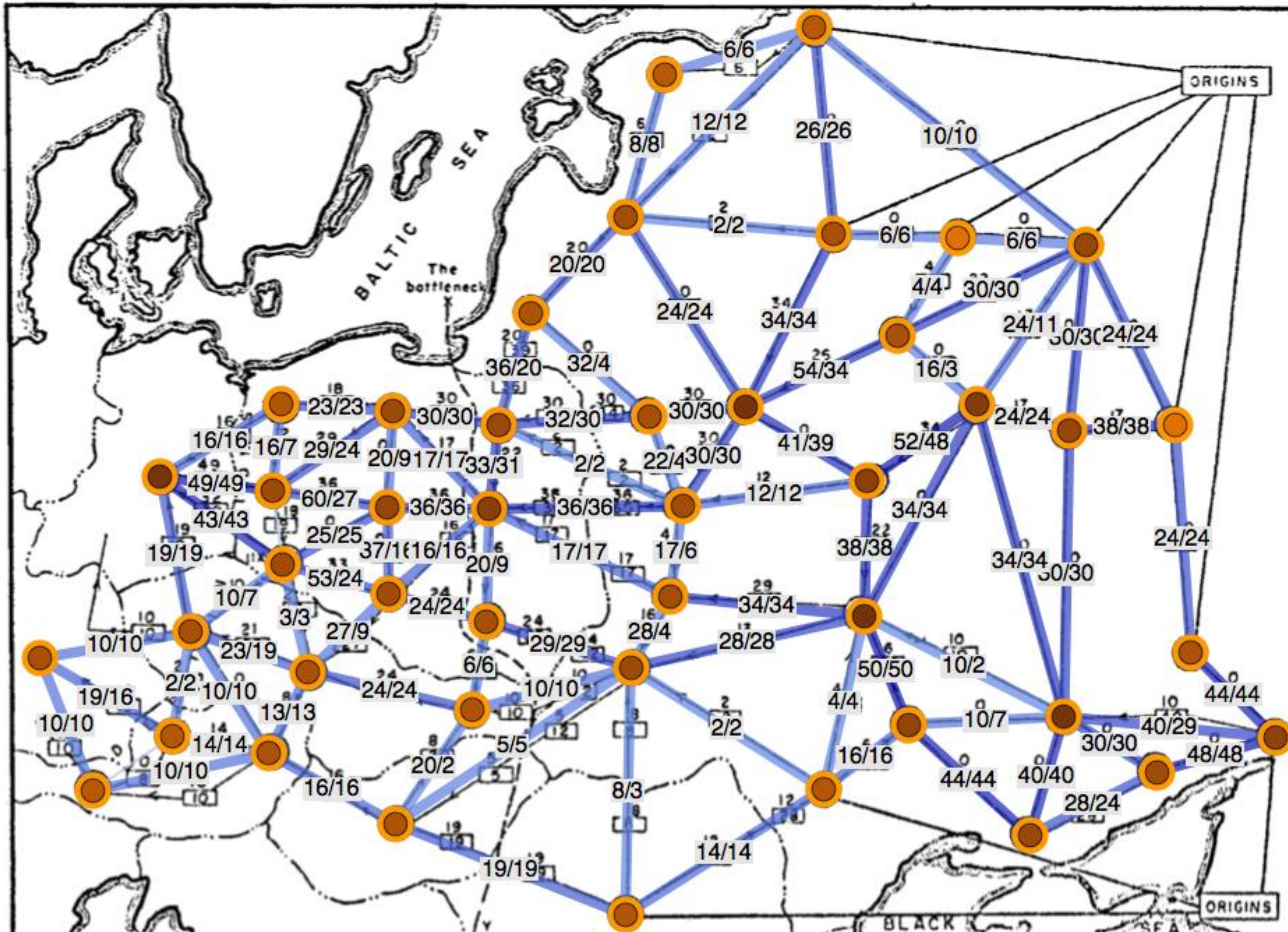
第7章 网络流算法

7.1 最大流问题
- 网络流及其性质
- Ford-Fulkerson算法
- Dinic有效算法

7.2 最小费用流
- Floyd算法
- 最小费用流的负回路算法
- 最小费用流的最短路径算法

7.3-7.4 网络流的应用



ORIGINS
BALTIC SEA
The bottleneck
BLACK SEA
ORIGINS
6/6
12/12
26/26
10/10
8/8
2/2
6/6
6/6
20/20
24/24
34/34
4/4
30/30
24/11
30/30
24/24
36/20
32/4
54/34
16/3
23/23
30/30
32/30
30/30
41/39
52/48
24/24
38/38
16/16
16/7
29/24
20/9
17/17
33/31
2/2
22/4
30/30
49/49
60/27
36/36
36/36
12/12
34/34
43/43
25/25
37/16
16/16
17/17
17/6
38/38
24/24
19/19
53/24
20/9
34/34
30/30
10/7
3/3
24/24
34/34
27/9
28/4
10/10
23/19
29/29
28/28
6/6
50/50
10/2
44/44
2/2
10/10
19/16
13/13
24/24
10/10
4/4
10/7
40/29
10/10
14/14
2/2
5/5
16/16
30/30
48/48
10/10
20/2
8/3
44/44
40/40
16/16
28/24
14/14
19/19

7.1 最大流算法

网络流及其性质

容量网络 $N=\langle V,E,c,s,t\rangle$,其中$N=\langle V,E\rangle$是有向连通图, 记 $n=|V|$, $m=|E|$, $c:E\to R^*$是边的容量, s 和 t 是两个特殊的顶点, s 称作发点(源), t 称作收点(汇), 其余顶点称作中间点.

设 $f:E\to R^*$满足下述条件:

(1) 容量限制 $\forall\langle i,j\rangle\in E,\ f(i,j)\le c(i,j)$,

(2) 平衡条件 $\forall i\in V-\{s,t\}$,

$$\sum_{\langle j,i\rangle\in E} f(j,i) = \sum_{\langle i,j\rangle\in E} f(i,j)$$

称 f 是N上的一个可行流, 称 s 的净流出量 $v(f)$为 f 的流量,即

$$v(f)=\sum_{\langle s,j\rangle\in E} f(s,j)-\sum_{\langle j,s\rangle\in E} f(j,s)$$

流量最大的可行流称作最大流.

最大流问题

最大流问题 求给定容量网络上的最大流.

$$\max\ v(f)$$

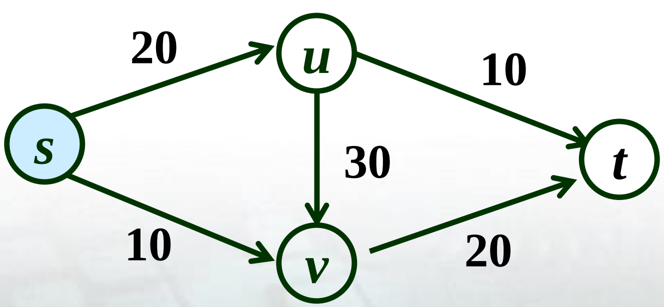
$$\text{s.t.}\quad f(i,j) \le c(i,j), \qquad <i,j>\in E$$

$$\sum_{<j,i>\in E} f(j,i) - \sum_{<i,j>\in E} f(i,j) = 0, \qquad i\in V-\{s,t\}$$

$$v(f) - \sum_{<s,j>\in E} f(s,j) + \sum_{<j,s>\in E} f(j,s) = 0$$

$$f(i,j) \ge 0, \qquad <i,j>\in E$$

$$v(f) \ge 0$$



最大流

$f(s,u)=20$, $f(u,t)=10$, $f(s,v)=10$,
$f(u,v)=10$, $f(v,t)=20$

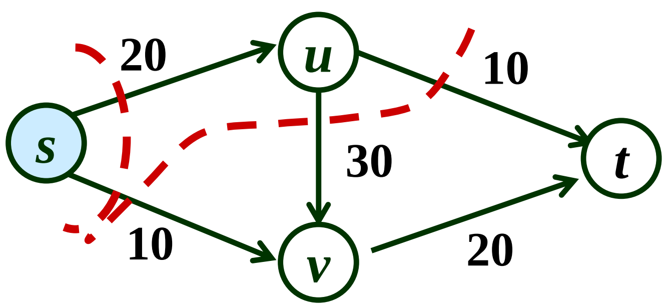
最小割

定义 设容量网络 $N=<V,E,c,s,t>$, $A\subset V$ 且 $s\in A$, $t\in V-A$, 称

$$(A,V-A)=\{<i,j> \mid <i,j>\in E \text{ 且 } i\in A, j\in V-A\}$$

为 N 的**割集**, $c(A,V-A)=\sum\limits_{<i,j>\in(A,\bar{A})} c(i,j)$ 为割集 $c(A,V-A)$ 的**容量**.

容量最小的割集称作**最小割集**.



$A=\{s,u\}$

$c(\{s,u\},\{v,t\})=50$

$A=\{s\}$

$c(\{s\},\{u,v,t\})=30$

引理1

引理1 设容量网络 $N=<V, E, c, s, t>$, f 是N上的任一可行流, $A\subset V$且 $s\in A$, $t\in V-A$, 则 $v(f)=\sum_{<i,j>\in(A,\bar{A})} f(i,j)-\sum_{<j,i>\in(\bar{A},A)} f(j,i)$

证

$$v(f)=\sum_{<s,j>\in E} f(s,j)-\sum_{<j,s>\in E} f(j,s)$$

$$=\sum_{<s,j>\in E} f(s,j)-\sum_{<j,s>\in E} f(j,s)+\sum_{i\in A-\{s\}}\left\{\sum_{<i,j>\in E} f(i,j)-\sum_{<j,i>\in E} f(j,i)\right\}$$

$$=\sum_{i\in A}\left\{\sum_{<i,j>\in E} f(i,j)-\sum_{<j,i>\in E} f(j,i)\right\}$$

$$=\sum_{i\in A}\sum_{<i,j>\in E} f(i,j)-\sum_{i\in A}\sum_{<j,i>\in E} f(j,i)$$

北京大学

引理

$$=\sum_{i\in A}\sum_{<i,j>\in E} f(i,j)-\sum_{i\in A}\sum_{<j,i>\in E} f(j,i)$$

$$=\sum_{\substack{i\in A\\ j\in A}}\sum_{<i,j>\in E} f(i,j)+\sum_{\substack{i\in A\\ j\in \overline{A}}}\sum_{<i,j>\in E} f(i,j)-\sum_{\substack{i\in A\\ j\in A}}\sum_{<j,i>\in E} f(j,i)-\sum_{\substack{i\in A\\ j\in \overline{A}}}\sum_{<j,i>\in E} f(j,i)$$

$$=\sum_{<i,j>\in(A,\overline{A})} f(i,j)-\sum_{<j,i>\in(\overline{A},A)} f(j,i)$$

引理2 设 f 是任一可行流, $(A,V-A)$ 是任一割集, 则

$$v(f)\le c(A,V-A)$$

引理3 设 f 是一个可行流, $(A,V-A)$ 是一个割集. 如果 $v(f)=c(A,V-A)$, 则 f 是最大流, $(A,V-A)$ 是最小割集.

i-j 增广链

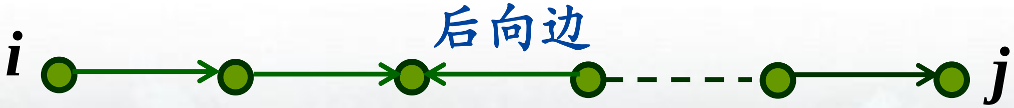
定义 设容量网络 $N=<V,E,c,s,t>$, f 是N上的一个可行流.

(1) N中流量等于容量的边称作**饱和边**, 流量小于容量的边称作**非饱和边**.

(2) 流量等于0的边称作**零流边**, 流量大于0的边称作**非零流边**.

(3) 不考虑边的方向, N中从顶点 i 到 j 的一条边不重复的路径称作 **i-j 链**. i-j 链的方向是从 i 到 j. 链中与链的方向一致的边称作**前向边**, 与链的方向相反的边称作**后向边**.

(4) 如果链中所有前向边都是非饱和的, 所有后向边都是非零流的, 则称这条链为 **i-j 增广链**.

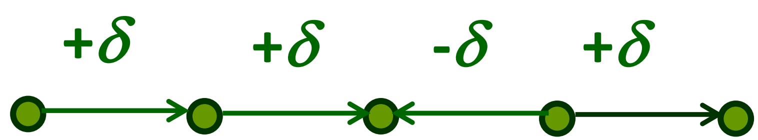


最大流的性质

定理1 可行流 f 是最大流 $\Leftrightarrow$ 不存在关于 f 的 s-t 增广链.

证 必要性. 假设P是一条可行流 f 的 s-t 增广链, 修改量δ等于P上所有前向边容量与流量之差及所有后向边流量的最小值. 令

$$f'(i,j)=\begin{cases} f(i,j)+\delta, & <i,j>\text{是}P\text{的前向边} \\ f(i,j)-\delta, & <i,j>\text{是}P\text{的后向边} \\ f(i,j), & \text{否则} \end{cases}$$



f' 是可行流, 且 $v(f') = v(f)+\delta$, 与 f 是最大流矛盾.

最大流的性质（续）

充分性．假设不存在关于 f 的 s-t 增广链. 令

$$A=\{\, j\in V \mid \text{存在关于} f \text{ 的 } s\text{-}j \text{ 增广链} \,\}$$

$s\in A$, $t\notin A$. $\forall\langle i,j\rangle\in(A,V-A)$, 必有 $f(i,j)=c(i,j)$. 否则, 可以把 s-i 增广链通过非饱和的$\langle i,j\rangle$延伸到 j, 与 $j\notin A$矛盾.

$\forall\langle j,i\rangle\in(V-A,A)$, 必有 $f(j,i)=0$. 否则, 可以把 s-i 增广链通过非零流的$\langle j,i\rangle$ (后向边)延伸到 j, 与 $j\notin A$矛盾. 于是, 由引理1

$$v(f)=\sum_{\langle i,j\rangle\in(A,\overline{A})} f(i,j)-\sum_{\langle j,i\rangle\in(\overline{A},A)} f(j,i)=c(A,\overline{A})$$

又由引理3, 得证 f 是最大流.

定理2 (最大流最小割集定理)

容量网络的最大流的流量等于最小割集的容量.

Ford-Fulkeson算法

设计思想

从给定初始可行流 (通常取零流) 开始

取当前可行流的 s-t 增广链 P, 修改P上流量得到新的可行流.

重复进行, 直到不存在 s-t 增广链为止.

标号法

从 s 开始给顶点作标号, 直到 t 得到标号为止.

顶点 j 得到标号表示已找到从 s 到 j 的增广链.

标号为 (l_j, δ_j)

δ_j 等于 s-j 链上可增加流量的最大值

$l_j = +i$ 表示链是从 i 到 j 的且$<i, j>$是前向边

$l_j = -i$ 表示链是从 i 到 j 的且$<j, i>$是后向边

顶点状态 已标号已检查的, 已标号未检查的, 未标号的.

FF算法伪码

Ford-Fulkerson算法

1. $f \leftarrow 0$ // 零流为初始可行流
2. $T \leftarrow \{s\}$, $R \leftarrow V - \{s\}$, $l_s \leftarrow \Delta$, $\delta_s \leftarrow +\infty$ // T:已标未查，R:未标，
3. while $T \neq \varnothing$ do
4. 取 $i \in T$, $T \leftarrow T - \{ i \}$ // 检查顶点 i
5. for 所有R中与 i 邻接的 j do
6. if $<i, j> \in E$ 且 $f(i, j) < c(i, j)$ then
7. $l_j \leftarrow +i$, $\delta_j \leftarrow \min\{\delta_i, c(i, j) - f(i, j)\}$, $R \leftarrow R - \{ j \}$, $T \leftarrow T \cup \{ j \}$
8. if $j = t$ then goto 13 // 找到 s-t 增广链
9. if $<j, i> \in E$ 且 $f(j, i) > 0$ then
10. $l_j \leftarrow -i$, $\delta_j \leftarrow \min\{\delta_i, f(j,i)\}$, $R \leftarrow R - \{ j \}$, $T \leftarrow T \cup \{ j \}$
11. if $j = t$ then goto 13 // 找到 s-t 增广链
12. reture f //无 s-t 增广链, 结束

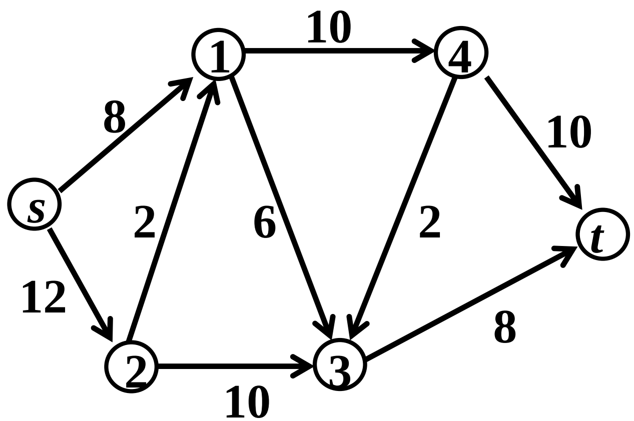
FF算法伪码（续）

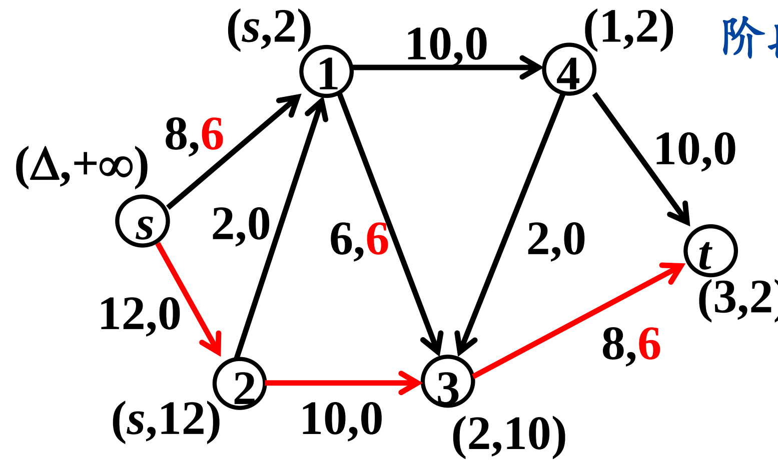
13. $\delta \leftarrow \delta_t$ // 沿增广链回溯, 修改流量
14. if $l_j = \Delta$ then goto 2 // 重新开始下一阶段标号
15. if $l_j = +i$ then
16. $f(i, j) \leftarrow f(i, j) + \delta$, $j \leftarrow i$
17. if $l_j = -i$ then
18. $f(j, i) \leftarrow f(j, i) - \delta$, $j \leftarrow i$
19. goto 14

每个阶段的工作：

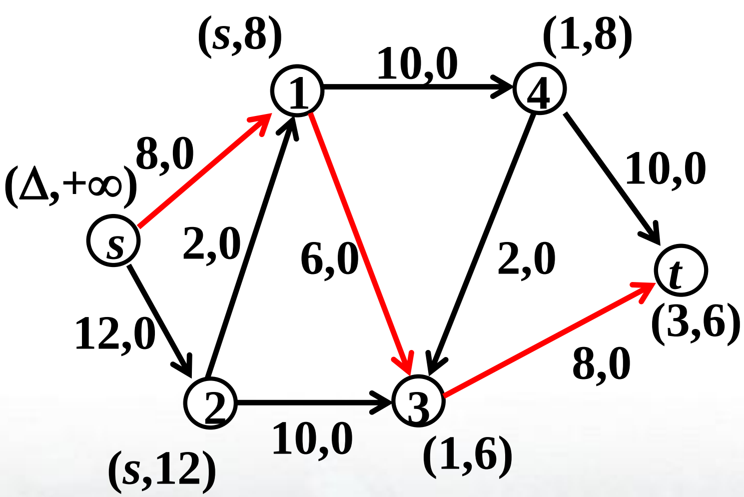
- 通过标记过程寻找一条 s-t 增广链
- 沿链回溯修改链上的流值

算法运行实例：例1

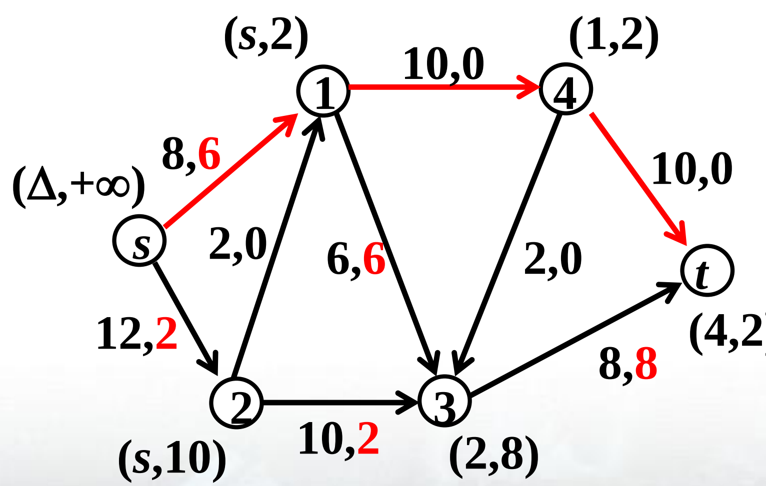




阶段2

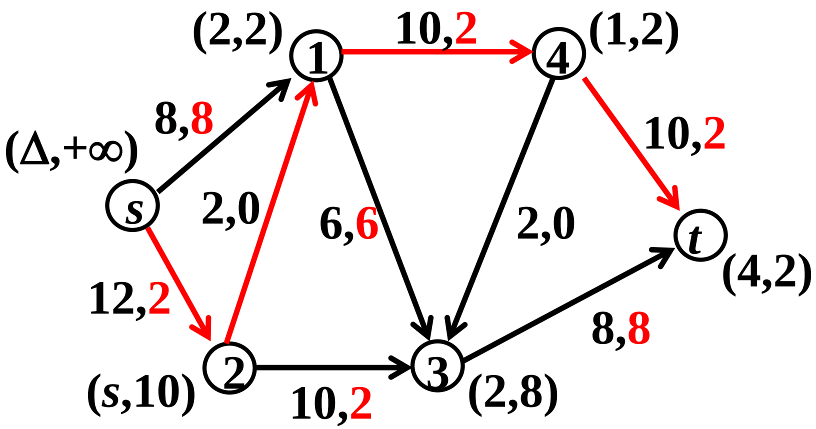


阶段1

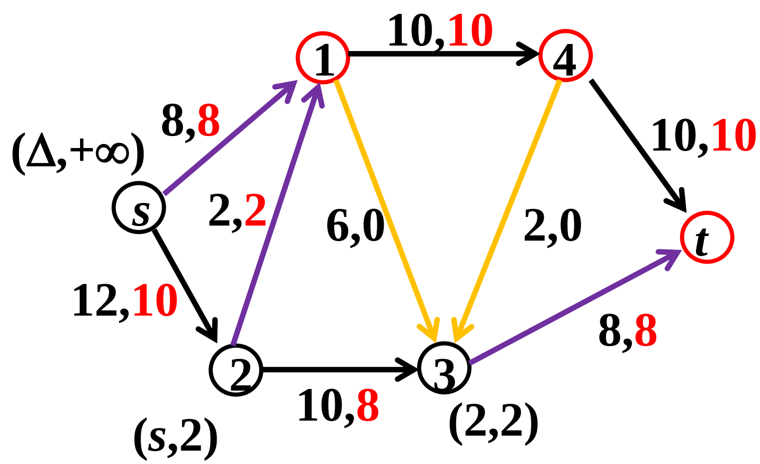


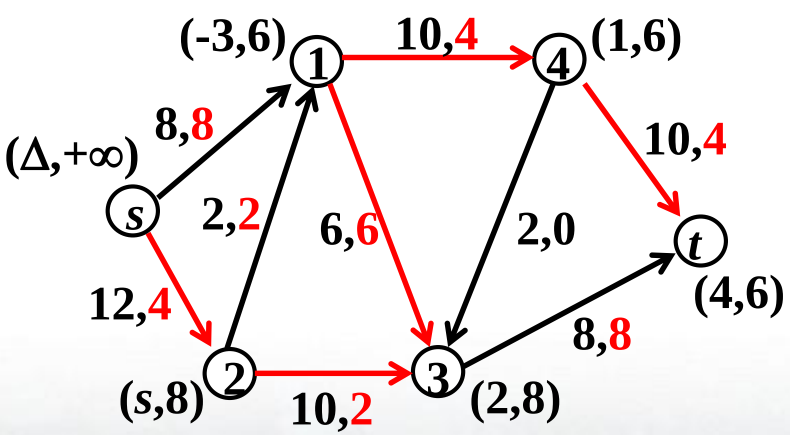
阶段3

例1(续)



阶段4





阶段5

解：$v(f)=18$

$f(s,1)=8$, $f(s,2)=10$, $f(1,3)=0$, $f(1,4)=10$, $f(2,1)=2$, $f(2,3)=8$, $f(3,t)=8$, $f(4,3)=0$, $f(4,t)=10$,

最小割 $(\{s,2,3\},\{1,4,t\})$

$c(\{s,2,3\},\{1,4,t\})=18$

算法运行时间

时间复杂度

假设所有容量都是正整数, 则算法时间复杂度为 $O(mC)$, 其中

$$C = \sum_{<s,j>\in E} c(s,j)$$

最大流量 $v^* \le C$. δ 是正整数, 每次流量至少加1, 至多C个阶段. 而每阶段标号和修改增广链流量需要 $O(m)$.

提高效率的途径

每次求最短的 s-t 增广链

一次标号修改尽可能多条 s-t 增广链上的流量

辅助网络

定义 设容量网络 $N=<V, E, c, s, t>$, f 是N上的一个可行流. 关于 f 的辅助网络 $N(f) = <V, E(f), ac, s, t>$, 其中

$$E^{+}(f) = \{<i, j> \mid <i, j>\in E \text{ 且 } f(i, j) < c(i, j) \}$$

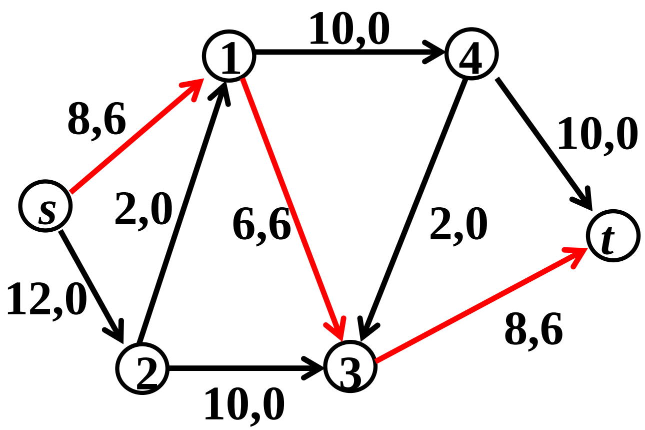
$$E^{-}(f) = \{<j, i> \mid <i, j>\in E \text{ 且 } f(i, j) > 0 \}$$

$$E(f) = E^{+}(f)\cup E^{-}(f)$$

$$ac(i, j) = \begin{cases} c(i, j) - f(i, j), & <i, j>\in E^{+}(f) \\ f(j, i), & <i, j>\in E^{-}(f) \end{cases}$$

ac 称作辅助容量. $N(f)$也是容量网络.

辅助网络的实例

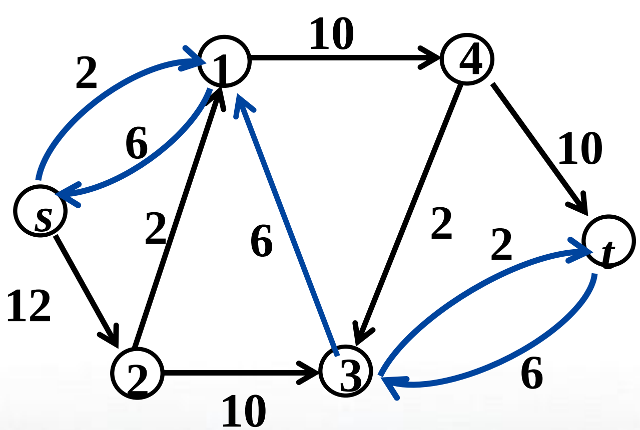


容量网络: $N = <V,E,c,s,t>$,

可行流 f:

$f(s,1)=6$, $f(1,3)=6$, $f(3,t)=6$

其余 $f(i,j)=0$



辅助网络:

$N(f)= <V,E(f),ac,s,t>$,

引理4

引理4 设 N 的最大流量为 v^*, f 是可行流, 则 $N(f)$ 的最大流量为 $v^*-v(f)$.

证 N中的割集$(A,V-A)$也是$N(f)$中的割集, 记作$(A,V-A)'$. $(A,V-A)'$由$(A,V-A)$中关于f的非饱和边E_1和$(V-A,A)$中关于f的非零流的反向边E_2组成, $(A,V-A)'$的容量

$$
\begin{aligned}
ac(A,\overline{A})' &= \sum_{<i,j>\in E_1}\{c(i,j)-f(i,j)\}+\sum_{<i,j>\in E_2} f(j,i)\\
&= \sum_{<i,j>\in(A,\overline{A})}\{c(i,j)-f(i,j)\}+\sum_{<j,i>\in(\overline{A},A)} f(j,i)\\
&= \sum_{<i,j>\in(A,\overline{A})} c(i,j)-\left\{\sum_{<i,j>\in(A,\overline{A})} f(i,j)-\sum_{<j,i>\in(\overline{A},A)} f(j,i)\right\}\\
&=c(A,V-A)-v(f)
\end{aligned}
$$

由最大流最小割集定理, 得证$N(f)$的最大流量等于$v^*-v(f)$.

引理5

定义 设 f 是N上的一个可行流, g是$N(f)$上的一个可行流, 定义 $f' = f + g$ 如下： $\forall \langle i,j\rangle \in E$,

$$f'(i,j) = f(i,j) + g(i,j) - g(j,i)$$

规定 $\langle i,j\rangle \notin E(f)$时, $g(i,j) = 0$.

引理5 设 f 是N上的一个可行流, g是$N(f)$上的一个可行流, 则 $f + g$是 N上的可行流, 且$v(f+g) = v(f) + v(g)$.

证 容量限制. $\forall \langle i,j\rangle \in E$,

$$0 \le g(i,j) \le c(i,j) - f(i,j) \qquad (1)$$

$$0 \le g(j,i) \le f(i,j) \Rightarrow -f(i,j) \le -g(j,i) \le 0 \qquad (2)$$

(1)+(2) $\quad -f(i,j) \le g(i,j) - g(j,i) \le c(i,j) - f(i,j)$

$+f(i,j) \quad 0 \le f'(i,j) = f(i,j) + g(i,j) - g(j,i) \le c(i,j)$

引理5（续）

平衡条件 $\forall i \in E-\{s,t\}$,

$$\sum_{<j,i>\in E} f'(j,i) = \sum_{<j,i>\in E}\{f(j,i)+g(j,i)-g(i,j)\}$$

$$= \sum_{<j,i>\in E} f(j,i) + \underline{\sum_{\substack{<j,i>\in E(f)\\ \wedge <j,i>\in E}} g(j,i) - \sum_{\substack{<i,j>\in E(f)\\ \wedge <j,i>\in E}} g(i,j)}$$

流入 i

$$\sum_{<i,j>\in E} f'(i,j) = \sum_{<i,j>\in E}\{f(i,j)+g(i,j)-g(j,i)\}$$

$$= \sum_{<i,j>\in E} f(i,j) + \underline{\sum_{\substack{<i,j>\in E(f)\\ \wedge <i,j>\in E}} g(i,j) - \sum_{\substack{<j,i>\in E(f)\\ \wedge <i,j>\in E}} g(j,i)}$$

从 i 流出

而

$$\sum_{\substack{<j,i>\in E(f)\\ \wedge <j,i>\in E}} g(j,i) - \sum_{\substack{<i,j>\in E(f)\\ \wedge <j,i>\in E}} g(i,j) - \sum_{\substack{<i,j>\in E(f)\\ \wedge <i,j>\in E}} g(i,j) + \sum_{\substack{<j,i>\in E(f)\\ \wedge <i,j>\in E}} g(j,i)$$

$$= \sum_{<j,i>\in E(f)} g(j,i) - \sum_{<i,j>\in E(f)} g(i,j) = 0$$

$$\sum_{<j,i>\in E} f'(j,i) - \sum_{<i,j>\in E} f'(i,j) = 0$$

$f' = f+g$ 是 N 的可行流

引理5（续）

$$v(f')=\sum_{<s,j>\in E}\{f(s,j)+g(s,j)-g(j,s)\}$$

$$-\sum_{<j,s>\in E}\{f(j,s)+g(j,s)-g(s,j)\}$$

$$=\sum_{<s,j>\in E}f(s,j)+\sum_{\substack{<s,j>\in E(f)\\ \wedge<s,j>\in E}}g(s,j)-\sum_{\substack{<j,s>\in E(f)\\ \wedge<s,j>\in E}}g(j,s)$$

$$-\sum_{<j,s>\in E}f(j,s)-\sum_{\substack{<j,s>\in E(f)\\ \wedge<j,s>\in E}}g(j,s)+\sum_{\substack{<s,j>\in E(f)\\ \wedge<j,s>\in E}}g(s,j)$$

$$=\{\sum_{<s,j>\in E}f(s,j)-\sum_{<j,s>\in E}f(j,s)\}+\{\sum_{<s,j>\in E(f)}g(s,j)-\sum_{<j,s>\in E(f)}g(j,s)\}$$

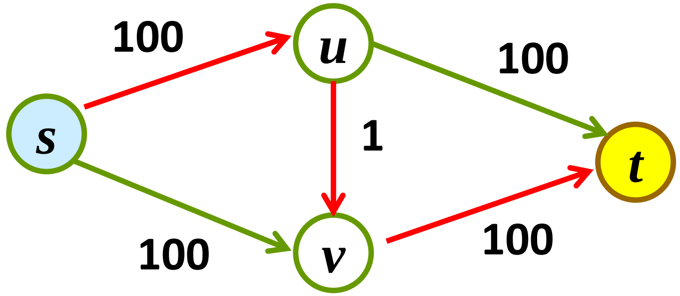
$$=v(f)+v(g)$$

算法实现

算法：
- 给出初始流
- 通过辅助网络选择增广路径以增加流值

问题：在存在多条增广路径情况下，如何选择 g？

一个坏的例子：



可能执行200次增广操作

解决方案：通过分层辅助网络，每次增广都选辅助网络中的极大流