韓國經營科學會誌
第26卷 第4號
2001年 12月


# 분할 루팅이 허용되는 링의 용량결정문제에 대한 개선된 해법*

명영수** · 김후곤***

# A Faster Algorithm for the Ring Loading problem with Demand Splitting

Young-Soo Myung** · Hu-Gon Kim***

■ Abstract ■

In the ring loading problem with demand splitting, traffic demands are given for each pair of nodes in an undirected ring network and a flow is routed in either of the two directions, clockwise and counter-clockwise. The load of a link is the sum of the flows routed through the link and the objective of the problem is to minimize the maximum load on the ring. The fastest algorithm to date is Myung, Kim and Tcha's algorithm that runs in $O(n|K|)$ time where $n$ is the number of nodes and $K$ is the index set of the origin-destination pairs of nodes having flow traffic demands. Here we develop an algorithm for the ring loading problem with demand splitting that improves the rerouting step of Myung, Kim and Tcha's algorithm and runs in $O(\min\{n|K|, n^2\})$ time.

Keyword : Ring loading problem with demand splitting ; Algorithm.

## 1. 서 론

링의 용량결정문제(RLP, Ring Loading Problem)는 다음과 같이 정의된다. 노드의 집합 $V=\{1,2,\cdots,n\}$와 링크의 집합 $L=\{(1,2),(2,3),\cdots,(n-1,n),(n,1)\}$로 이루어진 방향성이 없는 링 모양의 그래프 $R=(V,L)$이 주어져 있다고 가정하자. 링크 $(i,i+1)$과 링크 $i$는 동일한 링크를 지칭하는

논문접수일 : 2001년 6월 11일 논문게재확정일 : 2001년 11월 28일
* 이 논문은 2001년도 한국학술진흥재단의 연구비에 의하여 연구되었음.(KRF-2001-041-C00309)
** 단국대학교 경상학부
*** 경성대학교 경영학부

것으로 정의한다. 다만 링크 $(n,1)$이 링크 $n$으로 표시되는 것은 예외로 인정한다. $K$는 두 노드, 즉 노드 쌍들의 인덱스 집합이다. 각 노드 쌍 $k \in K$에 대하여, $r_k$의 통신 수요가 존재하고, $o(k)$와 $d(k)$ $(o(k)<d(k))$는 노드 쌍 $k$의 공급노드와 수요노드를 각각 표시하는 것으로 정의한다. $o(k)$와 $d(k)$ 사이의 흐름은 시계 방향과 반시계 방향의 양방향으로 보내질 수 있다. 다시 말해서 흐름이 $\{o(k), o(k)+1, \cdots, d(k)-1, d(k)\}$ 방향으로 움직이면 흐름이 시계 방향이라고 부르고 흐름이 $\{o(k), o(k)-1, \cdots, 1, n, \cdots, d(k)+1, d(k)\}$ 방향으로 움직이면 반시계 방향이라고 부른다. 링을 구성할 때는 모든 링크에 동일한 용량의 링크를 설치하여야 되기 때문에 링 구성에 필요한 용량은 링크 중에서 가장 많은 흐름이 통과하는 링크의 흐름량에 의하여 결정되게 된다. 링에서의 링크별 흐름의 양은 각 노드 쌍간의 통신 수요를 어떻게 분산해서 루팅(routing)하는가에 의하여 결정되므로 링의 용량결정문제는 링의 용량이 최소화 되게 해주는 루팅 방법을 찾는 문제가 된다.

양방향으로 루팅이 가능한 링에서는 두 노드간의 수요를 분할해서 동시에 양방향으로 전송할 수 있는 경우와 양방향 중 시계 방향이나 반시계 방향의 어느 한 방향으로만 전송해야 하는 경우로 나누어진다. 이러한 조건에 따라 링의 용량결정문제도 분할 루팅이 가능한 경우와 가능하지 않은 경우로 구분한다. 전자의 경우, 즉 분할 루팅이 허용되는 링에서의 용량결정문제를 RLPW(Ring Loading Problem With demand splitting)로 부르기로 한다. 경우에 따라서는 분할이 허용되는 경우도 전송은 꼭 정수 단위이어야 되는 경우도 있다. 왜냐하면 링크의 용량이 단위 용량의 정수 배로서만 설치될 수 있는 경우를 고려하기 때문이다. 분할이 허용되지 않는 경우 및 분할이 정수 단위로만 허용되는 문제에 대해서도 RLPW는 이 문제들의 완화문제(relaxation)가 되는 속성 때문에 RLPW의 해법은 나머지 문제들을 해결하는 데도 요긴하게 사용된다.

링의 용량결정문제는 양방향으로 운용하는 동기식 광전송망(SONET, Synchronous Optical Network)에서 주어진 통신 수요를 만족시키기 위해서 필요한 SONET링의 용량을 최소화하는 문제에 적용된다. SONET링에서는 광전송 장비인 ADM (Add-Drop Multiplexer)을 노드로 이를 연결하는 광케이블을 링크로 표현하는데, 전송이 단방향으로만 이루어지는 형태와 시계 방향과 반시계 방향의 양방향으로 이루어지는 형태의 두 가지가 있다. SONET링은 이를 구성하는 링크들이 모두 동일한 용량을 가져야 한다는 기술적 제약이 따르게 된다. 따라서 양방향으로 전송이 이루어지는 SONET링의 경우는 각 노드 쌍간의 통신 수요를 시계 방향과 반시계 방향으로 어떻게 루팅하느냐에 따라서 필요한 링의 용량이 달라지게 되므로, 링의 용량을 최소화하기 위한 최적의 루팅 방식을 구하는 링의 용량결정문제가 중요한 과제이다[4, 11]. 특히 SONET링에서는 수요의 단위가 일정 용량을 갖는 채널의 수로 나타나므로 링의 용량결정문제 중 분할이 허용되지 않거나 허용되더라도 정수 단위로 허용되는 문제가 응용대상이 된다.

SONET링에서 노드 쌍 $o(k)$와 $d(k)$에 통신 수요 $r_k$가 존재한다는 것은 $o(k)$에서 $d(k)$로의 통신 수요는 물론 $d(k)$에서 $o(k)$로의 통신 수요도 $r_k$임을 의미한다. 즉 양방향으로 동일한 통신 수요가 존재하는 것을 전제로 한다. 본 논문에서 다루는 링의 용량결정문제에서는 수요가 한쪽 방향으로만 존재하는 것처럼 가정하므로 SONET링의 용량결정문제와 차이가 나는 것으로 생각할 수 있다. 하지만, SONET링에서는 광케이블을 2개 또는 4개를 중첩하는 구조로 운용하기 때문에 본 논문에서 다루는 문제의 용량을 최소화하는 루팅 방식이 SONET링의 용량을 최소화시키는 해와 일치하게 된다. SONET링에 대한 자세한 설명과 링의 용량결정문제에 대한 응용에 대해서는 Cosares 등[4]과 Wu[11]의 연구를 참조하기 바란다.

RLPW는 이 문제 자체로도 응용성이 있지만 앞에서 언급한대로 분할이 허용되지 않는 경우 및 분

할이 정수 단위로만 허용되는 문제를 푸는 데도 요긴하게 사용되기 때문에 그동안 많은 연구가 이루어져 왔다. Cosares와 Saniee[5]는 휴리스틱을, Vachani 등[10]과 Dell'Amico 등[6]이 각각 $O(n^3)$의 계산 시간이 소요되는 최적해법을, Schrijver 등[9]이 $O(n^2|K|)$의 계산 시간이 필요한 최적해법을 개발하였고, Myung 등[7]이 이론적인 계산 시간에서는 가장 빠른 최적해법인 $O(n|K|)$의 계산 시간이 소요되는 해법을 개발하였다. 이들 최적해법들간의 특성 및 실제 계산능력에 대한 비교 분석이 명영수와 김후곤[2]에 의해서 연구되었다. 수요의 분할 처리가 허용되지 않는 경우를 살펴보면 Cosares와 Saniee[5] 및 Myung 등[7]이 휴리스틱을 제시하였다. 수요의 분할 처리가 허용되는 경우 중 정수 단위로만 분할이 허용되는 문제에 대해서는 Vachani 등[10]이 $O(n^3)$에 계산되는 최적해법을, Schrijver 등[9]은 노드간의 통신 수요가 1로 주어지는 경우에 대해서 $O(n^2|K|)$의 계산 시간이 필요한 해법을 제시하였고, Myung[1, 8]은 Myung 등[7]의 분할 루팅이 허용되는 문제의 최적해법을 이용하여 $O(n|K|)$ 계산 시간 내에 최적해를 구할 수 있는 해법을 제시하였다.

본 논문에서는 $O(\min\{n|K|, n^2\})$ 계산 시간 내에 RLPW의 최적해를 구할 수 있는 해법을 제시하고자 한다. 우리의 해법은 Myung 등[7]의 해법에서 재루팅(루팅 방법을 변경하는 것을 의미함) 방법을 개선한 것으로 이제까지 발표된 어떠한 해법보다도 더 빠르게 RLPW의 최적해를 구할 수 있다. 또한, SONET링의 설계에 응용성이 더욱 높은 정수 단위로만 분할이 허용되는 문제를 푸는 Myung[1, 8]의 해법에, Myung 등[7]의 해법 대신 여기서 제시된 해법을 사용한다면 $O(\min\{n|K|, n^2\})$에 정수 단위로만 분할이 허용되는 문제의 해를 구할 수 있게 된다. 본 논문은 다음과 같이 구성되어 있다. 2장에서는 사용된 기호를 정의하고 3장에서는 해법의 제시와 함께 계산 시간을 분석한다.

## 2. 용어의 정의

각 노드 쌍 $k\in K$에 대해 공급노드 $o(k)$로부터 수요노드 $d(k)$까지의 시계 방향으로의 경로 상에 존재하는 링크의 집합을 $L_k^+$로, 반시계 방향으로의 경로 상에 존재하는 링크의 집합을 $L_k^-$로 정의한다. 즉, $L_k^+=\{(i,i+1)\in L|o(k)\le i\le d(k)-1\}$이고 $L_k^-=L\backslash L_k^+$이다. 따라서 링크 $n=(n,1)$은 항상 모든 $k\in K$에 대하여 $L_k^-$에 포함된다. 또한 모든 링크 $l\in L$에 대해 링크 $l$을 시계 방향의 경로에 포함하는 노드 쌍들의 집합을 $K_l^+$로, 반시계 방향의 경로에 포함하는 노드 쌍들의 집합을 $K_l^-$로 정의한다. 즉, $K_l^+=\{k\in K|l\in L_k^+\}$이고 $K_l^-=\{k\in K|l\in L_k^-\}$ 이다. 즉 $K_l^-=K\backslash K_l^+$임을 쉽게 알 수 있다. 노드 쌍 $k$의 공급노드와 수요노드가 각각 $i$와 $j$일 때 노드 쌍 $k$를 $k$대신에 $\{i,j\}$로도 표기하기로 한다.

각 노드 쌍 $k\in K$에 대해 변수 $x_k$를 이용하여 $o(k)$와 $d(k)$ 사이의 수요 중 시계 방향으로 루팅되는 흐름의 양을 표시하기로 하자. 그리고 $X=\{\mathrm{x}\in R^{|K|}|0\le x_k\le r_k, \forall k\in K\}$를 정의하면 임의의 $\mathrm{x}\in X$는 주어진 수요를 만족하는 루팅 방법에 대응되는 벡터가 된다. 주어진 루팅 방법 $\mathrm{x}\in X$에 대해,

$$g(\mathrm{x},l)=\sum_{k\in K_l^+}x_k+\sum_{k\in K_l^-}(r_k-x_k),\quad l\in L$$

이라고 정의하면 $g(\mathrm{x},l)$은 x 에 따라 통신 수요를 루팅할 때 링크 $l$을 통과하는 흐름의 합을 의미한다. 각각의 링크 $l\in L$에 대해 용량이 $c_l$로 주어져 있다고 가정하자. 그러면 링크 $l$에서의 흐름의 합은 링크의 용량을 초과할 수 없으므로 모든 링크 $l\in L$에 대해서 $g(\mathrm{x},l)\le c_l$이 만족되어야 한다.

링의 용량결정문제는 주어진 수요를 만족하기

위하여 필요한 링크의 용량 중 최대용량을 최소화하는 루팅 방법을 결정하는 문제이므로 RLPW는 다음과 같이 정식화된다.

$$(RLPW) \quad z = \min_{x \in X} \max_{l \in L} g(x, l)$$

(RLPW)는 다음과 같이 선형계획모형으로도 표현할 수 있다.

$$(RLPW) \quad z = \min c$$
$$\text{s.t. } c \geq g(x, l), \quad l \in L$$
$$x \in X.$$

## 3. 새로운 해법

이 절에서는 $O(\min\{n|K|, n^2\})$ 계산 시간 내에 (RLPW)의 최적해를 구할 수 있는 해법을 제시하고자 한다. 우리의 해법은 Myung 등[7]의 해법(앞으로 MKT 해법이라고 부르기로 한다)의 가장 중요한 요소인 재루팅 방법을 개선하여 계산 시간의 효율성을 높인 해법이므로 우선 MKT 해법을 먼저 소개하고 개선된 부분을 소개함으로써 우리의 해법을 설명하기로 한다.

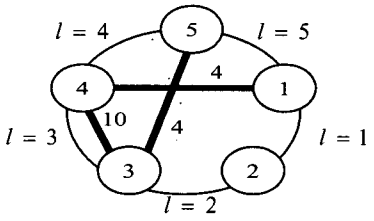
MKT 해법의 개요를 먼저 설명하기로 하자. MKT 해법은 우선 $K$에 속한 노드 쌍을 일정한 순서로 배열한 다음에 일단 모든 노드 쌍간의 수요를 시계 방향으로 루팅한다. 그리고 링의 용량을 최소화하기 위해서 재루팅이 이루어지는데 재루팅은 각 노드 쌍 $k \in K$에 대해 배열된 순서로 노드 쌍 $k$의 수요 중 일부 또는 전부를 링의 최대 용량이 감소되도록 반시계 방향으로 루팅을 변경한다. 루팅의 변경은 다음과 같이 간단히 수행된다. 현재의 루팅 방법 $x \in X$에 대하여 가장 흐름이 많은 링크들이 어떤 노드 쌍 $k \in K$의 시계 방향의 경로에 속하면, 즉 $\max_{l \in L_k^+} g(x, l) > \max_{l \in L_k^-} g(x, l)$이면 현재 시계 방향으로 루팅되고 있는 노드 쌍 $k$의 수요를 반시계 방향으로 재루팅하는 경우 링의 용량을 감소시킬 수 있다. 재루팅 단계에서 노드 쌍 $k$의 수요는 링의 용량이 최소가 되도록 재루팅되는데, 루팅이 변경되는 수요의 양은 상황에 따라 전체 수요가 될 수도 있고 수요의 일부만 재루팅될 수도 있다. 후자의 경우는 재루팅이 이루어진 뒤의 루팅 방법 $x \in X$에 대하여 $\max_{l \in L_k^+} g(x, l) = \max_{l \in L_k^-} g(x, l)$가 될 때까지 수요의 일부만 재루팅될 때 일어난다. MKT 해법은 다음과 같이 나타낼 수 있다.

**MKT 해법**

1. (노드 쌍의 순서 배열) $k_i$를 $K$에 속한 $i$번째 노드 쌍이라고 할 때 $o(k_i) < o(k_j)$이면 $k_i < k_j$ 되게 하고, $o(k_i) = o(k_j)$인 경우에는 $d(k_i) > d(k_j)$이면 $k_i < k_j$되도록 한다.
2. (초기해의 결정) 각 노드 쌍 $k \in K$에 대해 $x_k \leftarrow r_k$로 한다.
3. (재루팅) 각 노드 쌍 $k \in K$에 대해 배열된 순서로 다음을 수행한다
   $\delta \leftarrow \max_{l \in L_k^+} g(x, l) - \max_{l \in L_k^-} g(x, l)$
   를 계산한 다음에 $\delta > 0$이면
   $\Delta(k) \leftarrow \min\{\delta/2, r_k\}$, $x_k \leftarrow x_k - \Delta(k)$로 놓고 다음과 같이 $g(x, l)$을 재조정한다.

$$g(x, l) \leftarrow \begin{cases} g(x, l) - \Delta(k) & \forall l \in L_k^+ \\ g(x, l) + \Delta(k) & \forall l \in L_k^- \end{cases}$$

MKT 해법의 적용 예가 [그림 1]에 나타나 있다. [그림 1]은 5개의 노드로 구성된 링에서 3개의 노드 쌍이 통신 수요를 가진 예를 보여 주고 있다. [그림 1]의 (a)는 5개의 노드와 링크로 구성된 링을 나타내고 있고, 노드 쌍을 연결하는 굵은 직선은 통신 수요가 존재하는 노드 쌍 {1,4}, {3,4}, {3,5}를 나타내고, 굵은 직선 위의 숫자는 이들 노드 쌍의 통신 수요를 나타내고 있다. (b)는 모든 수요를 초기에 시계 방향으로 보내고, 재루팅 단계에서 링크의 용량을 재루팅을 통해 줄여 나가는 과정을 보여준다. 이때 MKT 해법에서의 재루팅 순서는 노드 쌍의 배열 순서인 {1,4}, {3,5}, {3,4}가 됨




(a)

| 재루팅 $\{i, j\}$ | 링크 용량 $g(x, l)$ | | | | | $\Delta(\{i, j\})$ |
|---|---|---|---|---|---|---|
| | $l = 1$ | 2 | 3 | 4 | 5 | |
| {1,4} | 4 | 4 | 18 | 4 | 0 | min {(18−4)/2, 4} = 4 |
| {3,5} | 0 | 0 | 14 | 8 | 4 | min {(14−4)/2, 4} = 4 |
| {3,4} | 4 | 4 | 10 | 4 | 8 | min {(10−8)/2, 10} = 1 |
| | 5 | 5 | 9 | 5 | 9 | |

(b)

[그림 1] MKT 해법의 예

에 주목하자. (b)의 테이블에서 첫 번째 열의 4, 4, 18, 4, 0은 초기 할당에 의한 각 링크의 흐름량의 합, 즉 필요한 링크 용량을 나타내고 있다. 단계 3에서 재루팅 순서에 의해 {1,4}를 먼저 재루팅하게 되는데, 노드 쌍 {1,4}의 시계 방향에 포함된 링크 1, 2, 3들은 음영으로 표시되어 있다. 그리고 반시계 방향의 링크 4, 5는 흰색으로 되어 있다. 시계 방향에 포함된 링크들의 현재 용량에서 최대 값은 $\max\{4,4,18\} = 18$, 반시계 방향에서의 최대값은 $\{4,0\} = 4$이다. 따라서 $\delta > 0$이 되므로 재루팅을 하게 되는데, 반시계 방향으로 보내는 수요량은 $r_{\{1,4\}} = 4$이므로 $\Delta(\{1,4\}) = \min\{(18-4)/2, 4\} = 4$가 된다. 따라서 4 만큼을 시계 방향의 링크(링크 1, 2, 3)에서는 빼주고, 반시계 방향의 링크(링크 4, 5)에서는 더해준 후 다음 재루팅 단계인 노드쌍 {3,5}로 넘어가게 된다. 동일한 과정을 노드 쌍 {3,5}와 {3,4}에 대해서도 반복하여 재루팅을 하게 되면 링크 용량은 네 번째 열인 5, 5, 9, 5, 9가 되고, 이때의 최대 링크 용량인 9는 링의 최적 용량이 된다. 단계 1의 재배열은 $O(|K|)$계산으로 가능하고[3], 단계 2의 초기해의 설정도 $O(|K|)$계산으로 가능하다. 단계 3의 반복과정을 어떤 노드 쌍 $k \in K$에 대해서 한 번 시행하는데 $O(n)$ 시간 소요되므로 단계 3 전체는 $O(n|K|)$시간 내에 수행된다. Myung 등[7]은 MKT 해법이 링크의 용량을 최소화하는 최적해를 구할 수 있음을 보였다. MKT 해법은 현재까지 발표된 RLPW의 해법들 중 이론적인 계산 시간에서 가장 우수하나 수요가 모든 노드 쌍에 존재한다면 $|K| = O(n^2)$이므로 MKT 해법도 Vachani 등[10]과 Dell'Amico 등[3]의 해법처럼 $O(n^3)$의 소요 시간이 필요함을 알 수 있다.

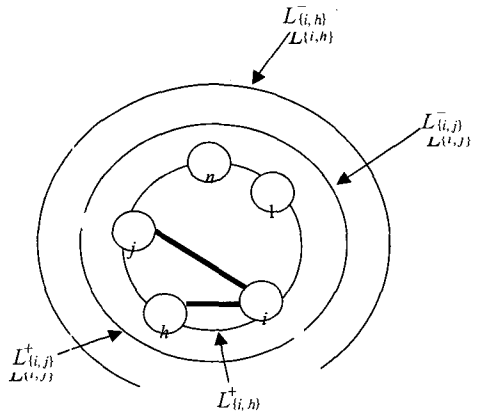
위에서의 분석처럼 MKT 해법의 소요 시간은 재루팅 단계의 계산에 좌우된다. 따라서 만약에 재루팅 단계의 계산을 좀 더 효율적으로 할 수 있다면 전체 시간도 개선될 수 있을 것이다. MKT 해법에서 하나의 노드 쌍에 대한 재루팅이 $O(n)$시간 소요되는 이유는 $\Delta(k)$의 계산에 모든 링크 $l \in L$에 대해서 $g(x,l)$의 정보가 필요하고 $x_k$의 변화에 따라 $g(x,l)$의 값을 조정하는데 $O(n)$ 시간이 필요하기 때문이다. 우리가 제시하는 새로운 방법은(MK 해법이라고 부르기로 한다.) 기본적으로 MKT 해법과 동일하게 진행되지만 재루팅 단계에서의 $\Delta(k)$ 계산을 효율적으로 함으로써 전체 계산시간을 줄일 수 있도록 고안되었다. 새로운 MK 해법을 설명하기 위해 다음과 같은 기호를 추가로 정의하자. 노드 쌍의 집합 $K$에 대해서 공급노드들

의 집합 $I=\{i\in N \mid o(k)=i$인 $k\in K$가 존재$\}$를 정의하자. 그리고 각 $i\in I$에 대해서 공급노드가 $i$인 노드 쌍의 집합을 $K(i)=\{k\in K \mid o(k)=i\}$로 정의한다. 새로 제시하는 MK 해법은 어떠한 $i\in I$에 대해서도 $K(i)$에 속한 노드 쌍의 재루팅을 $O(n)$시간 내에 완료함으로써 전체 계산 시간을 $O(n|I|)$ 내에 가능하게 한다. 즉 MK 해법은 MKT 해법과는 달리 $O(n)$시간 소요되는 반복과정을 $|K|$번이 아니라 $|I|$번만 반복하는 재루팅을 완료하게 된다. 이 때 $|I|\leq \min\{|K|, n\}$이므로 MK 해법에서 재루팅은 $O(\min\{n|K|, n^2\})$의 계산시간 내에 완료되는 것이다. 뒤에서 재루팅과정을 포함한 MK 해법의 전체 과정도 $O(\min\{n|K|, n^2\})$의 계산시간에 완료됨을 보일 것이나 우선은 재루팅 과정을 먼저 소개하기로 한다.

위에서 언급한대로 MK 해법은 각 $i\in I$에 대해서 $K(i)$에 속한 노드 쌍별로 재루팅이 이루어지는데 $K(i)$의 노드 쌍은 MKT 해법에서와 마찬가지로 배열시킨다. MK 해법에서는 $K(i)$에 속한 노드 쌍의 재루팅을 $O(n)$시간 내에 완료하기 위해서 $g(x,l)$ 값의 조정을 $K(i)$에 속한 모든 노드 쌍에 대한 재루팅이 완료된 뒤에 행한다. 이 경우 모든 링크 $l\in L$에 대한 $g(x,l)$ 값의 정보 없이도 $\Delta(k)$를 계산할 수 있어야 하는데 $K(i)$에 속한 노드 쌍들의 공급노드와 수요노드의 구조적 특성을 이용하면 가능함을 보일 수 있다. [그림 2]에서처럼 $k=\{i,j\}$와 $k+1=\{i,h\}$이 $K(i)$에 속한 연속한 두 노드 쌍이라고 가정하자. 노드 쌍의 배열 방법 때문에 $h<j$이다. 노드 쌍 $k$에 대하여 재루팅이 이루어지기 직전에 얻어진 루팅 방법을 $x\in X$라고 하고 $s_k=\max_{l\in L_k^+} g(x,l)$와 $t_k=\max_{l\in L_k^-} g(x,l)$를 정의하자. 즉 $s_k$는 노드 쌍 $k$에 대하여 재루팅이 이루어지기 직전에 얻어진 루팅 방법 x에 대해서 노드 쌍 $k$의 시계 방향에 속한 링크의 최대용량을, $t_k$는 반시계 방향에 속한 링크의 최대용량을 나타낸다. 따라서 $\Delta(k)$의 정의에 의해서 $s_k$와 $t_k$를 알면 $\Delta(k)$를 구할 수 있다. 이를 이용하여 노드 쌍 $k$에 대하여 재루팅이 이루어지고 재루팅이 이루어진 후에 얻어진 루팅 방법을 $x'\in X$이라고 하자. 만약에 $s_{k+1}$과 $t_{k+1}$을 구하기 위하여 모든 링크 $l\in L$에 대한 $g(x',l)$의 값을 새로 조정한다면 이 방법은 MKT 해법과 동일하게 진행될 것이다. 새로운 해법은 $s_k$와 $t_k$를 알고 있다면 일부 링크에 대해서만 $g(x',l)$의 값을 알아도 $s_{k+1}$과 $t_{k+1}$을 구할 수 있다는 아이디어에서 비롯되었다. 우선 $t_{k+1}$은 정의에 의해 다음과 같이 표시할 수 있다.

$$\begin{aligned} t_{k+1} &= \max_{l\in L_{k+1}^-} g(x',l) \\ &= \max[\max\{g(x,l) \mid h\leq l\leq j-1\}-\Delta(k), \\ &\quad \max\{g(x,l) \mid l\in L_k^-\}+\Delta(k)] \\ &= \max[\max\{g(x,l) \mid h\leq l\leq j-1\}-\Delta(k), \\ &\quad t_k+\Delta(k)] \end{aligned}$$

[그림 2]에 나타낸 것처럼 $L_k^-=\{1,\cdots,i-1,j,j+1,\cdots,n\}\subset L_{k+1}^-=\{1,\cdots,i-1,h,h+1,\cdots,j-1,j,\cdots,n\}$임을 고려하면 위의 관계는 쉽게 알 수 있다. 따라서 $t_{k+1}$을 구하기 위해서는 $t_k$와 $\Delta(k)$ 외에 일부 링크에 대한 $g(x',l)$의 값의 정보, 즉 $\max\{g(x',l) \mid h\leq l\leq j-1\}$만 알 수 있다면 $t_{k+1}$을 구할 수 있다.




[그림 2] 노드 쌍 $k=\{i,j\}$와 $k+1=\{i,h\}$의 관계

한편 $t_{k+1}$을 알면 $s_{k+1}$은 다음의 사실로부터 쉽게 유도할 수 있다.

**[정리 1]** $K(i)$에 속한 연속한 두 노드 쌍 $k=\{i,j\}$와 $k+1=\{i,h\}(h<j)$에 대해서 다음이 성립한다 : $s_{k+1}>t_{k+1}$이기 위한 필요충분조건은 $s_k-\Delta(k)>t_{k+1}$이고, 또한 $s_k-\Delta(k)>t_{k+1}$이면 $s_{k+1}=s_k-\Delta(k)$이다.

**(증명)**

MKT 해법의 성격에 의해 $s_k-\Delta(k)\geq t_{k+1}$가 항상 성립한다. 노드 쌍 $k$에 대하여 재루팅이 이루어지기 직전에 얻어진 루팅 방법을 $x\in X$라고 나타내자. 이제 $L_k^+$에 속한 링크 중 $g(x,l)$이 최대인 링크를 생각해 보자. 물론, 그러한 링크는 복수로 존재할 수도 있다. 이러한 링크 중 하나가 $L_k^+-L_{k+1}^+=\{h,h+1,\ldots,j-1\}$에 속하는 경우와 그렇지 못한 경우를 각각 생각해 보자. $L_k^+$에 속한 링크 중 $g(x,l)$이 최대인 링크 중 $L_k^+-L_{k+1}^+$에 속하는 링크 $l^*$가 존재한다고 가정해 보자. 즉, $g(x,l^*)=s_k(=\max_{l\in L_k^+}g(x,l))$이고 $l^*\in L_k^+-L_{k+1}^+=\{i,i+1,\cdots,h-1\}$이다. 그러면, $s_{k+1}$과 $t_{k+1}$의 정의에 의해서 $s_{k+1}\leq s_k-\Delta(k)=g(x,l^*)-\Delta(k)\leq t_{k+1}$이 성립한다. 만약에 $L_k^+$에 속한 링크 중 $g(x,l)$이 최대이면서 $L_k^+-L_{k+1}^+$에 속한 링크는 없다고 가정하자. 이 경우에는 $s_{k+1}=s_k-\Delta(k)$이 성립한다. 따라서, $s_{k+1}>t_{k+1}$이거나 $s_k-\Delta(k)>t_{k+1}$인 상황은 항상 후자의 경우에만 발생하고, 이 때 $s_{k+1}=s_k-\Delta(k)$이 성립하므로 정리 1이 성립한다. □

위의 정리에 의해서 노드 쌍 $k+1$에 대한 재루팅은 $s_k-\Delta(k)>t_{k+1}$일 때만 필요하게 되며 이 때 $s_{k+1}=s_k-\Delta(k)$로 구할 수 있어 $t_{k+1}$을 알고 있다면 다음 단계의 재루팅을 수행할 수 있게 된다. 다시 말해서 $K(i)$에 속한 각각의 노드 쌍들에 대한 재루팅을 행할 때는 전 단계에서 계산한 $s_k$와 $t_k$를 알고 있다면 모든 링크가 아닌 일부의 링크에 대한 $g(x',l)$의 값의 정보, 즉 $\max\{g(x',l)|h\leq l\leq j-1\}$을 구함으로써 다음 단계에서 재루팅이 필요한지 여부를 판단할 수 있고 재루팅을 해야 할 경우에도 재루팅에 필요한 $s_{k+1}$과 $t_{k+1}$를 구할 수 있다.

이제부터는 위의 사실에 근거하여 $K(i)$에 속한 모든 노드 쌍에 대한 재루팅이 $O(n)$ 시간 내에 완료될 수 있음을 보이기로 하자. 이를 위하여 각각의 $i\in I$에 대해서 $K(i)$에 속한 모든 노드 쌍에 대한 재루팅을 수행하는 알고리즘 REROUTING_SUB를 고려해 보자. REROUTING_SUB는 현재의 루팅 방법 $x\in X$와 이에 대응되는 모든 링크 $l\in L$에 대한 $g(x,l)$의 정보를 입력자료로 받아서 $K(i)$에 속한 모든 노드 쌍에 대한 재루팅을 수행한 뒤에 재루팅의 결과로 생성된 루팅 방법 $x\in X$와 이에 대응되는 모든 링크 $l\in L$에 대한 $g(x,l)$의 정보를 출력하는 알고리즘이다. 따라서 각 $i\in I$에 대해서 REROUTING_SUB를 반복하여 실행하게 되면 MKT 해법의 재루팅 단계를 수행하는 셈이 된다.

**알고리즘 REROUTING_SUB**$(x, g(x,l), i)$

(표현의 편의를 위해서 $K(i)=\{1,\cdots,p\}$라고 가정)

1. (초기화) $g_l\leftarrow g(x,l)\ \forall\, l\in L$, $s_1\leftarrow\max_{l\in L_1^+}g_l$, $t_1\leftarrow\max_{l\in L_1^-}g_l$, $\rho_1\leftarrow 0$.
2. (반복단계) 각 노드 쌍 $k=1,\cdots,p$에 대해 다음을 수행한다 :

   $\Delta(k)\leftarrow\min\left\{\dfrac{s_k-t_k}{2}, r_k\right\}$.

   ($s_k, t_k, r_k$의 성격상 항상 $\Delta(k)\geq 0$)

   만약에 $\Delta(k)=0$이면 단계 3으로 이동.

   $x_k\leftarrow x_k-\Delta(k)$.

   $\rho_{k+1}\leftarrow\max\{g_l-\sum_{t=1}^{k}\Delta(t)|d(k+1)\leq l\leq d(k)-1\}$.

   $s_{k+1}\leftarrow s_k-\Delta(k)$, $t_{k+1}\leftarrow\max\{\rho_{k+1}, t_k+\Delta(k)\}$.

3. ( $g(\mathrm{x}, l)$의 재조정)

$$g(\mathrm{x}, l) \leftarrow \begin{cases} g_l + \sum_{t=1}^{k} \Delta(t) \\ \forall l \le i-1,\ l \ge d(1) \end{cases} \quad (1)$$

$$\begin{cases} g_l - \sum_{t=1}^{k} \Delta(t) \\ \forall i \le l \le d(k)-1 \end{cases} \quad (2)$$

$$\begin{cases} g_l - \sum_{t=1}^{k'-1} \Delta(t) + \sum_{t=k'}^{k} \Delta(t) \\ \forall d(k') \le l \le d(k'-1)-1,\ 2 \le k' \le k \end{cases} \quad (3)$$

단계 2에서의 $\rho_k$는 앞에서 설명한 것처럼 $t_{k+1}$을 계산하는데 추가로 필요한 링크들에 대한 용량의 최대치를 구한 값이다. 단계 2의 $\rho_k$의 의미와 단계 3의 $g(\mathrm{x}, l)$의 재조정 방법의 이해를 돕기 위해서 $K(i)$의 첫 번째 노드 쌍부터 $k$ 번째 노드 쌍까지의 재루팅이 일어난 과정을 <표 1>에 표현하였다. <표 1>에서는 재루팅 단계에서 각 링크의 흐름량의 변화를 '+'와 '-'로 나타내고 있다. 각 노드 쌍의 시계 방향에 있는 링크의 흐름량은 감소하고 반시계 방향에 포함된 링크의 흐름량은 증가한다. 표에서 음영으로 표시된 부분은 반시계 방향에 포함된 링크들을 나타내고 있다. <표 1>을 참조하면 단계 2에서 $\rho_{k'}$를 계산할 때 $k'$ 번째까지 재루팅이 이루어진 뒤에 $d(k'+1) \le l \le d(k')-1$을 만족하는 링크 $l$들의 $g(\mathrm{x}, l)$의 재조정된 값을 구하는 방법과 단계 3에서 $k$까지 재루팅이 이루어진 뒤에 모든 링크 $l \in L$에 대한 $g(\mathrm{x}, l)$의 값을 구하는 방법에 대해서 이해할 수 있다.

이제 REROUTING_SUB의 계산 시간을 분석해 보자. 초기화에 $O(n)$ 시간이 소요됨은 자명하고 단계 2에서 하나의 링크는 서로 다른 노드 쌍의 $\rho_k$ 계산에 중복되어 포함되는 경우가 없으므로 $K(i)$의 각 노드 쌍에 대해서 반복단계 전체가 진행되는 동안 모든 $\rho_k$ 계산은 합해서 $O(n)$ 시간 내에 가능하다. 단계 2에서의 $\rho_k$ 외의 나머지 계산은 한 노드 쌍에 대해서 상수 시간만 소요되고 $|K(i)| \le O(n)$이므로 $O(n)$ 시간 내에 가능하다. 따라서 단계 2의 총 계산 시간은 $O(n)$ 시간 내에 완료된다. 단계 3에서의 $g(\mathrm{x}, l)$의 재조정도 식 (1)과 식 (2)의 링크에 대한 재조정은 $\sum_{t=1}^{k} \Delta(t)$를 한 번 계산하면 각 링크별로 상수 시간 소요되고 $k \ge 2$인 경우 식 (3)의 링크에 대한 재조정도 $2 \le k' \le k$인 노드 쌍 $k'$에 대해서 $d(k') \le l \le d(k'-1)-1$인 링크 $l$의 값을 정할 때 $-\sum_{t=1}^{k'-1} \Delta(t) + \sum_{t=k'}^{k} \Delta(t)$ 부분은 $k'$이 증가함에 따라 $2\Delta(k')$를 추가로 감소시키면 되므로 상수 시간에 가능하다. 따라서 단계 3의 전체 과정도 $O(n)$ 시간 내에 가능하다.

앞서 언급한대로 MK 해법은 MKT 해법과 동일하게 진행된다. 초기해를 각 노드 쌍 $k \in K$에 대해 $x_k \leftarrow r_k$로 정하고 초기해에 대한 $g(\mathrm{x}, l)$의 값을 구한 뒤, 각 $i \in I$에 대해서 REROUTING_SUB를 반복하는 것이다. MK 해법은 진행 과정이 MKT 해법과 동일하므로 최적해를 구하는 것이 보장된다. 이제 MK 해법의 계산시간을 분석하여 보자. 초기해를 구하는 것은 $O(|K|)$ 시간 내에 가능하

〈표 1〉 REROUTING_SUB에서의 재루팅에 따른 링크 흐름량의 변화

| 노드쌍 | | 링크 용량 변화 $g(\mathrm{x}, l)$ | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $l=1$ | ⋯ | $i-1$ | $i$ | | $d(k)$ | | $d(k'+1)$ | | $d(k')$ | | $d(k'-1)$ | | $d(1)$ | ⋯ | $n$ |
| 1 | $\Delta(1)$ | + | ⋯ | + | − | ⋯ | − | | − | | ⋯ | | − | − | + | ⋯ | + |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ | | ⋮ |
| $k'-1$ | $\Delta(k'-1)$ | + | ⋯ | + | − | ⋯ | ⋯ | | ⋯ | | − | − | + | ⋯ | + | | + |
| $k'$ | $\Delta(k')$ | + | ⋯ | + | − | ⋯ | ⋯ | | − | − | + | ⋯ | + | ⋯ | + | | + |
| $k'+1$ | $\Delta(k'+1)$ | + | ⋯ | + | − | ⋯ | ⋯ | − | + | ⋯ | | | | | | | |
| ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | ⋯ | ⋮ | + | ⋮ | | | | | | | | |
| $k$ | $\Delta(k)$ | + | ⋯ | + | − | − | + | | | | | | | | | | |

고 초기해에 대한 $g(\mathrm{x}, l)$의 값은 뒤에서 설명하겠지만 REROUTING_SUB의 단계 3의 $g(\mathrm{x}, l)$의 재조정 방법과 유사한 방법으로 $O(n|I|)$ 시간에 완료할 수 있다. 그러면 앞서 설명한대로 REROUTING_SUB가 $O(n)$ 시간에 수행 가능하므로 MK 해법은 $O(n|I|)$ 시간에 완료할 수 있다. 결론적으로 $|I| \le O(\min\{|K|, n\})$이므로 MK 해법은 $O(\min\{n|K|, n^2\})$ 계산 시간 내에 RLPW의 최적해를 구할 수 있게 된다. 아래에 소개하는 과정은 초기해에 대한 $g(\mathrm{x}, l)$의 값을 구하기 위한 것이다. 알고리즘 INIT는 REROUTING_SUB의 단계 3에서 $g(\mathrm{x}, l)$을 재조정하는 경우처럼 $O(n)$ 시간 내에 가능함을 쉽게 알 수 있다.

**초기해($x_k \leftarrow r_k \ \forall k \in K$)에 대한 $g(\mathrm{x}, l)$의 결정과정**

1. (초기화) $g(\mathrm{x}, l) \leftarrow 0 \ \forall l \in L$.
2. (반복단계) 각 $i \in I$에 대해서 INIT($g(\mathrm{x}, l)$, $i$)를 수행한다.

**알고리즘 INIT**($g(\mathrm{x}, l), i$)

(표현의 편의를 위해서 $K(i) = \{1, \cdots, p\}$라고 가정)

각 노드 쌍 $k = 1, \cdots, p$에 대해 다음을 순차적으로 수행한다 :

$$g(\mathrm{x}, l) \leftarrow \begin{cases} g(\mathrm{x}, l) + \sum_{t=1}^{k} r_t \\ \forall d(k+1) \le l \le d(k) - 1, \ 1 \le k \le p-1 \\ g(\mathrm{x}, l) + \sum_{t=1}^{p} r_t \\ \forall i \le l \le d(p) - 1 \end{cases}$$

## 4. 결 론

링의 용량결정문제는 현실적인 중요성때문에 많은 연구가 이루어져 왔다. 이 중에서 수요의 분할 처리가 허용되는 경우에 링의 용량을 최소화하는 루팅 방법을 찾는 RLPW의 해법은 동기식 광통신망 (SONET)의 용량결정에 직접 적용되며, 수요의 분할 처리가 허용되지 않거나 허용되더라도 정수 단위로만 가능한 문제에 대해서도 완화문제의 해법으로서 유용하게 사용될 수 있다. 이러한 응용성 때문에 RLPW에 대해서 최근에 Vachani 등의 해법, Schrijver 등의 해법, Myung 등의 해법 및, Amico 등의 해법 등이 동시 다발적으로 개발되었다. 본 논문에서는 이제까지 개발된 해법 중 이론적인 계산 시간이 $O(n|K|)$로 가장 빠른 Myung 등의 해법의 재루팅 과정을 개선함으로써 계산 시간이 $O(\min\{n|K|, n^2\})$으로 개선된 새로운 해법을 제시하였다. 한편으로는 수요의 분할 처리가 허용되는 경우 중 정수 단위로만 분할이 허용되는 문제에 대해서 Myung[1, 8]은 Myung 등의 MKT 해법을 이용하여 $O(n|K|)$ 계산 시간 내에 최적해를 구할 수 있는 해법을 제시하였는데 여기서 개발된 MK 해법을 같은 방법으로 이용한다면 정수 단위로만 분할이 허용되는 문제에 대해서도 $O(\min\{n|K|, n^2\})$이 가능하게 된다.

## 참 고 문 헌


[1] 명영수, "정수단위로만 루팅이 허용되는 SONET링의 용량결정문제", 「한국경영과학회지」, 제25권, 제1호(1998), pp.49-62.

[2] 명영수, 김후곤, "분할이 허용된 SONET 링의 루팅 해법들에 대한 비교 분석", 「경영과학지」, 제18권, 제2호(2001), pp.107-116.

[3] Cormen, T.H. C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, The MIT Press, Boston, MA, 1990, pp.175-177.

[4] Cosares, S. D.N. Deutsch, I. Saniee and O.J. Wasem, "SONET toolkit : a decision support system for designing robust and cost-effective fiber-optic networks," Interfaces 25(1995), pp.20-40.

[5] Cosares, S. and I. Saniee, "An optimization problem related to balancing loads on SONET rings," Telecommunication Systems,

3(1994), pp.165-181.

[6] Dell'Amico, M. M. Labbe, and F. Maffioli, "Exact Solution of the SONET Ring Loading Problem," Operations Research Letters, 26(1999), pp.119-129.

[7] Myung, Y.-S. H.-G. Kim and D.-W. Tcha, "Optimal load balancing on SONET bidirectional rings," Operations Research, 45(1997), pp.148-152.

[8] Myung, Y.-S. "An Efficient Algorithm for the Ring Loading Problem with Integer Demand Splitting," SIAM J. Discrete Math, 14(2001), pp.291-298.

[9] Schrijver, A. P. Seymour and P. Winkler, "The ring loading problem," SIAM J. Discrete Math., 11(1998) pp.1-14.

[10] Vachani, R. A. Shulman and P. Kubat, "Multicommodity flows in ring networks", INFORMS Journal on Computing, 8(1996), pp.235-242.

[11] Wu, T.H. "Fiber network service survivability," Artech House, Boston, MA, 1992.