Journal of the Korean Institute of Industrial Engineers
Vol. 25, No. 2, pp. 184-191, June 1999.

# 절단기수의 나열을 통한 최대유통문제에서 모든 $k$-치명호를 찾는 방법†

안재근[1] · 정호연[2] · 박순달[3]

[1]한경대학교 컴퓨터공학과 / [2]전주대학교 산업공학과 / [3]서울대학교 산업공학과

# A Method for Determining All the $k$ Most Vital Arcs in the Maximum Flow Problem by Ranking of Cardinality Cuts

Jaegeun Ahn[1] · Hoyeon Chung[2] · Soondal Park[3]

The $k$ most vital arcs ($k$-MVA) of a maximum flow problem is defined as those $k$ arcs whose simultaneous removal from the network causes the greatest decrease in the throughput capability of the remaining system between a specified pair of nodes. In this study, we present a method for determining all the $k$-MVA in maximum flow problem using a minimal cardinality cut algorithm and $k$-th minimal cut ranking algorithm. For ranking cardinality cuts, we use Hamacher's ranking algorithm for cut capacity and by comparing present residual capacity of cardinality cut with expected residual capacity of next cardinality cut, we also present termination condition for this algorithm.

While the previous methods cannot find all the alternatives for this problem, a method presented here has advantage of determining all the $k$-MVA.

## 1. 서 론

최대유통문제는 대표적인 네트워크문제 중의 하나로써, 시발점에서 종착점까지 네트워크를 통해 보낼 수 있는 최대의 유통량과 경로를 구하는 문제이다(Ahuja *et al.*, 1993). 이때 주어진 네트워크에서 우발적이거나 고의적인 여러 요인에 의해 호(arc)가 고장나거나 파괴될 수가 있는데, 어떤 호들의 고장이나 파괴가 시발점과 종착점 간의 최대유통량을 가장 크게 감소시키는가를 알아보는 문제를 치명호(most vital arc)문제라고 한다(Lubore *et al.*, 1971; Malik *et al.*, 1989; Ratliff *et al.*, 1975). 따라서 $k$개의 치명호($k$-most vital arcs : 이하 $k$-치명호)를 찾는 문제란 $k$개의 호들로 이루어진 호집합을 동시에 제거함으로써 시발점에서 종착점까지의 최대유통량을 가장 많이 줄이는 호집합을 찾는 문제를 의미한다(Ratliff *et al.*, 1975). 이러한 연구는 적의 공격하에 처해 있는 이해상충의 상황이나 물류 또는 통신네트워크에서 어느 호가 가장 치명적인지 알아내어 적의 공격으로부터 경계를 강화하거나 또는 어느 호를 파괴시켜야 적의 시스템의 효율성을 가장 크게 저하시킬 수 있는지 알고자 하는 문제에 잘 적용될 수 있다. 예를 들면, 지역정보와 각종 도변이 저장되어 있는 지리정보시스템(GIS)에서의 도로망, 상·하수도관망, 송전망 등의 설계 및 관리, 군사용 보급로나 물류에서의 수·배송망 관리, 통신망, 송·수신 네트워크 등 호의 제거가 전체 네트워크의 성능에 치명적으로 영향을 미치는 경우의 네트워크문제에 활용될 수 있다.

이에 대한 연구는 Wollmer와 Ratliff 등에 의해 수행되었다. Wollmer는 무방향(undirected)이고 평면(planar)인 네트워크에 대하여 위상적 쌍대 네트워크(topological dual network)를 구성하여 $k$-치명호를 찾는 해법을 제시하였다(Wollmer, 1964). Ratliff 외 2인 (1975)은 유방향인 네트워크에 대하여 적은 개수의 절단을 만들기 위해 순차적으로 네트워크를 변형하는 $O$(용량상한* $n^3$)인 변형네트워크 절차(modified network procedure)와 나무탐색법을 이용한 해법을 제시하였다(Ratliff *et al.*, 1975). 그러나 이 방법은 $k$-치명호에 대한 대안해가 존재할 경우에도 하나의 $k$-치명호밖에는 제시하지 못하는 단점을 가지고 있다.

$k$-치명호문제에 대해서는 최단경로문제에서의 $k$-치명호문제에 대해 Corley와 Sha, Malik 외 2인, Bar-Noy 외 2인 그리고 안재근에 의해 연구가 수행되었다. Corley와 Sha(1982)는 $K$번째 최

†본 연구는 정보통신부 '97년도 대학기초연구지원사업(97-G-0683)의 지원을 받음.

단경로($K$-th shortest path)를 구하는 해법을 이용하여, 최단경로 문제에서 1개의 치명호(1-치명호)를 찾는 해법과, $k$개의 치명호($k$-치명호)를 찾는 문제에 대한 충분조건을 제시하였다(Corley and Sha, 1982). Malik 외 2인은 Corley와 Sha의 방법을 개선하여 무방향 네트워크에서 1-치명호를 찾는 효율적인 해법을 제시하였고, $k$-치명호에 대해서는 이를 찾는 비다항식의 계산의 복잡도(non- polynomial time complexity)를 갖는 해법을 제시하였다(Malik *et al.*, 1989). 그러나 Malik 외 2인이 제시한 $k$-치명호를 찾는 해법은 Bar-Noy 외 2인에 의해 해결될 수 없는 반례가 제시되어, 해법에 오류가 있는 것으로 판명되었고, 또한 최단경로문제의 $k$-치명호를 찾는 문제의 계산의 복잡도가 NP-hard임이 증명되었다(Bar-Noy *et al.*, 1995). 그리고 안재근은 Malik 외 2인이 제시한 해법의 오류를 수정하면서 다수최단경로문제를 이용하여 $k$-치명호문제를 해결하기 위한 방법을 혼합정수계획법의 부분문제를 이용하여 제시하였다(안재근, 1997).

본 연구에서는 최대유통문제에서의 $k$-치명호를 찾는 문제를, 최단경로문제의 $k$-치명호를 찾는 문제에서 다수 번째 최단경로를 나열하면서 $k$-치명호에 해당하는 호를 선택하는 방법의 개념을 이용하여, 최대유통문제에서 다수 번째의 절단을 오름차순(ascending order)으로 나열하는 방법을 통해 존재하는 모든 $k$-치명호집합을 구하는 해법을 제시하고자 한다. 이 방법은 기존의 Ratliff 외 2인의 연구에서 적은 수의 호로 이루어진 절단을 우선적으로 고려하여 주는 변형네트워크 절차를 이용한 방법에 착안하여 용량절단(capacity cut)보다는 절단기수(cardinality of cut)를 고려하여 작은 절단기수로부터 큰 절단기수로 절단을 기수(cardinality)의 오름차순으로 나열하면서 존재하는 모든 $k$-치명호를 구하는 해법을 제시하고자 한다.

이를 위해 2절에서는 절단용량의 오름차순 나열 알고리즘의 개념을 통해 절단기수의 오름차순 나열 알고리즘을 제시하고, 이를 통해 3절에서 $k$-치명호를 찾는 문제의 해법을 제시하고자 한다. 그리고 마지막으로 4절에서 예제를 통해 절단기수의 오름차순 나열해법과 최대유통문제의 $k$-치명호를 찾는 문제의 해법의 적용예를 보이고자 한다.

## 2. 절단기수(cardinality of cut)의 나열(ranking) 방법

본 연구에서 다루는 네트워크에는 시발점( source node ; $s$ )과 종착점( sink node ; $t$ )이 지정되고, 각 호 $(i,j)$에는 비음수(non-negative) 파라미터인 용량상한값(upper bound) $u_{ij}$와 용량하한값(lower bound) $l_{ij}$가 주어진다. $u_{ij}$는 호 $(i,j)$를 통과하는 유통량(flow) $f_{ij}$의 최대허용값을 나타내며, $l_{ij}$는 $f_{ij}$의 최소허용값을 나타낸다(본 연구에서는 편의상 $l_{ij}$를 0으로 놓았음). 이때 각 마디(node)에서 흘러나가는 유통량(outflow)은 유통량보존법칙(flow conservation law)에 의해 그 마디로 흘러들어오는 유통량(inflow)과 같아야 한다(단, $s$와 $t$에서는 예외).

$s$에서 나간 모든 유통량은 $t$에 도착해야 하기 때문에, 이 유통량의 총량을 $v$라고 하면, $v$는 다음과 같이 표현된다.

$$\sum_k f_{sk} = v = \sum_j f_{jt}$$

이러한 용량제약이 있는 네트워크에서 최대유통문제는 시발점 $s$에서 종착점 $t$까지 네트워크를 통해 보낼 수 있는 최대의 유통량을 구하는 문제이며, 다음과 같이 모형화된다(Ahuja *et al.*, 1993; Murty, 1992).

$$\begin{aligned} &max \quad v \\ &s.t. \quad \sum_{i\in G,j\in A} f_{ij} - \sum_{i\in G,j\in A} f_{ji} = \begin{cases} v, & i = s \\ 0, & i = N-\{s,t\} \\ -v, & i = t \end{cases} \\ &\qquad 0 \le f_{ij} \le u_{ij},\ \forall (i,j) \in A \end{aligned}$$

여기서 $G=(N,A)$는 마디수가 $|N| = n$이고, 호의 수가 $|A| = m$인 유방향(directed) 네트워크라고 가정한다.

위의 $G=(N,A)$에서 마디집합 $N$을 $s$와 $t$를 중심으로 상호배반(mutually exclusive)인 부분집합 $X$와 $\overline{X}$로 분리했을 때 이 두 부분을 연결하는 호의 집합을 절단(cut) $(X,\overline{X})$라고 한다. 즉, $(X,\overline{X}) = \{(x,y) | (x,y)\in A, x\in X, y\in\overline{X}\}$, 단, $s\in X$, $t\in\overline{X}$, $X\cap\overline{X}=\phi$, $X\cup\overline{X}=N$. 여기서 절단용량(capacity of cut) $U(X,\overline{X})$는 $X$에서 $\overline{X}$로 향하는 모든 호의 용량상한값의 합에서 $\overline{X}$에서 $X$로 향하는 모든 호의 용량하한값의 합을 뺀 것으로 정의된다(Hamacher *et al.*, 1984). 이때 시발점에서 종착점까지의 최대유통량은 시발점과 종착점을 분리하는 절단 $(X,\overline{X})$의 최소용량(minimum cut)과 같다는 최대유통 최소절단정리(maximum flow minimum cut theorem)가 알려져 있다(Ahuja *et al.*, 1993; Murty, 1992).

본 연구에서는 절단 $(X,\overline{X})$을 구성하는 호들의 집합을 편의상 $D$라고 나타낸다. 그리고 호집합 $D$에 대한 절단용량을 $U(D)$, 호집합 $D$를 구성하고 있는 $q$개의 호를 $D^q$, 집합 $D$를 구성하고 있는 $q$개의 호들 중 용량이 큰 순서로 $k$개의 호로 구성된 집합을 $D_k^q$로 나타낸다.

이러한 최소절단을, 절단용량이 적은 것부터 큰 순서대로, 오름차순으로, 나열하는 해법에 대하여 Hamacher 외 2인이, $K$가 구하고자 하는 최소절단의 개수일 때, $O(Kn^4)$의 계산량(computation time)을 갖는 나열(ranking) 해법을 제시하였다(Hamacher *et al.*, 1984-b). 이들은 절단을 $(X,\overline{X}) = \{a\in A : t(a)\in X, h(a)\in\overline{X}\}$ (단, $t(a)$는 호 $a$의 시작마디, $h(a)$는 호 $a$의 도착마디)로 나타내고, 마디 $x$에 대한 특성변수 $z_x$를 다음과 같이 정의하여 절단집합을 표현하였다.

$$\text{특성변수} \quad z_x = \begin{cases} 1 & \text{if } x\in X \\ 0 & \text{if } x\notin X \end{cases} \text{(단, } z_s = 1,\ z_t = 0)$$

절단에 속한 호들의 용량의 합은 $U(X,\overline{X}) = \sum_{a\in A} u_a z_{t(a)} (1 - z_{h(a)})$로 나타내어, 최소절단문제가 이진 이차계획법문제(binary quadratic programming problem)인 $\min\{\sum_{a\in A} u_a z_{t(a)}$

$(1-z_{M(a)})$: $z\in D\}$, (단, $D=\{z\in\{0,1\}^n: z_s=1,\ z_t=0\}$)으로 표현될 수 있음을 밝혔다. 이들은 이 식을 다수 번째의 해를 나열하는 일반적인 Lawler와 Murty의 알고리즘에 적용하여 다수 번째의 절단을 오름차순으로 나열하는 해법을 제시하였다(Hamacher *et al.*, 1984-a, 1984-b; Lawler, 1972; Murty, 1968).

본 연구에서는 Hamacher 외 2인의 절단용량의 오름차순 나열해법의 개념을 이용하여 절단기수의 오름차순 나열 알고리즘을 제시하고자 한다. 절단기수(cardinality of cut)란 절단에 속한 호의 개수를 의미한다. 최대유통문제에서 호용량(arc capacity)의 상한을 0 또는 1로 놓고 문제를 푼 Even(Even S., 1979)의 0-1 최대유통문제와 같이 네트워크를 구성하는 모든 호의 용량을 1로 놓은 상태에서 Hamacher 외 2인의 절단용량에 대한 나열 알고리즘을 적용하면 절단기수에 대한 나열이 가능하게 된다. 이를 위해 계산법은 다음과 같다.

절단기수의 나열해법 (Ranking method for cardinality cuts)
단계 0 [초기화]
(0-1) 주어진 네트워크에서 모든 호의 용량상한을 1로 설정한다.
(0-2) 시발점과 종착점에 대해 $z_s=1$, $z_t=0$으로 설정한다.
(0-3) 시발점과 종착점을 제외한 다른 모든 마디에 대하여 절단 특성변수 $z$의 값을 고정하지 않은 0-1최대유통문제를 푼다. 이때 나온 절단 특성변수 $z_i$, $i\in N\setminus\{s,t\}$와 이때의 최대유통량을 LIST에 넣는다.
(0-4) $k=1$로 설정한다.

단계 1 [$k$ 번째의 해를 출력]
LIST에서 가장 최소의 최대유통값을 가지는 해를 꺼내어 출력한다. $z_x^{(k)}=(z_1^{(k)}, z_2^{(k)}, \cdots, z_n^{(k)})$

단계 2 [검사]
만일 $k=K$이면 끝낸다.

단계 3 [LIST를 확장]
(3-1) Lawler와 Murty의 알고리즘의 LIST 확장방법에 의해 문제를 분지한다.

분지할 때, 만일 $z^{(k)}$가 $z_1, z_2, \cdots, z_l$의 $l$개의 마디의 값을 0 또는 1로 고정하여 얻어진 값이라고 가정해도 일반성을 잃지 않는다($k=1$일 경우 $l=0$). 즉, $n-l$개의 마디는 시발점 또는 종착점쪽의 마디집합에 속함이 결정되지 않은 자유마디에 속하는 마디의 수이다. $k$번째의 절단에서 $(k+1)$번째의 절단을 구하기 위해서는 다음과 같은 부분문제로 분지시킨다.

(1) $z_{l-1}=1-z_{l+1}^{(k)}$

(2) $z_{l+1}=z_{l+1}^{(k)}$, $z_{l+2}=1-z_{l+2}^{(k)}$

(3) $z_{l+1}=z_{l+1}^{(k)}$, $z_{l+2}=z_{l+2}^{(k)}$, $z_{l-3}=1-z_{l+3}^{(k)}$
$\vdots$ $\vdots$ $\vdots$
$(n-l)$ $z_{l+1}=z_{l+1}^{(k)}$, $z_{l+2}=z_{l+2}^{(k)}$, $\cdots$, $z_n=1-z_n^{(k)}$

(3-2) LIST에 새로 들어온 부분문제에 대해 다음의 문제를 푼다.
주어진 네트워크에서 호의 용량상한을 1로 설정한다.
$l$개의 마디 중에서 $z_i=1$인 마디의 경우는 $S$에, $z_i=0$인 마디의 경우는 $T$의 집합에 각각 넣는다.

$S$에 속한 마디 $a$에 대하여,
마디 $a$로 들어오는 모든 호의 용량상한을 0으로 설정한다.
초시작마디(super source : $SS$)에서 마디 $a$로 가는 인공호(artificial arc)를 연결하고 호의 용량을 $\infty$로 설정한다.

$T$에 속한 마디 $b$에 대하여,
마디 $b$에서 나가는 모든 호의 용량상한을 0으로 놓는다.
마디 $b$에서 초종착마디(super sink: $ST$)로 가는 인공호를 연결하고 호의 용량을 $\infty$로 놓는다.
초시작마디($SS$)에서 초종착마디($ST$)로 가는 최대유통문제를 푼다.
결과로 얻어진 최소절단의 해인 특성변수 $z$와 이때의 최소절단값을 LIST에 넣는다.

(3-3) 단계 1로 간다.

위 해법의 계산의 복잡도(time complexity)는 단계 3에서 발생되는 매회 $(n-1)$개의 부분문제에 대하여 각각 0-1 최대유통문제를 풀 때 필요한 계산량 $O(n^{1/2}m)$이 소요되기 때문에(Ahuja *et al.*, 1993), 구하고자 하는 $K$가 정해져 있는 경우에 $O(Kn^{3/2}m)$만에 $K$번째의 절단기수를 구할 수 있다. 그러나 이미 $K-1$번까지의 절단기수의 집합을 유지하고 있을 때, 다음 번의 절단기수를 구하는 계산량은 $O(n^{3/2}m)$이 된다.

## 3. $k$-치명호를 결정하는 방법

최대유통문제에서 $k$-치명호를 찾는 문제에 대하여 기존의 연구에서 밝혀진 사실은, $k$-치명호를 찾는 문제가 NP-hard(Wood, 1993)라는 것과 $k$-치명호는 1-치명호를 순차적으로 $k$번 반복적용하여 구한 것과 같지 않다는 것이다(Malik *et al.*, 1989).

$k$-치명호를 찾는 문제가 NP-hard라는 사실은 Wood(1993)에 의하여 증명되었는데, 그는 $k$-치명호를 찾는 문제의 의사결정문제(decision problem)가 군집(clique) 문제로 변환됨을 보임으로써 $k$-치명호를 찾는 문제가 NP-hard임을 증명하였다. $k$-치명호를 찾는 문제가 1-치명호를 찾는 문제를 순차적으로 $k$번 반복적용하여 구한 것과 같지 않다는 것은 $k$-치명호를 찾는 문제가

단순히 1-치명호 찾는 문제를 $k$회 반복 적용하여 구할 수 없다는 사실을 말해 준다. Wood는 최대유통문제에서 $k$-치명호가 임의의 절단 중에서 절단에 속한 호들 중 용량이 큰 $k$개의 호를 제외한, 호들에 대한 절단용량이 가장 작은 호들로 구성되는 절단에서 존재함을 밝혔다(Wood, 1993).

그러므로 $k$-치명호를 구하기 위해서는, 절단용량을 최소절단으로부터 오름차순으로 나열하여 절단을 구하여, 구하여진 절단을 구성하고 있는 호들 중에서 $k$개를 제외한 호의 용량이 가장 작아지는 것을 찾아내어, 찾아진 절단의 용량이 큰 $k$개의 호들을 $k$-치명호로 선정하는 방법을 생각해 볼 수 있다.

그러나 이와 같은 접근법은 모든 절단을 나열하기 이전에 종료할 수 있는 종료조건이 존재하지 않는다. 그런데 일반적으로 주어진 네트워크에서 구별가능한 절단의 최대개수는 최대 $2^{|N|}$개, 이때 $|N|$은 마디의 집합의 개수, 정도인 것으로 알려져 있다. 그러므로 모든 절단을 나열하여야만 $k$-치명호를 구할 수 있는 경우는 모든 $k$-치명호를 구할 수 있는 장점 이외에는 해법의 속도의 측면에서 바람직하지 않을 수 있다.

그러므로 본 연구에서는 절단에 속하는 호의 개수를 오름차순으로 나열하는 해법에 대하여 고려하고자 한다. 즉, 절단기수(cardinality of cut)의 오름차순으로 절단을 나열하여 절단기수 중에서 용량이 큰 $k$개의 호를 제외한 호를 통해 흘려줄 수 있는 양이 가장 작은 호를 찾아내는 방법이다. 이 방법은 절단기수가 증가함에 따라 용량이 큰 $k$개의 호를 제외한 호를 통해 흘려줄 수 있는 용량의 합이 증가함에 반하여, 주어진 네트워크에서 최소한 흘려줄 수 있을 것으로 예상되는 용량은 증가하는 성질을 가질 수 있다는 사실에 착안한 것이다. 즉, 이 개념을 통해 모든 절단을 나열하지 않고도 $k$-치명호를 구할 수 있는 방법이 있을 수 있다.

제시하고자 하는 해법의 종료조건을 알아보기 위해 먼저 $q$를 고려중인 절단기수(a number of arcs consisting of cut : cardinality cut) $u_{(i)}$를, 절단 $D$에 속한 호들을 용량의 오름차순으로 $i$번째 호의 용량상한값이라고 하자. 그러면 절단 $D$를 구성하고 있는 $q$개의 호들 중에서 용량이 큰 $k$개의 호로 구성된 호집합 $D_k^q$의 용량 $U(D_k^q) = \sum_{i=q-k+1}^{q} u_{(i)}$이 된다. 반면에 절단 $D$를 구성하고 있는 $q$개의 호들 중에서 용량이 큰 $k$개의 호를 제외한 호들의 용량의 합 $U(D^q \setminus D_k^q) = \sum_{i=1}^{q-k} u_{(i)},\ i \in D$가 된다. 이를 현재 구하여진 절단 $D$의 절단기수의 잔여용량(residual capacity of the current cardinality cut $D$) $CCR(D_k^q)$라고 하자. 이 절단기수의 잔여용량을 이용하여 현재 고려하고 있는 절단기수보다 큰 기수에 나타나는 절단기수의 잔여용량을 계산할 수 있는데, 절단기수의 수가 $q$일 때 다음 번 절단기수의 잔여용량 $U(D^q \setminus D_k^q) = \sum_{i=1}^{q-k} u_{(i)},\ i \in A$가 된다. 이를 주어진 네트워크에서의 절단기수의 예상 잔여용량 $ER^q(A)$라고 정의하자.

위와 같은 방법으로 절단기수를 나열하면서 $k$-치명호를 구하는 데 사용할 수 있는 다음과 같은 보조정리를 도출할 수 있다.

[ 보조정리 1 ] 주어진 네트워크에서 절단기수의 예상 잔여용량 $ER^q(A)$는 $q$가 증가함에 따라 증가한다.

(증명) $q_1$과 $q_2$인 절단기수의 수가 주어졌다고 가정하자(단, $q_1 < q_2$). 그러면 절단기수 $q_2$의 예상 잔여용량 $ER^{q_2}(A)$는 $\sum_{i=1}^{q_2-k} u_{(i)},\ i \in A$이 되고, 절단기수 $q_1$의 예상 잔여용량 $ER^{q_1}(A)$는 $\sum_{i=1}^{q_1-k} u_{(i)},\ i \in A$가 된다. 이때 두 값의 차는 $q_1 < q_2$ 이므로 $\sum_{i=1}^{q_2-q_1} u_{(i)} > 0,\ i \in A$가 되어 항상 양의 값을 갖게 된다. 따라서 절단기수$q$의 예상 잔여용량 $ER^q(A)$은 절단기수의 개수 $q$가 증가함에 따라 항상 증가한다.

따라서 절단기수의 수가 $q$일 때 현재의 절단기수의 잔여용량 $CCR(D_k^q)$보다 기수의 예상 잔여용량 $ER^q(A)$의 값이 더 커진다면, 그 이후의 절단에서는 $k$-치명호가 존재하지 않게 되므로 알고리즘을 종료하면 된다. 즉, $U(D^q \setminus D_k^q) < U(D^q \setminus D_k^q)$이 성립할 때 알고리즘은 종료된다. 이를 보이면 다음의 정리와 같다.

[ 정리 2 ] 절단기수의 오름차순으로 절단들을 나열하고 있다고 가정하자. 그리고 현재 고려중인 절단 $D$의 기수를 $q$라고 하자. 만일 현재까지 고려한 모든 절단들의 기수의 잔여용량의 최소값이 기수 $q$의 예상 잔여용량의 값보다 더 작아진다면, 현재의 절단 이후의 절단에서는 $k$-치명호가 존재하지 않는다.

(증명) 절단 $D$의 기수 $q$의 잔여용량 $CCR(D_k^q)$은 기수 $q$의 예상 잔여용량 $ER^q(A)$보다 항상 크거나 같다. 왜냐하면 $CCR(D_k^q) = \sum_{i=1}^{q-k} u_{(i)},\ i \in D$이고, $ER^q(A) = \sum_{i=1}^{q-k} u_{(i)},\ i \in A$인데, 두 값의 차이는 고려하고 있는 호의 집합이 $CCR(D_k^q)$의 경우는 절단을 구성하는 대상이 되는 호가 $ER^q(A)$의 경우는 네트워크의 모든 호이기 때문이다.

가정에 의해, 현재 구하여진 절단 이전의 절단에서는 절단기수가 $q$보다 크지 않다. 그러므로 기수가 $q$ 이하인 경우의 절단의 기수의 잔여용량의 최소값이 현재까지 주어진 네트워크를 통해 흘려줄 수 있는 양이 가장 큰 절단에서 $k$개를 제외한 호를 통해 흘려보낼 수 있는 양의 최소값이다. 그리고 [보조정리]에 의해 현재의 절단기수 $q$의 예상 잔여용량은 이후의 절단기수의 증가를 통해서 얻을 수 있는 절단기수의 예상 잔여용량값의 최소값이다. 그러므로 $q$보다 큰 기수의 잔여용량은 예상잔여용량보다 작을 수 없으므로 이후의 절단에 대해 $k$-치명호를 구하는 과정에서 고려할 필요가 없다.

위의 [ 정리 2 ]를 이용하여 기수의 잔여용량과 기수의 예상 잔여용량의 관계를 통한 종료조건을 사용하여 $k$-치명호를 찾는 해법을 제시하면 다음과 같다.

모든 $k$-치명호 계산법(A method for determining all the $k$-MVA)

사용지수(Notation)

$p$ : 계산횟수를 나타내는 지수

$q$ : 절단기수의 수

$k$ : 구하고자 하는 치명호의 갯수

단계 0 : [초기화]

네트워크를 구성하는 호의 지수(index)를 호 용량의 오름차순으로 정렬한다.

즉, $u_{(1)} \le u_{(2)} \le \cdots \le u_{(m)}$.

절단의 잔여용량 = $\infty$.

$p = 1$.

첫번째 절단기수를 구하여 이를 $q$라고 놓는다.

단계 1 : [자명해 판정]

만일 $q < k$ 라면,

$k$-치명호는 $q$개의 절단기수에 속한 호와 $k-q$개의 임의의 호로 구성된 모든 조합에 대해 $k$-치명호로 선정한다. 단계 4로 간다.

만일 $q = k$ 일 경우,

$k$-치명호는 절단기수에 속한 호로 구성된다. 단계 4로 간다.

아니면 (즉, $q > k$ 인 경우),

만일 이전단계의 $q$가 $k$와 같았으면, 끝낸다.

아니면, 단계 2로 간다.

단계 2 : [절단의 잔여용량계산]

절단의 잔여용량 = 절단에 속한 호 중에서 첫번째 호부터 $q-k$번째 호까지의 용량의 합.

만일 절단의 잔여용량이 기존에 구해진 잔여용량보다 작으면,

$k$-치명호 후보집합 = 현재의 절단기수에서 지수의 순서가 높은 $k$개의 호

만일 절단의 잔여용량이 기존에 구해진 잔여용량과 같으면,

$k$-치명호 후보집합에 현재의 절단기수에서 지수의 순서가 높은 $k$개의 호를 추가한다.

단계 3 : [최적판정]

만일 (기수 $q$의 예상 잔여용량의 합) > (현재 고려중인 절단의 잔여용량)이면 현재의 후보집합이 $k$-치명호가 됨. 끝.

아니면, 단계 4로 간다.

단계 4 : [다음 번 최소절단을 구함]

$p = p + 1$.

$p$번째 절단기수를 구한다.

단계 2로 간다.

위의 계산법에서 절단 $D$의 기수 $q$의 잔여용량 $CCR(D_q^*)$과 현재의 절단기수의 잔여용량 $CCR(D_q^*)$의 계산의 효율을 위해 단계 0에서 용량상한의 내림차순으로 정렬한다. 그리고 이 해법을 $k$-치명호 중에서 하나만 계산하기 위해서는 단계 1의 ($q < k$) 인 경우에 대해서 $q$개의 절단기수에 속한 호와 $k-q$개의 임의의 호로 구성되는 조합 중의 임의의 하나를 $k$-치명호로 선정하고 끝내면 된다. 마찬가지로 단계 1에서 ($q = k$)인 경우에 대하여 단계 4로 가서 다음의 절단의 고려 없이 바로 끝내면 되게 수정하면 된다.

## 4. 예제

다음 <그림 1>에서 시발점을 1, 종착점을 6으로 하는 최대유통문제에서 2-치명호를 결정하는 문제를 고려해 보자.

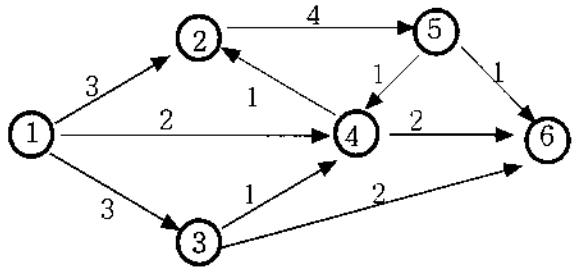


그림 1. 최대유통문제의 예제(Murty, 1992).

• 절단기수의 나열방법

먼저 절단기수의 나열을 위해 <그림 1>의 네트워크에서 호의 용량상한을 1로 설정한다.

$z_1 = 1$, $z_6 = 0$으로 설정한 다음 0-1 최대유통문제를 푼다.

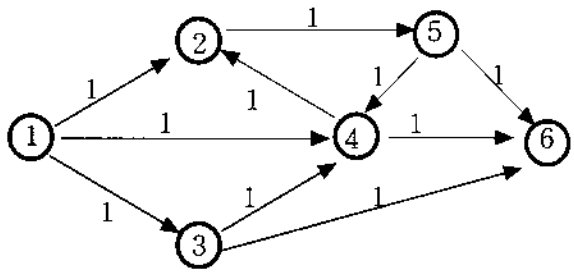


그림 2. 상한이 1인 네트워크(부분문제 0).

그 결과는 다음과 같다. 이 결과를 가지고 나열해법에 대해 살펴보면 다음과 같다.

표 1. 초기 LIST

| LIST | 절단기수 | 절단기수상의 호 |
|---|---|---|
| (부분문제 0) | 3 | (1,2)(1,3)(1,4) |

(1회)

$k=1$ 일 때의 절단기수는 3이며, 이때의 절단을 구성하는 호는 (1,2)(1,3)(1,4)이다. 부분문제 0을 LIST에서 제거하고 부분문제 1, 2, 3, 4로 다음과 같이 분지시킨다.

(부분문제 1) $z_2=1$

(부분문제 2) $z_2=0,\ z_3=1$

(부분문제 3) $z_2=0,\ z_3=0,\ z_4=1$

(부분문제 4) $z_2=0,\ z_3=0,\ z_4=0,\ z_5=1$

<그림 2>의 (부분문제 0)을 부분문제로 나누면 다음과 같

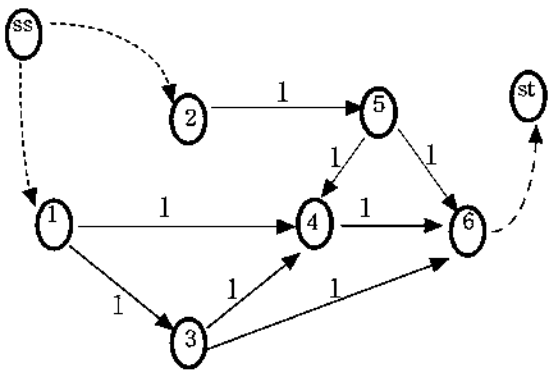


그림 3. (a) 부분문제 1.

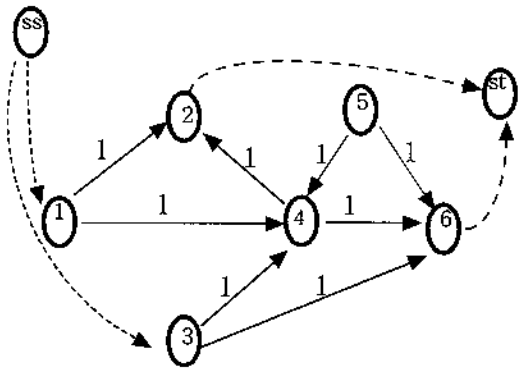


그림 3. (b) 부분문제 2.

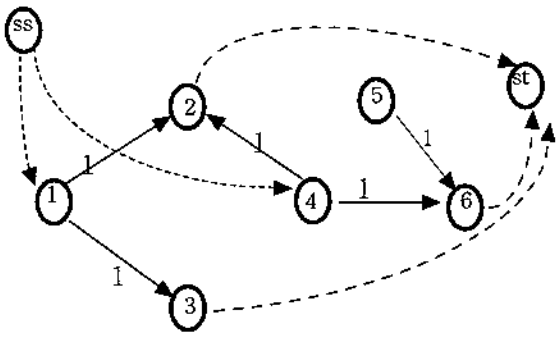


그림 3. (c) 부분문제 3.

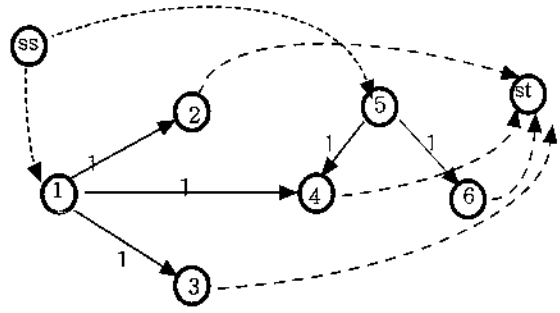


그림 3. (d) 부분문제 4.

다. 이때 점선으로 표시한 초마디( $ss$, $st$) 인공호에서 나가는 호를 의미하며, 이 인공호의 용량상한은 무한대이다.

(2회)

$k=2$ 일 때의 LIST는 다음과 같다.

표 2. 2회의 LIST의 내용

| LIST | 절단기수 | 절단기수상의 호 |
|---|---|---|
| (부분문제 1) | 3 | (1,3)(1,4)(2,5) |
| (부분문제 2) | 4 | (1,2)(1,4)(3,4)(3,6) |
| (부분문제 3) | 4 | (1,2)(1,3)(4,2)(4,6) |
| (부분문제 4) | 5 | (1,2)(1,3)(1,4)(5,4)(5,6) |

$k=2$ 일 때의 절단기수는 3이며, 이때의 절단을 구성하는 호는 (1,3)(1,4)(2,5)이다.

그리고 LIST에서 부분문제 1을 제거하고 부분문제 5,6,7로 다음과 같이 분지시킨다.

(부분문제 5) $z_3=1$

(부분문제 6) $z_3=0,\ z_4=1$

(부분문제 7) $z_3=0,\ z_4=0,\ z_5=1$

(3회)

$k=3$일 때의 LIST는 다음과 같다.

$k=3$ 일 때의 절단기수는 3이며, 이때의 절단을 구성하는

표 3. 3회의 LIST의 내용

| LIST | 절단기수 | 절단기수상의 호 |
|---|---|---|
| (부분문제 2) | 4 | (1,2)(1,4)(3,4)(3,6) |
| (부분문제 3) | 4 | (1,2)(1,3)(4,2)(4,6) |
| (부분문제 4) | 5 | (1,2)(1,3)(1,4)(5,4)(5,6) |
| (부분문제 5) | 3 | (1,4)(2,5)(4,6) |
| (부분문제 6) | 3 | (1,3)(2,5)(4,6) |
| (부분문제 7) | 4 | (1,3)(1,4)(5,4)(5,6) |

호는 (1,4)(2,5)(4,6)이다. LIST에서 부분문제 5를 제거하고 부분문제 8, 9로 분지시킨다. 이러한 절차를 구하고자 하는 $K$개만큼 반복한다. 전체 계산과정을 표로 정리하면 다음 <표 4>와 같다.

표 4. 절단기수의 순위

| LIST | 절단기수 | 절단상의 호 | 분지되는 부분문제 | 순서 |
|---|---|---|---|---|
| (부분문제 0) | 3 | (1,2)(1,3)(1,4) | 부분문제 1,2,3,4 | 1 |
| (부분문제 1) | 3 | (1,3)(1,4)(2,5) | 부분문제 5,6,7 | 2 |
| (부분문제 2) | 4 | (1,2)(1,4)(3,4)(3,6) | 부분문제 11,12 | 7 |
| (부분문제 3) | 4 | (1,2)(1,3)(4,2)(4,6) | 부분문제 13 | 8 |
| (부분문제 4) | 5 | (1,2)(1,3)(1,4)(5,4)(5,6) | | 12 |
| (부분문제 5) | 3 | (1,4)(2,5)(4,6) | 부분문제 8,9 | 3 |
| (부분문제 6) | 3 | (1,3)(2,5)(4,6) | 부분문제 10 | 4 |
| (부분문제 7) | 4 | (1,3)(1,4)(5,4)(5,6) | | 9 |
| (부분문제 8) | 4 | (1,4)(2,5)(3,4)(3,6) | 부분문제 14 | 10 |
| (부분문제 9) | 3 | (3,6)(4,6)(5,6) | | 5 |
| (부분문제 10) | 3 | (1,3)(4,6)(5,6) | | 6 |
| (부분문제 11) | 4 | (1,2)(4,2)(3,6)(4,6) | 부분문제 15 | 11 |
| (부분문제 12) | 5 | (1,2)(1,4)(3,4)(3,6)(5,6) | | 13 |
| (부분문제 13) | 5 | (1,2)(1,3)(4,2)(4,6)(5,6) | | 14 |
| (부분문제 14) | 5 | (1,4)(3,4)(3,6)(5,4)(5,6) | | 15 |
| (부분문제 15) | 5 | (1,2)(3,6)(4,2)(4,6)(5,6) | | 16 |

(2회)에서 구한 부분문제 1, 2, 3, 4 의 절단기수는 각각 3, 4, 4, 5이다. 따라서 현재의 LIST에 있는 부분문제 1, 2, 3, 4 중에서 절단기수의 수가 가장 작은 (부분문제 1)이 선택되고, 이 (부분문제1)은 LIST에서 제거된다. 이때 (부분문제 1)이 제거되면서 (부분문제 1)로부터 부분문제 5, 6, 7이 분지되어 LIST에 추가된다. (3회)에서 LIST에 존재하는 부분문제는 부분문제 2,3,4,5,6,7이며, 이 부분문제에 대한 절단기수의 수는 4, 4, 5, 3, 3, 4이다. 따라서 절단기수의 수가 가장 작은 (부분문제 5)가 LIST에서 선택되고, 이 (부분문제 5)가 제거되면서 (부분문제 5)로부터 부분문제 8, 9가 분지되어 LIST에 추가된다. (4회)에서 LIST에 존재하는 부분문제는 부분문제 2, 3, 4, 6, 7, 8, 9이며, 이 부분문제에 대한 절단기수는 4, 4, 5, 3, 4, 4, 3이므로 (부분문제 6)이 LIST에서 선택되어 제거되고 다시 (부분문제 6)으로 부터 (부분문제 10)이 분지되어 LIST에 추가된다. 이를 반복하여 절단기수의 순위를 정한 결과가 위의 <표 4>와 같다.

▪ 2-치명호 해법

절단의 잔여용량은 절단에 속한 호 중 용량이 큰 2개의 호를 제외한 호들의 절단용량의 합이므로 (부분문제 0)에 대한 잔여용량은 호 (1,4)의 용량인 2가 되고, (부분문제 5)에 대한 잔여용량은 호 (4,6)의 용량인 2가 된다. 여기서 절단기수가 3인 경우 절단에 속한 호 중 용량이 큰 2개의 호를 제외하면 하나의 호에 대한 용량상한만이 고려된다. 주어진 네트워크에서 호의 용량을 오름차순으로 정렬하면 1, 1, 1, 1, 2, 2, 2, 3, 3, 4이므로 절단기수가 3인 경우 $q(3)$-$k(2)$ = 1 이 되어, 예상 잔여용량은 1이 된다. 그러나 일곱 번째 최소절단인 (부분문제 2)를 고려하게 되면 절단기수가 4이기 때문에 2개의 호에 대한 용량상한을 고려해야 하며, 이때의 최소용량 $q(4)$-$k(2)$ = 2이므로 호를 통해 흐를 수 있는 예상 잔여용량 또한 2가 된다. 따라서 여덟 번째 최소절단인 (부분문제 2)를 고려하기 이전에 절단의 잔여용량이 1인 (부분문제 9)와 (부분문제 10)의 절단에서 용량이 큰 두 개의 호 (3,6)과 (4,6) 그리고 (1,3)과 (4,6)이 2-치명호로 결정되게 된다.

위와 같은 절차를 반복 적용하면서 종료조건에 부합되게 되면 계산이 끝나게 된다.

계산과정을 정리하면 다음 <표 5>와 같다.

## 5. 결 론

본 연구는 $k$개의 호들로 이루어진 호집합을 동시에 제거함으로써 시발점에서 종착점까지의 최대유통량을 가장 많이 줄이는 호집합을 찾는, $k$-치명호문제의 모든 해를 구하는 해법을 제시하기 위한 것이다.

이를 위해 본 연구에서는 최대유통문제에서 오름차순으로 절단용량을 나열한 다음, 이 절단에 속한 호 중 용량이 큰 $k$개의 호를 제외한 호들의 절단용량이 가장 작은 호에 해당하는 절단집합에 $k$-치명호가 존재한다는 사실을 이용하여 $k$-치명호를 구하는 방법을 제시하였다.

절단기수의 나열방법은 Hamacher 외 2인의 절단용량에 대한 나열 알고리즘을 변형하여 이용하였고, 현재의 절단기수의 잔여용량과 기수의 예상 잔여용량의 값의 비교를 통해 알고리즘

표 5. 2-치명호를 구하는 해법의 계산절차

| 순위 | LIST | 절단을 구성하는 호 | 호의 용량 | 현재 절단의 잔여용량 | 기수의 예상 잔여용량 |
|---|---|---|---|---|---|
| 1 | (부분문제 0) | (1,2)(1,3)(1,4) | 3 3 ② | 2 | 1 |
| 2 | (부분문제 1) | (1,3)(1,4)(2,5) | 3 ② 4 | 2 | 1 |
| 3 | (부분문제 5) | (1,4)(2,5)(4,6) | ② 4 2 | 2 | 1 |
| 4 | (부분문제 6) | (1,3)(2,5)(4,6) | 3 4 ② | 2 | 1 |
| 5 | (부분문제 9) | (3,6)(4,6)(5,6) | 2 2 ① | 1 | 1 |
| 6 | (부분문제 10) | (1,3)(4,6)(5,6) | 3 2 ① | 1 | 1 |
| 7 | (부분문제 2) | (1,2)(1,4)(3,4)(3,6) | 3 3 ① ② | 3(1+2) | 2(1+1) 종료 |
| 8 | (부분문제 3) | (1,2)(1,3)(4,2)(4,6) | 3 3 ① ② | 3(1+2) | |
| 9 | (부분문제 7) | (1,3)(1,4)(5,4)(5,6) | 3 2 ① ① | 2(1+1) | |
| 10 | (부분문제 8) | (1,4)(2,5)(3,4)(3,6) | 2 4 ① ② | 3(1+2) | |
| 11 | (부분문제 11) | (1,2)(4,2)(3,6)(4,6) | 3 ① ② 2 | 3(1+2) | |
| 12 | (부분문제 4) | (1,2)(1,3)(1,4)(5,4)(5,6) | 3 3 2 ① ① | 2(1+1) | |
| 13 | (부분문제 12) | (1,2)(1,4)(3,4)(3,6)(5,6) | 3 2 ① 2 ① | 2(1+1) | |
| 14 | (부분문제 13) | (1,2)(1,3)(4,2)(4,6)(5,6) | 3 3 ① 2 ① | 2(1+1) | |
| 15 | (부분문제 14) | (1,4)(3,4)(3,6)(5,4)(5,6) | 2 ① 2 ① 1 | 2(1+1) | |
| 16 | (부분문제 15) | (1,2)(3,6)(4,2)(4,6)(5,6) | 3 2 ① 2 ① | 2(1+1) | |

이 종료되는 종료조건도 제시하였다.

이 논문에서 제시한 해법은, 기존의 방법과 다르게, $k$-치명호에 대한 대안해가 존재할 경우 이에 대한 모든 대안해를 제시해 줄 수 있는 해법이다.

## 참고문헌


박순달 (1997), *경영과학*, 3판, 민영사.

안재근 (1997), *네트워크문제에서의 치명호에 대한 연구*, 서울대학교 박사학위논문.

Ahuja, R. K., Magnanti, T. I. and Orlin, J. B. (1993), *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall.

Bar-Noy, A.,Khuller, S. and Schieber, B. (1995), The complexity of finding most vital arcs and nodes, *Univ. of Maryland Technical Reports*, CS-TR-3539.

Corley, H. W. and Sha, D. Y. (1982), Most vital links and nodes in weighted networks, *O. R. Letters*, 1(4), 157-160.

Even, S. (1979), *Graph Algorithm*, Computer Science Press.

Hamacher, H. W., Picard, J. C. and Queyranne, M. (1984-a), On finding the K best cuts in a network, *O. R. Letters*, 2(6), 303-305.

Hamacher, H. W., Pacard, J. C. and Queyranne, M. (1984-b), Ranking the cuts and cut-sets of a network, *Annals of Discrete Mathematics*, 19, 183-200.

Lawler, E. L. (1972), A procedure for computing the $K$ best solutions to discrete optimization problems and its application to the shortest path, *Management Science*, 18(7), 401-405.

Lubore, S. H., Ratliff, H. D. and Sicillia, G. T. (1971), Determining the most vital link in a flow network, *N. R. L. Q.*, 18(4), 497-502.

Malik, K., Mittal, A. K. and Gupta, S. K. (1989), The k most vital arcs in the shortest path problem, *O. R. Letters*, 8, 223-227.

Murty, K. G. (1992), *Network Programming*, Prentice-Hall.

Murty, K. G. (1986), An algorithm for ranking all the assignments in order of increasing cost, *Operations Research*, 16, 682-687.

Ratliff, H. D., Lubore, S. H. and Sicillia, G. T. (1975), Finding the n most vital links in a flow network, *Management Science*, 21(5), 531-539.

Wollmer, R. (1964), Removing arcs from a network, *Operations Research*, 934-940.

Wood, R. K. (1993), Deterministic network interdiction, *Math. Comput. Modeling*, 17(2), 1-18.