韓國컴퓨터情報學會 論文誌
第16卷 第5號, 2011. 05.

2011-16-5-5-2

최대 수용량-기반 최소절단 알고리즘

이 상 운*

Maximum Capacity-based Minimum Cut Algorithm

Sang-Un, Lee *

요 약

최소 절단 문제는 공급처 S에서와 수요처 T로의 흐름 용량이 최소가 되는 지점들을 절단하는 문제이다. 망의 병목지점을 찾는 방법은 대부분 유동망을 계산하여 최소 절단값을 찾는 유동-기반 알고리즘이 적용되고 있다. 이 알고리즘은 최소절단은 제시하지 않는 단점이 있다. 본 논문은 유동망을 구하지 않고 망으로부터 직접 최대 수용량을 가진 정점을 인접한 S 또는 T로 병합하는 방법으로 최소 절단값을 찾는 간단한 알고리즘이다. 13개의 한정된 그래프에 적용한 결과 제안된 알고리즘은 간단하면서도 정확하게 최소 절단 값 $_{\min}c(S,T)$을 찾을 수 있었다.

▸ Keyword : 최대 유동/최소 절단, 유동 기반, 결정 알고리즘, 축약 알고리즘, 구성 알고리즘

Abstract

The minimum cut problem is to minimize $c(S,T)$, that is, to determine source S and sink T such that the capacity of the $S-T$ cut is minimal. The flow-based algorithm is mostly used to find the bottleneck arcs by calculating flow network, and does not presents the minimum cut. This paper suggests an algorithm that simply includes the maximum capacity vertex to adjacent set S or T and finds the minimum cut without obtaining flow network in advance. On applying the suggested algorithm to 13 limited graphs, it can be finds the minimum cut value $_{\min}c(S,T)$ with simply and correctly.

▸ Keyword : Max-flow/Min-cut, Flow-based, Deterministic Algorithm, Contraction Algorithm, Construction Algorithm

• 제1저자 : 이상운
• 투고일 : 2011. 01. 14, 심사일 : 2011. 02. 02, 게재확정일 : 2011. 02. 10.
* 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Science, Gangneung-Wonju National University)

I. 서 론

출발지 (source, s)와 목적지 (sink, t)가 주어진 방향 그래프 $D=(N,A), n\in N, a\in A$로 표현되는 도로, 통신 또는 전자회로 설계분야에서 현재 개설된 망의 최대 유동량 (maximum flow)을 파악하고, 병목 (bottleneck)이 발생하는 지점을 확장하면 최대 유동량을 증가시켜 효과를 극대화 시킬 수 있다.[1] 이 문제는 모든 유동 (흐름)은 s에서 t로만 흐른다고 가정한다. 망의 각 노드들 간에 흐를 수 있는 최대 용량인 수용량 (capacity)들이 모두 다를 경우 병목 지점들이 존재한다. 이러한 복잡한 망에서 병목지점들을 어떻게 찾을 수 있는가? 이 문제는 최대 유동/최소 절단 (Max-Flow/Min-Cut) 이론으로 설명된다.[2,3] 즉, s에서 t로의 유동이 있는 망에서 최대로 흐를 수 있는 최대 유동량은 그래프의 노드 집합 N을 병목지점들의 최소 수용량 합인 최소절단 (minimum cut, $_{\min}c(S,T)$을 기준으로 $s\in S$와 $t\in T$의 공통원소를 갖지 않는 2개 집합으로 분리시키는 문제이다. 결국, 망의 최대 유동량은 최소 절단값이 결정한다.

최소 절단 문제 알고리즘에는 유동망을 구하고 병목지점을 찾는 유동-기반 알고리즘 (flow-based algorithm)이 일반적으로 적용되고 있다. 가장 간단한 유동-기반 알고리즘은 망의 s에서 t로의 모든 경로를 찾아 병목 현상이 발생하는 호 (수용량이 가장 작은 호)들을 찾는 방법이다. 그러나 이 방법은 망의 크기가 큰 경우 모든 병목지점을 찾는 것은 현실적으로 어려우며, 또한 모든 병목지점들이 최소 절단값으로 사용되지 않는다.

보다 효율적인 유동-기반 알고리즘은 수용량 (c)을 가진 망의 가능한 경로에 유동량 (f)을 할당하고 잉여 수용량 $(c-f)$ 망으로 병목지점들을 얻는 방법으로 Ford-Fulkerson, Edmonds-Karp, Brute-Force Search, Linear Programming, Relabel-to-Front 알고리즘이 있으며, Ford-Fulkerson 알고리즘[2-6]이 대표적이다. 이 방법들은 병목지점들은 찾을 수 있지만 최소절단을 결정해 주지 않는 단점이 있다.

본 논문은 s와 t가 주어진 방향 그래프 망에서 최소 절단값을 직접 구하는 알고리즘을 제안한다. 제안된 알고리즘의 특징은 망의 최대 수용량 호를 내림차순으로 정렬시키고, 최대 수용량 호부터 차례로 인접한 $s\in S$ 또는 $t\in T$로 병합하면서 $s\in S$ 또는 $t\in T$가 최소 절단 값을 갖는지를 검증하는 방법이다. 기존 알고리즘과의 차이점은 다음과 같다. 첫 번째로, Ford-Fulkerson 알고리즘은 유동망을 구하는 반면에 제안된 알고리즘은 유동망을 구하지 않는다. 두 번째로, Ford- Fulkerson 알고리즘은 병목지점들만을 제시하며 최소절단은 시각적으로 추가로 결정해야 한다. 반면에 제안된 알고리즘은 병목지점들을 제시하지 않고 최소 절단을 제시한다.

2장에서는 최대 유동량/최소 절단 개념, 유동망과 병목지점들을 찾는 방법을 고찰한다. 3장에서는 최대 수용량 호를 선택하여 최소 절단값을 찾는 알고리즘을 제안한다. 4장에서는 제안된 알고리즘의 적용성과 성능을 평가해 본다.

II. 관련 연구

유동-기반의 최소절단 알고리즘을 적용하기 위해 관련 용어들을 살펴보자.

망 (network): 방향 그래프 $D=(N,A)$의 호 $a=(u,v)$, $u,v\in N$가 수용량 $c(u,v)$를 갖고 있다.

유동망 (flow network): s에서 t로의 임의의 경로 (path, p)에 대해 최대로 흐를 수 있는 유동량을 계산하여 망의 각 호에 유동량 $f(u,v)$를 할당한 그래프.

잉여 수용량 (residual capacity): $c_f(u,v)=c(u,v)-c(u,v)-f(u,v)$.

잉여 망 (residual network) $D_f(N,A_f)$: 호가 $c_f(u,v)$를 갖고 있는 망.

증대 경로 (augmenting path) $(s,u_1,u_2,\cdots,t)$, $c_f(u_i,u_{i+1})>0$: 추가로 유동량을 보낼 수 있는 경로.

망의 하나의 경로에서 흐를 수 있는 유동량은 잉여 수용량이 최소인 호 (병목지점)에 의해 결정된다.

절단 (cut): 모든 $s,t\in N$에 대해 모든 노드 N을 공통원소를 갖지 않는 S $(s\in S)$와 T $(t\in T)$의 두 집합으로 분할하는 것을 말하며, 병목지점들에서 발생한다. 이를 호 절단 (arc cut)이라 하며, 이 호들의 수용량 합을 절단 값 (cut value)이라 한다.

절단 (S,T)의 수용량: $c(S,T)=\sum_{u\in S,v\in T|(u,v)\in A} c(u,v)$로 S 집합 영역에서 T 집합 영역으로의 절단면에 대한 모든 호들의 수용량 합이다.

최소 절단: 절단들 중 가장 작은 수용량 합을 가진 절단으로 $_{\min}c(S,T)$로 표기한다.

Menger 정리에 따르면 "망의 최대 유동량은 S와 T 영역을 분리하는 최소 절단의 수용량 합 $_{\min}c(S,T)$과 동일하다." 이를 최대 유동/최소 절단이론이라 한다. 즉, 망에서 최소 절

단 값을 갖도록 $s \in S, t \in T$인 N을 공통 원소를 갖지 않는 두 집합으로 정확히 분할하였을 경우, 최소 절단 값은 망의 최대 유동량 값이 된다. s에서 t로의 총 유동량은 $_{\max}f(S,T) \le {}_{\min}(\sum S_o, \sum T_i)$가 성립한다. 여기서 $\sum S_o$은 S의 총 유출 량 (total outgoing flow), $\sum T_i$은 T의 총 유입 량 (total incoming flow)이다.

Ford-Fulkerson 알고리즘[2-6]은 $s-t$의 임의의 경로를 랜덤하게 찾아 유동량을 계산하고 $c-f=0$인 병목지점을 삭제하고 다시 $s-t$의 가능한 경로를 찾아 유동량을 계산한다. 이 방법을 모든 가능한 $s-t$ 경로가 존재하지 않을 때까지 수행한다. Ford-Fulkerson 알고리즘은 "가능 경로 탐색 유동량 계산 반복 방법"으로 알고리즘 복잡도는 $O(E \cdot f)$로 알려져 있다. 여기서 f는 최대 유동량 값이다. Ford-Fulkerson 알고리즘은 그림 1과 같다.

① 모든 간선 (u,v)에 대해 $f(u,v)=0$ 설정 (초기치)
② s에서 t로의 잉여 수용량 $c_f(u,v)>0$인 경로 p를 임의로 선택
$c_f(u,v)>0$인 경로가 더 이상 없으면 ⑤ 수행
③ 경로 p에 대한 최대 유동량 (Maximum Flow, f) 결정
$c_f(p) = {}_{\min}\{c_f(u,v) \mid (u,v) \in p\}$로 Bottleneck Arc 결정
④ 각 호 $(u,v) \in p$에 대해 Flow Augmenting Algorithm 수행, ②로 복귀
$f(u,v) = f(u,v) + c_f(p)$, $f(v,u) = f(v,u) - c_f(p)$
⑤ Maximum Flow = 각 경로 p에서 얻어진 Flow의 합

Minimum Cut = $c_f(u,v)=0$인 간선들로 분할된 (S,T)

그림 1. Ford-Fulkerson 알고리즘
Fig. 1. Ford-Fulkerson Algorithm

Ford-Fulkerson 알고리즘은 너비우선탐색 (breadth-first-search, BFS) 또는 깊이우선탐색 (depth-first-search, DFS)으로 임의의 증대 경로를 찾는다. 반면에, Edmonds-Karp 알고리즘은 Ford-Fulkerson 알고리즘과 동일하지만 임의의 증대경로들 중에서 최단경로를 찾는 차이점이 있다.

그림 2의 N_1 망 (방향 그래프)에 대해 모든 흐름이 s에서 시작하여 t로 흐를 수 있는 최대 유동량을 결정하는 최소 절단을 찾아 보자. 망에서 노드 i에서 j로 흐를 수 있는 수용량 c는 $c(i,j)$로 표기되어 있다. 이 망은 Chinneck[1]에서 인용되었다.

s에서 t로의 가능한 경로를 모두 열거하면 16개가 존재한다. 그러나 Ford-Fulkerson 알고리즘을 적용하되, 위에서 아래로 경로를 찾는 방법을 적용하면 그림 3과 같다. 여기서는 f/c로 표기하였다.

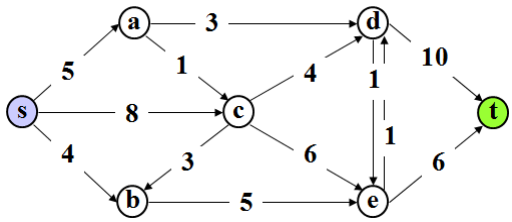
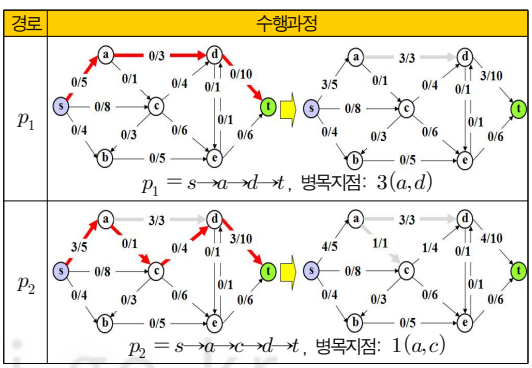


그림 2. N_1 망
Fig. 2. N_1 Network

모든 호의 유동량을 $f=0$으로 초기치를 설정한다. 첫 번째로 $p_1 = s \rightarrow a \rightarrow d \rightarrow t$으로 결정한 경우, $s \xrightarrow{0/5} a \xrightarrow{0/3} d \xrightarrow{0/10} t$으로 이 경로를 따라 흐를 수 있는 최대 흐름양은 병목지점 $3(a,d)$가 결정한다. 따라서 $s \xrightarrow{3/5} a \xrightarrow{3/3} d \xrightarrow{3/10} t$이 되며, (a,d)는 더 이상 흐를 수 없으므로 망에서 삭제된다. 이와 같이 계속적으로 잉여 수용량을 가진 호들로 경로를 찾으면 6개를 찾을 수 있다. 알고리즘을 수행한 결과 얻은 병목지점들은 $(a,d), (a,c), (c,d), (e,d), (s,c), (e,t)$의 6개이다. 이들 6개 병목지점들 중에서 $N=\{s,a,b,c,d,e,t\}$를 공통원소를 갖지 않는 $s \in S$와 $t \in T$로 분리시키면서 최소값을 가지는 절단을 시각적으로 찾으면 $(a,d), (c,d), (e,d), (e,t)$이다. 이때 최소 절단 값 $_{\min}c(S,T) = 3+4+1+6 = 14$가 되며, $S=\{s,a,b,c,e\}$, $T=\{d,t\}$이다. 결국, 초기에는 s에서는 $8+5+4=17$을 보내고, t에서는 $10+6=16$을 받아 $_{\min}\{17,16\}=16$을 망에서 흐를 수 있지만 망의 중간 과정에서 병목지점들로 인해 $_{\min}c(S,T)=14$이상은 흐를 수 없음을 알 수 있다.



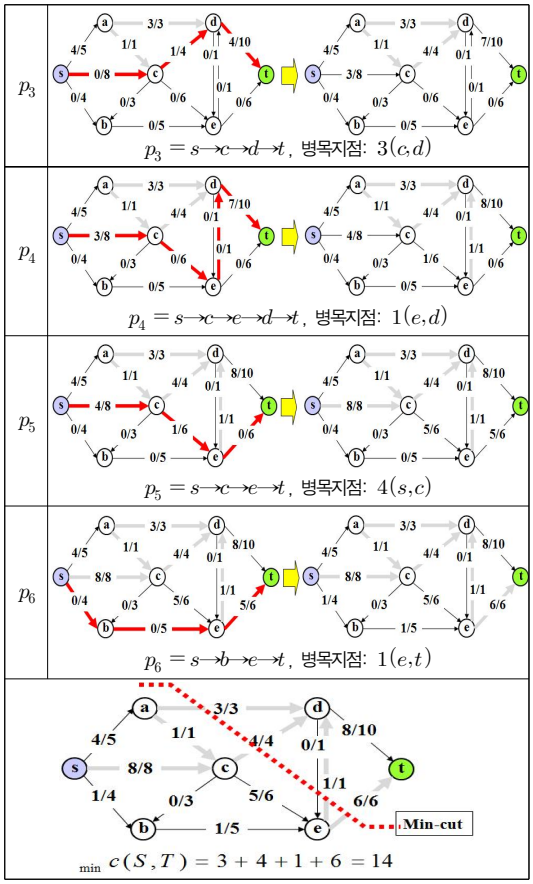


그림 3. N_1 망의 Ford-Fulkerson 알고리즘 병목지점
Fig. 3. Ford-Fulkerson Algorithm Bottleneck for N_1 Network

Ford-Fulkerson 알고리즘은 흐름증대경로의 각 호에 대해 $s \rightarrow t$ 의 순방향이면 $c(u,v)-f(u,v)$, $s \leftarrow t$의 역방향이면 $c(u,v)+f(u,v)$의 부기 방식으로 계산한다. 그러나 (그림 3)의 과정에서는 모두 순방향만 존재하여 역방향은 계산되지 않았다.

망의 최대 흐름양은 $\min c(S,T) \leq \min(\sum S_o = 17, \sum T_i = 16)$ 법칙을 만족시킨다. 즉, t의 유입 호를 확장하지 않는 한, 현재의 최소 절단 호들을 2 증가시키면 최대 16까지 흐를 수 있게 할 수 있다.

최소 수용량을 가진 절단을 찾는 방법으로는 열거 방법 보다 최적으로 알려진 방법이 없다.[7] 유동-기반 알고리즘은 많은 연구가 되었지만 아직까지 수행 복잡도가 $O(VE)$ 보다 좋은 알고리즘이 제안되지 않고 있다.[8] 또한, 병목지점들만을 찾을 수 있으므로 최소절단은 추가적으로 구해야한다.

Ford-Fulkerson 알고리즘은 가장 오래되고 비교적 간단한 장점이 있다. 이후에 Deng et al.[9]은 증대경로 선택시 많은 반복시간이 소요되는 점을 개선한 연구를 수행하였다. Goldberg[10]는 사전 흐름 (preflow) 개념을 도입하여 증대경로를 한 번에 하나씩 또는 모든 최단 증대경로를 한 번에 찾는 방법을 연구하였다. 이후에도 Blocking Flows & Fujishige's, Glodberg-Tarjan, Gomery-Hu Tree 알고리즘들이 제안되었다.[2-3] 그러나 Ford-Fulkerson 알고리즘의 마지막 단계인 병목지점들 중에서 최소절단을 찾는 방법은 연구결과를 확인할 수 없었다.

III. 최대 수용량 기반 최소 절단 알고리즘

s와 t가 주어진 망에서 $s-t$ 최소 절단은 망의 노드 집합 N의 원소들을 최소 절단값을 기준으로 정확히 공통원소를 갖지 않는 두 집합인 $s \in S$ 와 $t \in T$로 분할하는 문제이다. 일반적으로 적용되고 있는 Ford-Fulkerson이나 Edmonds-Karp 알고리즘은 유동망과 증대경로를 찾는 복잡한 과정을 수행함에도 불구하고 병목지점들만을 제시하고 있으며 최소 절단을 직접 구하지 못한다. 본 장에서는 이러한 문제점을 해결하는 간단한 알고리즘을 제안한다. 제안된 알고리즘은 다음 특징에 기반한다.

(1) $s-t$ 망에 대한 최소절단은 공통원소를 갖지 않도록 망을 S와 T로 양분하는 것이다.

(2) 망의 최대 지름 관로는 보다 작은 지름 관로들 보다 흐름의 여유가 있어 병목현상이 발생하지 않아 최소절단에 기여하지 못한다. 따라서 최대 지름 관로로 연결된 양 끝단의 두 정점은 하나의 정점으로 통합할 수 있다.

(3) 망의 최대지름 관로부터 내림차순으로 S 또는 T로 병합시키는 과정에서 반드시 우리가 찾고자 하는 최소절단을 얻을 수 있다.

위와 같은 최소절단 특성에 기반하여 제안된 알고리즘은 단순히 최대 수용량 호의 두 정점을 하나로 통합하면서 최소절단을 찾는 간단한 방법이다. 망의 모든 호 수용량에 대해 내림차순으로 정렬시키고, 최대 수용량 호 (u,v)를 선택하여 (S,v)이면 $S=S+v$로, (u,T)이면 $T=T+u$로 선택하고 절단값을 계산하는 방식이다. 이를 위해 초기 값으로 $S=\{s\}$, $T=\{t\}$로 설정하고, s는 유출만 발생하므로 $\sum S_o$ 즉, S 집합의 유출 호들의 수용량 합을 구한다. 마찬가지로 t는 유입만 발생하므로 $\sum T_i$ 즉, T 집합의 유입 호들의 수용량 합을 구한다. 제안된 알고리즘은 그림 4에 제시되어 있으며, 다음의 기준이 적용된다.

[기준 1] S로 유입되는 호와 T에서 유출되는 호는 무시된다.

〔기준 2〕만약, (u,v)가 S 와 T에 모두 인접하지 않으면 나중에 포함될 집합을 결정하기 위하여 임시로 H 집합(Holding Set)으로 남겨 놓는다.

〔기준 3〕만약 동일한 수용량 호가 다수 존재시, $S>T$ 순으로 선택한다.

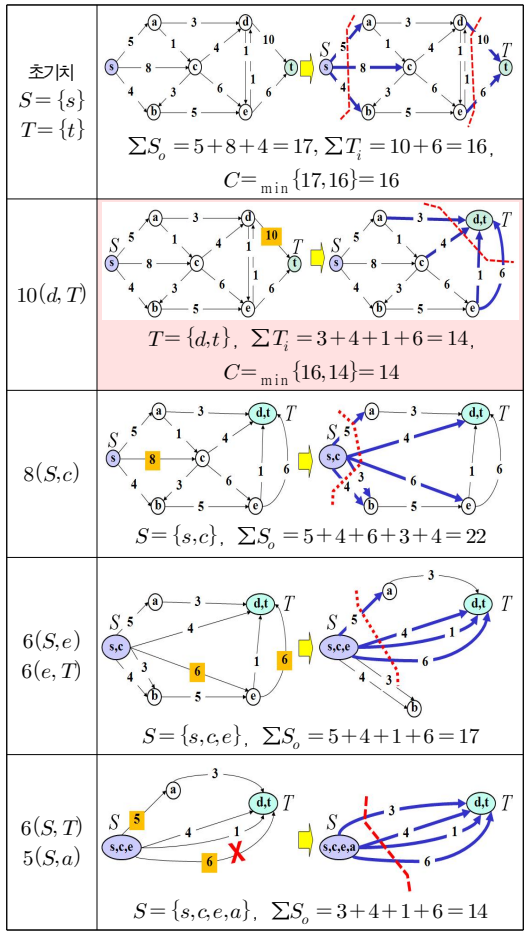
〔기준 4〕$(u,T),(v,T)$로 t의 유입 차수가 2인 경우, 만약, $T=\{u,t\}$로 병합된 상태라면 (v,t)에 대해서는 $T=\{u,v,t\}$ 대신 $T=\{v,t\}$로 다시 수행한다. 만약, $H=\{u,v\}$라면 (u,t)에 대해 $T=\{u,v,t\}$를 적용한다.

망 $D=(N,A)$.
초기값 : $S=\{s\}, T=\{t\}, H=\{\varnothing\}, N=N\setminus\{s,t\}$.
$\Sigma S_o, \Sigma T_i$ 계산.
$C=\min\{\Sigma S_o, \Sigma T_i\}$. /* C : 최소 절단 값.
모든 호 수용량 $c(u,v)$를 내림차순 정렬 /* 복잡도: $O(a\log a)$
for $N=\{\varnothing\}$ do /* 복잡도 : $O(a)$
  최대값 $\max c$ 호 (u,v) 선택.
  if (S,S), (S,T), or (T,T) then skip
  else if (S,v) then $S=S+v$, ΣS_o 계산
    $C=\min\{C,\Sigma S_o\}$, $N=N\setminus v$
  else if (S,H) then $S=S+H$, ΣS_o 계산,
  $C=\min\{C,\Sigma S_o\}$
  else if (u,T) then $T=T+u$, ΣT_i 계산
    $C=\min\{C,\Sigma T_i\}$, $N=N\setminus u$
  else if (H,T) then $T=T+H$, ΣT_i
  계산, $C=\min\{C,\Sigma T_i\}$
    단, $(x,t),(y,t)$로 $d_G(t)=2$이고, $T=\{t,x\}$인 경우,
      $T=\{t,x,y\}$가 아닌 $T=\{t,y\}$ 수행.
  else if (u,v) then $H=\{u,v\}$, $N=N\setminus\{u,v\}$
  else if (u,H) or (H,v) then $H=H+\{u|v\}$,
  $N=N\setminus\{u|v\}$.
  S 유입 호와 T 유출 호 삭제, 유출 호가 없는 정점 삭제
end

그림 4. 최대 수용량 기반 최소 절단 알고리즘
Fig. 4. Minimum-Cut Algorithm based on Maximum Capacity

제안된 알고리즘을 "최대 수용량 기반 최소절단 알고리즘"이라 하자. 제안 알고리즘의 특징은 노드들을 병합하지 않고 N개 노드를 S 또는 T집합으로 선택하는 ${}_NC_2$의 조합문제이며, 선택 조건은 가능한 모든 경우수 2^N을 모두 고려하는 전수검사 방식이 아니라 최대 수용량 호 선택 기준을 적용한다.

그림 2의 N_1 망에 대해 최대 수용량 기반 최소절단 알고리즘을 적용한 결과는 그림 5에 제시하였다. 초기치로, $S=\{s\}$, $T=\{t\}$로 설정하고 절단 값을 계산한 $C=\min\{17,16\}=16$을 얻었다. $10(d,T)$부터 시작하여 $8(S,c)$, $6(S,e)$, $5(S,a)$로 선택한 결과 $10(d,T)$에서 $N=\{s,a,b,c,d,e,t\}$를 $S=\{s,c,e,a,b\}$와 $T=\{d,t\}$의 공통원소를 갖지 않는 두 집합으로 정확히 절단할 수 있었으며, 최소 절단치 $\min c(S,T)=14$를 얻었다. 제안된 알고리즘은 단지 4개의 호를 선택하는 과정에서 최소 절단값을 얻었다. 반면에, Ford-Fulkerson 알고리즘은 6개의 증대경로를 선택하는 탐색 과정이 필요하며, 이 과정에서 수용량-흐름양의 여유 유동량을 계산해야 한다. 따라서 제안된 알고리즘을 적용하는 것이 보다 효율적이라 할 수 있다.



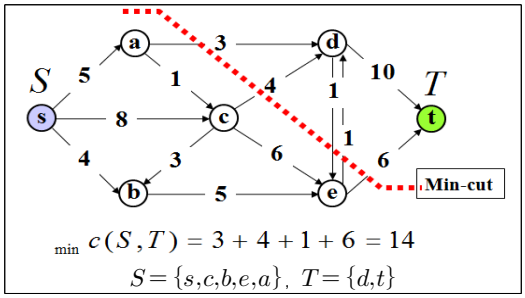
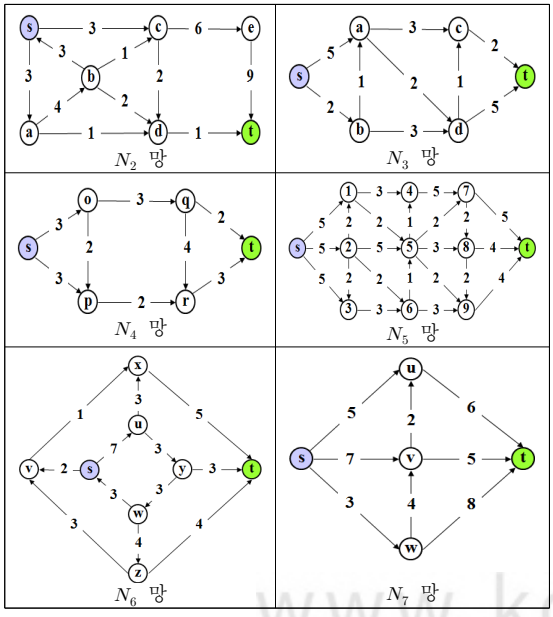


그림 5. N_1 망의 최대 수용량 기반 최소 절단 알고리즘
Fig. 5. Min-Cut Algorithm for N_1 Network based on Max Capacity

IV. 적용 및 결과 분석

제안된 알고리즘이 실제 망에 적용하여 최소 절단 값을 정확히 찾을 수 있으며, 망의 모든 노드 집합 N을 $s \in S, t \in T$인 공통 원소를 갖지 않는 두 집합으로 정확히 분할할 수 있는지 고찰해 보자. 가능한 다양한 형태의 망에 대해 제안된 알고리즘이 적합한지 검증하기 위해 실험 데이터를 수집하였다. 실험에 적용된 망은 12개로 그림 6에 제시되어 있다. N_2, N_4는 Wikipedia[11]에서, N_3는 Wagner[4] N_5, N_6, N_7, N_8, N_9는 Ikeda[12]에서, $N_{10}, N_{11}, N_{12}, N_{13}$은 Chen[13]에서 인용되었다.



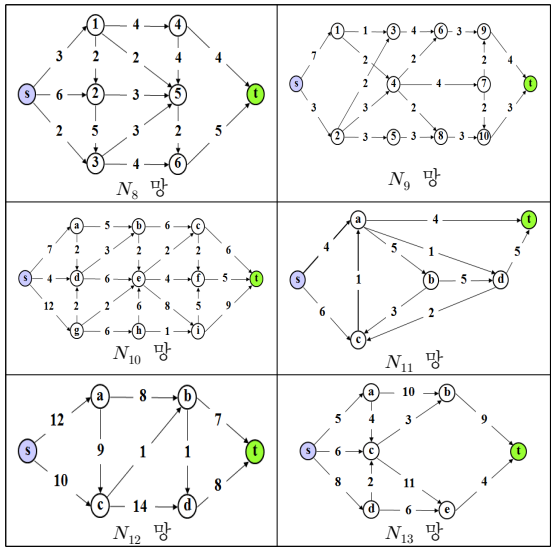


그림 6. 실험 데이터
Fig. 6. Numerical Examples

그림 6의 실험 망에 대해 제안된 알고리즘을 수행하는 과정과 결과는 그림 7~그림 18에 제시하였다. 제안된 알고리즘은 N_8 망에 대해서는 $d_G(t)\,d_G(t)=2$, $T=\{6,t\}$에서 $T=\{4,6,t\}$가 아닌 $T=\{4,t\}$로 할 경우에만 최소 절단을 구하는 경우가 적용되었으며, 나머지 모든 망에 대해서는 일반적인 법칙이 적용되었다.

호	V	S	T	$\sum S_o$, $\sum T_i$
초기치	$\{a,b,c,d,e\}$	$\{s\}$	$\{t\}$	$\sum S_o = 3+3=6$ $\sum T_i = 9+1=10$ $C = \min\{6,10\} = 6$
$(e,t)=9$	$\{a,b,c,d\}$	-	$\{e,t\}$	$\sum T_i = 6+1=7$
$(c,e)=6$	$\{a,b,d\}$	-	$\{c,e,t\}$	$\sum T_i = 3+1+1=5$ $C = \min\{6,5\} = 5$
$(a,b)=4$	$\{d\}$	-	-	H 집합
$(s,c)=3$ $(s,a)=3$	X $\{d\}$	- $\{s,a,b\}$	- -	$\sum S_o = 7$
$(b,d)=2$	$\{\varnothing\}$	$\{s,a,b,d\}$	-	$\sum S_o = 3+1+1=5$ $C = \min\{5,5\} = 5$
$s-t$ min cut		$\{s,a,b,d\}$	$\{c,e,t\}$	$\min c(S,T) = 5$

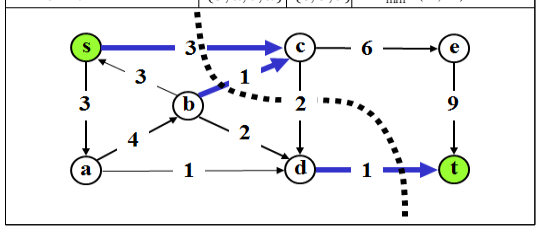


그림 7. N_2 망의 최대 수용량-기반 최소 절단
Fig. 7. Min-Cut Algorithm for N_2 Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{a,b,c,d\}$	$\{s\}$	$\{t\}$	$\sum S_o = 5+2 = 7$ $\sum T_i = 2+5 = 7$ $C = \min\{7,7\} = 7$
$(s,a) = 5$ $(d,t) = 5$	$\{b,c,d\}$	$\{s,a\}$	-	$\sum S_o = 3+2+2 = 7$
$(d,t) = 5$	$\{b,c\}$	-	$\{d,t\}$	$\sum T_i = 3+2+2 = 7$
$(a,c) = 3$ $(b,d) = 3$	$\{b\}$	$\{s,a,c\}$	-	$\sum S_o = 2+2+2 = 6$ $C = \min\{7,6\} = 6$
$(b,d) = 3$	$\{\varnothing\}$	-	$\{b,d,t\}$	$\sum T_i = 2+2+2 = 6$ $C = \min\{6,6\} = 6$
$s-t$ min cut		$\{s,a,c\}$	$\{b,d,t\}$	$\min c(S,T) = 6$

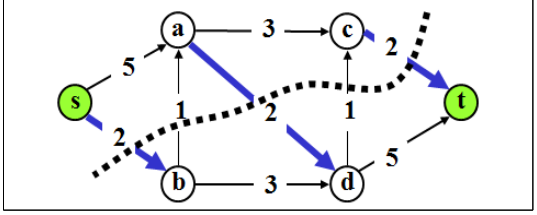


그림 8. N_3 망의 최대 수용량-기반 최소 절단
Fig. 8. Min-Cut for N_3 Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{1,2,3,\cdots,8,9\}$	$\{s\}$	$\{t\}$	$C = 13$
$(7,t) = 5$ $(4,7) = 5$ $(s,1) = 5$ $(s,2) = 5$ $(s,3) = 5$ $(2,5) = 5$	$\{1,2,\cdots,6,8,9\}$ $\{1,2,3,5,6,8,9\}$ $\{2,3,5,6,8,9\}$ $\{3,5,6,8,9\}$ $\{5,6,8,9\}$ $\{6,8,9\}$	- - $\{s,1\}$ $\{s,1,2\}$ $\{s,1,2,3\}$ $\{s,1,2,3,5\}$	$\{7,t\}$ $\{4,7,t\}$ - - - -	$\sum T_i = 15$ $\sum T_i = 14$ $\sum S_o = 15$ $\sum S_o = 19$ $\sum S_o = 15$ $\sum S_o = 16$
$(8,t) = 4$ $(9,t) = 4$	$\{6,9\}$ $\{6\}$	- -	$\{4,7,8,t\}$ $\{4,7,8,9,$	$\sum T_i = 13$ $\sum T_i = 14$
$(5,8) = 3$ $(6,9) = 3$	$\{6\}$ $\{\varnothing\}$	-	$\{4,5,7,8,$ $\{4,5,\cdots,9$	$\sum T_i = 14$ $\sum T_i = 15$
$s-t$ min cut		$\{s,1,\cdots,9\}$ $\{s,1,2,3,6,9\}$	$\{t\}$ $\{4,7,8,t\}$	$\min c(S,T) = 13$

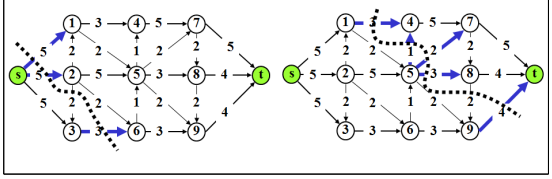


그림 10. N_5 망의 최대 수용량-기반 최소 절단
Fig. 10. Min-Cut for N_5 Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{o,p,q,r\}$	$\{s\}$	$\{t\}$	$\sum S_o = 6,\ \sum T_i = 5$ $C = \min\{6,5\} = 5$
$(q,r) = 4$	$\{o,p\}$	-	-	$H = \{q,r\}$
$(s,o) = 3$	$\{p\}$	$\{s,o\}$	-	$\sum S_o = 3+2+3 = 8$ $C = \min\{5,8\} = 5$
$(s,p) = 3$	$\{\varnothing\}$	$\{s,o,p\}$	-	$\sum S_o = 3+2 = 5$ $C = \min\{5,5\} = 5$
$s-t$ min cut		$\{s,o,p,q,r\}$ $\{s,o,p\}$	$\{t\}$ $\{q,r,t\}$	$\min c(S,T) = 5$

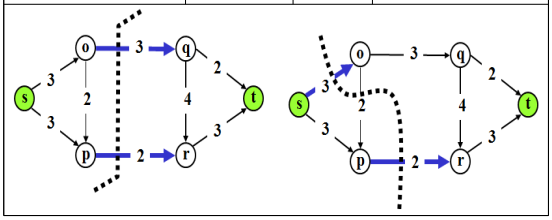


그림 9. N_4 망의 최대 수용량-기반 최소 절단
Fig. 9. Min-Cut for N_4 Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{u,v,\cdots,z\}$	$\{s\}$	$\{t\}$	$\sum S_o = 2+7 = 9$ $\sum T_i = 5+3+4$ $C = 9$
$(s,u) = 7$	$\{v,w,x,y,z\}$	$\{s,u\}$	-	$\sum S_o = 2+3+3$ $C = \min\{9,8\} = 8$
$(x,t) = 5$	$\{v,w,y,z\}$	-	$\{x,t\}$	$\sum T_i = 1+3+3$
$(z,t) = 4$ $(w,z) = 4$	$\{v,w,y\}$ $\{v,y\}$	- -	$\{x,z,t\}$ $\{x,w,z,t\}$	$\sum T_i = 1+3+3$ $\sum T_i = 1+3+3$
$(y,t) = 3$	$\{v\}$	-	$\{x,w,y,z,t\}$	$\sum T_i = 1+3+3$ $C = \min\{8,7\} = 7$
$(s,v) = 2$	$\{\varnothing\}$	$\{s,u,v\}$	-	$\sum S_o = 1+3+3$ $C = \min\{7,7\} = 7$
$s-t$ min cut		$\{s,u,v\}$	$\{x,w,y,z,t\}$	$\min c(S,T) = 7$

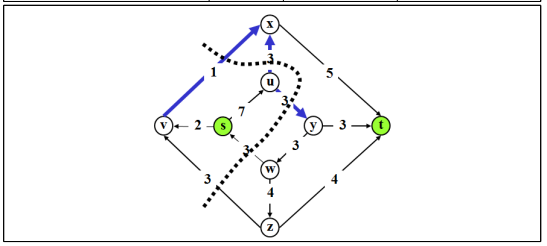


그림 11. N_6 망의 최대 수용량-기반 최소 절단
Fig. 11. Min-Cut for N_6 Network based on Max Capacity

호	V	S	T	ΣS_o, ΣT_i
초기치	$\{u,v,w\}$	$\{s\}$	$\{t\}$	$C=\min\{15,19\}=15$
$(w,t)=8$	$\{u,v\}$	-	$\{w,t\}$	$\Sigma T_i=3+5+6=14$ $C=\min\{15,14\}=14$
$(s,v)=7$	$\{u\}$	$\{s,v\}$	-	$\Sigma S_o=15$
$(u,t)=6$	$\{\varnothing\}$	-	$\{u,w,t\}$	$\Sigma T_i=15$
$s-t$ min cut		$\{s,u,v\}$	$\{w,t\}$	$\min c(S,T)=14$

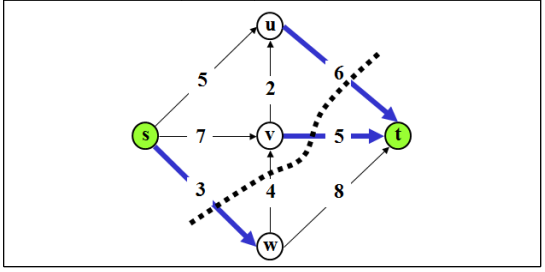


그림 12. N_7 망의 최대 수용량-기반 최소 절단
Fig. 12. Min-Cut for N_7 Network based on Max Capacity

호	V	S	T	ΣS_o, ΣT_i
초기치	$\{1,2,\cdots,6\}$	$\{s\}$	$\{t\}$	$C=\min\{11,9\}=9$
$(s,2)=6$	$\{1,3,4,5,6\}$	$\{s,2\}$	-	$\Sigma S_o=13$
$(6,t)=5$ $(2,3)=5$	$\{1,3,4,5\}$ $\{1,4,5\}$	- $\{s,2,3\}$	$\{6,t\}$ -	$\Sigma T_i=10$ $\Sigma S_o=13$
$(4,t)=4$ $(1,4)=4$ $(4,5)=4$ $(3,6)=4$	$\{1,5\}$ $\{5\}$ $\{\varnothing\}$ -	- - - -	$\{4,t\}$ $\{1,4,t\}$ $\{1,4,5,$ $\{3,6,t\}$	$\Sigma T_i=4+5=9$ $\Sigma T_i=3+5=8$ $\Sigma T_i=14$ $\Sigma T_i=13$
$s-t$ min cut		$\{s,2,3,5,6\}$	$\{1,4,t\}$	$\min c(S,T)=8$

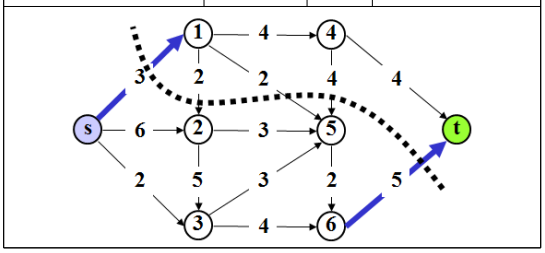


그림 13. N_8 망의 최대 수용량-기반 최소 절단
Fig. 13. Min-Cut for N_8 Network based on Max Capacity

호	V	S	T	ΣS_o, ΣT_i
초기치	$\{1,2,\cdots,10\}$	$\{s\}$	$\{t\}$	$C=7$
$(s,1)=7$	$\{2,3,\cdots,9,10\}$	$\{s,1\}$	-	$\Sigma S_o=6$ $C=\min\{7,6\}=6$
$(9,t)=4$ $(3,6)=4$ $(4,7)=4$	$\{2,3,\cdots,8,10\}$ $\{2,4,5,7,8,10$ $\{2,5,8,10\}$	- - -	$\{9,t\}$ - -	$\Sigma T_i=8$ $H=\{3,6\}$ $H=\{3,6\},\{4,7\}$
$(10,t)=3$ $(8,10)=3$ $(5,8)=3$ $(2,5)=3$	$\{2,5,8\}$ $\{2,5\}$ $\{2\}$ $\{\varnothing\}$	- - - -	$\{10,t\}$ $\{8,10,t\}$ $\{5,8,10,t\}$ $\{2,5,8,10,t\}$	$\Sigma T_i=9$ $\Sigma T_i=11$ $\Sigma T_i=11$ $\Sigma T_i=11$
$(s,2)=3$ $(2,3)=3$ $(2,4)=3$ $(3,6)=3$ $(6,9)=3$	- - - - -	- - - - -	- - - - -	- - - - -
$s-t$ min cut		$\{s,1\}$	$\{2,3,\cdots,10,t\}$	$\min c(S,T)=6$

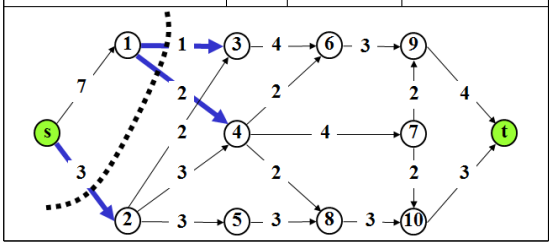


그림 14. N_9 망의 최대 수용량-기반 최소 절단
Fig. 14. Min-Cut for N_9 Network based on Max Capacity

호	V	S	T	ΣS_o,ΣT_i
초기치	$\{a,b,\cdots,h,i\}$	$\{s\}$	$\{t\}$	$C=20$
$(s,g)=12$	$\{a,b,\cdots f,h,i\}$	$\{s,g\}$	-	$\Sigma S_o=21$
$(i,t)=9$	$\{a,b,\cdots f,h\}$	-	$\{i,t\}$	$\Sigma T_i=20$
$(e,i)=8$	$\{a,b,c,d,f,h\}$	-	$\{e,i,t\}$	$\Sigma T_i=28$
$(s,a)=7$	$\{b,c,d,f,h\}$	$\{s,a,g\}$	-	$\Sigma S_o=21$
$(c,t)=6$ $(b,c)=6$ $(h,e)=6$ $(g,h)=6$ $(d,e)=6$	$\{b,d,f,h\}$ $\{d,f,h\}$ $\{d,f\}$ - $\{f\}$	- - - - -	$\{c,e,i,t\}$ $\{b,c,e,i,t\}$ $\{b,c,e,h,i,t\}$ - $\{b,c,d,e,h,i,t\}$	$\Sigma T_i=28$ $\Sigma T_i=28$ $\Sigma T_i=27$ - $\Sigma T_i=26$
$(f,t)=5$ $(i,f)=5$ $(a,b)=5$	$\{\varnothing\}$ - -	- - -	$\{b,c,d,e,f,h,i,t\}$	$\Sigma T_i=21$
$s-t$ min cut		$\{s,a,\cdots,i\}$ $\{s,a,\cdots,h\}$	$\{t\}$ $\{i,t\}$	$\min c(S,T)=20$

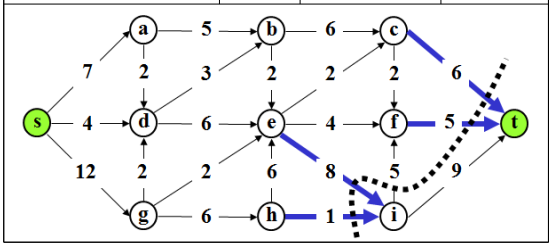


그림 15. N_{10} 망의 최대 수용량-기반 최소 절단
Fig. 15. Min-Cut for N_{10} Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{a,b,c,d\}$	$\{s\}$	$\{t\}$	$C=\min\{10,9\}=9$
$(s,c)=6$	$\{a,b,d\}$	$\{s,c\}$	-	$\sum S_o=4+1=5$ $C=\min\{9,5\}=5$
$(d,t)=5$ $(b,d)=5$ $(a,b)=5$	$\{a,b\}$ $\{a\}$ $\{\varnothing\}$	- - -	$\{d,t\}$ $\{b,d,t\}$ $\{a,b,d,t\}$	$\sum T_i=12$ $\sum T_i=10$ $\sum T_i=5$
$s-t$ min cut		$\{s,c\}$	$\{a,b,d,t\}$	$\min c(S,T)=5$

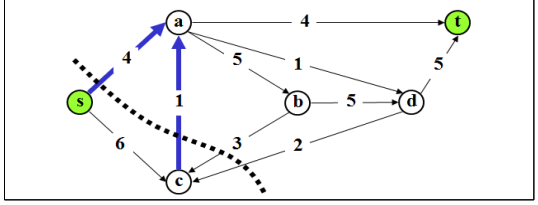


그림 16. N_{11} 망의 최대 수용량-기반 최소 절단
Fig. 16. Min-Cut for N_{11} Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{a,b,c,d\}$	$\{s\}$	$\{t\}$	$C=\min\{22,15\}=15$
$(c,d)=14$	$\{a,b\}$	-	-	$H=\{c,d\}$
$(s,a)=12$	$\{b\}$	$\{s,a\}$	-	$\sum S_o=10+9+8=27$
$(s,c)=10$	$\{b\}$	$\{s,a,c,d\}$	-	$\sum S_o=8+1+8=17$
$(a,c)=9$	$\{b\}$	x	-	-
$(d,t)=8$ $(a,b)=8$	$\{b\}$ $\{\varnothing\}$	x $\{s,a,b,c,d\}$	- -	- $\sum S_o=7+8=15$
$s-t$ min cut		$\{s,a,b,c,d\}$	$\{t\}$	$\min c(S,T)=15$

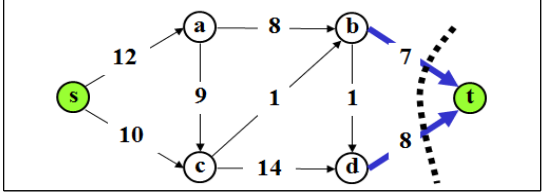


그림 17. N_{12} 망의 최대 수용량-기반 최소 절단
Fig. 17. Min-Cut for N_{12} Network based on Max Capacity

호	V	S	T	$\sum S_o,\ \sum T_i$
초기치	$\{a,b,c,d,e\}$	$\{s\}$	$\{t\}$	$C=\min\{19,13\}=13$
$(c,e)=11$	$\{a,b,d\}$	-	-	$H=\{c,e\}$
$(a,b)=10$	$\{d\}$	-	-	$H_1=\{c,e\},H_2=\{a,b\}$
$(b,t)=9$	$\{d\}$	-	$\{a,b,t\}$	$\sum T_i=5+3+4=12$ $C=\min\{13,12\}=12$
$(s,d)=8$	$\{\varnothing\}$	$\{s,d\}$	-	$\sum S_o=5+6+2+6=19$
$s-t$ min cut		$\{s,c,d,e\}$	$\{a,b,t\}$	$\min c(S,T)=12$

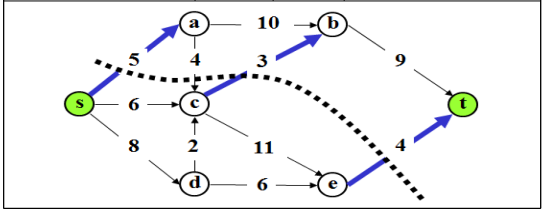


그림 18. N_{13} 망의 최대 수용량-기반 최소 절단
Fig. 18. Min-Cut for N_{13} Network based on Max Capacity

최대 수용량 기반 알고리즘을 12개 망에 적용한 결과 모든 망에서 S와 T 집합을 정확히 분리할 수 있었으며, $\min c(S,T)$를 얻는데 성공하였다. 알고리즘 수행 과정에서 비교되는 호의 수는 최소 $|N|-2$에서 최대 $|N|+1$개만이 사용되었다. 제안된 알고리즘은 수행 복잡도가 $O(a\log a)$로 기존의 Ford-Fulkerson 알고리즘의 증대경로를 찾고 잉여량을 계산하는 방법에 비해 간단하면서도 $\min c(S,T)$도 제공함을 알 수 있다.

제안된 알고리즘은 13개의 한정된 망에 대해 적용하여 성능이 우수함을 검증하였다. 추후에는 최소절단 문제가 제기된 기원인 냉전시대에 구 소련에서 동구권 소련연방의 여러 나라로 물자를 철도로 수송하는 문제[14]에 대한 최소절단 값을 구하는데 본 알고리즘을 적용할 예정이다. 이 데이터는 구 소련에는 9개의 공급지가 있는 반면 목적지는 명시되지 않은 경우로 본 알고리즘의 단일 s와 단일 t가 정해진 문제와는 차이점이 있다. 따라서 본 알고리즘을 변형시켜 적용해야 할 것으로 예상된다.

V. 결론 및 추후 연구과제

s와 t가 주어진 망에 대한 최소 절단을 찾는 Ford-Fulkerson 알고리즘은 유동망을 구하고, 단지 s에서 t로의 증대경로에서 병목지점들을 제시하고 있으며, 최소 절단을 결정하지 못하는 단점이 있었다. 본 논문은 유동망을 구하지 않고 직접 최소절단을 구하는 알고리즘을 제안하였다. 제안된 방법은 최대 수용량 호에서는 병목현상이 발생하지 않으며, 최대 수용량 호의 양 끝단 정점은 하나의 정점으로 통합시킬 수 있다는 특징에 기반하였다. 따라서 단순히 최대 수용량 호를 인접한 $s\in S$ 또는 $t\in T$의 원소로 포함시키는 기준을 적용하여 최소 절단을 간단히 구하였다. 13개의 한정된 그래프에 대해서는 제안된 알고리즘의 적합성이 검증되었다. 추후 구 소련의 철도망에 대해 적용성 검증이 필요할 것으로 판단된다.

참고문헌

[1] J. W. Chinneck, "Practical Optimization: A Gentle Introduction," http://www.sce.carleton.ca/faculty/chinneck/po.html, 2000.

[2] T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein, "Introduction to Algorithms," 2nd Ed., Section 26: Maximum Flow," pp. 643-698, MIT Press, 2005.

〔3〕 B. Korte and J. Vygen, "Combinatorial Optimization: Theory and Algorithm, Section 8.1 Maximum-Flow Min-Cut Theorem," 4th Ed., pp. 166-186, Springer-Verlang, 2008.
〔4〕 D. Wagner, "CS 170: Efficient Algorithms and Intractable Problems," UC Berkeley, 2003.
〔5〕 H. Nagamochi and T. Ibaraki, "Computing Edge Connectivity in Multigraphs and Capacitated Graphs," SIAM Journal of Discrete Mathematics, Vol. 5, No. 1, pp. 54-66, 1992.
〔6〕 D. R. Karger and C. Stein, "A New Approach to the Minimum Cut Problem," Journal of the ACM, Vol. 43, Issue. 4, pp. 601-640, 1996.
〔7〕 D. S. Hochbaum, "The Minimum/Maximum Cut Problem, " RICT, UC Berkeley, http://riot.ieor.berkeley.edu/riot/ Applications/ WeightedMinCut/, 1999.
〔8〕 M. S. Levine, "Experimental Study of Minimum Cut Algorithms," Master Thesis, MIT, 1997.
〔9〕 G. Deng, M. Tang, and G. Chen, "An Improved Algorithm for Maximum Flow Problem," International Conference on Communications, Circuits and Systems(ICCAS), pp. 591-594, 2009.
〔10〕 A. V. Goldberg, "A New Approach to the Maximum Flow Problem," Journal of the ACM, Vol. 35, pp. 921-940, 1988.
〔11〕 Wikipedia,"Edmonds-KrapAlgorithm," http://en.wikipedia.org/wiki/Edmonds_Krap_Algorithm, Wikimedia Foundation Inc., 2007.
〔12〕 K. Ikeda, "Mathematical Programming," Dept. Information Science and Intelligent Systems, The University of Tokushima, http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/maxflow/MaxflowApp. shtml. 2005.
〔13〕 WWL.Chen,"Discrete Mathematics," Department of Mathematics, Division of ICS, Macquarie University, Australia, http://www.maths.mq.edu.au/~wchen/ /lndmfolder/ lndm.html, 2003.
〔14〕 A. Schrijver, "On the History of the Transportation and Maximum Flow Problems," http://homepages. cwi.nl/~lex/files/histrpclean.pdf.Departmentof Mathematics, University of Amsterdam, Netherlands, 2004.

저 자 소 개

이 상 운

1983년~1987년 : 한국항공대학교 항공전자공학과 (학사)
1995년~1997년 : 경상대학교 컴퓨터과학과 (석사)
1998년~2001년 : 경상대학교 컴퓨터과학과 (박사)
2003년 : 강원도립대학 컴퓨터응용과 전임강사
2004년~2007.2 : 국립 원주대학 여성교양과 조교수
2007.3~현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수
관심분야 :소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도(소프트웨어 규모, 개발 노력, 개발기간, 팀 규모), 분석과 설계 방법론, 소프트웨어 시험 및 품질 보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지,, 그래프 알고리즘
e-mail : sulee@gwnu.ac.kr