흐름량을 고려한 네트워크 생존도 계산방법에 관한 연구*

명영수** · 김현준***

An Algorithm for Calculating Flow-based Network Survivability*

Young-Soo Myung** · Hyun-joon Kim***

Abstract

Survivability of a network is one of the most important issues in designing present-day communication networks. The *k*-edge survivability of a given network is defined as the percentage of total traffic surviving the worst case failure of *k* edges. Although several researches calculated *k*-edge survivability on small networks by enumeration, no prior research has considered how to calculate *k*-edge survivability on large networks. In this paper, we develop an efficient procedure to obtain lower and upper bounds on the *k*-edge survivability of a network.

Keywords : Network survivability, Combinatorial optimization, Communication netwerks

1. 서 론

광섬유의 도입과 더불어 비롯된 광통신기술의 발달은 정보통신기술의 혁신적인 발전으로 이어졌다. 정보기술의 디지털화와 더불어 인터넷기반 서비스가 폭발적으로 확산되고 고속 교환 및 대용량 전송기술을 이용하는 다양한 정보서비스의 보급이 이루어지고 있다. 한편으로는 통신망의 구조도 고속의 교환 및 전송장비의 도입으로 인하여 과거와 같은 복잡한 그물구조의 통신망이 보다 단순화된 구조의 통신망으로 대체될 수 있게 되었다. 트리(tree) 또는 링(ring)과 같은 단순한 망(network)의 구조를 사용하여 효율적인 통신망 구현이 가능하게 된 것이다. 이러한 변화는 망 구축비용을 극소화한다는 측면에서 긍정적이지만, 예측하기 힘든 장애가 발생할 때 서비스의 안정성이 취약해진다

논문접수일 : 2001년 1월 15일 논문게재확정일 : 2001년 8월 30일
* 이 논문은 1999년도 한국학술진흥재단의 연구비에 의하여 연구되었음(KRF-1999-041-C00308).
** 단국대학교 경상학부
*** 울산대학교 경영학부

는 약점을 가지게 된다[11].

이러한 맥락에서 사용자의 요구와 이에 따른 통신망의 성능 평가 기준도 변화하였다. 즉, 서로 격리된 지점간의 단순 연결만으로도 충분했던 단계에서 보다 안정적인 서비스 전달을 요구하는 단계를 거쳐 언제 어떠한 상황에서도 지속적인 서비스가 보장되는 단계로 사용자의 요구가 변화하게 된 것이다. 이에 따라 망의 성능 지표도 단순한 연결가능성이나 접근가능성을 고려하는 지표보다 서비스 제공의 신뢰도 즉 통신망의 위기대처 능력을 평가하는 지표의 비중이 높아지게 되었다.

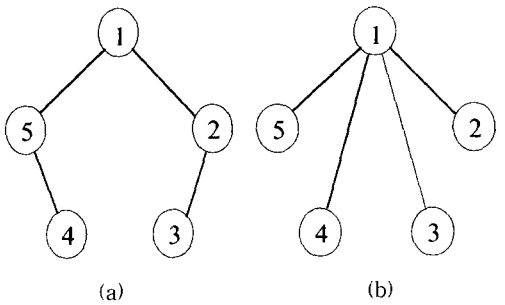
이처럼 통신망의 효율적인 구축 및 안정적인 유지관리를 위한 통신망의 생존도(survivability)가 통신망의 성능을 평가하는 중요한 지표로 떠오르고 있다. 통신망의 생존도란 일반적으로 어떠한 상황에서도 지속되는 서비스의 제공, 즉 통신망의 장애대비 능력을 측정하는 지표를 의미하는데, 이는 통신망 구성요소 장애를 전제로 하는 한계적인 상황에서 통신망의 서비스 제공이 가능한 정도를 측정하는 지표라 할 수 있다. 특히 통신 서비스에의 의존도가 늘어가고 있는 오늘날의 사회에서는 비용 면에서 효율적이면서도 일부 구성요소의 장애에도 지속적인 서비스의 제공이 가능한 안정적인 통신망의 구축이 필요하며, 통신망 생존도의 연구는 이를 위한 필수불가결의 요소라 할 수 있다.

본 연구의 관심은 네트워크의 외형구조(topology)에 따른 생존도의 차이여서 네트워크를 구성하는 설비의 용량은 고려하고 있지 않다. 일반적으로 통신망을 설계할 때는 네트워크의 외형구조의 결정과 일단 결정된 구조를 기반으로 장애가 발생했을 때 우회 경로 확보를 위하여 필요한 용량까지 포함하는 포괄적 용량 결정과정을 분리하여 다룬다. 왜냐하면 이 두 단계를 하나로 묶어서 최적의 통신설계모형을 결정하는 것은 현실적으로 불가능하기 때문이다. 본 논문은 첫째 단계에서 고려해야 되는 네트워크의 외형구조와 생존도의 관계를 다루고 있다.

일반적으로 통신망을 그래프로 표현하면, 교환설비는 노드(node)로 전송설비는 링크(link) (망(network)에서는 가지(edge)가 더 일반적인 용어이나 본 논문에서는 링크를 사용하기로 함)로 표현된다. 그리고 노드 사이에는 통신요구량 (traffic requirement)이 주어져 있다. 통신요구량이 존재하는 두 노드를 o-d쌍(origin-destination pair)으로 부르기로 한다. 앞서 언급한대로 우리의 모델은 네트워크의 외형구조와 생존도의 관계를 다루기 때문에 링크는 용량에 제한이 없는 무방향 링크 (undirected link)를 가정한다. 또한 o-d쌍의 통신요구량은 두 노드 사이에 교환이 필요한 양쪽 방향의 통신량의 합을 의미한다. 통신망의 생존도는, 노드 또는 링크로 표현된 망의 요소에 장애가 발생한 경우에도 여전히 처리 가능한 통신요구량이 정상적인 상황에서 네트워크가 처리해야 될 총 통신요구량에 대해 얼마만큼의 비율인지를 나타내는 용어이다. 장애의 대상에 따라 링크 생존도, 노드 생존도 등으로 구분하며, 장애가 일어난 노드 또는 링크의 개수를 k로 표시하는 경우 k-링크 생존도 또는 k-노드 생존도 등의 용어를 사용한다. Brush와 Marlow[1], Cardwell 등[2] 및 Wu 등 [12]은 통신망의 구조에 따른 신뢰성과 비용간의 관계를 비교하기 위해서 망의 구조별로 k-링크 생존도를 계산하여 신뢰성의 지표로 사용하였다. 그러나, 이러한 연구들에서는 작은 규모의 망을 대상으로 생존도를 일일이 헤아리는 방법(enumeration method)으로 계산하였으며, 대규모의 망에서도 k-링크 생존도를 계산할 수 있는 체계적인 방법을 제시한 연구는 아직 찾아보기 어렵다.

한편으로 기존의 많은 연구들 중에는 망의 연결도(connectivity)를 이용하여 생존도를 간접적으로 표현하기도 하였다[3, 4, 7, 8]. 여기서 연결도란 통신망의 노드간에 존재하는 링크가 중복되지 않는 경로(link-disjoint path)의 수를 나타낸다. 연결도 역시 통신망의 장애대비 능력을 표현하게 된다. 즉, 특정 통신망의 연결도가 k라고 하면 어떠한 두 노드 사이에도 k개 이상의 링크가 중복되지 않는 연결경로가 존재함을 의미하는데 이 경우 링크의

용량이 충분히 크다고 가정하면 k개 미만의 링크 장애가 발생하더라도 해당 통신망은 모든 노드간의 통신요구량을 처리해 줄 수 있는 것이다. 그러나, 이 경우에 k개 이상의 링크 장애가 발생하는 경우는 얼마나 많은 통신요구량이 손실되는 지를 설명해 주지는 못하기 때문에 연결도는 제한적인 생존도 지표라 할 수 있다. 이러한 점은 [그림 1]의 망 모형을 통해 쉽게 설명될 수 있다. (a)와 (b)의 그래프는 모두 동일한 수준의 연결도를 가지고 있다. 둘 다 연결도는 1이다. 그러나 1-링크 생존도에 있어서는 상당한 차이를 보여주고 있다. 모든 노드 사이의 통신요구량이 1이라고 가정하면 한 개의 링크가 손상되는 경우 손실 가능한 최대 통신요구량이 (a)는 6이고 (b)는 4이다. 모든 노드 사이에 통신요구량이 1이므로 총 통신요구량은 10이 되므로 1-링크 생존도는 (a)는 0.4이고 (b)는 0.6이다.



[그림 1] 두 개의 서로 다른 망 구조

생존도는 노드간의 통신요구량, 즉 흐름량(flow)을 기준으로 분석하므로, 망의 신뢰성을 좀 더 정확하게 평가할 수 있다는 장점을 가지지만, 한편으로는 흐름량을 고려하기 때문에 정확한 값을 구하기가 어렵다는 단점도 있다. 실제로 Myung 등[9]은 생존도를 고려하는 망설계 모형이 연결도를 고려하는 망설계 모형을 특수한 경우로 포함하는 좀 더 세밀한 모형이라는 사실과 함께, 생존도 계산문제가 NP-hard에 속하는 어려운 문제임을 밝혔다. 이러한 수리적 계산의 복잡성 때문에 작은 규모의 망을 대상으로 일일이 헤아리는 방법(enumeration)에 의해 생존도를 계산한 연구는 있었으나[6, 12], 일반적인 망에서의 k-링크 생존도의 계산에 관한 연구는 아직까지 다루어진 적이 없다. 다만 노드간의 연결도가 $k-1$ 이상인 그래프에서 k-링크 생존도를 계산하는 제한적 방법은 Myung 등[9]에 의하여 제시되었다.

본 연구에서는 특별한 구조적 특성을 갖지 않는 일반적인 그래프에서 k-링크 생존도를 계산하는 해법을 제시하고자 한다. 앞서 언급한대로 k-링크 생존도의 계산문제는 NP-hard에 속하는 문제이므로, 다항시간(polynomial time)안에 이 문제를 풀 수 있는 해법을 기대하기는 어렵다. 따라서 바람직한 접근방법은 원 문제의 최적해에 가까운 실행가능해를 구하는 것이며 특히 원 문제의 목적함수 값에 대한 상한과 하한을 구할 수 있다면 구해진 해의 품질을 평가하는데 이용할 수 있다. 또한 분지해법(branch and bound method)을 이용하여 최적해를 구하고 싶은 경우에도 상한과 하한은 요긴하게 사용된다. 본 연구에서는 k-링크 생존도를 계산하기 위한 수리적인 모형을 제시하고 주어진 모형을 이용하여 k-링크 생존도의 상한과 하한을 구하는 방법을 제시하기로 한다. 또한 주어진 그래프의 사전처리(preprocessing)를 통하여 그래프의 크기를 줄이는 방법도 개발하기로 한다. 이를 위해서 2절에서는 링크 생존도의 계산을 위한 문제를 기술하고, 3절에서 정수계획모형의 도출, 사전처리를 통한 문제의 단순화방법, k-링크 생존도의 상한과 하한의 산출을 위한 절차를 소개한다. 4절에서는 개발된 해법을 현실의 통신망과 추가로 생성된 망에 적용한 계산결과를 통해서 개발된 해법의 효율성을 분석하기로 한다.

2. 용어의 정의 및 문제의 특성

주어진 그래프에서 노드 집합을 $V=\{1,\cdots,n\}$, 무방향 링크의 집합을 $E=\{1,\cdots,m\}$라 정의하면 그래프 $G=(V,E)$는 통신망의 구조를 나타내게 된다. 또 노드 $i\in V$와 노드 $j\in V$로 이루어진 o-d쌍을 $\{i,j\}$로, o-d쌍 $\{i,j\}$사이에 존재하는 통신요

구량을 t_{ij} 또는 t_{ji}로 나타내고 (즉 $t_{ij}=t_{ji}, \forall\{i,j\}\subseteq V$), 그리고 통신요구량이 존재하는 o-d쌍의 집합을 $T=\{\{i,j\}|t_{ij}>0\}$로 표시하기로 한다. o-d쌍과 링크를 구분하기 위해서 두 노드 $i\in V$와 $j\in V$에 걸쳐있는 링크를 $e=(i,j)$로 표시하기로 하고 두 노드 i와 j를 링크 e의 종단노드(end nodes)라고 부르기로 한다. 주어진 그래프 G에는 두 노드를 동일한 종단노드로 하는 복수의 링크(multiple link)는 존재할 수 있으나 동일한 노드를 종단노드로 하는 링크(self-loop)는 존재하지 않는 것으로 가정한다. 그래프 G에서 임의의 두 노드 i와 j 사이의 링크가 중복되지 않는 경로의 수, 다시 말해서 두 노드 i와 j 사이의 연결도를 $\lambda(i,j,G)$로 표시한다.

노드의 집합 V에 대해서 $V=V_1\cup\cdots\cup V_p$, 이고 $V_i\cap V_j=\emptyset, \forall i,j$을 만족하는 V의 부분집합들로 구성되는 $(V_1,\cdots,V_p)$를 V의 분할(partition)이라고 정의한다. 그리고 $(V_1,\cdots,V_p)$에 대해서 $\delta(V_1,\cdots,V_p)$는 종단노드가 서로 다른 노드의 부분집합에 속하게 되는 링크의 집합을 정의하기로 한다. V의 임의의 부분집합 W에 대해 $\delta(W,V\backslash W)$는 표기의 편의를 위해서 $\delta(W)$로도 표시하기로 한다. 그러면 $\delta(W)$는 양쪽 종단노드 중 하나의 노드는 W에 다른 노드는 $V\backslash W$에 속하는 링크의 집합, 즉 노드집합을 둘로 분리시키는 컷(cut)을 나타내게 되며, 만약에 $i\in W$이고 $j\in V\backslash W$이면 $\delta(W)$는 $i-j$ 컷이라고 부른다. $\delta(V_1,\cdots,V_p)$는 컷의 확장된 개념이므로 다중컷(multicut)이라고 부르기로 한다. 또한 $t(V_1,\cdots,V_p)$는 o-d쌍의 한 쪽 노드가 서로 다른 노드의 부분집합에 속하게 되는 o-d쌍의 통신요구량의 합으로 정의하고 V의 임의의 부분집합 W에 대해 $t(W,V\backslash W)$는 $t(W)$로도 표시하기로 한다. 그러면 $t(W)=\sum_{i\in W}\sum_{j\in V\backslash W}t_{ij}$이고, $t(V_1,\cdots,V_p)$는 다음과 같이 나타낼 수 있다.

$$t(V_1,\cdots,V_p)=\frac{1}{2}\sum_{i=1}^{p}t(V_i)$$

그래프 $G=(V,E)$와 노드사이의 통신요구량이 주어져 있을 때 k-링크 생존도는 어떠한 k개의 링크에 장애가 발생해도 여전히 처리되어질 수 있는 통신요구량의 총 통신요구량에 대한 비율로서 정의된다. 따라서 k-링크 생존도가 0.9인 통신망이 있다면 어떠한 k개의 링크에 장애가 발생하더라도 총 통신요구량의 90%이상이 처리되어질 수 있음을 의미하게 된다. 여기서 통신망 G의 총 통신요구량에 대한 k개의 링크장애로 인한 서비스의 최대손실을 $L_k(G)$로 정의하면, 다음과 같이 표현할 수 있다.

$$L_k(G)=\max\{t(V_1,\cdots,V_p):(V_1,\cdots,V_p)\text{는 } V\text{의 분할이고, } |\delta(V_1,\cdots,V_p)|\le k\}$$

k-링크 생존도는 $L_k(G)$의 값에 의해서 구해지므로 본 연구에서는 이 $L_k(G)$의 효율적인 계산방법을 다루기로 한다. 앞으로는 의미상의 혼동이 없는 범위 내에서, $L_k(G)$와 k-링크 생존도를 구별하지 않고 사용하기로 한다.

3. 링크 생존도의 산출

$L_k(G)$를 정확하게 산출하는 하나의 방법은 k개 이하의 링크로 구성되는 모든 다중컷에 대하여 그 단절에 따른 통신요구량의 손실을 계산하는 것이다. 그러나 대상이 될 수 있는 다중컷의 수는 매우 많기 때문에 모든 다중컷을 분석하는 것은 현실적으로 쉽지 않으며, 이러한 사실이 이 문제가 NP-hard에 속하는 문제임을 암시하고 있다. 본 연구에서는 현실적인 효율성을 감안하여 적절한 시간 내에 $L_k(G)$의 상한과 하한을 구하는 방법을 제시하고자 한다. 이 장에서는 $L_k(G)$를 산출하는 정수계획모형을 소개하고, 주어진 그래프의 크기를 줄일 수 있는 사전처리과정을 제시하며, 적정의 휴

리스틱을 통하여 $L_k(G)$의 하한을 구하는 과정과 정수계획모형의 선형계획 완화문제를 통하여 상한을 구하는 과정을 보이기로 한다.

3.1 정수계획모형

$L_k(G)$는 k-링크 장애상황에서의 최악의 통신요구량의 손실이므로, 그러한 손실을 주는 k-링크 집합을 찾으면 $L_k(G)$를 구하게 된다. 특정의 링크 e에 대해서 그 장애여부를 나타내는 변수 x_e와 두 노드 i와 j사이의 통신요구량의 손실여부를 나타내는 변수 y_{ij}를 다음과 같이 정의하자.

$$x_e=\begin{cases}1, & \text{링크 } e\text{에 장애가 일어났을 때}\\ 0, & \text{링크 } e\text{가 정상일 때}\end{cases}$$

$$y_{ij}=\begin{cases}1, & \text{노드 } i\text{와 노드 } j\text{의 연결이 끊어진 경우}\\ 0, & \text{노드 } i\text{와 노드 } j\text{의 연결이 유지되는 경우}\end{cases}$$

그러면 $L_k(G)$의 산출을 위한 문제는 다음과 같은 정수계획모형 (IP)로 나타낼 수 있다.

$$(IP)\quad max\ z=\sum_{(i,j)\in V} t_{ij}\,y_{ij} \tag{1}$$

$$s.t.\quad \sum_{e\in E} x_e\le k, \tag{2}$$

$$\sum_{e\in E(p)} x_e\ge y_{ij},\ \forall p\in P_{ij},\ \forall\{i,j\}\in T, \tag{3}$$

$$x_e\in\{0,1\},\quad \forall e\in E, \tag{4}$$

$$y_{ij}\in\{0,1\},\quad \forall\{i,j\}\in T, \tag{5}$$

여기서 P_{ij}는 그래프 G에 존재하는 노드 i와 노드 j사이의 모든 경로(path)의 집합을 나타내고 $E(p)$는 경로 p에 속한 링크의 집합을 나타낸다. 단절되는 두 노드 사이의 통신요구량은 충족되지 못하므로, 목적식 (1)은 이러한 충족되지 못한 통신요구량의 합을 최대화시키는 식이다. 제약식 (2)는 k개 이하의 링크장애만을 대상으로 하기 위하여 장애가 발생하는 링크의 수를 k개 이하로 제약하는 식이다. 그리고 제약식 (3)은 노드 i와 노드 j의 연결이 끊어지기 위해서는 두 노드 사이의 경로마다 적어도 한 링크는 장애가 있어야 됨을 의미하는 것이다. 제약식 (4)와 (5)는 변수들의 정수조건을 나타낸다.

모형 (IP)의 최적해를 구함으로써 k-링크 장애시의 최대 서비스 손실을 구할 수 있다. 그러나 대규모의 망에서는 제약식 (3)의 수가 매우 많아지므로 망의 규모가 클수록 (IP)의 최적해를 구하는 것은 어려운 문제가 된다. 그러나 주어진 그래프의 특성을 이용하면 그래프의 사전처리를 통하여 망의 규모를 줄일 수 있음을 보이려고 한다.

3.2 사전처리(preprocessing)를 통한 문제 크기의 축소과정

일반적으로 망을 대상으로 하는 문제를 푸는데 걸리는 시간은 망의 규모가 커짐에 따라 기하급수적으로 늘어난다. 따라서 배제하여도 최적해에 영향을 주지 않는 노드나 링크를 알 수 있다면 망의 규모를 축소할 수 있어서 문제 해결에 소요되는 시간을 줄일 수 있을 것이다.

우리 문제에서의 망의 축소는 망의 두 개의 노드를 축약(contraction)하는 과정을 통해서 이루어진다. 여기서 두 노드의 축약은 두 노드를 종단노드로 하는 링크들을 제거하고 두 노드를 하나의 노드로 합치는 것을 의미한다. 이 때, 두 노드 중 어느 한 노드에 연결되었던 링크는 새로이 합쳐진 노드로 연결되게 되고 축약의 결과로 동일한 노드를 종단노드로 하게 되는 링크(self-loop)는 그래프에서 제거된다. 또한 축약의 결과로 두 노드 사이에는 두 노드를 동일한 종단노드로 하는 복수의 링크(multiple link)가 새로 발생할 수도 있다. 두 노드 i와 j가 새로운 노드 l로 축약되는 경우, 노드 l과 다른 노드 l' 사이의 통신요구량은 $t_{il}+t_{jl}$으로 대체된다. 본 연구에서 제시하는 문제의 사전처리는 다음과 같은 관찰에 근거하고 있다.

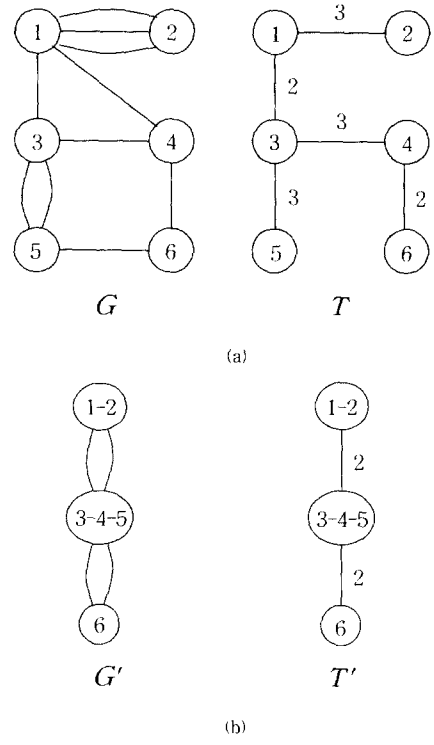
관찰 1. 주어진 그래프 G에서 두 노드사이의 연결도가 k보다 큰 두 노드 i와 j를 축약하여 얻어

진 그래프를 G'이라고 하자. 그러면 G에 포함되어 있던 k개 이하의 링크로 구성된 모든 다중컷은 G'에도 그대로 존재하게 되고 $L_k(G)=L_k(G')$이 만족된다.

증명 : G에 포함되어 있는 k개 이하의 링크로 구성된 임의의 다중컷 $\delta(V_1,\cdots,V_p)$를 고려해 보자. 이 다중컷은 $(V_1,\cdots,V_p)$를 구성하는 임의의 두 부분집합을 분할하는 컷을 포함하고 있으므로 두 노드사이의 연결도가 k 보다 큰 두 노드 i와 j는 $V_1,\cdots,V_p$ 중 동일한 부분집합에 속할 수밖에 없다. 따라서 $\delta(V_1,\cdots,V_p)$를 구성하는 모든 링크는 G'에도 존재하게 되고 $L_k(G)=L_k(G')$이 만족된다. □

관찰 1에 따라 k보다 큰 연결도를 가지는 두 노드는 k-링크 생존도 계산에 영향을 주지 않으므로 하나의 노드로 축약시킬 수 있다. 그래프 G에서 임의의 두 노드 i와 j사이의 연결도는 각 링크의 용량(capacity)을 1로 놓고 구한 두 노드 사이의 최대흐름량(maximum flow)과 같다. 본 연구에서는 주어진 그래프에서 k보다 큰 연결도를 가지는 모든 노드쌍을 찾기 위하여 Gomory-Hu의 방법을 사용하였다[5]. Gomory-Hu의 방법은 임의의 무방향 그래프 G와 링크별 용량을 입력자료로 Gomory-Hu 컷-트리(cut tree)를 생성하는데, 이 컷-트리를 이용하면 원래의 그래프 G에서의 임의의 두 노드 i와 j사이의 최대흐름량을 구할 수 있다. 컷-트리를 구성하는 Gomory-Hu의 자세한 방법에 대해서는 지면상의 이유로 소개하지 못하고 여기서는 컷-트리의 성격에 대해서만 소개하기로 한다. Gomory-Hu의 방법에 대해서는 Hu [5]를 참조하기 바란다.

생성된 컷-트리의 노드는 주어진 그래프 G의 노드와 일치하고 컷-트리에 포함되는 링크는 원 그래프의 링크와는 관계없이 생성되며 링크에는 가중치가 구해지는데 다음과 같은 성질을 만족하게 된다. 원래의 그래프 G에서 두 노드 i와 j사이의 최대흐름량은 컷-트리에서 i와 j를 연결하는 경로(트리이므로 유일하게 존재)에 포함되는 링크들의 가중치 중 최소값으로 나타난다. 또한 컷-트리에서 최소가중치의 링크를 제거하면 노드들은 두 그룹으로 분할되는데, 원 그래프에서 분할된 그룹사이에 걸쳐있는 링크들이 두 노드 i와 j사이의 최소컷(minimum cut)을 이루게된다. [그림 2]의 (a)에 주어진 그래프 G에 대한 컷-트리 T가 예시되어 있다. 예에서 노드 2 와 3 사이의 최대흐름량은 T에서 노드 2 와 3 사이의 유일한 경로에 포함된 링크들의 가중치 중 최소값인 2가 된다. 또한 최소값을 갖는 링크 (1,3)을 T에서 제거하면 노드가 두 그룹 {1, 2}와 {3, 4, 5, 6}으로 분할되는데 그래프 G에서 컷 (1, 2|3, 4, 5, 6) (노드의 부분집합 {1,2}와 {3, 4, 5, 6}을 분할하는 컷을 표시)이 노드 2와 3 사이의 최소컷을 이루게 된다.



[그림 2] Gomory-Hu의 컷-트리

이러한 Gomory-Hu 방법의 장점은 최대흐름문

제를 $\binom{n}{2}$번이 아니라 n-1 번만 풀어서 모든 노드쌍 사이의 최대흐름량을 구할 수 있다는데 있다. 왜냐하면 컷-트리를 만들기 위해서 최대흐름문제를 n-1번 풀면 되기 때문이다. 우리의 사전처리과정은 Gomory-Hu 컷-트리에서 가중치가 k보다 큰 모든 링크에 대해서 해당 링크의 두 종단노드를 원래의 그래프에서 축약시키는 것이다.

사전처리(Preprocessing)

Input : $G=(V,E)$

Output : 축약된 그래프 G'

(단계 1) 그래프 G에서 모든 링크의 용량을 1로 정한 후, Gomory-Hu의 컷-트리를 구한다.

(단계 2) G'을 G로 초기화하고 컷-트리에 포함된 각 링크에 대해서 다음을 수행한다 : 링크의 가중치가 k보다 크면 링크의 두 종단노드를 G'에서 축약시키고 축약된 노드와 다른 노드간의 통신요구량을 조정해 준다.

단계 2에서 노드 i와 j를 축약시킬 때, i나 j가 (또는 두 노드 모두가) 전 단계에서 이미 다른 노드들과 축약되어 있을 수 있다. 이러한 경우에 축약할 노드는 i나 j를 포함하는 축약된 노드가 된다. [그림 2]의 (b)에 주어진 그래프 G로부터 사전처리를 통해서 생성된 축약된 그래프 G'이 예시되어 있다. 사전처리와 컷-트리의 특성상 다음과 같은 사실이 성립됨을 쉽게 알 수 있다.

관찰 2. 원래의 그래프 G에서 노드 i와 j사이의 연결도가 k보다 크다면 컷-트리에 두 노드를 종단노드로 하는 링크가 존재하지 않더라도 단계 2의 연속적인 축약과정을 통해서 i와 j는 G'에서는 하나의 노드로 축약되게 된다. 왜냐하면 컷-트리에서 i와 j를 연결하는 유일한 경로에 포함된 모든 링크는 가중치가 k보다 클 것이기 때문이다. 다르게 표현하면, 컷-트리의 임의의 두 노드 i와 j가 사전처리과정에서 축약되었다면 i와 j를 연결하는 유일한 경로에 포함된 모든 노드는 i,j와 함께 하나의 노드로 축약되게 된다.

이 절을 종료하기 전에 다음 절에서 $L_k(G)$의 하한을 구할 때 유용하게 사용할 수 있는 사실을 하나 소개하기로 한다. 우리의 해법에서는 $L_k(G)$의 하한을 구할 때 컷-트리를 사용하게 된다. 따라서 사전처리를 거쳐서 축약된 새로운 그래프가 구해지면 이 그래프에 대해서 컷-트리를 다시 구해야 한다. 그러나, 다음의 정리를 이용하면 별도의 Gomory-Hu 방법을 수행하지 않고도 사전처리를 위해서 구했던 원래의 그래프의 컷-트리로부터 쉽게 원하는 컷-트리를 유도할 수 있다.

정리 1. 원래의 그래프를 G, 사전처리 후 축약된 그래프를 G', 그리고 G와 G'에 해당하는 컷-트리를 각각 T와 T'이라고 하자. 그러면 G에서 G'을 생성할 때 사전처리과정에서 축약된 노드쌍, 즉 T의 링크 중 가중치가 k보다 큰 링크의 종단노드쌍을 T에서 축약시키면 T'을 얻을 수 있다.

증명 : T'이 G'의 컷-트리임을 증명하기 위해서 G'의 임의의 두 노드 V_a와 V_b 사이의 최소컷이 T'에서 컷-트리의 정의대로 구해질 수 있음을 보이기로 한다. 표현의 편의를 위해서 V_a와 V_b는 G'의 노드를 나타냄과 동시에 V_a와 V_b에 축약되어 들어간 G의 노드의 부분집합도 나타내는 것으로 정의하자. G의 두 노 $i\in V_a$와 $j\in V_b$의 최소컷이 T의 링크 e에 대응된다고 가정하자. 즉, T에서 e를 제거하면 노드들은 두 그룹 X와 $V\backslash X$로 분할되고, G에서 X와 $V\backslash X$ 사이에 걸쳐있는 링크들이 두 노드 i와 j사이의 최소컷이 된다. 당연히 e의 가중치는 k이하이다. 아니면 관찰 2의 사실 때문에 i와 j는 G'에서 하나의 노드로 축약되었을 것이다. 또한 관찰 2의 마지막 문장 내용 때문에 T'의 하나의 노드에는 T의 부분트리(subtree) (하나의 노드로 구성될 수도 있다)가 대응된다. 따라서 e는 T'에서도 V_a와 V_b 사이의 유일한 경로에 포함되는 링크 중 가중치가 최

소인 링크가 되고, T'에서 e를 제거하면 노드들은 X와 $V\backslash X$로 (G의 노드를 기준으로 했을 때) 분할되게 된다. 이제 남은 것은 G'에서 V_a와 V_b 사이의 최소컷이 G'의 노드들을 X와 $V\backslash X$로 분할하는 컷임을 보이면 된다. G에서 X에 포함된 노드와 $V\backslash X$에 포함된 노드 사이의 연결도는 k이하이므로 축약되는 일은 없다. 따라서 X와 $V\backslash X$ 사이의 링크의 집합은 G와 G'에서 동일하고 G'에서도 V_a와 V_b 사이의 최소컷이 된다. □

예로서 [그림 2]에 나타나 있는 것처럼 G'의 컷-트리 T'은 G에서 G'을 생성할 때 사전처리과정에서 축약된 노드쌍을 T에서 축약시킴으로써 얻을 수 있다. 정리 1은 직관적으로 자명하게 생각될 수 있으나 G'이 G의 임의의 노드쌍을 축약한 그래프인 경우에는 성립하지 않는다.

이제 남은 두 절에서는 $L_k(G)$의 상한과 하한을 구하는 방법을 설명하기로 하는데 이 때 대상이 되는 그래프 G는 이미 사전처리를 거쳐서 생성된 그래프로 가정한다. 유의할 것은 사전처리를 거쳤으므로 모든 노드 사이의 연결도는 k개 이하가 된다는 점이다.

3.3 $L_k(G)$의 하한

이제 $L_k(G)$의 하한을 구하는 과정을 살펴보자. (IP)에서 보는 바와 같이 $L_k(G)$를 구하는 문제는 최대화문제이므로 (IP)의 제약식을 만족하는 어떠한 실행가능해라도 $L_k(G)$의 하한이 된다. 따라서 그래프 G에서 k개 이하의 링크로 구성되는 임의의 다중컷을 찾으면 하나의 하한을 얻게 되는 것이다. 가능한 많은 다중컷을 고려할 수록 얻게되는 하한의 효율은 좋아지게 될 것이지만 소요되는 시간은 길어지게 되므로 얻고자 하는 해의 품질과 투입되는 시간을 적절히 조화시키는 것이 중요하다.

k개 이하의 링크로 구성되는 임의의 다중컷을 선택하기 위해서 여러 가지가 다양한 방법을 사용할 수 있으나, 본 연구에서는 다음과 같이 선택할 링크를 추가해 가는 방식의 휴리스틱(add heuristic)을 사용하였다. 다중컷은 성격상 항상 적절한 컷의 합집합으로 구성됨을 쉽게 알 수 있다. 즉, $\delta(V_1,\cdots,V_p)=\delta(V_1)\cup\cdots\cup\delta(V_p)$이다. 우리의 해법은 $|\delta(V_1,\cdots,V_p)|\le k$가 될 때까지 컷, $\delta(V_1)$, $\cdots,\delta(V_p)$를 추가해 가는 것이다. 추가할 컷의 선택을 위해서 우리는 컷-트리를 이용한다. 앞 절에서 사전처리 후에 얻어진 그래프에 대한 컷-트리를 쉽게 구하는 방법에 대해서 언급하였었다. 또한 컷-트리의 링크별로 컷이 대응된다는 사실도 설명하였었다. 우리는 컷-트리에서 얻을 수 있는 $n-1$개의 컷들 중에서 다중컷을 구성하는데 포함시킬 컷을 선택하게 된다. 사전처리를 거친 그래프의 성격상 컷-트리에서 얻을 수 있는 $n-1$개의 컷들은 모두 k개 이하의 링크만을 포함하게 된다.

우리의 해법에서 다중컷을 선택하기 위해서 후보가 되는 $n-1$개의 컷들을 어떤 순서로 어떻게 추가해 가는 지 설명하기로 하자. 어떤 컷을 먼저 선택할 것인지를 정하는 기준으로 여러 가지 방법을 고려할 수 있고 선택된 기준에 따라서 얻어지는 해도 달라지게 된다. 본 연구에서는 링크당 손실되는 통신요구량을 기준으로 컷을 선택하였다. 즉, 고려하는 컷의 링크를 모두 단절했을 때에 손실되는 통신요구량을 컷에 포함된 링크수로 나눈 비율이 높은 컷을 우선적으로 선택하는 것이다. [그림 2]의 (a)에 예시한 그래프와 컷-트리를 예로 들어보자. 통신요구량은 모든 노드쌍에서 1이라고 가정하자. 그러면 컷-트리의 링크 (1,3)에 대응되는 컷 (1, 2 | 3, 4, 5, 6)에 대해서 그래프 G에 포함된 링크의 수는 2개이고 이 컷이 제거되었을 때의 손실되는 통신요구량은 8로서 컷-트리에 포함되어 있는 5개의 컷 중 링크당 손실되는 통신요구량이 4로서 가장 크다. 물론 이 방법이 항상 최선이라고 할 수는 없지만, 4절의 계산결과에서 제시되는 바와 같이 이 방법에 의한 휴리스틱이 작은 문제뿐만 아니라 현실규모의 문제들에 대해서도 만족할 만한 해를 제공하였고 개념적으로도 단순하여 본 연구의

해법에서 채택되었다.

컷-트리에서 구해지는 $n-1$개의 컷에 포함된 G의 링크들은 서로 중복될 수 있다. 따라서 현재까지 선택한 컷의 합집합에 포함된 링크의 수가 k개 미만이어서 컷을 추가하는 경우에 추가로 선택할 컷의 링크당 손실되는 통신요구량의 계산은 최초의 컷을 선택할 때와는 달리 순수 증가 부분만을 계산하게 된다. 즉 컷을 합집합에 추가함으로써 늘어나는 통신 요구량의 손실과 추가로 제거되는 링크의 비율에 의해서 다음 선택될 컷을 정하게 된다. [그림 2]의 (a)를 이용한 예에서 $k=4$인 경우를 고려해 보자. 처음에 컷 (1, 2 | 3, 4, 5, 6)을 선택한 후에 아직 2개의 링크가 더 추가되어도 가능하다. 이 때 컷-트리의 노드 3과 4 사이의 링크에 해당하는 컷 (1, 2, 3, 5 | 4, 6)의 경우, 이 컷이 추가로 제거되면 통신요구량의 손실은 4가 증가되고, G에서 추가로 제거되는 링크의 수는 3이 아니라 2가 된다. 전자의 경우는 노드 3,5와 4,6의 추가 단절 때문이고 후자는 G에서 링크 (1,4)는 이미 처음에 선택한 컷 (1, 2 | 3, 4, 5, 6)에 포함되어 있기 때문이다. 따라서 컷 (1, 2, 3, 5 | 4, 6)의 링크당 손실되는 통신요구량은 4를 2로 나눈 2로 계산된다.

3.4 $L_k(G)$의 상한 도출

모형 (IP)에서 제약식 (4)와 (5)를 각각 $0 \le x_e \le 1, \forall e \in E,\ 0 \le y_{ij} \le 1, \forall \{i,j\} \in T$로 대체함으로써 ($IP$)의 선형계획 완화문제(linear programming relaxation problem)를 얻을 수 있다. 이를 (LP)라 하자. 일반적으로 최대화하는 정수계획모형의 선형계획 완화문제는 원문제의 최적해의 목적함수값에 대한 상한(upper bound)을 제공한다. 즉, 선형계획 완화문제의 최적해를 구함으로써, k-링크 장애 때의 최대손실 $L_k(G)$의 상한을 구할 수 있는 것이다. 그러나 제약식 (3)의 수가 너무 많기 때문에, 모든 제약식을 전부 포함한 선형계획문제를 풀어서 상한을 구하기에는 현실적인 무리가 따른다.

본 연구에서는 제약식 (3) 중 필요한 식만 선택적으로 포함하는 선형계획문제의 해결방법을 사용한다. 이 방법은 우선 제약식 (3) 중 일부의 식만을 포함한 선형계획문제를 풀고, 얻어진 해가 (LP)의 최적해인지를 확인한 다음, 필요하면 다시 제약식을 추가하여 반복하는 단계적 절차로 구성되어 있다. 반복의 단계마다 제약식의 추가에 의해서 실행가능영역(feasible region)을 줄여 가는 이러한 방법을 절단면해법(cutting plane algorithm)이라고 부른다. 우리의 절단면해법에서 중요한 것은 현재의 선형계획문제의 해가 모든 제약식 (3)을 만족하는지를 판정하고 아닌 경우에 현재의 해가 만족시키지 못하는 제약식이 어떤 것인지를 결정하는 분리문제(separation problem)를 쉽게 풀 수 있는가의 여부이다. 절단면해법과 분리문제에 대한 자세한 설명은 Nemhauser와 Wolsey [10]를 참조하기 바란다. 제약식 (3)에 해당하는 분리문제는 아래의 절차 단계 ②에서 설명하는 것과 같이 최단경로문제(shortest path problem)를 풀어서 해결할 수 있다.

$L_k(G)$의 상한 계산절차

① 목적식과 제약식 (2), $0 \le x_e \le 1, \forall e \in E,\ 0 \le y_{ij} \le 1, \forall \{i,j\} \in T$ 만을 포함하는 선형계획문제를 구성한다.

② 최적해 $\{x_e, y_{ij}\}$를 구하여, x_e를 링크 e의 길이로 하는 그래프에서 모든 o-d쌍 $\{i,j\}$에 대해서 최단경로를 구하고 그 길이 l_{ij}를 구하여, 만일 $l_{ij} < y_{ij}$이면 이 최단경로 p에 대해서 제약식 (3)을 추가한다.

③ 추가될 제약식이 없으면 현재의 해가 최적이므로 종료하고, 추가된 제약식이 있으면 추가된 제약식을 포함하여 선형계획문제를 푼 뒤에 다시 ②로 간다.

제약식 (3)에 해당하는 분리문제의 근거는 다음과 같다. x_e를 링크 길이로 하는 그래프에서, 임의의 두 노드 i와 j사이의 최단 경로의 길이가 y_{ij} 이상이면 i와 j사이의 모든 경로의 길이가 y_{ij}이

상이므로 현재의 선형계획문제의 해 $\{x_e, y_{ij}\}$는 노드쌍 i와 j에 대한 모든 제약식 (3)을 만족하게 되고, 만약 어떤 경로 p에 대한 경로의 길이가 y_{ij} 미만이면 $\{x_e, y_{ij}\}$는 해당 경로에 대해서 (3)을 만족시키지 못하게 된다. 이 과정을 반복하여 (LP)에 대한 최적해를 얻게되면, 이 때의 목적함수의 값이 $L_k(G)$의 상한이 된다. 이와 같이 제약식 (3)에 대한 분리문제를 최단경로문제의 해법을 이용하여 다항시간(polynomial time)에 풀 수 있음으로써 선형계획완화문제 (LP)도 다항시간에 풀 수 있음을 알 수 있다.

4. 계산실험 및 결과 분석

본 연구에서 제시된 k-링크 장애 때의 최대손실 $L_k(G)$의 상한과 하한을 구하기 위한 절차, 그리고 문제의 단순화 절차는 C 언어를 통하여 프로그램으로 구현되었다. 물론 계산실험에 적용할 대상문제를 만드는 프로그램도 같이 구현되었으며, 최대손실의 상한을 구하는 선형계획 완화문제를 반복적으로 풀어 가기 위해서 CPLEX 라이브러리를 이용하였다. 현실적 규모의 통신망을 대상으로 하는 실험계산을 위해서 48개 교환국으로 구성된 서울지역의 전화망 구조를 기반으로 대상문제를 만들었으며, 다양한 구조의 통신망을 대상으로 하는 계산실험을 위하여 30개에서 100개까지의 노드로 구성되는 가상의 문제를 추가로 만들었다. 가상의 문제를 만들기 위해서는 먼저 (100×100) 사각형 모양의 격자에 필요한 수만큼의 노드를 임의로 위치시키고, 다음에 노드간에 적정수의 링크설비를 위치시켰다. 우선 노드들이 상호 연결되도록 하나의 트리로 연결한 후, 지정된 수만큼의 추가적인 링크를 설치하였다. 노드간의 통신요구량은 일정한 범위 내에서 임의추출방식(random sampling)으로 생성하였다.

작성된 코드는 150Mhz CPU를 가진 펜티엄급 PC에서 실행되어 졌다. 모두 540문제를 대상으로 시험계산이 이루어졌으며, <표 1>~<표 4>는 그 결과를 보여주고 있다. 표들에 나타난 모든 결과는 각각 10문제에 대한 결과의 평균값이며, 분석이 용이하도록 상한, 하한 및 최대손실량을 총 통신요구량으로 나눈 비율로 결과를 표현하였다. 이 결과들은 전체적으로 본 연구에서 제시된 절차들이 상당히 큰 규모의 현실문제에도 효율적으로 적용될 수 있음을 보여주고 있다.

<표 1>은 서울지역의 노드수 48개, 링크수 87개로 고정된 하나의 망을 대상으로 통신요구량을 변화시켜 본 결과이다. 망의 구조가 상대적으로 단순하기 때문에 일일이 헤아리는 방법(enumeration

〈표 1〉 서울지역망에 대한 시험적용 결과

	문제크기			사전처리			최적해	하한		상한		상한－하한
k	|V|	|E|	|T|	|V|	|E|	CPU	%	%	CPU	%	CPU	하한
2	48	87	96	10	17	0.46	9.9	9.9	0.01	9.9	0.24	0.0
			384			0.44	8.3	8.3	0.01	8.3	0.18	0.0
			672			0.44	8.1	8.1	0.01	8.1	0.21	0.0
3	48	87	96	28	54	0.42	20.5	20.1	0.08	21.1	0.39	0.05
			384			0.44	18.8	18.8	0.11	18.8	0.76	0.0
			672			0.46	19.3	19.3	0.09	19.3	1.14	0.0
4	48	87	96	38	72	0.43	28.7	27.3	0.31	29.7	1.25	0.09
			384			0.44	26.8	25.7	0.29	27.1	5.48	0.05
			672			0.50	26.0	23.0	0.35	26.1	11.34	0.13
5	48	87	96	46	86	0.43	34.9	30.2	0.71	39.4	3.19	0.30
			384			0.45	31.7	27.3	0.74	36.6	23.76	0.34
			672			0.55	31.5	26.5	0.74	35.8	54.42	0.35

〈표 2〉 소규모망에의 적용결과

	문제크기			사전처리			최적해	하한		상한		상한－하한
k	|V|	|E|	|T|	|V|	|E|	CPU	%	%	CPU	%	CPU	하한
2	30	40	120	17.3	24.0	0.12	31.3	30.5	0.05	31.5	0.25	0.03
		40	240	17.3	24.0	0.11	28.6	26.3	0.06	28.8	0.37	0.10
		50	120	9.6	14.1	0.12	16.5	16.2	0.03	17.0	0.19	0.05
		50	240	9.6	14.1	0.11	16.6	16.0	0.02	17.1	0.17	0.07
3	30	40	120	26.6	36.8	0.12	42.6	38.5	0.19	43.5	0.94	0.13
		40	240	26.6	36.8	0.11	42.1	39.2	0.17	42.5	1.91	0.08
		50	120	19.3	35.2	0.13	23.8	22.6	0.08	25.1	0.43	0.11
		50	240	19.3	35.2	0.12	23.4	22.3	0.07	24.1	0.71	0.08
4	30	40	120	29.9	39.9	0.12	53.1	47.6	0.29	54.3	1.75	0.14
		20	240	29.9	39.9	0.11	50.5	47.4	0.28	51.6	4.45	0.09
		50	120	26.4	46.8	0.12	31.3	29.4	0.20	33.8	1.73	0.15
		50	240	26.4	46.8	0.14	29.8	29.4	0.20	32.4	4.17	0.10
5	30	40	120	29.9	39.9	0.12	61.1	56.5	0.33	62.0	1.74	0.10
		40	240	29.9	39.9	0.11	58.2	53.6	0.35	59.6	4.40	0.11
		50	240	29.3	49.6	0.13	39.4	33.1	0.33	41.3	3.08	0.25
		50	240	29.3	49.6	0.15	37.4	32.0	0.34	39.2	7.60	0.23

method)에 의해 최대손실의 정확한 값까지 구하여 하한 및 상한과 대비시켰다. <표 2>는 노드 수 30개인 시험망을 대상으로 한 결과이며, 상대적으로 작은 규모이므로 링크수 및 통신요구량을 다양하게 변화시켜보았다. <표 1>에서와 마찬가지로 헤아리는 방법으로 구한 최대손실의 정확한 값과 우리의 해법으로 구해진 상한, 하한을 대비시켜 보았다. <표 3>과 <표 4>는 큰 규모의 시험망을 대상으로 한 결과이며, 최대손실의 정확한 값을 구하는 것이 사실상 불가능했기 때문에 상한과 하한의 차이를 구하여 정리하였다.

각각의 표에는 3.2절에서 소개한 사전처리과정의 효율을 보여주기 위하여, 원래 망의 규모와 사전처리가 이루어진 이후 망의 규모를 대비하였고 소요된 시간을 표시하였다. 예상할 수 있는 것처럼 상대적으로 적은 k값(2 또는 3)에 대해서는 사전처리는 상당한 효과를 보이고 있으며, k가 커짐(4 또는 5)에 따라 그 효과는 줄어들고 있음을 알 수 있다. 또 그래프의 밀도(density)가 높은 통신망의 경우에 그 효과가 보다 잘 나타났다.

〈표 3〉 대규모망에의 적용결과(노드수 80)

	문제크기			사전처리			하한		상한		상한－하한
k	|V|	|E|	|T|	|V|	|E|	CPU	%	CPU	%	CPU	하한
2	80	100	100	50.3	69.3	1.42	18.9	0.69	19.5	0.66	0.03
	80	150	100	17.8	29.3	1.65	6.8	0.04	7.0	0.20	0.03
	80	200	100	7.6	12.3	1.88	5.9	0.01	6.1	0.17	0.03
3	80	100	100	73.4	96.5	1.38	25.3	2.63	28.9	4.27	0.14
	80	150	100	39.1	79.4	1.65	9.2	0.52	10.3	0.66	0.12
	80	200	100	16.9	39.2	1.90	7.2	0.04	8.6	0.19	0.19
4	80	100	100	79.7	99.9	1.38	32.4	3.76	38.6	7.92	0.19
	80	150	100	59.8	126.0	1.64	11.6	2.18	13.7	3.11	0.18
	80	200	100	33.9	94.8	1.91	9.7	0.38	10.8	0.48	0.11
5	80	100	100	80.0	100.0	1.38	37.4	4.48	46.9	7.89	0.25
	80	150	100	71.2	143.0	1.65	14.0	3.89	17.1	10.19	0.22
	80	200	100	52.7	152.0	1.91	11.6	1.48	13.0	2.29	0.12

〈표 4〉 대규모망에의 적용결과(노드수 100)

	문제크기			사전처리			하한		상한		상한－하한
k	|V|	|E|	|T|	|V|	|E|	CPU	%	CPU	%	CPU	하한
2	100	120	150	70.5	93.5	2.58	19.3	1.59	20.0	1.52	0.04
	100	150	150	41.7	63.9	2.72	9.3	0.36	9.9	0.40	0.06
	100	200	150	18.3	30.6	3.10	5.1	0.04	5.2	0.20	0.02
3	100	120	150	94.1	117.7	2.55	26.6	4.60	29.5	9.08	0.11
	100	150	150	73.8	121.9	2.74	13.8	2.24	14.1	3.86	0.02
	100	200	150	41.9	90.9	3.10	7.1	0.55	7.7	0.54	0.08
4	100	120	150	99.6	119.9	2.55	33.0	6.82	38.8	21.23	0.18
	100	150	150	91.6	144.8	2.75	17.3	5.66	18.6	22.09	0.08
	100	200	150	67.5	154.5	3.10	9.2	2.59	10.3	4.69	0.12
5	100	120	150	100.0	120.0	2.55	38.8	7.85	46.8	21.87	0.21
	100	150	150	98.3	149.2	2.74	20.8	8.06	22.9	68.79	0.10
	100	200	150	85.2	186.3	3.11	11.3	6.04	12.7	30.55	0.12

제시되는 하한 및 상한의 성능을 기준으로 할 때, 상대적으로 적은 k인 경우 보다 나은 효율을 보여 주었다. 많은 경우에 거의 최적에 가까운 하한과 상한을 얻음으로써, 최대손실의 정확한 값을 찾을 수 있었으며, k가 커짐에 따라 상한과 하한의 차이가 증가함을 알 수 있었다. 휴리스틱에 의한 하한의 효율도 떨어지며, 선형계획 완화문제를 통한 상한의 효율도 떨어지고 있다. 현재의 방법들이 대규모의 망에서도 실행시간의 면에서는 만족스럽다고 평가할 수 있으나 좀 더 나은 상한과 하한을 얻기 위해서는 보다 세밀한 형태의 휴리스틱의 개발이나, 유효부등식(valid inequalities)를 이용한 선형계획 완화문제의 상한의 향상 등을 고려해 볼 수도 있을 것으로 판단된다.

5. 결 론

본 연구에서는 통신망의 k-링크 생존도를 계산하기 위하여 k개의 링크장애로 인한 통신요구량의 최대손실을 산출하는 해법을 제시하였다. k-링크 생존도는 통신망이 k개 이하의 링크 장애에 대처하는 능력으로서 최악의 상황에서도 여전히 처리될 수 있는 서비스의 비율을 의미하며 이러한 생존도의 정의는 실제 통신망의 설계자들에 의해서 사용되었던 개념이다. k-링크 생존도를 계산하는 것은 NP-hard 문제로 알려져 있으므로 본 연구에서는 k개의 링크장애로 인한 통신요구량의 최대손실을 계산할 수 있는 수리모형을 제시하고, 제시된 모형의 하한을 산출하는 휴리스틱과 선형계획 완화문제를 이용한 상한을 도출하는 절차를 개발였다. 또한 주어진 그래프의 특성을 이용하여 문제의 크기를 줄이는 과정을 제시하였고, 폭넓은 범위의 문제들을 대상으로 한 계산실험을 통하여 제시된 해법의 현실문제에의 적용가능성을 확인하였다. k 값에 따라 계산결과의 효율성은 다소 차이가 있으나 원래 문제의 난이도를 감안할 때 상대적으로 규모가 큰 망에서도 해법의 효율은 만족스럽다고 판단된다. 다만 k가 커지는 경우에 상한과 하한의 차이가 빠르게 증가함에 비추어 향후 하한을 구하기 위한 다양한 휴리스틱을 개발하고 문제의 다면체적 구조(polyhedral structure)의 분석을 통해서 선형계획 완화문제의 상한을 향상시키는 연구가 이루어질 필요가 있다고 판단된다.

참 고 문 헌

[1] Brush, G.G. and N.A. Malrow, "Assuring the dependability of telecommunications networks and services," *IEEE Network magazine* Jan. (1990), pp.29-34.

[2] Cardwell, R., C.L. Monma, and T. Wu, "Computer-aided design procedures for survivable fiber optic networks," *IEEE J. SAC* 7, (1989), pp.1188-1197.

[3] Gavish, B., P. Trudeau, M. Dror, M. Gendreau and L. Mason, "Fiberoptic circuit network design under reliability constraints," *IEEE J. SAC* 7, (1989). pp.1181-1187.

[4] Grötschel, M., C.L. Monma, and M. Stoer, "Design of survivable networks," in *Network Models*, M.O. Ball et al. (eds.), North-Holland, Amsterdam 1995.

[5] Hu, T.C., *Combinatorial Algorithms*, Addison-Wesley, Reading, 1982.

[6] Kolar, D.J. and T. Wu, "A study of survivability versus cost for several fiber network architectures," *Proceedings of IEEE Int'l Conference on Communications*, (1988), 61-66.

[7] Monma, C.L., B.S. Munson and W. R. Pulyblank, "Minimum-weighted two-connected spanning networks," *Mathematical Programming*, 46, (1990). pp.153-171.

[8] Monma, C.L. and D.F. Shallcross, "Methods for designing communications networks with certain two-connected survivability constraints," *Operations Research*, 37, (1989), pp.531-541.

[9] Myung, Y.-S., H.-J. Kim, and D.-W. Tcha, "Design of communication networks with survivability constraints," *Management Science*, 45, (1999), pp.238-252.

[10] Nemhauser, G.L. and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York 1988.

[11] Wu, T., *Fiber network survivability*, Artech House, Boston 1992.

[12] Wu, T., D.J. Kolar and R.C. Cardwell, "Survivable network architectures for broadband fiber optic network : model and performance comparison," *Journal of Lightwave Technology*, 6, (1988) pp.1698 1709.