4.1> 예제 망

이때 총 상태 수는, $\prod_{i=1}^{N} M_i = 3 \cdot 3 \cdot 3 = 27$ 이며, 이 예제에서의 유닛들은 <표 4.2>와 같이 확률 비에 의해서 레이블링되며 그 결과는 <표 4.3>과 같이 원 유닛 번호와 그 상태의 관계를 의미한다. 이때 알고리즘의

<표 4.1> 유닛번호, 상태, 유닛 용량, 상태확률

유닛 i	상태	용량 $C_{i,j}$	확률 $p_{i,j}$
1	0	4	0.9
	1	3	0.05
	2	0	0.05
2	0	2	0.8
	1	1	0.15
	2	0	0.05
3	0	3	0.7
	1	2	0.2
	2	0	0.1

<표 4.2> 상태, 확률, 확률비, 순위 (M_i =3, i=1,2,3)

유닛 i	상태	확률	확률비	순위
1	0	0.9	-	
	1	0.05	0.0555	6
	2	0.05	0.0555	5
2	0	0.8	-	
	1	0.15	0.1875	2
	2	0.05	0.0625	4
3	0	0.7	-	
	1	0.2	0.2857	1
	2	0.1	0.14286	3

<표 4.3> 순위, e_i, (유닛 i, 상태 j)

순위	e_i	(i,j)
1	1	(3,1)
2	2	(2,1)
3	3	(3,2)
4	4	(2,2)
5	5	(1,2)
6	6	(1,1)

일부 연산 결과인 반복 5회까지 얻어진 MPS와 힙으로부터 MPS를 추출 후, 힙에 남아있는 노드들의 결과는 <표4.4>와 같다. 반복 열의 0회째는 가장 확률이 큰 상태와 그 다음으로 큰 상태를 초기 값으로 취한 것으로 <표 4.3>으로 부터 ϕ는 $(0,0,0)$, {1}은 $(0,0,1)$을 의미한다. Gabler와 Chen은 상태를 벡터형태로 표기하나 이 예제에서는 비교를 위해서 제안된 알고리즘의 형태로 바꾸어 표현하였다. <표 4.4>의 결과로부터 제안된 방법이 힙에 남아있는 노드 수가 다소

<표 4.4> 반복, 현 MPS, 잔여노드

반복	현재 MPS	MPS 추출후 힙의 잔여 노드	
		제안된 방법	Gabler와 Chen
0	ϕ {1}	-	-
1	{2}	{1,2}	{3},{1,2}
2	{3}	{1,3}, {2,3}	{1,2},{4},{5}, {2,5}
3	{4}	{1,2},{2,3}, {3,4}	{1,2},{2,3},{2,5}, {5}
4	{5}	{1,2},{2,3}, {3,4},{4,5}	{1,2},{2,3},{2,5}, {4,5}
5	{6}	{1,2},{2,3}, {3,4},{4,5}	{1,2},{2,3},{2,5}, {4,5}

<표 4.5> 평균잔여노드, 힙의 크기

$\sum_i P(S_i)$	MPS 개수	제안된 방법		Gabler et al.	
		잔여 노드	최대 크기	잔여 노드	최대 크기
0.95	10	3.25	5	4.125	7
0.99	17	4.13	6	4.93	7
0.999	24	3.68	6	4.22	7
1	27	3.28	6	3.76	7

적은 것으로 나타났다. <표 4.5>는 각 상태 공간의 포함정도(coverage)인 $\sum_i P(S_i)$ 값을 기준으로 하여 각각 그 시점까지의 반복 횟수에 대한 평균 잔여 노드 수와 그 반복시까지의 힙의 최대 크기를 비교한 결과이다. 이때 각각의 상태공간 포함정도 별로 다소 우수한 것을 알 수 있다. <표 4.5>의 노드의 크기와 평균 잔여노드수 는 상태의 표현방법에 따른 실제 메모리 크기로 계량화 한 것이 아니라 각 방법별로 노드의 수만을 비교한 것이며, Gabler와 Chen방법은 상태 표현에 있어 X_i 의 형태로 노드를 구성하며, 제안된 방법은 S_i의 형태로 구성되므로 실제로는 기존의 방법이 더 많은 메모리 자원을 소요하게 된다. 예제에는 나타나 있지는 않으나 망 유닛 개수는 동일하나 각 유닛별 상태수, M_i 가 증가 될 때, Gabler와 Chen의 방법은 적은 수의 MPS를 생성할 때 비교적 힙의 크기가 증가하는 경향이 있으며, 제안된 방법은 많은 수의 MPS를 얻을 때 힙의 크기가 증가하는 특징이 있다. 그러한 것은 Gabler와 Chen의 방법은 각 유닛중 가장 확률이 큰 상태(상태 0)와 그 다음으로 큰 상태(상태 1)만을 이용하여 레이블링을 하고

<표4.6> 망의 MPS와 망 성능의 상, 하 한계

반복	순위	S_i	X_i	$P(S_i)$	$R(S_i)$	$\overline{R}_L$	$\overline{R}_U$
0	1	ϕ	(0,0,0)	0.504	4	2.016	4
0	2	{1}	(0,0,1)	0.144	4	2.592	4
1	3	{2}	(0,1,0)	0.0945	4	2.97	4
2	4	{3}	(0,0,0)	0.072	2	3.114	3.865
3	5	{4}	(0,0,2)	0.0315	3	3.2085	3.8245
4	6	{5}	(1,0,0)	0.028	3	3.2925	3.7965
5	7	{6}	(2,0,0)	0.028	0	3.2925	3.6845
6	8	{1,2}	(0,1,1)	0.027	3	3.3735	3.6575
7	9	{2,3}	(0,1,2)	0.0135	1	3.387	3.617
8	10	{1,4}	(0,2,1)	0.009	2	3.405	3.599
9	11	{1,5}	(1,0,1)	0.008	3	3.429	3.591
10	12	{1,6}	(2,0,1)	0.008	0	3.429	3.599
11	13	{2,5}	(1,1,0)	0.00525	3	3.44475	3.55375
12	14	{2,6}	(2,1,0)	0.00525	0	3.44475	3.53275
13	15	{3,4}	(0,2,2)	0.0045	0	3.44475	3.51475
14	16	{3,5}	(1,0,2)	0.004	2	3.45275	3.50675
15	17	{3,6}	(2,0,2)	0.004	0	3.45275	3.49075
16	18	{4,5}	(1,2,0)	0.00175	3	3.458	3.489
17	19	{4,6}	(2,2,0)	0.00175	0	3.458	3.482
18	20	{1,2,5}	(2,1,1)	0.0015	3	3.4625	3.4805
19	21	{1,2,6}	(1,1,1)	0.0015	0	3.4625	3.4745
20	22	{2,3,5}	(2,1,2)	0.00075	1	3.46325	3.47225
21	23	{2,3,6}	(1,1,2)	0.00075	0	3.46325	3.46925
22	24	{1,4,5}	(2,2,1)	0.0005	2	3.46425	3.46825
23	25	{1,4,6}	(1,2,1)	0.0005	0	3.46425	3.46625
24	26	{3,4,5}	(2,2,2)	0.00025	0	3.46425	3.46525
25	27	{3,4,6}	(1,2,2)	0.00025	0	3.46425	3.46425

후보 상태들을 생성하기 때문에 적은 수의 MPS를 얻을 때 더 많은 수의 후보를 생성하게 되는 것이다. 따라서 제안한 방법은 유닛들의 각 상태가 클 때 강점이 있다. <표 4.6>은 <그림 4.1>의 예제 망에 대해 각 단계별로 도출된 MPS와 그에 대응되는 망의 원 상태 벡터 및 단계별 망 신뢰도(성능)에 대한 식(2.1), (2.2)로부터의 상 · 하 한계와 전체 망 상태를 모두 생성한 후 정확한 망 신뢰도(성능)는, $\overline{R}$=3.46425인 계산 결과를 보인 것이다.

5. 결 론

본 논문에서는 다중상태(모드)를 갖는 유닛들로 구성된 망 시스템에 관해, 임의 망 측도와 유닛들의 상태확률이 주어질 때 망 신뢰도의 근사 값이나 그 상 · 하 한계를 계산하기 위해서 가장 확률이 큰 상태들을 그

확률의 내림차순으로 생성하는 알고리즘을 제안하였다. 알고리즘 효율성의 비교에서는 그 수행시간은 힙의 크기에 상당히 의존적이며 그 외는 수행시간에 많은 영향을 주지 않기 때문에, 예제를 통해서 제안된 방법과 Gabler와 Chen의 방법을, 메모리 소요량 측면에서 힙의 크기와 상태공간 포함정도에 따른 평균 잔여노드 수를 비교하여 알고리즘의 효율성을 검토하였다. 수치 예제를 통해 Gabler와 Chen의 방법보다 다소 좋은 결과를 보였으며 Gabler와 Chen의 방법은 상태 벡터형태에서 후보를 생성함으로써 상태 정보를 리스트로 표현하고 이를 검색하는 계산상의 단점을 갖고 있기 때문에 수행시간 측면에서도 다소 단축되리라 본다. 그리고 계산 실험을 통한 바로는 유닛의 수 가 크고 MPS를 많이 생성하는 경우는 Gabler와 Chen의 방법보다 메모리 소요량이 크게 향상되지는 않지만 각 유닛별 상태의 수가 크며 MPS를 많이 생성하지 않을 때 장점을 갖고 있다. 현 MPS로부터 후보상태를 생성하는 방법에서, 예로써, 힙의 후보 상태 {2}, {1,2}로부터 현 MPS, {2}로 부터, {3}과 {2,3}을 생성하게 되는데 상태집합 {2,3}은 {1,2}보다 확률계산 없이도 대수적으로 그 값이 작은 상태집합이며 이를 미리 생성하게 되어 궁극적으로 메모리 소요량을 증가시키는 문제점을 갖고 있다. 향후의 과제로 이러한 문제를 보완하는 것이 남아 있다.

참고문헌

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

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

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

[4] Shen-Neng Chiou and V.O.K. Li(1986), "Reliability Analysis of a Communication Network with Multimode Components",*IEEE Journal on Selected Areas in Communications,* Vol. SAC-4, No. 7, pp. 1156-1161.

[5] R.F. Gabler and R.J.Chen(1987), "An Efficient Algorithm for Enumerating States of a System with Multimode Unreliable Components", *U.S. Sprint Communications, Overland Park, Kansas, Technical Report.*

[6] E. Horowitz and S. Sahni(1976), "Fundamentals of a Data Structure", Pitman.