經營學硏究 第17卷 第2號(1988年 月)

最大흐름問題의 解法에 관한 硏究

—A Study on Algorithms for the Maximal Flow Problem—

金 基 錫*

≪目 次≫

Ⅰ. 序 言

最大흐름問題(maximal flow problem)는 $\{N, A, s, t, c\}$로 표현되는 네트워크에서 정의된다. 여기서 N은 마디集合, A는 가지集合, s는 源泉(source), t는 下溝(sink), 그리고 c는 각 가지의 흐름容量(flow capacity)을 나타낸다. 예를 들어 〈그림 1〉의 네트워크에서 $N=\{s, a, b, c, d, e, t\}$, $A=\{(s,a),(s,b),\cdots\cdots(e,t)\}$, $c(s,a)=5$, $c(s,b)=4,\cdots\cdots c(e,t)=8$이다.

주어진 네트워크에서 각 가지에 대한 "흐름(flow)" f는 다음 두 요건을 갖추어야 한다. 즉,

(i) $\sum_i f(i,j)-\sum_k f(j,k)=\begin{cases}-V, & j=s \text{ 일때}\\ V, & j=t \text{ 일때}\\ 0, & j\neq s,t \text{ 일때}\end{cases}$

(ii) $0\leq f(i,j)\leq c(i,j),\ \forall(i,j)\in A$

이때 V를 흐름값(flow value)이라 하며, 最大흐름問題는 바로 이 흐름값을 最大로 하는 흐름 f^*를 찾는 것이다.

한편 주어진 네트워크에서 "切斷(cut)"이란, 마디集合을 $s\in X$, $t\in X'$가 되도록 X와 X'로 兩分했을 때의 가지集合(X, X')을 가리킨다. 이때 $u(X,X')=\sum_{i=X}\sum_{j=X'}c(i,j)$를 그 切斷의 容量(capacity)이라 하고, 모든 切斷중에서 容量이 最小인 것을 最小切斷(minimal cut)이라 한다. 最大흐름값 V^*와 最小切斷容量 u^*는 항상 일치하며, 이를 "最大흐름 最小切斷의 定理(Max-Flow Min-Cut Theorem)"라 한다.

* 釜山大 경영학과
† 學會原稿 接受日 2月 3日

잘 알려진 바와 같이 最大흐름問題는 다음과 같은 線型計劃模型으로 定式化될 수 있다.

최 대 화 : V

제약조건 :

$$\sum_i f(i,s)-\sum_k f(s,k)+v=0$$

$$\sum_i f(i,t)-\sum_k f(t,k)-v=0$$

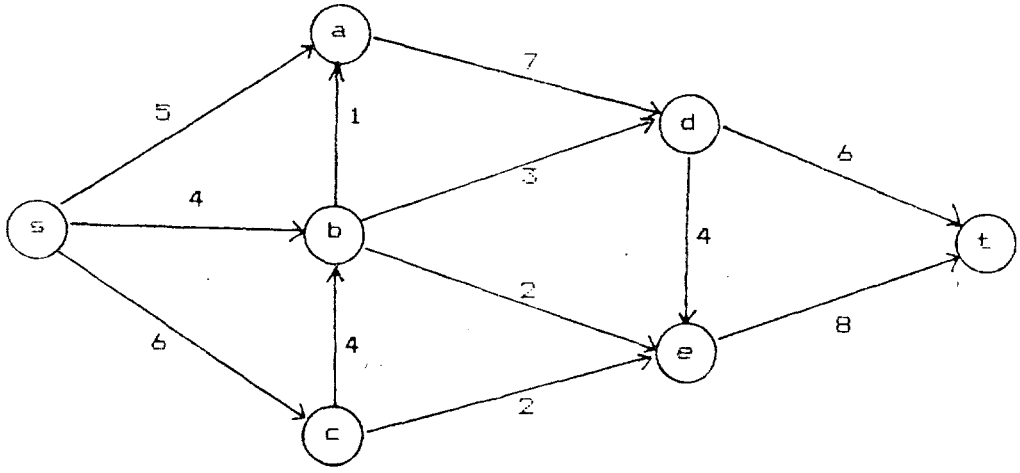
$$\sum_i f(i,j)-\sum_k f(j,k)=0,\ \forall_j\in N-\{s,t\}$$

$$f(i,j)\leq c(i,j),\ \forall(i,j)\in A$$

$$f(i,j)\geq 0,\ \forall(i,j)\in A$$

最大흐름問題가 처음으로 제기된 것은 1955년 Harris에 의해서였다. 즉 그는 Ford와 Fulkerson에게, 철도네트워크에 있어서 다수의 중계역을 거치는 두 도시간의 最大수송가능량과 그 經路를 알아내는 問題를 제기하였던 것이다[10]. 이와 유사한 最大흐름問題들은 실제로 여러가지 네트워크에서 발생한다. 가령 通信네트워크에서의 情報흐름, 航空네트워크에서의 旅客운송, 上水道네트워크에서의 給水 등의 最大化問題가 그 例이다. 그러나 最大흐름問題의 應用分野는 이와 같은 직관적인 네트워크흐름의 最大化에 그치지 않고, 일견 관련이 없어 보이는 問題들에 대해서 간접적인 해결방안을 제공하기도 한다[11, chapter Ⅱ].

이와같이 광범위한 應用分野를 가진 最大흐름問題의 우수한 解法을 창안 또는 발굴해 내는 것은 매우 중요하다고 할 수 있다. 本 硏究에서는, 지금까지 개발되어온 最大흐름問題의 解法들을 類型별로 구분하여 각 類型의 典型的인 解法절차를 例示하고, 主要 解法들의 理論的인 效率性과 實際的인 效率性을 비교·검토함으로써 效率的인 最大흐름問題解法에 관하여 종합적인 결론을 내리고자 한다.



〈그림 1〉 最大흐름問題 네트워크

Ⅱ. 最大흐름問題의 諸解法

Ⅱ.1 심플렉스解法

이미 序言에서 보인 바와 같이, 最大흐름問題는 線型計劃問題로 定式化할 수 있으므로, 모든 最大흐름問題는 線型計劃問題의 一般解法인 심플렉스解法으로 해결할 수 있다. 그러나 最大흐름問題는 線型計劃問題이면서도 특수한 구조를 갖고 있기 때문에, 標準심플렉스解法을 이 구조에 적합하도록 변형하면 最大흐름問題만을 위한 特殊심플렉스解法의 개발이 가능하게 된다. 이와같은 特殊심플렉스解法의 개발은 지금까지 몇차례 試圖되었으며, 그 최초의 것은 Fulkerson-Dantzig[12]의 "나무解法(tree method)"이었다.

최근에는 Glover-Klingman-Mote-Whitman[17](GKMW라 略함)들이 最大흐름問題를 위한 심플렉스解法의 새로운 變形(variant)을 개발하였으며, 또한 그 우수성을 경험적으로 입증하였다. GKMW의 심플렉스解法은, 原네트워크에 模造마디(dummy node) r을 도입하고 無限흐름容量을 가진 가지 (t, r)과 (r, s)를 추가함으로써 가능해진다. 이와같이 수정된 네트워크에서의 最大흐름問題를 定式化하면 다음과 같이 된다.

최 대 화 : V_1

제약조건 :

$$
\begin{aligned}
&-\sum_k f(s,k)+v_1=0\\
&\sum_i f(i,t)-v_2=0\\
&\sum_i f(i,j)-\sum_k f(j,k)=0,\quad \forall j\in N-\{s,t\}\\
&-v_1+v_2=0\\
&0\leq f(i,j)\leq c(i,j),\quad \forall (i,t)\in A\\
&v_1, v_2\geq 0
\end{aligned}
$$

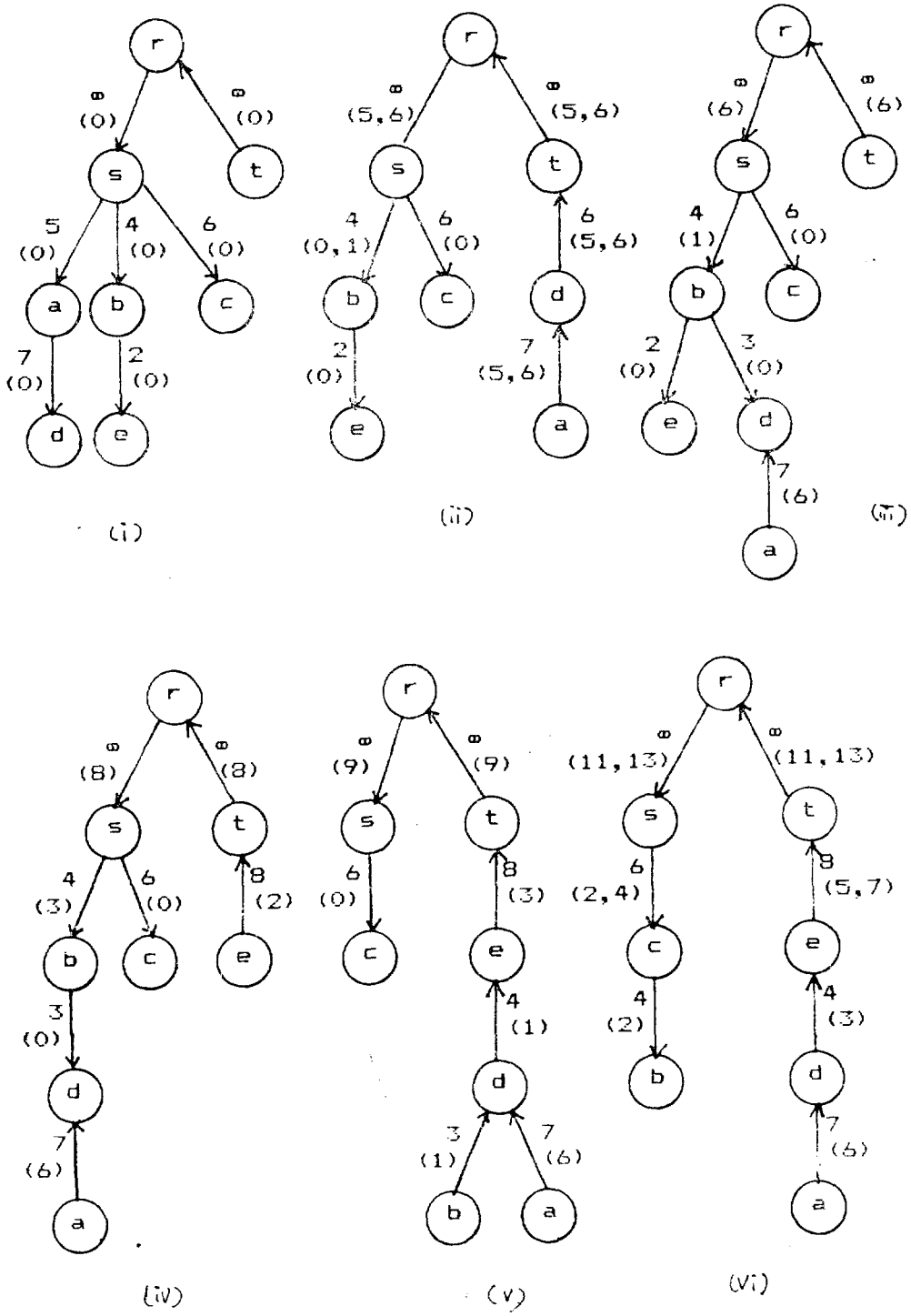
여기서 마디 j에 관한 雙大變數를 $\Pi(j)$라 두면, 相補餘裕性(complementary slackness)에 의해 다음과 같은 연립방정식 $[E]$를 얻는다.

$$
[E]\begin{cases}
-\Pi(r)+\Pi(s)=1\\
-\Pi(t)+\Pi(r)=0\\
-\Pi(i)+\Pi(j)=0,\quad \forall (i,j)\in A_T-\{(r,s),(t,r)\}
\end{cases}
$$

위 式에서 A_T는 基底나무(basis tree)에 속하는 가지들의 集合을 의미한다[4, chapter 17].

이제 GKMW의 特殊심플렉스解法을 단계별로 요약하면 다음과 같다.

단계 Ⅰ : 마디 r을 뿌리로 하고 마디 s와 t가 r에 직접 연결된 최초의 基底나무를 구성한다.


r
s
t
a
b
c
d
e
∞ (0)
∞ (0)
5 (0)
4 (0)
6 (0)
7 (0)
2 (0)
(i)
∞ (5,6)
∞ (5,6)
4 (0,1)
6 (0)
6 (5,6)
2 (0)
7 (5,6)
(ii)
∞ (6)
∞ (6)
4 (1)
6 (0)
2 (0)
3 (0)
7 (6)
(iii)
∞ (8)
∞ (8)
4 (3)
6 (0)
8 (2)
3 (0)
7 (6)
(iv)
∞ (9)
∞ (9)
6 (0)
8 (3)
4 (1)
3 (1)
7 (6)
(v)
∞ (11,13)
∞ (11,13)
6 (2,4)
8 (5,7)
4 (2)
4 (3)
7 (6)
(vi)

〈그림 2〉 심플렉스解法

단계 Ⅱ : 현재의 基底나무를 이용하여 연립방정식 $[E]$를 푼다. 이때 $\Pi(r)=0$이라 둔다.

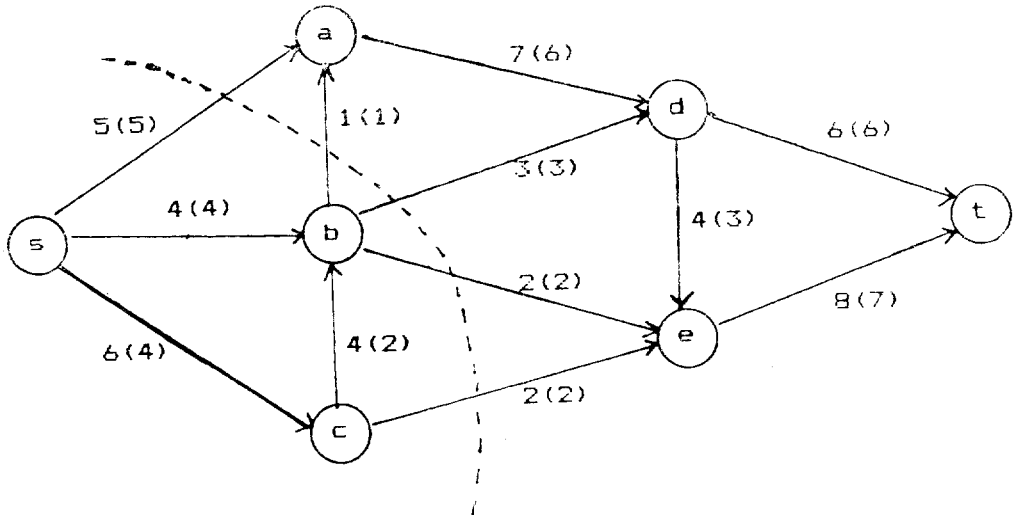
단계 Ⅲ : (가) $\Pi(i)=1$, $\Pi(j)=0$, $f(i,j)=0$ 또는 $\Pi(i)=0$, $\Pi(j)=1$, $f(i,j)=c(i,j)$를 만족시키는 非基低가지 (i,j)를 찾는다. 만약 찾을 수 없으면 현재解가 最適解이다.

단계 Ⅳ : 단계 Ⅲ에서 찾은 가지 (i,j)를 현재 基底나무에 추가함으로써 증가시킬 수 있는 最大의 흐름量을 △라 하자. 만약 $\triangle=c(i,j)$이면 단계 V-2로 가고, 아니면 단계 V-1로 간다.

단계 V-1 : 가지 (i,j)를 추가하여 생긴 순환經路上의 가지흐름을 △만큼 증가시킨다. 이 經路上의 가지들 중에서 △의 크기를 제한했던 가지를 현재의 基底나무에서 제거하고, 대신 (i,j)를 도입함으로써 새 基底나무를 구성하고 단계 Ⅱ로 간다.

단계 V-2 : 가지 (i,j)를 추가하여 생긴 순환經路上의 가지흐름을 △만큼 증가시킨다. 基底나무의 변화없이 단계 Ⅲ으로 간다.

〔例題〕 GKMW의 심플렉스解法을 〈그림 1〉의 네트워크로 정의된 最大흐름問題에 적용하면 〈그림 2〉의 (i)~(vi)과 같은 基底나무의 변화를 거쳐 最適解에 도달하게 된다. 여기서 각 가지에 附記한 숫자들은 흐름容量과 현재흐름(괄호안)을 나타내며, 괄호안의 두 숫자는 단계 V-2가 적용되어 基底나무의 변화없이 흐름값만 증가했을 경우이다. 이렇게 구한 最大흐름을 原네트워크上에 표시하면 〈그림 3〉의 괄호안 숫자들과 같으며, 이때의 最大흐름값은 13이다. 이 그림에서 점선은, 최종基底나무의 구조에 따라 마디들을 分割한 것이며 最小切斷에 속하는 가지들을 가로지르고 있다.



〈그림 3〉 最大흐름問題의 最適解

Ⅱ.2 레이블解法

最大흐름問題의 固有解法으로서 처음 개발된 것은 Ford-Fulkerson[10](FF라 略함)의 레이블解法이었다. FF의 解法은 레이블(label)을 이용하여 "흐름증가經路(flow augmenting path)"를 반복적으로 발견함으로써 最大흐름을 구하는 방법이다. FF의 레이블解法을 略記하면 다음과 같다.

단계Ⅰ : 마디 s에는 레이블 $[-,\infty]$을 부여하고 나머지 마디들은 레이블을 부여하지 않는다.

단계Ⅱ : 有레이블 마디 중에서 點檢(scan) 않은 마디를 선택한다. 만약 그런 마디가 없으면 현재의 흐름이 最大흐름이다.

단계Ⅲ : 단계Ⅱ에서 선택한 마디(i라 하자)를 點檢(scan)하여 (가) $f(i,j)<c(i,j)$ 또는 (나) $f(j,i)>0$을 만족시키는 無레이블 마디 j를 모두 찾아, (가)의 경우에는, 레이블 $[i^+,e(j)]$를 부여하고 (나)의 경우에는 $[i-,e(j)]$를 부여한다. 여기서 $e(j)$는 (가)의 경우 $\min\{e(i),\ c(i,j)-f(i,j)\}$로, (나)의 경우 $\min\{e(i),\ f(j,i)\}$로 정의된다. 마디 t가 레이블을 부여받았으면 단계Ⅳ로 가고, 그렇지 않으면 단계Ⅱ로 돌아간다.

단계Ⅳ : 레이블의 前半部를 따라서, 마디 t로부터 마디 s에 이르는 흐름증가經路를 발견하고 그 經路上의 모든 가지흐름을 $e(t)$만큼 변경시킨다.

단계Ⅴ : 마디 s의 레이블만 남기고 나머지 레이블을 모두 제거한 뒤 단계Ⅱ로 간다.

FF의 레이블解法은 흐름容量이 모두 整數이면 문제가 없으나, 無理數를 허용하면 最大흐름을 발견하지 못하는 경우가 발생한다[11, pp.21~22]. 이 문제점은 그후 Edmonds-Karp[6]에 의해 간단히 해결되었다. 즉, 그들은 FF解法의 단계Ⅱ에서 點檢할 마디의 선택기준으로 "레이블이 부여된 順(즉, breadth-first-search)"을 채택하기만 하면 항상 最大흐름에 도달할 수 있음을 증명하였다. 또한 Tucker[31]는 다른 선택기준으로 "레이블이 부여된 逆順(즉, depth-first-search)"을 채택해도 반드시 最大흐름에 도달할 수 있음을 증명하였다(두 선택기준을 채택한 FF解法의 變形을 각각 FFB와 FFD로 略함).

〔例題〕 FFB와 FFD를 이용하여 〈그림 1〉의 最大흐름問題를 다시 풀어보면, 흐름증가總路별 레이블 부여결과는 〈표 1〉의 (가), (나)와 같다. 이 표에서 (가)의 FFB에 의한 첫번째 흐름증가經路는 s-a-d-t이고, 이 經路를 통한 흐름증가는 5임을 알 수 있다. 한편 (나)의 FFD에 의한 첫번째 흐름증가經路는 s-a-d-e-t이고 이때의 흐름증가는 4이다. FFB와 FFD 모두 7回의 흐름증가로 最大흐름에 도달하게 되었으며, 그 결과는 심플렉스解法의 경우와 같이 〈그림 3〉으로 나타난다.

FF의 레이블解法을 개선한 것으로서 또하나 특기할만한 變形은 Fong-Rao[9]의 "레이블存續解法"이다. 이 解法의 특징은 FF解法의 단계Ⅴ에서 레이블을 제거할 때, 다음 흐름증가經路의

〈표 1〉 레 이 블 解 法

(가) FFB

흐름증가경로	s	a	b	c	d	e	t
1	$(-,\infty)$	$(s+,5)$	$(s+,4)$	$(s+,6)$	$(a+,5)$	$(b+,2)$	$(d+,5)$
2	$(-,\infty)$	$(b+,1)$	$(s+,4)$	$(s+,6)$	$(b+,3)$	$(b+,2)$	$(d+,1)$
3	$(-,\infty)$	$(b+,1)$	$(s+,3)$	$(s+,6)$	$(b+,2)$	$(b+,2)$	$(e+,2)$
4	$(-,\infty)$	$(b+,1)$	$(s+,1)$	$(s+,6)$	$(b+,2)$	$(c+,2)$	$(e+,2)$
5	$(-,\infty)$	$(b+,1)$	$(s+,1)$	$(s+,4)$	$(b+,1)$	$(d+,1)$	$(e+,1)$
6	$(-,\infty)$	$(b+,1)$	$(c+,4)$	$(s+,4)$	$(b+,1)$	$(d+,1)$	$(e+,1)$
7	$(-,\infty)$	$(b+,1)$	$(c+,3)$	$(s+,3)$	$(a+,1)$	$(d+,1)$	$(e+,1)$
8	$(-,\infty)$		$(c+,2)$	$(s+,2)$			

(나) FFD

흐름증가경로	s	a	b	c	d	e	t
1	$(-,\infty)$	$(s+,5)$			$(a+,5)$	$(d+,4)$	$(e+,4)$
2	$(-,\infty)$	$(s+,1)$			$(a+,1)$		$(d+,1)$
3	$(-,\infty)$	$(b+,1)$	$(s+,4)$		$(a+,1)$		$(d+,1)$
4	$(-,\infty)$		$(s+,3)$		$(b+,3)$		$(d+,3)$
5	$(-,\infty)$	$(d-,2)$	$(c+,4)$	$(s+,6)$	$(e-,2)$	$(b+,2)$	$(d+,1)$
6	$(-,\infty)$	$(d-,1)$	$(c+,3)$	$(s+,5)$	$(e-,1)$	$(b+,1)$	$(e+,1)$
7	$(-,\infty)$	$(d-,2)$	$(c+,2)$	$(s+,4)$	$(e-,2)$	$(c+,2)$	$(e+,2)$
8	$(-,\infty)$		$(c+,2)$	$(s+,2)$			

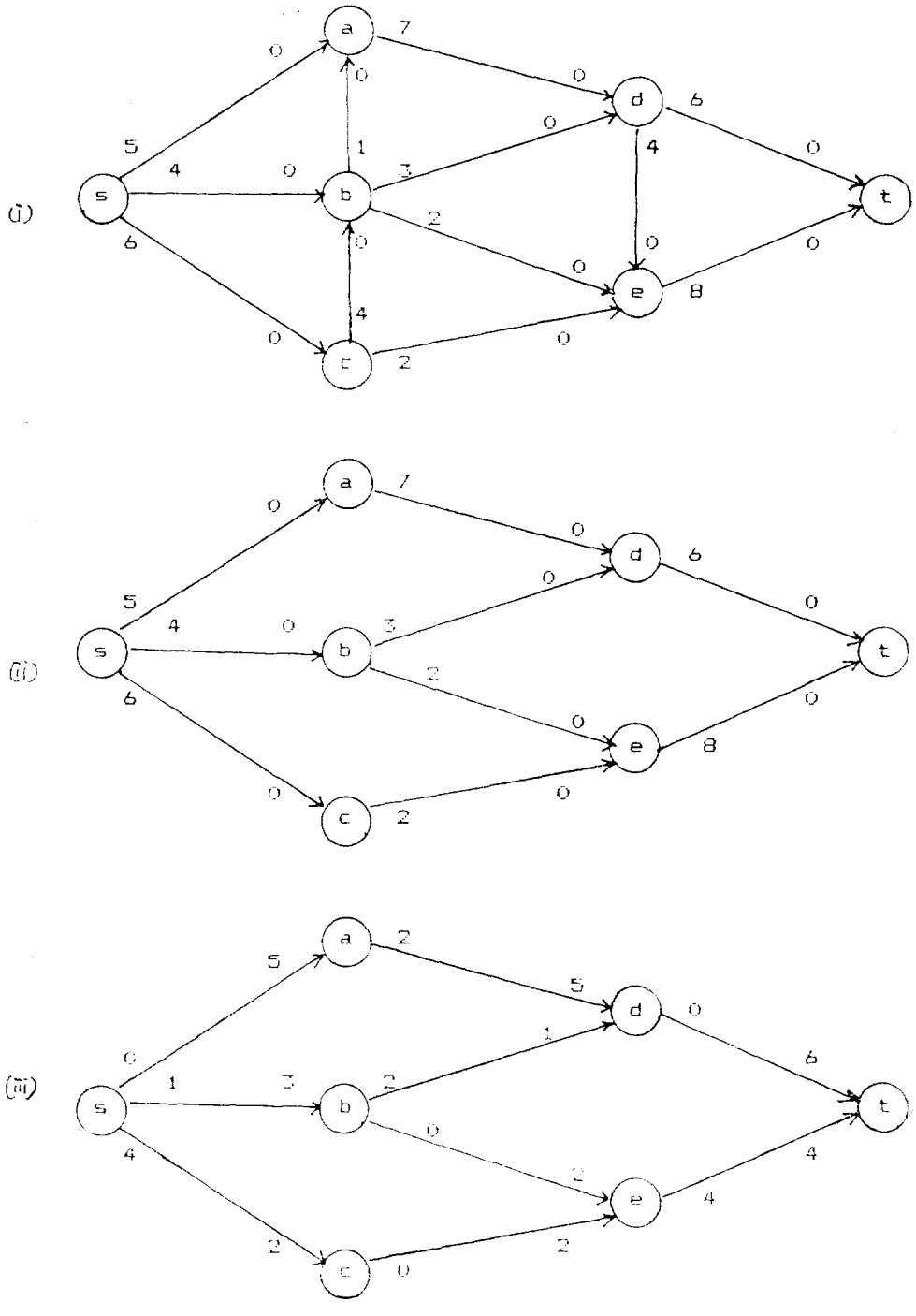
발견에 다시 사용할 수 있는 레이블들을 存續(retain) 시키는 점이다. 따라서 중복되는 레이블 부여의 노력을 절약함으로써 FF 解法보다 높은 效率性을 성취한다는 것이다(이 방법은 FFB와 FFD에 모두 적용가능하며, 이때 두 變形을 각각 FFBR과 FFDR이라 略함). 예를들어 앞의 例題에서 FFB대신 FFBR을 적용한다면, (가)의 첫번째 흐름증가經路를 통해 흐름을 증가시킨 후 마디 s의 레이블은 물론이고 마디 b,c,e들의 레이블도 存續시켜 두번째 흐름증가經路의 발견에 다시 사용하게 된다.

Ⅱ.3 封鎖흐름解法

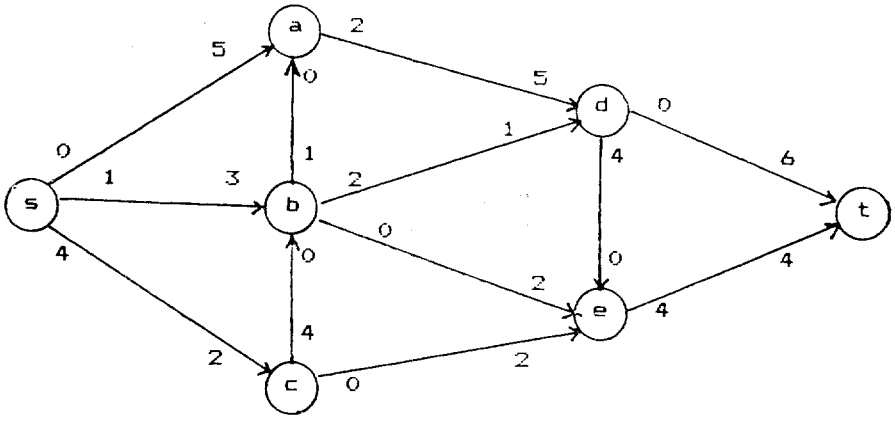
最大흐름問題解法의 세번째 類型은 "封鎖흐름(blocking flow)"의 반복적인 발견으로 最大흐름에 도달하는 방법이다. 封鎖흐름解法의 논의를 위해서는 중간과정의 네트워크를 $\{N, A, s, t, \delta\}$로 표현하는 것이 편리하다. 여기서 δ는 "純흐름容量(net flow capacity)"이라 불리며, 각 가지(i,j)에 대해 한 雙이 정의된다 : $\delta(i,j)=c(i,j)-f(i,j)$, $\delta(j,i)=f(i,j)$.

封鎖흐름이란, 다음 두 요건을 갖춘 "層化네트워크(layered network)" $W^L=\{N^L, A^L, s, t, \delta\}$

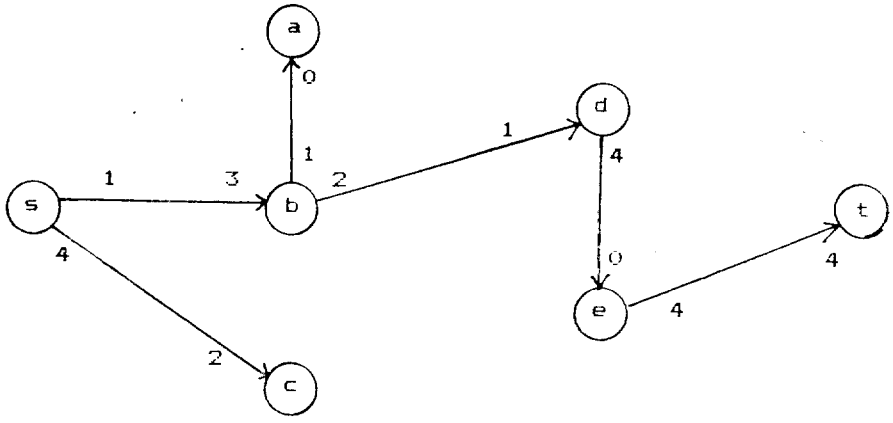
〈그림 4〉 封鎖흐름解法


(i)
a
b
c
d
e
s
t
7
5
4
0
6
3
2
8
(ii)
a
b
c
d
e
s
t
7
5
4
0
6
3
2
8
(iii)
a
b
c
d
e
s
t
2
5
0
1
3
4
2
6

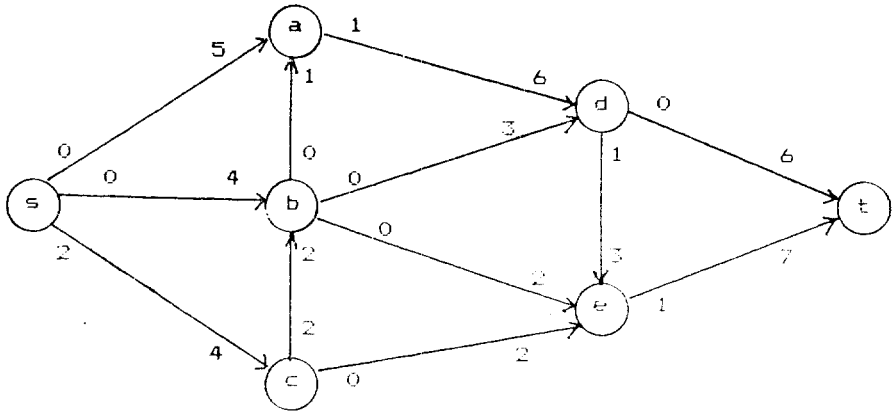
(iv)


a
d
s
b
t
e
c
5
2
5
0
0
1
1
4
6
1
3
2
0
0
2
4
4
4
2
2
0

(v)


a
d
s
b
t
e
c
0
1
1
4
1
3
2
0
4
4
2

(vi)


a
d
s
b
t
e
c
5
1
1
6
0
0
0
3
4
0
1
6
0
2
2
2
3
7
4
2
1
2
0

에서 더이상의 흐름이 封鎖된 상태의 흐름을 가리킨다.

(i) $N^L \subseteq N$, $N^L = L_0 U\ L_1 U \cdots U L_d$ 이며, L_i 들은 "마디層(layer)"이라 불리우는 N^L 의 分割(partition)로서 특히 $L_0 = \{s\}$, $L_d = \{t\}$ 이다.

(ii) $A^L \subseteq A$ 이며, 만약 $(u, v) \in A^L$ 라면 (가) $u \in L_j$, $v \in L_{j+1}$, $\delta(u, v) > 0$ 또는 (나) $u \in L_{j+1}$, $v \in L_j$, $\delta(v, u) > 0$ 이다.

封鎖흐름의 개념을 활용한 最大흐름問題의 解法에는 여러가지 變形들이 있으나, 모두 다음과 같은 공통된 절차를 따르고 단지 단Ⅲ계에서 封鎖흐름을 발견하는 방법만 相異하다.

단계Ⅰ : 최초의 네트워크 $\{N, A, s, t, \delta\}$를 작성한다.

단계Ⅱ : 현재 純흐름容量 δ에 따라 새로운 層化네트워크 W^L을 구성한다. 이때 마디 t가 W^L에서 연결되어있지 않으면 현재의 흐름이 最大흐름이다.

단계Ⅲ : 새로 구성된 層化네트워크에서의 封鎖흐름 g^*를 구한다.

단계Ⅳ : 原네트워크의 純흐름容量을 g^*만큼 수정하고 단계Ⅱ로 간다.

封鎖흐름解法의 창시자인 Dinic[5]은 단계Ⅲ에서 封鎖흐름을 구하기 위하여 FFD를 적용하였다.

〔例題〕 Dinic의 解法을 例示하기 위하여 〈그림 1〉의 最大흐름問題를 다시 도입해 보자. 먼저 단계Ⅰ에 의해 〈그림 4〉의 (i)과 같은 수정네트워크를 얻게된다. 이에 따라 단계Ⅱ에서 層化네트워크를 구성하면 (ii)와 같이 되며, 이때 마디層들은 $L_0 = \{s\}$, $L_1 = \{a, b, c\}$, $L_2 = \{d, e\}$, $L_3 = \{t\} = L_d$ 이다. 단계Ⅲ에서 FFD를 (ii)에 적용하면 4回의 흐름증가經路를 발견하여 (iii)으로 된다. 여기서 封鎖흐름은 4回의 흐름증가를 더한 것이며 $g^*(s, a) = 5$, $g^*(s, b) = 3$, $\therefore g^*(e, t) = 4$ 이다. 이제 단계Ⅳ에 의하여 原네트워크 (i)을 g^*만큼 수정하면 (iv)를 얻게 된다. 이렇게 한 차례의 시행이 끝나면 다시 단계Ⅱ로 돌아가서 두번째 層化네트워크인 (v)를 구성할 수 있다. (v)의 마디層들은 모두 다섯으로서 $L_0 = \{s\}$, $L_1 = \{b, c\}$, $L_2 = \{a, d\}$, $L_3 = \{e\}$, $L_4 = \{t\} = L_d$ 와 같다. 이와같은 과정을 계속하면 모두 4回의 封鎖흐름을 발견한 뒤 (vi)과 같은 네트워크에 도달한다. 이때 層化네트워크를 구성해보면 $L_0 = \{s\}$, $L_1 = \{c\}$, $L_2 = \{b\}$로 그쳐, 마디 t에 연결될 수 없으므로 이미 最大흐름에 도달했음을 알 수 있다. 最大흐름은 (i)과 (vi)을 비교하여 그 차이로서 알 수 있고, 그 결과는 역시 〈그림 3〉과 같다.

위의 단계Ⅲ에서 封鎖흐름을 발견하는 방법으로는 Dinic의 FFD 이외에도 여러가지가 있다. 예를들면 Karzanov[18]는 가능한 모든 "事前흐름(preflow)"을 증가시킨 뒤(push), 흐름의 요건을 갖추는 일(balance)을 반복함으로써 封鎖흐름을 구하였다. 또한 Malhotra-Kumar-Maheshwari[22]는 각 마디를 통과할 수 있는 흐름可能量(flow potential)을 비교하여 그것이 最小인 마디(reference node)를 중심으로 흐름을 증가시키는 封鎖흐름解法을 고안하였다.

Ⅲ. 最大흐름問題法의 效率性

Ⅲ.1 解法의 效率性 評價

일반적으로 어떤 解法의 效率性을 評價하기 위해서는

(i) 解의 正確度(accuracy)

(ii) 解의 導出에 소요되는 컴퓨터 計算時間

(iii) 解의 導出에 소요되는 컴퓨터 記憶容量

등의 기준을 고려해야 한다. 여기서 (i)解의 正確度란, 특정解法으로 구한 解가 실제 最適解와 近接한 정도를 의미하며 近似解法(approximation algorithm)의 評價에 중요한 기준이 된다. 한편, 最適解法(optimization algorithm)의 評價에 중요시되는 기준은 (ii)와 (iii)이며 특히 (ii) 소요計算時間이다. 지금까지 개발된 最大흐름問題의 解法들은 모두 最適解法이므로, 本稿에서 解法의 效率性이라고 하면 바로 소요計算時間을 그 尺度로 한다.

解法의 效率性은 다시 理論的 效率性(theoretical efficiency)과 實際的 效率性(practical efficiency)으로 구분할 수 있다. 이때 어떤 解法의 實際的인 效率性을 評價하기 위해서는, 그 解法의 컴퓨터 프로그램을 작성한 뒤 특정 컴퓨터를 이용하여 다수의 實驗問題들을 해결하는 데 소요된 cpu 時間을 사용하는 것이 보통이다. 이와같은 實際的 效率性의 短點은, cpu 소요時間이 解法자체의 效率性외 다른 要因들에 의해서도 상당한 영향을 받을 수 있다는 점이다. 예를들면 프로그래밍技法, 컴퓨터 運營體制(operating system), 實驗問題 등의 要因을 들 수 있다.

한편 어떤 解法의 理論的인 效率性은 入力資料의 크기가 N인 問題를 푸는 데 소요되는 基本演算(加, 減, 乘, 除, 比較 등)의 數 $T(N)$으로 評價하며, $T(N)=O(f(N))$과 같이 표시한 것을 그 解法의 "計算裝雜性(computational complexity)"이라 부른다[1 chapter 1]. 이때 N은 충분히 큰 수로 가정하며 問題의 형태를 最惡의 경우로 가정하는 데, 여기에 바로 理論的 效率性의 限界가 있다.

이와같이 두 效率性은 각각 短點을 갖고 있으며, 서로의 短點을 해결함으로써 長短으로 삼고 있다. 또한 두 解法을 비교할 경우, 理論的인 效率性이 우수한 解法이 반드시 實際的인 效率性도 우수한 것으로 나타나지 않을 수 있다.

本稿에서는, 여러가지 最大흐름問題解法들의 理論的 效率性을 먼저 검토하고(Ⅲ.2), 몇 가지 주요解法들에 대해서는 實際的 效率性의 비교를 試圖함으로써(Ⅲ.3), 效率的인 最大흐름問題解法에 관하여 종합적인 결론을 내리고자 한다.

Ⅲ.2 最大흐름問題解法의 理論的 效率性

最大흐름問題의 入力資料는, 주어진 네트워크의 마디數 $|N|$과 가지數 $|A|$에 의하여 그 크기가 결정된다. 한편 가지數 $|A|$는, 네트워크의 마디들이 서로 연결된 정도에 따라 최소 $|N|-1$부터 최대 $|N|\cdot(|N|-1)$ 사이의 값을 가진다.

일반적인 線型計劃問題를 위한 심플렉스解法의 理論的인 效率性은 變數의 數에 관하여 指數函數인 것으로 알려져 있다. 그러나 最大흐름問題를 위하여 수정된 特殊심플렉스解法의 理論的 效率性 역시 指數函數인지는 아직 명확히 알려지지 않았다(筆者는 多項函數일 것으로 추측함).

Ford-Fulkerson의 레이블解法은 이미 지적한 대로 最大흐름을 구하지 못한 채 무한히 반복할 경우가 있다. 그러나 Edmonds-Karp의 FFB解法은 理論的 效率性이 $O(|N|\cdot|A|^2)$인 것으로 증명되었다. 즉, 레이블이 부여된 順에 의해 새로운 흐름증가經路를 발견하는 데는 $O(|A|)$의 시간이 소요되고, 最大흐름에 도달하기 까지 필요한 흐름증가經路의 수는 $O(|N|\cdot|A|)$이므로, 총 計算時間은 $O(|N|\cdot|A|^2)$이 된다.

封鎖흐름解法의 類型에 속하는 最大흐름解法들은 모두 동일한 방법, 즉 레이블 부여順을 이용하여 하나의 層化네트워크를 구성하게 되는데, 그 소요計算時間은 $O(|A|)$이다. 이와같은 層化네트워크는 最惡의 경우 $|N|-1$回 작성하게 되므로 $O(|N|)$으로 표시된다. 주어진 層化네트워크에서 封鎖흐름을 발견하기 위하여, Dinic은 FFD를 사용함으로써 $O(|N|\cdot|A|)$의 時間을 소모하였으므로 最大흐름을 발견하기 위한 총 計算時間은 $O(|N|)\cdot[O(|A|)+O(|N|\cdot|A|)]=$

〈표 2〉 最大흐름問題解法들의 理論的 效率性

Fulkerson-Dantzig[1955]	—
Glover-Klingman-Mote-Whitman1[984]	—
Ford-Fulkerson[1956]	∞
Edmonds-Karp[1972]	$O(\|N\|\|A\|^2)$
Dinic[1970]	$O(\|N\|^2\|A\|)$
Karzanov[1974]	$O(\|N\|^3)$
Cherkasky[1977]	$O(\|N\|^2\|A\|^{1/2})$
Malhotra-Kumar-Maheshwar[1978]	$O(\|N\|^3)$
Shiloach[1978]	$O(\|N\|\|A\|\log^2\|N\|)$
Galil[1980]	$O(\|N\|^{5/3}\|A\|^{2/3})$
Galil-Naamad[1980]	$O(\|N\|\|A\|\log^2\|N\|)$
Sleator[1980]	$O(\|N\|\|A\|\log\|N\|)$
Gabow[1983]	$O(\|N\|\|A\|\log\|N\|)$
Tarjan[1984]	$O(\|N\|^3)$

$O(|N|^2 \cdot |A|)$이 된다. Dinic 이후 지금까지 많은 학자들이 더 우수한 理論的 效率性을 갖는 封鎖흐름解法을 개발하기 위하여 경쟁적으로 노력해 오고 있다.

〈표 2〉는 대표적인 最大흐름問題解法들의 理論的 效率性을 類型別, 年度別로 정리한 것이다. 만약 주어진 네트워크의 가지들이 稀少(sparse)하다면, $|A|=O(|N|)$이 되고 가장 우수한 理論的 效率性은 Sleator[28]와 Gabow[13]에 의한 $O(|N|^2 \log |N|)$이다. 한편 가지들이 稠密(dense)하다면, $|A|=O(|N|^2)$ 이므로 최선의 計算時間은 $O(|N|^3)$가 되고 Karzanov[18], Malhotra-Kumar-Maheshwari[22], 또는 Tarjan[29]에 의해 성취될 수 있다.

Ⅲ.3 最大흐름問題解法의 實際的 效率性

最大흐름問題解法의 理論的 效率性에 관한 연구는 지금까지 매우 활발했던 반면에, 實際的인 效率性에 관한 연구는 드물어서 Cheung[3]과 Glover 등[16]의 연구 외에는 찾아보기 어렵다. Cheung은 레이블解法 3種과 封鎖흐름解法 5種의 實際的 效率性을 비교한 결과, Dinic의 封鎖흐름解法이 가장 우수하다는 결론을 내렸다. 그러나 Glover 등은 70여 가지의 심플렉스解法, 레이블解法, 封鎖흐름解法들의 效率性을 비교한 결과 그들이 개발한 GKMW 심플렉스解法(Ⅱ.1 참조), 이 實際的인 면에서 가장 우수하다는 결론을 얻었다. 그러나 解法의 實際的 效率性은 외부要因들에 의해 歪曲되기 쉬우므로, 최종 결론을 내리기 위해서는 신중을 기할 필요가 있다.

本節에서는 GKMW 심플렉스解法의 實際的 效率性을 다른 解法들의 效率性과 새로운 각도에서 다시 한번 비교하기 위하여, GKMW 解法의 FORTRAN 프로그램을 原著者들로 부터 入手하였다. 한편 레이블解法에 속하는 FFB와 FFBR, 그리고 Malhotra-Kumar-Maheshwari(MKM이라 略함)의 封鎖흐름解法은 筆者가 FORTRAN으로 프로그램을 작성하였다. 또한 Karzanov(KAR라 略함)의 封鎖흐름解法은 Nijenhuis-Wilf[24]의 FORTRAN 프로그램을 사용하였다.

實驗을 위한 最大흐름問題들로는 Faaland-Schmitt[7]에 나타난 日程計劃問題(SCH라 略함), Lerchus-Grossmann[20]에 나타난 露天採鑛問題(OPM이라 略함), 그리고 Mamer-Smith[23]에 나타난 修繕道具問題(FRK라 略함)와 같이 實際應用問題들을 채택하였다. 각 應用問題에 대해서는 〈표 3〉과 같이 크기와 구조를 달리하는 12 問題를 구성하고, 이들 36問題(3×12) 각각에 대해서 다시 擬似亂數(pseudorandom number)를 사용하여 最大흐름問題를 4回 生成시켰다.

〈표 3〉에서 解法란에 기재된 숫자들은, 中型컴퓨터인 VAX 11/780을 이용하여 36實驗問題를 각기 4回 푼 결과의 平均 cpu 時間(秒)이다. 實驗결과를 분석해보면 다섯 最大흐름問題解法의 實際的 效率性에 관하여 다음과 같은 결론을 얻을 수 있다.

(i) GKMW 심플렉스解法의 效率性은 本 실험에서도 매우 우수한 것으로 나타났다. 다만 日程計劃問題(SCH)에서는 FFBR 解法 다음으로 우수했다.

(ii) FFB 레이블解法은 가장 效率性이 낮은 것으로 나타났다. 특히 問題의 크기 즉 $|N|$과

〈표 3〉 最大흐름問題解法들의 實際的 效率性(VAX 11/780 컴퓨터의 cpu 秒)

實驗問題			解法				
問題	$\|N\|$	$\|A\|$	GKMW	FFB	FFBR	MKM	KAR
SCH1	112	215	0.07	0.29	0.04	0.67	1.44
SCH2	304	602	0.37	3.19	0.18	5.19	5.62
SCH3	508	1,009	0.55	6.60	0.28	9.94	10.64
SCH4	757	1,509	1.41	17.69	0.79	26.21	19.28
SCH5	1,005	2,005	2.02	28.87	1.05	25.76	27.48
SCH6	3,005	6,006	21.23	323.22	10.50	417.50	164.87
SCH7	5,001	10,013	51.56	&	23.93	&	426.64
SCH8	7,508	15,011	95.44	&	41.66	&	&
SCH9	10,008	20,012	141.90	&	82.74	&	&
SCH10	12,512	25,021	282.96	&	137.98	&	&
SCH11	15,026	30,045	410.33	&	235.71	&	&
SCH12	20,026	40,047	696.08	&	386.24	&	&
OPM1	232	770	0.13	0.83	0.09	0.66	5.94
OPM2	232	770	0.16	1.24	0.20	0.90	6.05
OPM3	412	1,490	0.44	4.92	0.54	2.72	13.70
OPM4	412	1,490	0.45	5.92	1.07	3.07	14.22
OPM5	482	1,620	0.32	3.20	0.30	1.68	14.17
OPM6	482	1,620	0.40	5.41	0.69	2.10	14.87
OPM7	912	3,340	1.10	19.68	1.84	6.79	35.68
OPM8	912	3,340	1.27	28.67	4.99	7.23	37.63
OPM9	982	3,320	0.86	12.39	1.07	3.70	32.94
OPM10	982	3,320	1.06	20.52	3.15	4.99	34.53
OPM11	1,912	7,040	3.56	98.73	9.01	20.82	97.71
OPM12	1,912	7,040	4.04	137.58	23.43	28.44	101.86
FRK1	202	422	0.11	1.14	0.30	0.83	3.33
FRK2	202	422	0.11	1.07	0.32	0.70	3.30
FRK3	402	851	0.28	5.53	1.34	2.04	7.97
FRK4	402	851	0.25	4.99	1.29	1.78	7.74
FRK5	802	1,716	0.74	22.90	5.88	6.01	19.52
FRK6	802	1,716	0.72	22.07	5.72	6.03	19.51
FRK7	1,602	3,435	2.22	97.50	24.50	18.30	50.58
FRK8	1,602	3,455	2.15	90.13	24.86	16.49	49.05
FRK9	2,402	5,163	4.34	222.69	56.59	36.34	89.97
FRK10	2,402	5,163	3.91	208.84	55.58	32.16	88.67
FRK11	3,202	6,896	7.37	409.56	100.45	54.96	137.97
FRK12	3,202	6,896	6.78	377.82	100.69	50.87	137.59

註 : & 는 600 秒 이상

$|A|$가 증가할수록 效率性은 더욱 악화되었다.

(iii) FFBR 레이블解法은 FFB 를 부분적으로 변형함으로써 效率性이 현저하게 개선되었다. 특

히 日程計劃問題에 있어서는 가장 우수한 결과를 보여주었고, 나머지 問題들에서도 封鎖흐름解法에 필적하는 效率性을 보였다.

(iv) 封鎖흐름解法들인 MKM과 KAR은 理論的 效率性의 우수성에 비하여 그다지 만족스러운 결과를 보여주지 못했다. 또한 두 解法의 效率性 間에도 상당한 차이가 나타났다.

Ⅳ. 結 言

本稿 Ⅱ장에서는 最大흐름問題의 諸解法을 세 類型으로 구분하여, 類型別로 典型的인 解法절차를 例示하고 여러가지 變形들을 소개하였다. 그러나 Kim[19]은 이들 類型間에도 共通點 내지는 類似한 點이 있음을 보여주었다. 특히 레이블解法을 적절히 變形시키면 심플렉스解法과 同等하게(equivalent) 될 수 있다는 사실은 주목할 만하다.

本 硏究의 결과, 最大흐름問題에 있어서 理論的으로 우수한 解法이 實際的으로도 반드시 우수한 解法이 아닐 수 있다는 사실을 확인하였다. 또한 實際 最大흐름問題를 效率的으로 해결하기 위해서는, 理論的으로 우수한 解法보다 GKMW와 같은 심플렉스解法을 사용하는 것이 현명하다고 볼 수 있다. 그러나 당면한 最大흐름問題의 크기나 네트워크의 구조또는 應用分野 등에 따라서는 다른 결론이 나올 수도 있다. 예컨대 日程計劃問題를 반복해서 풀어야 한다면 FFBR 解法을 채택하는 것이 좋을 것이며, 問題의 크기가 매우 크다면 封鎖흐름解法을 사용하는 것이 유리할 수도 있다.

本 硏究와 관련된 向後 硏究課題들로는 다음과 같은 것들이 있다.

(i) 새로운 最大흐름問題解法의 개발을 통한 理論的 效率性 증대 : 예컨대 Tarjan[30]은 $O(|N|\cdot|A|\log|N|^2/|A|)$의 計算時間으로 最大흐름의 발견이 가능하다고 예측하였다.

(ii) 기존解法의 變形, 複合 또는 새로운 資料構造(data structure)의 導入등을 통한 實際的 效率性 증대 : 가령 간단한 레이블解法으로 初期흐름을 발견한 뒤, 심플렉스解法 또는 封鎖흐름으로 最大흐름에 도달하는 방법도 고려해 볼 수 있다.

(iii) 竝列컴퓨터(parallel computer)를 위한 竝列解法의 개발 및 활용 : 예컨대 Shiloach-Vishkin[27]은 K個의 프로세서(processor)를 사용할 때 $O(\{|N|^3\log|N|\}/K)$의 計算時間으로 最大흐름을 발견하는 解法을 발견하였다.

(iv) 實際的 效率性에 관한 새로운 試圖 : PC를 이용한 實驗, Sleator 또는 Gabow 解法의 프로그램 작성 및 實驗 등.

<參 考 文 獻>

1. Aho, A.H.: Hopcroft, J.E.; and Ullman, J.D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass.: 1974.

2. Cherkasky, B.V. "An Algorithm for Construction of Maximal Flows in Networks with Complexity $O(V^2JE)$ Operations." *Mathematical Methods for the Solution of Economic Problems* 7(1977) : 117~125(in Russian).
3. Cheung, T. "Computational Comparison of Eight Methods for the Maximum Network Flow Problem." *ACM Transactions on Mathematical Software* 6(March, 1980) : 1~16.
4. Dantzig, G.B. *Linear Programming and Extensions.* Princeton University Press, Princeton: 1963.
5. Dinic, E.A. "Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation." *Soviet Mathematics Doklady* 11(1970) : 1277~1280.
6. Edmonds, J., and Karp, R.M. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems." *JACM* 19(April 1972) : 248~264.
7. Faaland, B., and Schmitt, T. "Scheduling Tasks under Due Dates in a Fabrication/Assembly Process."(to appear in *Operations Research*).
8. Faaland, B.; Kim, K.; Schmitt, F. "Maximal Flow Aloorithms for the Closure Problem." Working Paper, University of Washington, 1987.
9. Fong, C.O., and Rao, M.R. "Accelerated Labeling Algorithms for the Maximum Flow Problem with Applications to Transportation and Assignment Problems." Working Prape 7222, Graduate School of Management, University of Rochester, 1974.
10. Ford, L.R., and Fulkerson, D.R. "Maximal Flow Through a Network." *Canadian Journal of Mathematics* 8(1956) : 399~404.
11. ……. *Flows in Networks.* Princeton University Press, Princeton: 1962.
12. Fulkerson, D.R., and Dantzig, G.B. "Computation of Maximal Flows in Networks." *Naval Research Logistics Quarterly* 2(1955) : 277~283.
13. Gabow, H.N. "Scaling Algorithms for Network Problems." *Proceedings of the 24th Annual Symposium on Foundations of Computer Science* (1983) : 248~257.
14. Galil, Z. "An $O(V^{5/3}E^{2/3})$ Algorithm for the Maximal Flow Problem." *Acta Informatica* 14 (1980) : 221~242.
15. Galil, Z., and Naamad, A. "An $O(EV\log^2 V)$ Algorithm for the Maximal Flow Problem." *Journal of Computer and System Sciences* 21(1980) : 203~217.
16. Glover, F.; Klingman, D.; Mote, J.; and Whitman, D. "Comprehensive Computer Evaluation and Enhancement of Maximum Flow Algorithms." *Applications of Management Science* 3(1983) : 109~175.
17. ……. "A Primal Simplex Variant for the Maximum-Flow Problem." *Naval Research Logistics Quarterly* 31(1984) : 41~61.
18. Karzanov, A.V. "Determining the Maximal Flow in a Network by the Method of Preflows." *Soviet Mathematics Doklad* 15(1974) : 434~437.
19. Kim, K. "On the Maximal Flow Problem and Some Related Problems." Working Paper, Pusan National University, 1987.
20. Lerchs, H., and Grossmann, I.F. "Optimum Design of Open-Pit Mines." *Canadian Mining and Metallurgical Bulletin* 58(1965) : 47~54.
21. Lin, P.M., and Leon, B.J. "Improving the Efficiency of Labeling Algorithms for Maximum Flow in Networks." *Proceedings of IEEE International Symposium on Circuits and Systems*(1974) :

162~166.

22. Malhotra, V.M.; Kumar, M.P.; and Maheshwari, S.N. "An $O(|V|^3)$ Algorithm for Finding Maximum Flows in Networks." *Information Processing Letters* 7(1978) : 277~278.
23. Mamer, J.W., and Smith, S.A. "Optimizing Field Repair Kits Based on Job Completion Rate." *Management Science* 28(1982) : 1328~1333.
24. Nijenhuis, A., and Wilf, H.S. *Combinatorial Algorithms for Computers and Calculators*, Second Edition. Academic Press, New York: 1978.
25. Queyranne, M. "Theoretical Efficiency of the Algorithm 'Capacity' for the Maximum Flow Problem." *Mathematics of Operations Research* 5(1980) : 258~266.
26. Shiloach, Y. "An $O(n \cdot I \log^2 I)$ Maximum-Flow Algorithm." Technical Report STAN-CS-78-702, Computer Science Department, Stanford University, 1978.
27. Shiloach, V., and Vishkin, U. "An $O(n^2 \log n)$ Parallel Max-Flow Algorithm." *Journal of Algorithms* 3(1982) : 128~146.
28. Sleator, D.D. "An $O(nm \log n)$ Algorithm for Maximum Network Flow." Ph.D. Dissertation, Stanford University, 1980.
29. Tarjan, R.E. "A Simple Version of Karzanov's Blocking Flow Algorithm." *Operations Research Letters* 2(1984) : 265~268.
30. ……. "Algorithms for Maximum Network Flow." *Mathematical Programming Study* 26(1986) : 1~11.
31. Tucker, A. "A Note on Convergence of the Ford-Fulkerson Flow Algorithm." *Mathematics of Operations Research* 2(1977) : 143~144.
32. Yao, F. "Maximum Flows in Networks." *Proceedings of Symposia in Applied Mathematics* 26 (1982) : 31~43.
33. Zadeh, N. "Theoretical Efficiency of the Edmonds-Karp Algorithm for Computing Maximal Flows." *JACM* 19(1972) : 184~192.