http://dx.doi.org/10.7236/JIWIT.2012.12.5.129
JIWIT 2012-5-17

최대 인접 병합 방법을 적용한 방향 그래프의 병목지점 탐색 알고리즘

A Bottleneck Search Algorithm for Digraph Using Maximum Adjacency Merging Method

이상운*

Sang-Un, Lee

요 약 공급처 s와 수요처 t, 호가 수용량을 갖고 있는 방향 그래프 망 $D=(N,A), n\in N, a=c(u,v)\in A$에 대해, 공급처 s에서 수요처 t로의 최대 흐름양은 N을 $s\in S$와 $t\in T$의 집합으로 분리시키는 최소절단값이 결정한다. 최소절단을 찾는 대표적인 알고리즘으로는 수행복잡도 $O(NA^2)$의 Ford-Fulkerson이 있다. 이 알고리즘은 가능한 모든 증대경로를 탐색하여 병목지점을 결정한다. 알고리즘이 종료되면 병목지점들의 조합으로 $N=S+T$의 절단이 되는 최소 절단을 결정해야 한다. 본 논문은 $S=\{s\}, T=\{t\}$를 초기값으로 설정하고, 망의 최대 수용량 호 $\max c(u,v)$를 인접한 S나 T로 병합시키고 절단값을 구하는 최대인접병합 알고리즘을 제안하였다. 최대인접병합 알고리즘은 $n-1$회를 수행하지만 알고리즘 수행 과정에서 최소절단을 찾는 장점을 갖고 있다. Ford-Fulkerson과 최대인접병합 알고리즘을 다양한 8개의 방향 그래프에 적용한 결과 제안된 알고리즘은 수행복잡도 $O(N)$인 $n-1$회 수행 과정에서 최소절단을 쉽게 찾을 수 있었다.

Abstract Given digraph network $D=(N,A), n\in N, a=c(u,v)\in A$ with source s and sink t, the maximum flow from s to t is determined by cut (S,T) that splits N to $s\in S$ and $t\in T$ disjoint sets with minimum cut value. The Ford-Fulkerson (F-F) algorithm with time complexity $O(NA^2)$ has been well known to this problem. The F-F algorithm finds all possible augmenting paths from s to t with residual capacity arcs and determines bottleneck arc that has a minimum residual capacity among the paths. After completion of algorithm, you should be determine the minimum cut by combination of bottleneck arcs. This paper suggests maximum adjacency merging and compute cut value method is called by MA-merging algorithm. We start the initial value to $S=\{s\}, T=\{t\}$, Then we select the maximum capacity $\max c(u,v)$ in the graph and merge to adjacent set S or T. Finally, we compute cut value of S or T. This algorithm runs $n-1$ times. We experiment Ford-Fulkerson and MA-merging algorithm for various 8 digraph. As a results, MA-merging algorithm can be finds minimum cut during the $n-1$ running times with time complexity $O(N)$.

Key Words : Max-flow/Min-cut, nodes partition, augmenting path, Maximum Adjacency (MA)-merging

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2012년 6월 18일, 수정완료 : 2012년 8월 23일
게재확정일자 : 2012년 10월 12일

Received: 18 June 2012 / Revised: 23 August 2012 / Accepted: 12 October 2012
**Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

Ⅰ. 서 론

Menger[1]의 "최대유동/최소절단 (Max-Flow/ Min-Cut) 정리"는 주어진 망 $D=(N,A), n\in N, a=(u,v)\in A$에 대해 공급처 (source, s)와 수요처 (sink, t)가 주어지며, u에서 v로의 호 $a=c(u,v)$는 수용량을 가중치로 갖고 있다. 이 경우 s에서 t로 흐를 수 있는 양을 최대 유동량 (maximum flow)이라 하며, 이 양은 모든 노드들 N을 최소의 수용량 합을 기준으로 $N=S+T$, $s\in S$, $t\in T$의 2개 집합으로 양분시키는 절단인 병목지점이 결정한다. 이를 최소 절단 (minimum cut)이라 하며, $_{\min}c(S,T)$로 표기한다. 즉, 망에서 최대로 흐를 수 있는 유동량은 병목현상이 발생하는 지점들의 수용량 합이 최소가 되는 최소 절단값 $_{\min}c(S,T)$이 결정한다. 따라서 망의 최대 유동량을 구하기 위해서는 망의 노드들 N을 최소 절단 값을 갖는 면을 기준으로 S와 T의 두 집합으로 정확히 분리시켜야 한다.

주어진 망에서 최소절단을 찾는 이유는 해당 최소절단의 수용량을 증가시키면 전체 망의 흐름양을 증가시킬 수 있기 때문이다. 이는 도로, 철도 등의 물류분야와 수도, 전기, 통신 등의 분야에 활용될 수 있는 중요한 문제이다.

주어진 복잡한 망의 최소절단을 찾기 위해 지금까지는 공급처 s에서 수요처 t로 추가로 흐를 수 있는 수용량을 가진 가능한 경로인 증대 경로 (augmenting path)를 찾아 이 경로상의 병목지점을 결정하는 방법을 주로 연구하고 있다. 이에 대한 대표적인 알고리즘으로 Ford-Fulkerson[2,3]과 Edmonds-Karp[4,5]이 있다. Ford-Fulkerson 알고리즘은 증대경로를 무작위로 찾는 방법이며, Edmonds-Karp 알고리즘은 가능한 증대경로들 중 최단경로를 찾는 차이점이 있다. Lee[6]는 증대경로를 찾지 않고 망의 호들 수용량을 내림차순으로 정렬하고 최대 수용량 호부터 선택하여 절단값을 구하는 수용량 역-추가 알고리즘을 제안하였다. s와 t가 주어지지 않은 무방향 가중 그래프에 대한 알고리즘으로는 Stoer-Wagner[7]가 있다.

본 논문에서는 s와 t가 주어진 경우에 한정한다. 이 문제에 대해, Ford-Fulkerson과 Edmonds-Karp 알고리즘은 s에서 t로의 증대경로를 계속적으로 찾아야 하는 어려움이 있으며, 알고리즘 수행 후에 결정된 병목지점들을 조합하여 최소 절단을 다시 결정해야 한다.

본 논문에서는 단일 s와 t가 주어진 방향 가중 그래프에서 대 수용량을 가진 노드를 인접한 $s\in S$나 $t\in T$로 단순히 병합시키고 절단값을 계산하는 방법을 제안한다. 이 방법을 최대인접병합 (maximum adjacent-ordering, MA-ordering) 방법이라 하자. 2장에서는 Ford- Fulkerson 알고리즘을 고찰한다. 3장에서는 최대인접병합을 단순히 $n-1$회 수행하여 최소절단을 찾는 알고리즘을 제안한다. 4장에서는 제안 알고리즘을 Ford- Fulkerson 알고리즘과 비교하여 성능을 평가해 본다.

Ⅱ. 관련 연구와 연구 배경

Ford-Fulkerson 알고리즘[2,3]은 $s\rightarrow t$의 임의의 증대경로 (p)를 찾아 증대경로 상의 최소 수용량 (병목지점) 값 $c_f(p)$으로 추가로 흐를 수 있는 유동량 f를 계산하고 $c-f=0$인 병목지점을 삭제하고 다시 $s\rightarrow t$의 가능한 증대경로를 찾아 유동량을 계산한다. 이 방법을 모든 가능한 $s\rightarrow t$ 증대경로가 존재하지 않을 때까지 수행한다. 이때 증대 경로상의 임의의 호가 $s\rightarrow t$의 순방향이면 $f=f+c_f(p)$, $t\rightarrow s$의 역방향이면 $f=f-c_f(p)$로 계산한다. Ford-Fulkerson 알고리즘은 그림 1에 제시하였다.

```
MinimumCut (G,c,s,t)
모든 간선 방향 (u,v)에 f(u,v)=0로 초기화.
WHILE 잉여 수용량 c_f(u,v)>0을 가진 s에서 t로의 경로 p 존재
    병목 수용량 c_f(p)=min{c_f(u,v)|(u,v)∈p} 결정.
    각 간선 (u,v)∈p에 대해
    if      f(u,v) then f(u,v)=f(u,v)+c_f(p) /* s→t 방향
    else if f(v,u) then f(v,u)=f(v,u)-c_f(p) /* s←t 방향
c-f=0인 간선들을 대상으로 최소절단을 구함.
```

그림 1. Ford-Fulkerson 최소 절단 알고리즘
Fig 1. Ford-Fulkerson Minimum-cut Algorithm

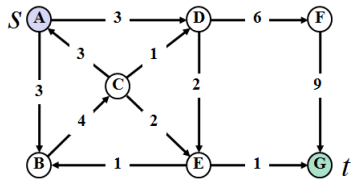


그림 2. G_1 그래프
Fig 2. G_1 Graph

그림 2의 G_1 그래프에 대해 Ford-Fulkerson 알고리즘으로 최소절단을 찾은 결과는 그림 3과 같다.[5] 그림 3에서 가능한 증대경로 4개를 찾아 각 증대경로에서의 병목지점 $(E,G),(A,D),(C,D),(E,D)$를 찾았으며, (A,G)의 최소절단 $_{\min}c(S,T)$는 $(E,G),(C,D),(A,D)$로 $3+1+1=5$를 다시 결정해야 한다.

이 그림에서 $s\rightarrow t$로의 순방향 경로는 $A\rightarrow D\rightarrow F\rightarrow G$, $A\rightarrow D\rightarrow E\rightarrow G$, $A\rightarrow B\rightarrow C\rightarrow D\rightarrow F\rightarrow G$, $A\rightarrow B\rightarrow C\rightarrow D\rightarrow E\rightarrow G$와 $A\rightarrow B\rightarrow C\rightarrow E\rightarrow G$의 5개가 존재한다.

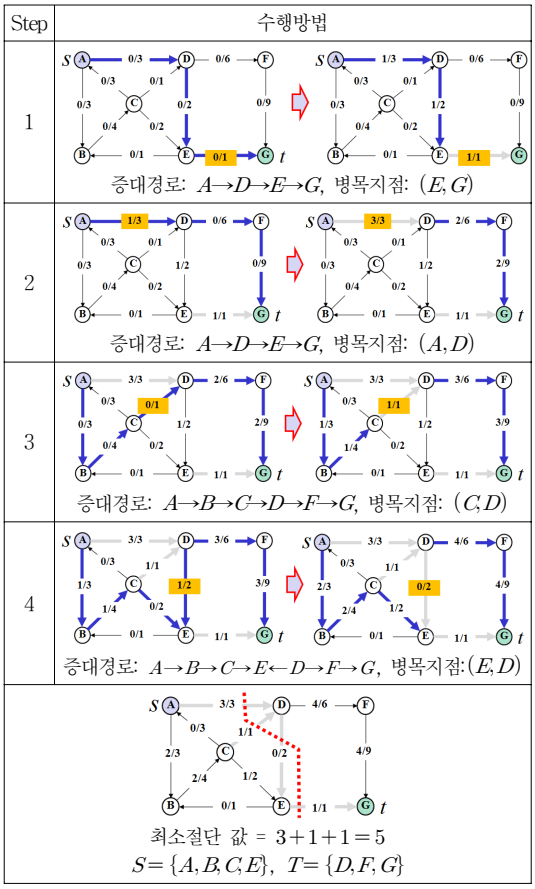


그림 3. G_1 그래프의 Ford-Fulkerson 최소 절단 알고리즘
Fig 3. Ford-Fulkerson Minimum-cut Algorithm for G_1 Graph

Ford-Fulkerson 알고리즘을 적용할 경우 증대경로를 탐색하는 과정이 쉽지 않다. 왜냐하면 경로상의 모든 간선의 방향이 $s\rightarrow t$의 순방향이면 쉽게 찾지만 Step 4의 $A\rightarrow B\rightarrow C\rightarrow E\leftarrow D\rightarrow F\rightarrow G$ 경로에서 $s\leftarrow t$역방향인 $E\leftarrow D$를 고려한 증대경로를 찾기는 쉽지 않기 때문이다.

또한, Ford-Fulkerson 알고리즘은 증대경로에서 병목지점은 찾을 수 있지만 더 이상의 증대경로가 없어 알고리즘이 종료되면 그 동안 찾은 병목지점들의 조합으로 최소절단을 결정해야만 한다. 엄밀히 말하면 Ford-Fulkerson 알고리즘은 병목지점들만 찾는 방법이라 할 수 있다.

또한, 이상운[6]은 증대경로를 찾지 않고 망의 호들 수용량을 내림차순으로 정렬하고 최대 수용량 호부터 선택하여 절단값을 구하는 그림 4의 수용량 역-추가 알고리즘을 제안하였다.

그래프 $D=(N,A)$.
초기값 : $S=\{s\}$, $T=\{t\}$, $U=\{\varnothing\}$, $N=N\setminus\{s,t\}$.
$\Sigma S_o, \Sigma T_i$ 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 : $O(n)$
최대값 $_{\max}c$ 간선 (u,v) 선택.
if $(u\in S)\cap(v\in S)$ or $(u\in T)\cap(v\in T)$ or $(u\in S)\cap(v\in T)$ then skip
else if $(u\in S)\cap(v\notin U)$ then $S=S+v$, ΣS_o 계산
$C=_{\min}\{C,\Sigma S_o\}$, $N=N\setminus v$
else if $(u\in S)\cap(v\in U)$ then $S=S+v+U$, ΣS_o 계산
$C=_{\min}\{C,\Sigma S_o\}$, $N=N\setminus v$
else if $(u\in T)\cap(v\notin U)$ then $T=T+v$, ΣT_i 계산
$C=_{\min}\{C,\Sigma T_i\}$, $N=N\setminus v$
else if $(u\in T)\cap(v\in U)$ then $T=T+v+U$, ΣT_i 계산
$C=_{\min}\{C,\Sigma T_i\}$, $N=N\setminus v$
else if $(u\notin S)\cap(v\notin T)$ then
if $(u\notin U)\cap(v\notin U)$ then
$U=\{u,v\}$,
$N=N\setminus\{u,v\}$
else if $(u\in U)\cap(v\notin U)$ then
$U=U+v$, $N=N\setminus v$.
end

그림 4. 수용량 역 추가-기반 최소 절단 탐색 알고리즘
Fig 4. Capacity Reverse Added-based Minimum-cut Algorithm

그림 4의 수용량 역-추가 알고리즘을 적용하는 과정에서 t에 그림 4의 방법을 적용하면 최소절단을 찾지 못하는 경우가 발생하여 다음과 같이 예외 기준을 적용하였다. $(u,t),(v,t)$로 t의 유입 차수가 2인 경우, 만약,

$T=\{u,t\}$로 병합된 상태라면 (v,t)에 대해서는 $T=\{u,v,t\}$ 대신 $T=\{v,t\}$를 수행한다. 만약, $U=\{u,v\}$로 병합된 상태라면 (u,t)에 대해서는 $T=\{u,v,t\}$를 적용한다. 또한 동일한 가중치 간선이 존재할 경우 선택 우선순위를 부여하였다.

Ⅲ. 최대인접병합 최소 절단 알고리즘

단일 s와 t가 주어진 방향 가중 그래프에 대한 최소 절단을 찾는 최대인접병합 알고리즘의 수행 방법은 다음과 같으며, 그림 5에 제시되어 있다.

(1) s는 유출간선만 존재하므로 유입간선은 삭제한다. 반대로, t는 유입간선만 존재하므로 유출간선은 삭제한다.

(2) S,T,U와 N 집합을 사용하며, 초기치는 $S=\{s\}$, $T=\{t\}$, $N=N\setminus\{s,t\}$, $U=\{\varnothing\}$이다. 여기서 S는 공급지 집합, T는 목적지 집합, U는 S와 T로의 병합이 미확정된 집합이다. N은 그래프의 모든 정점들 중에서 병합되지 않고 남아 있는 정점들의 집합이다. 망의 최대 흐름은 $\min\{\sum S,\sum T\}$를 초과할 수 없기 때문에 S의 유출 호 수용량 합 $\sum S$와 T의 유입 호 수용량 합 $\sum T$를 구한다.

(3) 망의 최대 수용량 간선 $\max c(u,v)$를 선택한다. 만약, 최대 수용량이 동일한 다수의 간선이 존재하면 S,T,U 순서로 선택한다. 민약, (S,T)이면 다음 최대 수용량 간선을 선택한다.

(4) 선택된 간선의 2개 정점 중 어느 하나가 S와 T 중 어느 집합의 원소인지 결정하여 해당 집합에 병합시킨다. 이를 최대인접 병합방법이라 한다. 인접정점들과의 간선 값을 변경한다. 이때 (1) 조건을 적용한다. 병합된 집합의 부속 간선 가중치 합을 절단값으로 계산한다.

최대인접병합 알고리즘과 Ford-Fulkerson 알고리즘의 차이점은 다음과 같다. Ford-Fulkerson 알고리즘은 s에서 t로의 증대경로를 찾아 잉여 수용량 (수용량-흐름양)을 계산하여 병목 지점을 결정한다. 더 이상의 증대경로가 없으면 병목 지점들을 대상으로 최소 절단을 결정한다. 반면에, 제안된 알고리즘은 단순히 그래프의 최대 수용량 간선 $\max c(u,v)$을 S나 T로 병합시키고 절단값을 계산한다. 즉, 제안된 알고리즘은 "망의 최대 수용량 간선은 최소절단에 포함되지 않는다."는 가정에 기반하고 있다.

MinCut(D,c,s,t)
  $S=\{s\}$, $T=\{t\}$, $N=N\setminus\{s,t\}$, $U=\{\varnothing\}$. /* 초기치 결정
  S의 유입 간선과 T의 유출간선 삭제.
  $\sum S$와 $\sum T$ 계산. /* $\sum S$: S의 유출간선 수용량 합
  /* $\sum T$: T의 유입간선 수용량 합
  $C=\min\{\sum S,\sum T\}$. /* 초기 최소 절단 값
  for $i=1$ to $n-1$, $n=|N|$ /* MA-merging
    최대 수용량 간선 $\max c(u,v)$선택 (단, 동일 수용량 간선이 다수 존재시 S,T,U 순서로 선택).
    if $u\in S, v\in T$ then skip
    else if $u\in N, v\in N$ then $U=\{u,v\}$, $N=N\setminus u,v$
    else if $u\in S, v\in N$ then $S=S+v$, $N=N\setminus v$
    else if $u\in S, v\in U$ then $S=S+U$, $U=\{\varnothing\}$
    else if $u\in T, v\in N$ then $T=T+v$, $N=N\setminus v$
    else if $u\in T, v\in U$ then $T=T+U$, $U=\{\varnothing\}$
    else if $u\in U, v\in N$ then $U=U+v$, $N=N\setminus v$.
    병합된 집합과 인접한 정점들 간의 간선들 수용량 재계산.
    (S의 유입 간선 삭제, T의 유출 간선 삭제)
    if 병합된 집합 = S 또는 T then
      절단 값 계산, $C=\min\{C,C_i\}$.
  end
최소 절단 값 : $\min c(S,T)=C$.

그림 5. 최대인접병합 알고리즘
Fig 5. Maximum Adjacency Merging Algorithm

7개의 노드를 갖고 있는 그림 3의 G_1 그래프에 대해 최대인접병합 알고리즘을 적용한 결과는 그림 6에 제시하였다. 제안 알고리즘은 첫 번째로 $s\in S, t\in T$에 대해, $c(S)=9, c(T)=10$를 계산하여 최소값 $c(S)=9$를 초기 최소 절단값으로 설정한다. 즉, 이 망은 공급지에서는 9를 흘려보내지만 수요지는 10을 받을 수 있는 용량을 갖고 있다. 따라서 공급지와 수요지만을 고려하면 이 망은 최대 9의 유동량을 갖고 있음을 알 수 있다. 다음으로, 망의 최대 수용량인 $(e,t)=9$를 선택하여 노드 e를 $t\in T$로 병합한 후 $c(T=\{e,t\})=7$을 계산한다. 이 값은 초기값 $c(S=\{s\})=9$보다 작으므로 $T=\{e,t\}$로 병합된 현 시점에서의 최소 절단값은 $c(T=\{e,t\})=7$로 변경된다. 이와 같은 방법으로 계속적으로 망의 최대 수용량 호의 두 노드를 병합하면서 병합된 S의 유출이나 T의 유입 호 가중치 (수용량) 합을 구하면서 최소절단값을 보다 작은 값으로 대체시키면서 $n-1$회에서 수행하면 최소절단값이

결정된다.

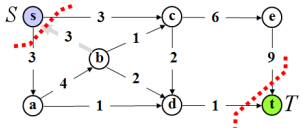
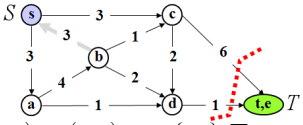
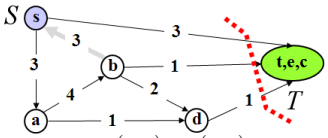
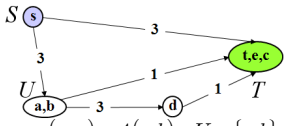
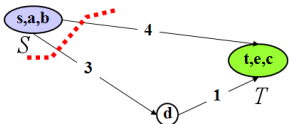
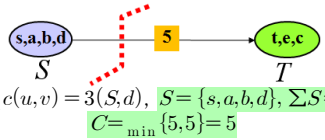
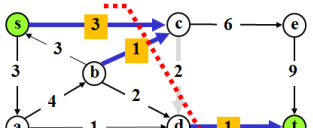
Step	수행 방법
1	 $S=\{s\},\ T=\{t\}$ $\sum S=3+3+3=9,\ \sum T=1+9=10,$ $C=\min\{9,10\}=9$
2	 $\max c(u,v)=9(e,T),\ \ T=\{t,e\},\ \sum T=1+6=7$ $C=\min\{9,7\}=7$
3	 $\max c(u,v)=6(c,T),$ $T=\{t,e,c\},\ \sum T=3+1+1=5,\ C=\min\{7,5\}=5$
4	 $\max c(u,v)=4(a,b),\ \ U=\{a,b\}$
5	 $\max c(u,v)=3(S,U),\ \ S=\{s,a,b\},\ \sum S=3+4=7$ $C=\min\{5,7\}=5$
6	 $\max c(u,v)=3(S,d),\ \ S=\{s,a,b,d\},\ \sum S=5$ $C=\min\{5,5\}=5$
	 최소절단 $\min c(S,T)=5,\ \ S=\{s,a,b,d\},\ T=\{c,e,t\}$

그림 6. G_1 **그래프의 최대인접병합 알고리즘**
Fig 6. Maximum Adjacency Merging Algorithm for G_1 Graph

Step 5에서는 $\max c(u,v)=3(S,T),3(S,U),3(U,d)$가 존재하였다. 여기서 S와 T는 병합할 수 없으며, S,T,U 순으로 수행하므로 $3(S,U)$가 병합되었다. Step 6도 동일한 기준에 의해 결정되었다. 결국, 제안된 알고리즘은 $\min c(S,T)=5$로 $N=\{s,a,b,c,d,e,t\}$를 $\min c(S,T)=5$인 $3(s,c),1(b,c),1(d,t)$를 기준으로 $S=\{s,a,b,d\},\ T=\{c,e,t\}$로 분할하였다.

Ⅳ. 알고리즘 적용 및 분석

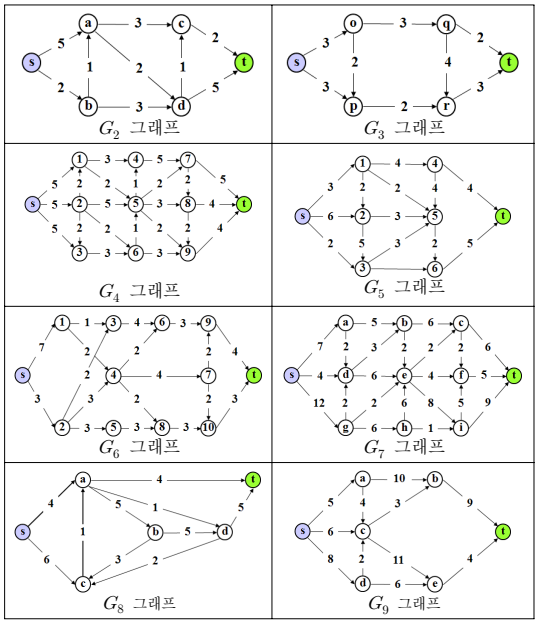
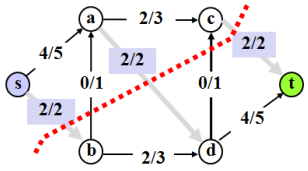


그림 7. 알고리즘 검증에 적용된 데이터
Fig 7. Experimental Data

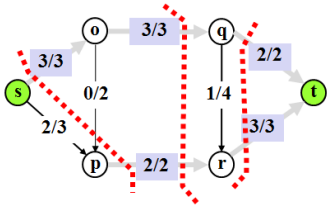
본 장에서는 제안된 알고리즘이 방향 그래프의 모든 노드 집합 N을 $s\in S, t\in T$인 공통 원소를 갖지 않는 두 집합으로 정확히 분할하는 최소 절단을 쉽게 찾을 수 있는지 고찰해 보자. 실험에 적용된 그래프는 8개로 그림 7에 제시되어 있다. G_2는 Wagner[9]에서, G_3는 Wikipedia[8]에서, G_4, G_5, G_6은 Ikeda[10]에서, G_7, G_8, G_9는 Chen[11]에서 인용되었다. s에서 t로의 순방향에 대해서만 Ford-Fulkerson 알고리즘을 적용하여 병목지점들을 찾은 결과에 대해 최소절단을 결정한 결과는 그림 8에 제시되어 있다. Ford-Fulkerson 알고리즘은 가능한 최대 증대경로 개수 f에 대해 알고리즘 수행 복잡도는 $O(Af)$이며,

Edmonds-Karp 알고리즘의 수행 복잡도는 $O(NA^2)$으로 Ford-Fulkerson 알고리즘 보다 효율적인 것으로 알려져 있다. 따라서 Ford-Fulkerson 알고리즘도 수행 복잡도는 $O(NA^2)$이라 할 수 있다.



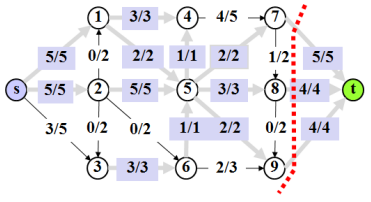
1. $s \xrightarrow{2/5} a \xrightarrow{2/3} c \xrightarrow{2/2} t$, 병목지점 : (c,t)
2. $s \xrightarrow{2/3} a \xrightarrow{2/2} d \xrightarrow{2/5} t$, 병목지점 : (a,d)
3. $s \xrightarrow{2/2} b \xrightarrow{2/3} d \xrightarrow{2/3} t$, 병목지점 : (s,b)

(a) G_2 그래프



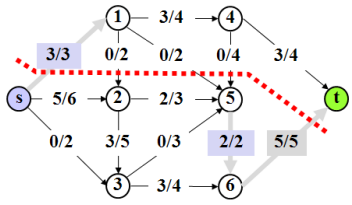
1. $s \xrightarrow{2/3} o \xrightarrow{2/3} q \xrightarrow{2/2} t$, 병목지점 : (b,t)
2. $s \xrightarrow{1/1} o \xrightarrow{1/1} q \xrightarrow{1/4} r \xrightarrow{1/3} t$, 병목지점 : $(s,o),(o,q)$
3. $s \xrightarrow{2/3} p \xrightarrow{2/2} r \xrightarrow{2/2} t$, 병목지점 : $(p,r),(r,t)$

(b) G_3 그래프



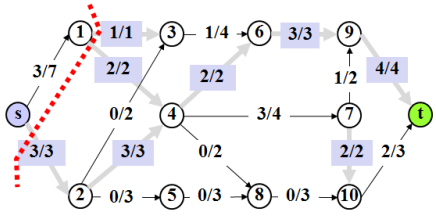
1. $s \xrightarrow{3/5} 1 \xrightarrow{3/3} 4 \xrightarrow{3/5} 7 \xrightarrow{3/5} t$, 병목지점 : $(1,4)$
2. $s \xrightarrow{1/2} 1 \xrightarrow{1/2} 5 \xrightarrow{1/1} 4 \xrightarrow{1/2} 7 \xrightarrow{1/2} t$, 병목지점 : $(5,4)$
3. $s \xrightarrow{1/1} 1 \xrightarrow{1/1} 5 \xrightarrow{1/2} 7 \xrightarrow{1/1} t$, 병목지점 : $(s,1),(1,5),(7,t)$
4. $s \xrightarrow{1/5} 2 \xrightarrow{1/5} 5 \xrightarrow{1/1} 7 \xrightarrow{1/2} 8 \xrightarrow{1/4} t$, 병목지점 : $(5,7)$
5. $s \xrightarrow{3/4} 2 \xrightarrow{3/4} 5 \xrightarrow{3/3} 8 \xrightarrow{3/3} t$, 병목지점 : $(5,8),(8,t)$
6. $s \xrightarrow{1/1} 2 \xrightarrow{1/1} 5 \xrightarrow{1/2} 9 \xrightarrow{1/4} t$, 병목지점 : $(s,2),(2,5)$
7. $s \xrightarrow{1/5} 3 \xrightarrow{1/3} 6 \xrightarrow{1/1} 5 \xrightarrow{1/1} 9 \xrightarrow{1/3} t$, 병목지점 : $(6,5),(5,9)$
8. $s \xrightarrow{2/4} 3 \xrightarrow{2/2} 6 \xrightarrow{2/3} 9 \xrightarrow{2/2} t$, 병목지점 : $(3,6),(9,t)$

(c) G_4 그래프



1. $s \xrightarrow{3/3} 1 \xrightarrow{3/4} 4 \xrightarrow{3/4} t$, 병목지점 : $(s,1)$
2. $s \xrightarrow{2/6} 2 \xrightarrow{2/3} 5 \xrightarrow{2/2} 6 \xrightarrow{2/5} t$, 병목지점 : $(5,6)$
3. $s \xrightarrow{3/4} 2 \xrightarrow{3/5} 3 \xrightarrow{3/4} 6 \xrightarrow{3/3} t$, 병목지점 : $(6,t)$

(d) G_5 그래프

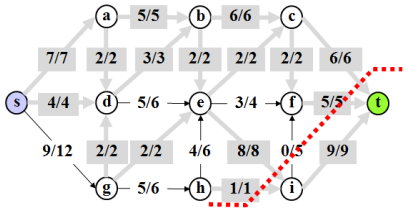


1. $s \xrightarrow{1/7} 1 \xrightarrow{1/1} 3 \xrightarrow{1/4} 6 \xrightarrow{1/3} 9 \xrightarrow{1/4} t$, 병목지점 : $(1,3)$
2. $s \xrightarrow{2/6} 1 \xrightarrow{2/2} 4 \xrightarrow{2/2} 6 \xrightarrow{2/2} 9 \xrightarrow{2/3} t$.

   병목지점 : $(1,4),(4,6),(6,9)$
3. $s \xrightarrow{1/3} 2 \xrightarrow{1/3} 4 \xrightarrow{1/4} 7 \xrightarrow{1/2} 9 \xrightarrow{1/1} t$, 병목지점 : $(9,t)$
4. $s \xrightarrow{2/2} 2 \xrightarrow{2/2} 4 \xrightarrow{2/3} 7 \xrightarrow{2/2} 10 \xrightarrow{2/3} t$,

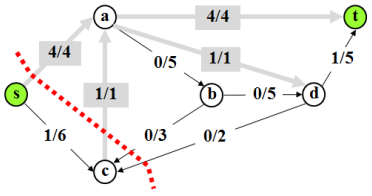
   병목지점 : $(s,2),(2,4),(7,10)$

(e) G_6 그래프



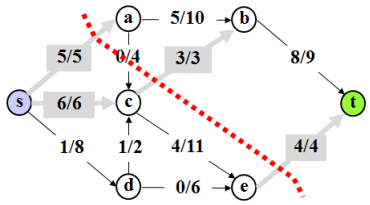
1. $s \xrightarrow{5/7} a \xrightarrow{5/5} b \xrightarrow{5/6} c \xrightarrow{5/6} t$, 병목지점 : (a,b)
2. $s \xrightarrow{1/2} a \xrightarrow{1/2} d \xrightarrow{1/3} b \xrightarrow{1/1} c \xrightarrow{1/1} t$, 병목지점 : $(b,c),(c,t)$
3. $s \xrightarrow{1/1} a \xrightarrow{1/1} d \xrightarrow{1/2} b \xrightarrow{1/2} e \xrightarrow{1/2} c \xrightarrow{1/2} f \xrightarrow{1/5} t$,
   병목지점:$(s,a),(a,d)$
4. $s \xrightarrow{1/4} d \xrightarrow{1/1} b \xrightarrow{1/1} e \xrightarrow{1/1} c \xrightarrow{1/1} f \xrightarrow{1/4} t$,
   병목지점 : $(d,b),(b,e),(e,c),(c,f)$
5. $s \xrightarrow{3/3} d \xrightarrow{3/6} e \xrightarrow{3/4} f \xrightarrow{3/3} t$, 병목지점 : $(s,d),(f,t)$
6. $s \xrightarrow{2/12} g \xrightarrow{2/2} d \xrightarrow{2/3} e \xrightarrow{2/8} i \xrightarrow{2/9} t$, 병목지점 : (g,d)
7. $s \xrightarrow{2/10} g \xrightarrow{2/2} e \xrightarrow{2/6} i \xrightarrow{2/7} t$, 병목지점 : (g,e)
8. $s \xrightarrow{4/8} g \xrightarrow{4/6} h \xrightarrow{4/6} e \xrightarrow{4/4} i \xrightarrow{4/5} t$, 병목지점 : (e,i)
9. $s \xrightarrow{1/4} g \xrightarrow{1/2} h \xrightarrow{1/1} i \xrightarrow{1/1} t$, 병목지점 : $(h,i),(i,t)$

(f) G_7 그래프



1. $s \xrightarrow{4/4} a \xrightarrow{4/4} t$, 병목지점 : $(s,a),(a,t)$
2. $s \xrightarrow{1/6} c \xrightarrow{1/1} a \xrightarrow{1/1} d \xrightarrow{1/5} t$, 병목지점 : $(c,a),(a,d)$

(g) G_8 그래프

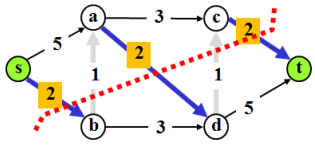


1. $s \xrightarrow{5/5} a \xrightarrow{5/10} b \xrightarrow{5/9} t$, 병목지점 : (s,a)
2. $s \xrightarrow{3/6} c \xrightarrow{3/3} b \xrightarrow{3/4} t$, 병목지점 : (c,b)
3. $s \xrightarrow{3/3} c \xrightarrow{3/11} e \xrightarrow{3/4} t$, 병목지점 : (s,c)
4. $s \xrightarrow{1/8} d \xrightarrow{1/2} c \xrightarrow{1/8} e \xrightarrow{1/1} t$, 병목지점 : (e,t)

(h) G_9 그래프

그림 8. 실험 데이터의 Ford-Fulkerson 알고리즘
Fig 8. Ford-Fulkerson Algorithm for Experimental Data

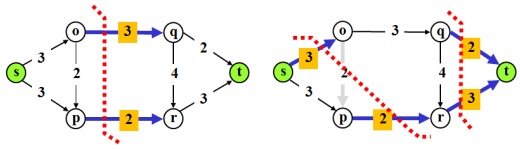
그림 9는 제안된 최대인접병합 알고리즘을 그림 7의 실험 데이터에 적용한 결과이다.



1. $S=\{s\},\ T=\{t\}$
   $\sum S=5+2=7, \sum T=2+5=7, C=_{\min}\{7,7\}=7$
2. $_{\max}c(u,v)=5(S,a),\ S=\{s,a\}, \sum S=2+2+3=7$
   $C=_{\min}\{7,7\}=7$
3. $_{\max}c(u,v)=5(d,T),\ T=\{t,d\}, \sum T=2+2+3=7$
   $C=_{\min}\{7,7\}=7$
4. $_{\max}c(u,v)=3(S,c),\ S=\{s,a,c\}, \sum S=2+4=6$
   $C=_{\min}\{7,6\}=6$
5. $_{\max}c(u,v)=3(b,T),\ T=\{t,d,b\}, \sum T=6$
   $C=_{\min}\{6,6\}=6$

   최소절단 $_{\min}c(S,T)=6,\ S=\{s,a,c\},\ T=\{b,d,t\}$

(a) G_2 그래프

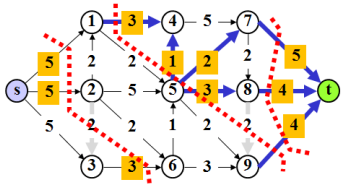


1. $S=\{s\}$, $T=\{t\}$
   $\sum S=3+3=6$, $\sum T=2+3=5, C=\min\{6,5\}=5$
2. $\max c(u,v)=4(q,r)$, $U=\{q,r\}$
3. $\max c(u,v)=5(U,T)$, $T=\{t,q,r\}, \sum T=3+2=5$
   $C=\min\{5,5\}=5$
4. $\max c(u,v)=3(S,o)$, $S=\{s,o\}, \sum S=3+5=8$
   $C=\min\{5,8\}=5$
5. $\max c(u,v)=5(S,p)$, $S=\{s,o,p\}, \sum S=5$,
   $C=\min\{5,5\}=5$

   최소절단 $\min c(S,T)=5$

   $S=\{s,o,p,q,r\}$, $T=\{t\}$ or $S=\{s,o,p\}$, $T=\{q,r,t\}$

(b) G_3 그래프



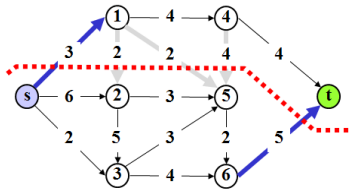
1. $S=\{s\}$, $T=\{t\}$
   $\sum S=5+5+5=15$, $\sum T=5+4+4=13$,
   $C=\min\{15,13\}=13$
2. $\max c(u,v)=5(S,1)$, $S=\{s,1\}, \sum S=3+2+5+5=15$
   $C=\min\{13,15\}=13$
3. $\max c(u,v)=5(S,2)$, $S=\{s,1,2\}$,
   $\sum S=3+7+2+7=19, C=\min\{13,19\}=13$
4. $\max c(u,v)=7(S,3)$, $S=\{s,1,2,3\}$,
   $\sum S=3+7+5=15, C=\min\{13,15\}=13$
5. $\max c(u,v)=7(S,5)$, $S=\{s,1,2,3,5\}$,
   $\sum S=4+2+3+2+5=16, C=\min\{13,16\}=13$
6. $\max c(u,v)=5(S,6)$, $S=\{s,1,2,3,5,6\}$,
   $\sum S=4+2+3+5=14, C=\min\{13,14\}=13$
7. $\max c(u,v)=5(S,9)$, $S=\{s,1,2,3,5,6,9\}$,
   $\sum S=4+2+3+4=13, C=\min\{13,13\}=13$
8. $\max c(u,v)=5(7,T)$, $T=\{t,7\}$,
   $\sum T=5+4+6=15, C=\min\{13,15\}=13$
9. $\max c(u,v)=5(4,T)$, $T=\{t,7,4\}$,
   $\sum T=4+10=14, C=\min\{13,14\}=13$
10. $\max c(u,v)=4(8,T)$, $T=\{t,7,4,8\}$,
    $\sum T=13, C=\min\{13,13\}=13$

    최소절단 $\min c(S,T)=13$,

    $S=\{s,1,2,3,5,6,9\}$, $T=\{4,7,8,t\}$ or

    $S=\{s,1,2,3,4,5,6,7,8,9\}$, $T=\{t\}$

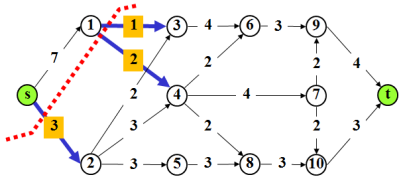
(c) G_4 그래프



1. $S=\{s\}$, $T=\{t\}$
   $\sum S=3+6+2=11, \sum T=4+5=9, C=\min\{11,9\}=9$
2. $\max c(u,v)=6(S,2)$, $S=\{s,2\}, \sum S=3+3+7=13$
   $C=\min\{9,13\}=9$
3. $\max c(u,v)=7(S,3)$, $S=\{s,2,3\}, \sum S=3+6+4=13$
   $C=\min\{9,13\}=9$
4. $\max c(u,v)=6(S,5)$, $S=\{s,2,3,5\}, \sum S=3+6=9$
   $C=\min\{9,9\}=9$
5. $\max c(u,v)=6(S,6)$, $S=\{s,2,3,5,6\}, \sum S=3+5=8$
   $C=\min\{9,8\}=8$
6. $\max c(u,v)=4(4,T)$, $T=\{t,4\}, \sum T=4+5=9$
   $C=\min\{8,9\}=8$
7. $\max c(u,v)=4(1,T)$, $T=\{t,4,1\}, \sum T=8$
   $C=\min\{8,8\}=8$

최소절단 $\min c(S,T)=8$, $S=\{s,2,3,5,6\}$, $T=\{1,4,t\}$

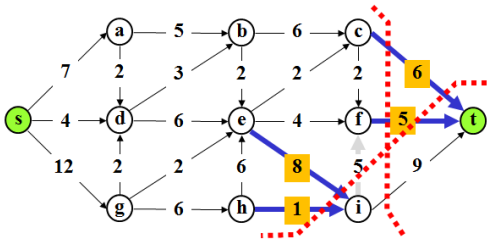
(d) G_5 그래프



1. $S=\{s\}$, $T=\{t\}$
   $\sum S=7+3=10, \sum T=4+3=7, C=\min\{10,7\}=7$
2. $\max c(u,v)=7(S,1)$, $S=\{s,1\}, \sum S=1+2+3=6$
   $C=\min\{7,6\}=6$
3. $\max c(u,v)=4(9,T)$, $T=\{t,9\}, \sum T=3+2+3=8$
   $C=\min\{6,8\}=6$

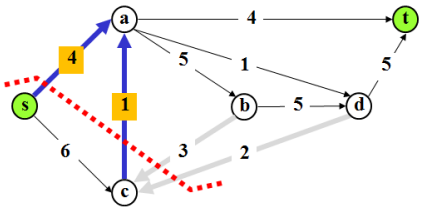
4. $\max c(u,v)=4(3,6),\ U_1=\{3,6\}$
5. $\max c(u,v)=4(4,7),\ U_2=\{4,7\}$
6. $\max c(u,v)=3(S,2),\ S=\{s,1,2\},\ \sum S=3+5+3=11$
   $C=\min\{6,11\}=6$
7. $\max c(u,v)=5(S,U_2),\ S=\{s,1,2,4,7\},$
   $\sum S=5+2+2+2+3=14,\ C=\min\{6,14\}=6$
8. $\max c(u,v)=5(S,U_1),\ S=\{s,1,2,4,7,3,6\},$
   $\sum S=5+2+2+3=12,\ C=\min\{6,12\}=6$
9. $\max c(u,v)=3(S,5),\ S=\{s,1,2,4,7,3,6,5\},$
   $\sum S=5+2+5=12,\ C=\min\{6,12\}=6$
10. $\max c(u,v)=5(S,8),\ S=\{s,1,2,4,7,3,6,5,8\},$
    $\sum S=5+5=10,\ C=\min\{6,10\}=6$
11. $\max c(u,v)=5(S,10),\ S=\{s,1,2,4,7,3,6,5,8,10\},$
    $\sum S=8,\ C=\min\{6,8\}=6$
    최소절단 $\min c(S,T)=6$
    $S=\{s,1\},\ T=\{2,3,4,5,6,7,8,9,10,t\}$

(e) G_6 그래프



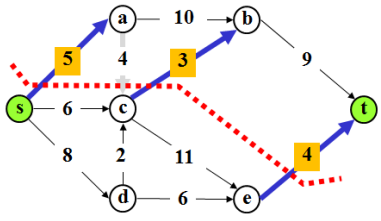
1. $S=\{s\},\ T=\{t\}$
   $\sum S=7+4+12=23,\ \sum T=6+5+9=20$
   $C=\min\{23,20\}=20$
2. $\max c(u,v)=12(S,g),$
   $S=\{s,g\},\ \sum S=7+6+2+6=21$
   $C=\min\{20,21\}=20$
3. $\max c(u,v)=9(i,T),\ T=\{t,i\}$
   $\sum T=6+5+8+1=20,\ C=\min\{20,20\}=20$
4. $\max c(u,v)=8(e,T),\ T=\{t,i,e\}$
   $\sum T=6+5+2+6+2+7=28,\ C=\min\{20,28\}=20$
5. $\max c(u,v)=7(S,a),\ S=\{s,g,a\},$
   $\sum S=5+8+2+6=21,\ C=\min\{20,21\}=20$
6. $\max c(u,v)=8(S,d),\ S=\{s,g,a,d\},$
   $\sum S=8+8+6=22,\ C=\min\{20,22\}=20$
7. $\max c(u,v)=8(S,b),\ S=\{s,g,a,d,b\},$
   $\sum S=6+10+6=22,\ C=\min\{20,22\}=20$
8. $\max c(u,v)=7(h,T),\ T=\{t,i,e,h\}$
   $\sum T=6+5+16=27,\ C=\min\{20,27\}=20$
9. $\max c(u,v)=6(S,c),\ S=\{s,g,a,d,b,c\},$
   $\sum S=2+22=24,\ C=\min\{20,24\}=20$
10. $\max c(u,v)=5(f,T),\ T=\{t,i,e,h,f\}$
    $\sum T=24,\ C=\min\{20,24\}=20$
    최소절단 $\min c(S,T)=20$
    $S=\{s,a,b,c,d,e,f,g,h,i\},\ T=\{t\}$ or
    $S=\{s,a,b,c,d,e,f,g,h\},\ T=\{i,t\}$

(f) G_7 그래프



1. $S=\{s\},\ T=\{t\}$
   $\sum S=4+6=10,\ \sum T=4+5=9,\ C=\min\{10,9\}=9$
2. $\max c(u,v)=6(S,c),\ S=\{s,c\},\ \sum S=5$
   $C=\min\{9,5\}=5$
3. $\max c(u,v)=5(S,a),\ S=\{s,c,a\},\ \sum S=4+5+1=10$
   $C=\min\{5,10\}=5$
4. $\max c(u,v)=5(S,b),\ S=\{s,c,a,b\},\ \sum S=4+6=10$
   $C=\min\{5,10\}=5$
5. $\max c(u,v)=6(S,d),\ S=\{s,c,a,b,d\},\ \sum S=9$
   $C=\min\{5,9\}=5$
   최소절단 $\min c(S,T)=5,\ S=\{s,c\},\ T=\{a,b,d,t\}$

(g) G_8 그래프



1. $S=\{s\},\ T=\{t\}$
   $\sum S=5+6+8=19,\ \sum T=9+4=13,$
   $C=\min\{19,13\}=13$
2. $\max c(u,v)=11(c,e),\ U_1=\{c,e\}$
3. $\max c(u,v)=10(a,b),\ U_2=\{a,b\}$

4. $_{\max}c(u,v)=9(U_2,T)$, $T=\{t,a,b\}$,
   $\sum T=5+7=12$, $C=_{\min}\{13,12\}=12$
5. $_{\max}c(u,v)=8(S,d)$, $S=\{s,d\}$,
   $\sum S=5+14=19$, $C=_{\min}\{12,19\}=12$
6. $_{\max}c(u,v)=14(S,U_1)$, $S=\{s,d,c,e\}$,
   $\sum S=12$, $C=_{\min}\{12,12\}=12$

   최소절단 $_{\min}c(S,T)=12$

   $S=\{s,c,d,e\}$, $T=\{a,b,t\}$

(h) G_9 그래프

그림 9. 실험 데이터의 최대인접병합 알고리즘

Fig 9. Maximum Adjacent Merging Algorithm for Experimental Data

Ford-Fulkerson알고리즘은 수행 복잡도가 $O(NA^2)$이며, 알고리즘 완료 이후 $c-f=0$인 병목지점들을 대상으로 S와 T를 분할하는 최소 절단을 찾기 위해서는 $T\rightarrow S$의 역방향 호를 무시하여야 시각적으로 찾을 수 있다. 반면에 제안된 알고리즘은 최소 절단을 N개의 정점에 대해 첫 번째로 s와 t에 대해 절단 값을 계산하고, 나머지 $N-2$개 정점을 결합시키면서 절단 값을 계산하기 때문에 알고리즘 수행 복잡도는 $O(N)$이며, 가장 큰 장점은 알고리즘 수행 과정에서 최소 절단 값과 최소 절단면을 찾을 수 있다는 점이다.

최대인접 병합 알고리즘은 항상 노드수 N에 대해 $N-1$회를 수행하여 수행 복잡도는 $O(N)$으로 Ford-Fulkerson이나 Edmonds-Karp 알고리즘의 수행 복잡도 $O(NA^2)$을 크게 향상시켰다.

V. 결론

본 논문에서는 Ford-Fulkerson 알고리즘의 s에서 t로의 증대경로를 탐색하는 방법과는 전혀 다른 방법을 적용하였다. 제안된 알고리즘은 망의 최대 수용량 호 $_{\max}c(u,v)$를 $s\in S$ 나 $t\in T$ 집합으로 단순히 병합하는 방법으로 최소절단을 구하였다. 제안 알고리즘은 최대 수용량이 인접한 노드를 병합하는 방법으로 최대인접병합 알고리즘이라 칭하였다.

최대인접병합 알고리즘은 망의 최대 수용량 호를 $n-1$회 찾아 해당 호의 두 노드를 병합하므로 수행 복잡도는 $O(N)$이다. 반면에, Ford-Fulkerson 알고리즘은 증대경로를 찾는데 많은 수행횟수가 요구되어 수행 복잡도는 $O(NA^2)$으로 알려져 있다. 또한, 역 방향 호를 포함한 증대경로도 고려해야 하는 어려움이 있다. 따라서 제안된 알고리즘은 증대경로를 찾는 어려움을 해결하였으며, 병목지점들의 조합으로 최소절단을 결정하는 과정도 수행하지 않고 단순히 망의 최대 수용량 호를 $n-1$개만 선택하면 되는 장점이 있다.

참 고 문 헌

[1] Wikipedia, "Max-flow Min-cut Theorem," http://en.wikipedia.org/wiki/Max_flow Min_cut Theorm, Wikimedia Foundation Inc., 2012.

[2] Wikipedia, "Ford-Fulkerson Algorithm," http://en.wikipedia.org/wiki/Ford_Fulkerson_Algorithm, Wikimedia Foundation Inc., 2012.

[3] L. R. Ford and D. R. Fulkerson, "Maximal Flow Through a Network," Canadian Journal of Mathematics, Vol. 8, pp. 399-404, 1956.

[4] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," Journal of the ACM, Vol. 19, Issue. 2, pp. 248-264, April 1972.

[5] Wikipedia, "Edmonds-Krap Algorithm," http://en.wikipedia.org/wiki/Edmonds_Krap_Algorithm, Wikimedia Foundation Inc., 2012.

[6] S. U. Lee, "Maximum Capacity-based Minimum Cut Algorithm," The Korea Society of Computer and Information, Vol. 16, No. 5, pp. 153-162, May 2011.

[7] M. Stoer and F. Wagner, "A Simple Min-Cut Algorithm," Journal of the ACM, Vol. 44, Issue. 4, pp. 585-591, 1997.

[8] Wikipedia, "Maximum Flow Problem" http://en.wikipedia.org/wiki/Maximum_flow_Problem, Wikimedia Foundation Inc., 2012.

[9] D. Wagner, "CS 170: Efficient Algorithms and Intractable Problems," UC Berkeley, 2003.

[10] K. Ikeda, "Mathematical Programming," Dept. Information Science and Intelligent Systems, The University of Tokushima, http://www-b2.is.tokushima-u.c.jp/ikeda/suuri/maxflow/MaxflowApp.shtml.

2005.

[11] WWL. Chen, "Discrete Mathematics," Department of Mathematics, Division of ICS, Macquarie University, Australia, http://www.maths.mq.edu.au/~wchen/lnd mfolder/lndm.html, 2003.

저자 소개

이 상 운

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007년 2월 : 국립 원주대학 여성교양과 조교수
- 2007년 3월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 부교수

<관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 소프트웨어 척도, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘>

- e-mail : sulee@gwnu.ac.kr