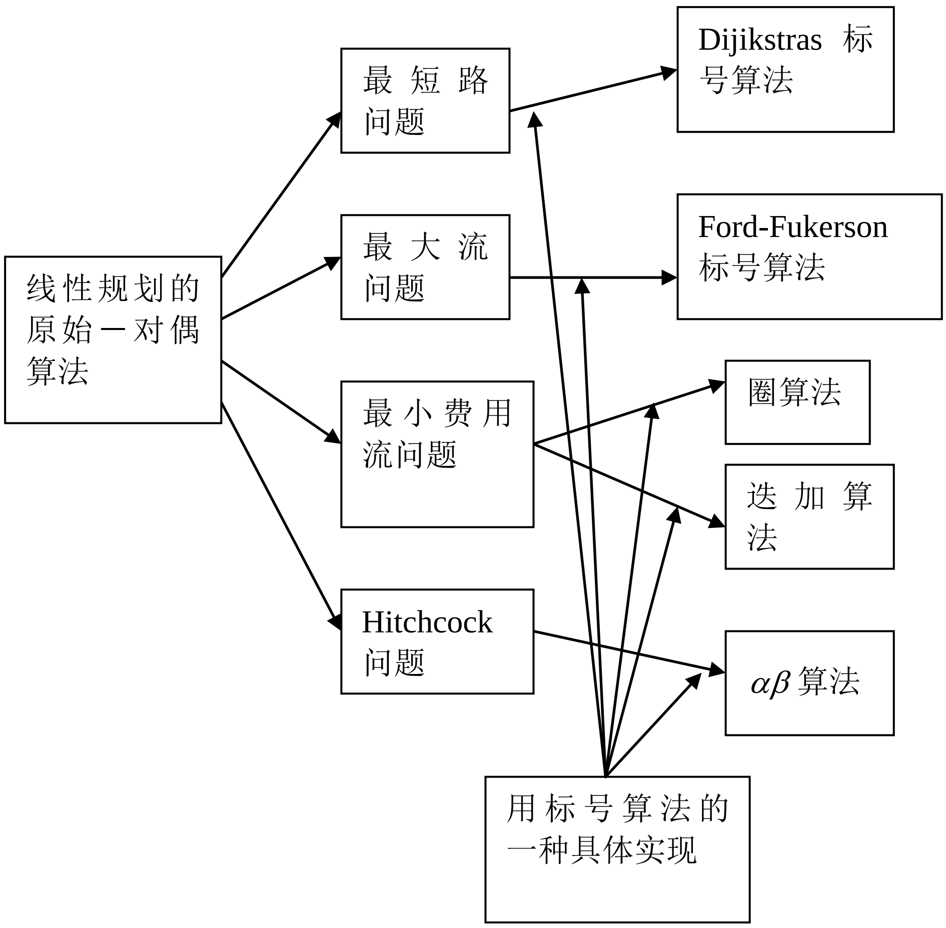
第六章 组合优化问题的有效算法

6.2 最大流问题的有效算法



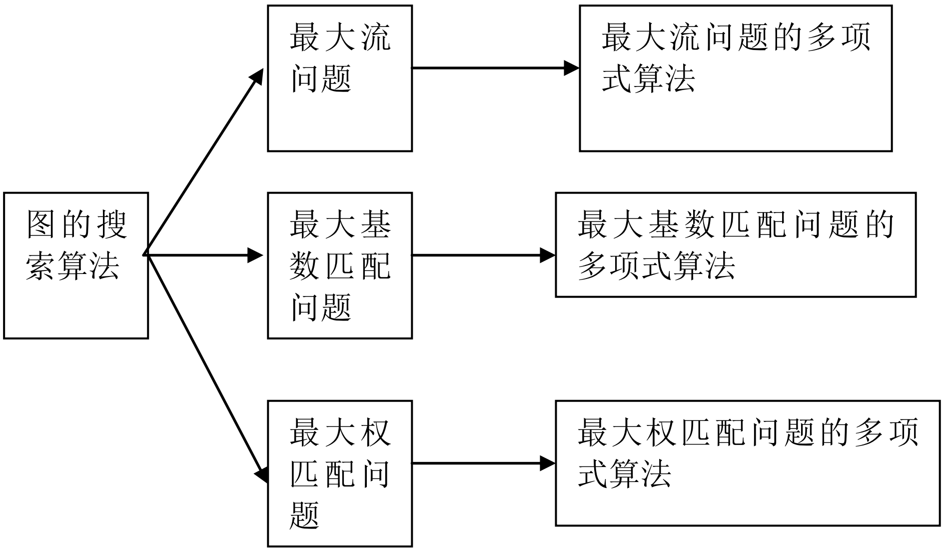
线性规划问题的单纯形算法不是多项式算法。
问题：线性规划问题存在多项式算法吗？
回答：是。
线性规划问题的多项式算法－内点算法。

最大流问题的 Ford-Fulkerson 标号算法不是多项式算法。
问题：最大流问题存在多项式算法吗？
回答：是。
下面进行讨论。

事实上，将最大流问题的 Ford-Fulkerson 标号算法做微小的修改，即可使之成为多项式算法。

除最大流问题外，这一思想也适用于有效的求解某些有意义的组合优化问题。

首先考察一个基本的图的算法－搜索算法。各种不同的搜索是许多图算法的核心。



一、图的搜索

一个图$G=(V,E)$可以用它的邻接表$A(v), v\in V$表示，其中$A(v)$表示与v邻接的点的集合，即对于$v\in V$， $A(v)=\{u:(v,u)\in E\}$。

一个合理的假设：两个节点之间平均至少有一条边相连，即我们总假定：

$$|E|\geq\frac{1}{2}|V|\text{，或者}|V|\leq 2|E|\text{。}$$

搜索算法的基本思想：从一个节点v_1开始，给它一个“标记”，然后给v_1的邻点以标记，再给它的邻点的邻点以标记，一直下去，直到没有节点可标记为止。

搜索算法：

Input: 用邻接表表示的图G；节点v_1（开始标记的点）

Output: 自v_1有路可到达的节点所组成的图

begin

$\boxed{Q=\{v_1\}}$

while $Q \neq \phi$ do

begin

$\forall v \in Q;$

$\boxed{\text{标记}\, v}$

$Q = Q \setminus v$

for all $v^{'} \in A(v)$ do

$\boxed{\text{if } v^{'} \text{没有标记，then } Q := Q \bigcup \{v^{'}\}}$

end

end

Q中保存与已标号的节点相邻且尚没有被标号的节点。当$Q=\phi$时，算法就终止。

定理 6.2 用上述搜索算法标记图$G=(V,E)$中所有与v_1相连通的节点所用的时间为$O(|E|)$。

证明：假定节点v与v_1有路相连，则可以对路长用归纳法证明v必然将被标记；另一方面，假如v和v_1无路相连，则可以证明v一定不能被标记。

上述搜索算法的计算复杂性由三部分组成：

（1） 开始步骤：所需要的计算量为常数；

（2） 保存集合Q：显然加到Q和从Q里去掉元素的次数之和最多为$2|V|$，而$|V| \leq 2|E|$。每一次将v加到Q或从Q里去掉的操作可以通过两个或三个初等运算来完成；

将 v 加到 Q 里的操作过程

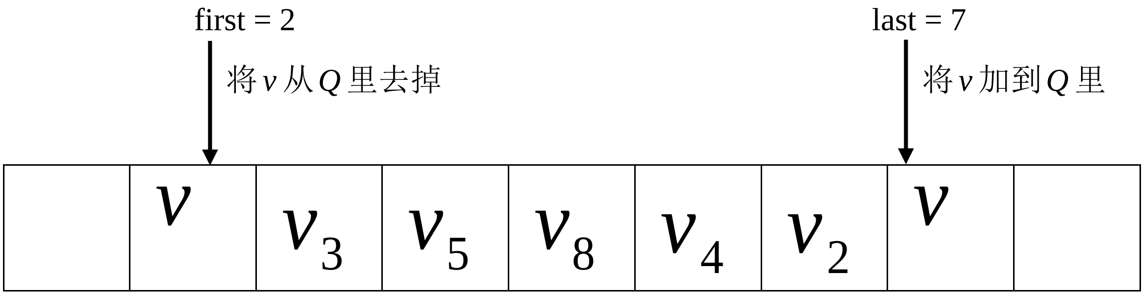
last := last + 1;

Q[last] := v;

将 v 从 Q 里去掉的操作过程

first := first + 1;

v := Q[first];



Q 中的元素按到达次序排列

开始：last＝first＝0，Q 是 $|V|$ 个元素的数组；$Q=\phi \Leftrightarrow$ first＝last。

（3） 搜索邻接表：对每一个邻接表中的每一个元素的搜索时间为常量，而这些邻接表长度之和为 $2|E|$，故所需时间为 $O(|E|)$

所以，总的时间界为 $O(|E|)$ 。

例 1： 搜索算法可以用于发现一个图是否连通，如果不连通，则可以找出图的所有连通分图（极大连通子图）：

（1）图 G 连通 $\Leftrightarrow$ 当搜索后所有的节点都得到了标记。

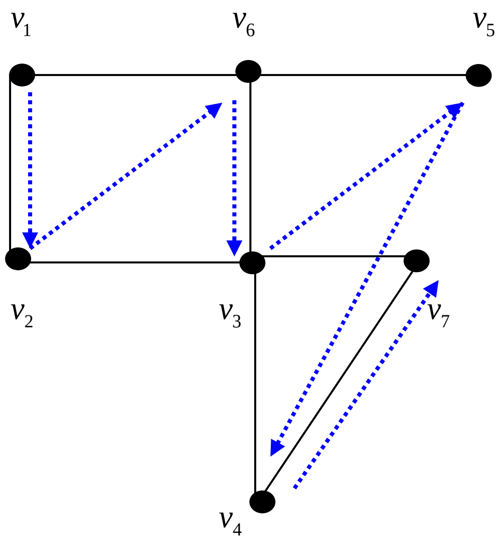
（2）如果图 G 连通，从 v_1 开始标记，当搜索后，所有得到标记的节点构成了 v_1 所在的图 G 极大连通子图

注：上述搜索算法不是完全确定的，循环的每一步必须确定选取 Q 中元素的方法。

选取 Q 中元素的法则：

- 广探法（BFS）：把 Q 设想为顾客排队，总是选取排队时间最长的顾客从 Q 中去掉（FIFO）。
- 深探法（DFS）：按后进先出（LIFO）方式进行搜索。这个算法为向纵深方向搜索，从而产生一条尽可能长的路；仅当这条路不能再延长时，才能转到一个新的探索方向，所以称为深探法。

例 2 用 BFS 给出节点的访问顺序：



$Q=\{v_1\}$ v_1

$Q=\{v_2,v_6\}$ $v_1\to v_2$

$Q=\{v_6,v_3\}$ $v_1\to v_2\to v_6$

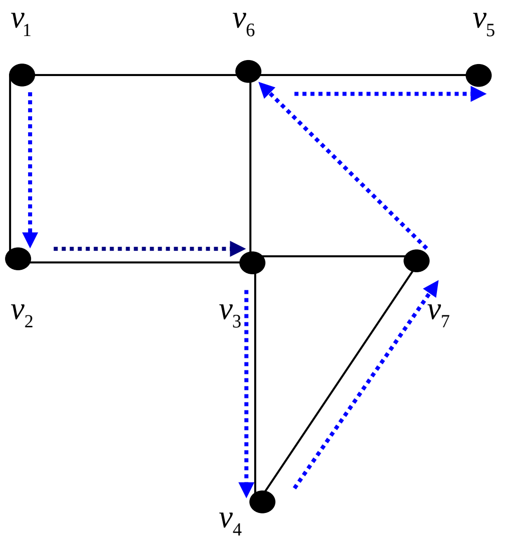
$Q=\{v_3,v_5\}$ $v_1\to v_2\to v_6\to v_3$

$Q=\{v_5,v_6,v_7\}$ $v_1\to v_2\to v_6\to v_3\to v_5$

$Q=\{v_6,v_7\}$ $v_1\to v_2\to v_6\to v_3\to v_5\to v_6$

$Q=\{v_7\}$ $v_1\to v_2\to v_6\to v_3\to v_5\to v_6\to v_7$

用 DFS 给出节点的访问顺序：



$$Q=\{v_1\} \qquad v_1$$

$$Q=\{v_6,v_2\} \qquad v_1\to v_2$$

$$Q=\{v_6,v_3\} \qquad v_1\to v_2\to v_3$$

$$Q=\{v_6,v_7,v_4\} \qquad v_1\to v_2\to v_3\to v_4$$

$$Q=\{v_6,v_7\} \qquad v_1\to v_2\to v_3\to v_4\to v_7$$

$$Q=\{v_6\} \qquad v_1\to v_2\to v_3\to v_4\to v_7\to v_6$$

$$Q=\{v_5\} \qquad v_1\to v_2\to v_3\to v_4\to v_7\to v_6\to v_5$$

图的搜索算法可以用于有向图：有向图也可以用它的邻接表 $A(v)$ 表示，其中 $A(v)=\{v'\in V:(v,v')\in A\}$。因此，搜索算法（BFS 和 DFS）可以不加改变地应用到有向图。

搜索算法的重要性不在于它是一个算法，而是**搜索算法是一类算法：**改变方框里的程序，就能得到同一个图的各种功能的算法。

例如：

begin

$Q=\{v_1\} \longrightarrow$ count: = 0

while $Q \neq \phi$ do

begin

$\forall v \in Q;$

标记 v $\longrightarrow$ count := count +1 及 order[v] := count

$Q=Q \setminus v$

for all $v^{'} \in A(v)$ do

if $v^{'}$ 没有标记，then $Q:=Q \bigcup \{v^{'}\}$

end

end

得到一个新算法，记录了被访问节点的顺序。

例如：**最大流的 Ford-Fulkerson 的标号算法，是搜索算法的更复杂的变种**。

有向图中寻求路的算法：

假定 $D=(V,A)$ 是一个有向图，$S,T \subseteq V$ 是节点的两个集合，分别称为发点集合和收点集合。

目标：在 D 中找一条自 S 中任一节点到 T 中任意一个节点的路。

对搜索算法进行修正：置 Q 的初始值为 S

Input: 有向图 $D=(V,A)$；V 的两个子集 S 和 T 。

Output: 如果 D 中自 S 到 T 有路，则输出自 S 中一节点到 T 中一个节点的路

```
begin
    for all v ∈ S do label[v]:=0. if v ∈ T then return(v); Q := S
while Q ≠ φ do
    begin
      ∀v ∈ Q;
        for all v' ∈ A(v) do
          if v' 没有标号，then
             begin
                label[v'] := v;
                if v' ∈ T then return Path(v')
                else Q := Q ∪ {v'}
             end
      Q = Q \ v
    end
    return "D 中不存在 S—T 路"
end
procedure Path(v') //得到从某一 s ∈ S 到节点 v 的节点序列
    if label[v] = 0 then return(v);
    else return path(label[v] )‖ v // path 是递归的，"‖"表示路的连接号。
```

注：在有向图中求路的这一方法，在解某些更复杂的组合优化问题中，有许多意想不到的应用。在随后，我们将对适合于一类增广路相继迭代求解的某些组合优化问题中，给出一些有效算法。在所有这些问题中，我们的算法都归结为在某个辅助有向图中，求自某发点集合到某收点集合的路。因此，图的搜索算法模型，统一了这些分散的算法。

二、标号算法存在的问题

网络 $N=(s,t,V,A,b)$

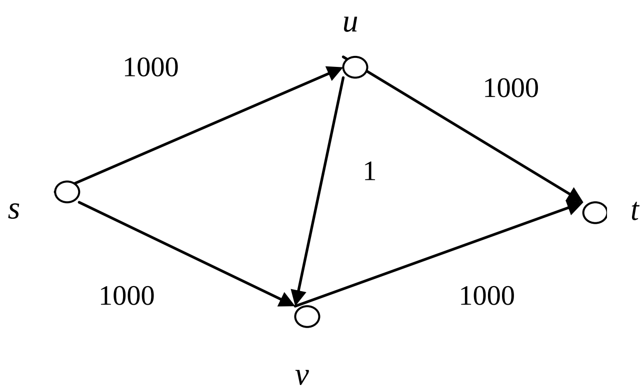
最大流的标号算法：每个阶段所需要的时间为 $O(|A|)$，

整个算法的复杂性为 $O(S\cdot|A|)$，其中 S 为流的增广次数。

所以，当网络的容量为整数时，$S\le|f^*|$（流的最大值）。

问题 1： S **能与** $|f^*|$ **相等吗？**

例如：



最大流值为 2000。从 0 流开始，应用标号算法：

第 1 次迭代，得到增广路 (s,u,v,t)，流增加 1，流值变为 1；

第 2 次迭代，得到增广路 (s,v,u,t)，流增加 1，流值变为 2；

第 3 次迭代，得到增广路 (s,u,v,t)，流增加 1，流值变为 3；

第 4 次迭代，得到增广路 (s,v,u,t)，流增加 1，流值变为 4；

⋮

⋮

第 2000 次迭代，得到增广路 (s,v,u,t)，流增加 1，流值变为 2000；

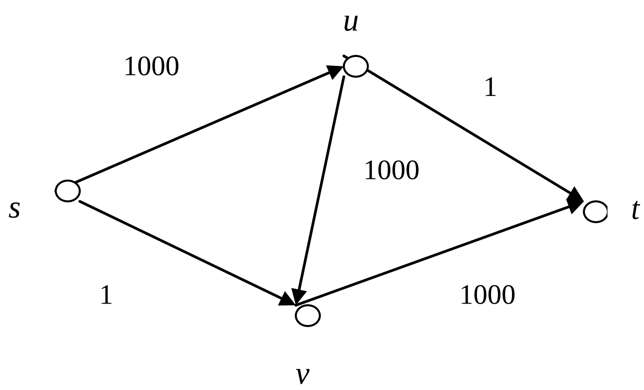
算法经过 2000 次迭代后终止。若将 1000 改为 M，则 Ford-Fulkerson 标号算法的迭代步数为 $2M$。所以，在最坏情况下，Ford-Fulkerson 标号算法所需要的迭代步数是指数的

结论：若每次增广路的选择都是不利的话，则 $S=|f^*|$ **是可能的**

改进措施：若用增广路 (s,v,t) 代替 (s,u,v,t)，则增广路的长度变短了，而且流的值增加更大，从而迭代步数减少。

即：在每次迭代时，找一条关于当前流的最短增广路（复杂性没有增加，可以用广探法进行）。

注意：并不是说沿最短的增广路的增加值，一定必沿长的增广路流的增加值增加值大。如：



显然，沿 (s,u,v,t) 流增加值>沿 (s,u,t) 或 (s,v,t) 流增加值。

事实上，可以证明，如果每次迭代都取当时最短的增广路，那么算法最多经过 $|V|\cdot|A|$ 次迭代可以结束。

问题 2：每次迭代，多次增广

在一个阶段结束时，标号算法要求抹去所有标号，然后进行下一阶段。

改进措施：**仅当能肯定已有的标号没有潜力可用时，我们才进行下一次迭代。**

改进的标号算法：

沿最短增广路增加流和每次迭代多次增广流——最大流的一个有效算法。

三、网络标号与有向图的搜索

为了实现上述标号算法的改进，并且要保证每一步迭代的复杂性没有本质的改变，我们从另一角度考察标号算法的过程。把标号算法用到网络 $N=(s,t,V,A,b)$

1. 当初始流为 0 时：在网络中找增广路，等于找有向图 (V,A) 中自 $S=\{s\}$ 到 $T=\{t\}$ 的路；

2． 以后各个阶段：关于流 f 的网络 N 中寻找增广路，等价于把求路的算法应用到网络 $N(f)$，其中 $N(f)$ 的定义如下：

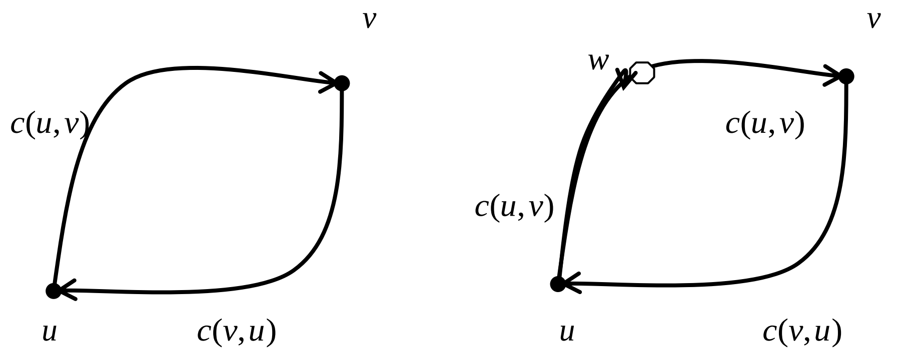
定义 6.1：给定网络 $N=(s,t,V,A,b)$ 和 N 上的一个可行流 f，定义一个新的网络 $N(f)=(s,t,V,A(f),ac)$，其中 $A(f)$ 的定义如下：

1） 若 $(u,v)\in A$ 且 $f(u,v)<b(u,v)$（非饱和弧），则 $(u,v)\in A(f)$ 且有 $ac(u,v)=b(u,v)-f(u,v)$

2） 若 $(u,v)\in A$ 且 $f(u,v)>0$，则 $(v,u)\in A(f)$ 且有 $ac(v,u)=f(u,v)$

称 $ac(u,v)$ 是弧 $(u,v)\in A(f)$ 增广容量。

注：$N(f)$ 中可能有多重弧，可以用 $(u,w),(w,v)$ 来代替 (u,v) 以避免多重弧：



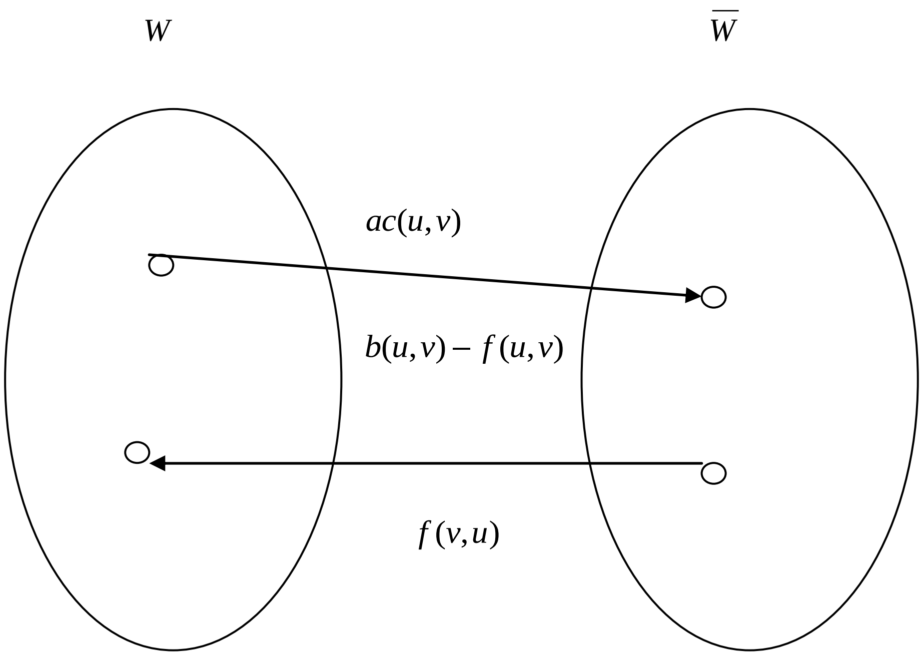
所以，我们可以假定 $N(f)$ 中没有多重弧。

网络 $N(f)$ 的性质：任取 $N(f)$ 一个 $s-t$ 截集 $(W,\overline{W})$，这个截集的容量等于 $N(f)$ 中自 W 到 $\overline{W}$ 的所有弧的增广容量之和。

这个截集中的每一条弧 (u,v)：

它或者是前向的：容量为 $ac(u,v)=b(u,v)-f(u,v)$

它或者是后向的：容量为 $ac(u,v)=f(v,u)$



$$(W,\overline{W})\text{ 在 } N(f) \text{ 中的截量 } = \sum_{\substack{(u,v)\text{前向} \\ \text{（在}N(f)\text{中）}}} ac(u,v)$$

$$= \sum_{\substack{(u,v)\text{前向} \\ \text{（在}N\text{中）}}} ac(u,v) + \sum_{\substack{(u,v)\text{后向} \\ \text{（在}N\text{中）}}} ac(u,v)$$

$$= \sum_{\substack{(u,v)\text{前向} \\ \text{（在}N\text{中）}}} b(u,v) - \sum_{\substack{(u,v)\text{前向} \\ \text{（在}N\text{中）}}} f(u,v) + \sum_{\substack{(u,v)\text{后向} \\ \text{（在}N\text{中）}}} f(v,u)$$

$$= (W,\overline{W})\text{ 在 } N \text{ 中的截量} - \left(\sum_{\substack{(u,v)\text{前向} \\ \text{（在}N\text{中）}}} f(u,v) - \sum_{\substack{(u,v)\text{后向} \\ \text{（在}N\text{中）}}} f(v,u) \right)$$

＝ $(W,\overline{W})$ **在** N **中的截量－** $|f|$

其中 $|f|$ 表示 f 的流值 $\sum_{(s,u)\in A} f(s,u) = \sum_{(v,t)\in A} f(v,t)$

所以，如果 N 中一个截的截量为 c，则它在 $N(f)$ 中的截量为 $c-|f|$；

如果 N 中最小截的截量为 $c*$，则它在 $N(f)$ 中的最小截量为 $c*-|f|$

根据最大流和最小截定理，在这两个网络中，都有最大流值＝最小截量

结论：如果 $|f*|$ 是网络 N 中最大流的值，则网络 $N(f)$ N 中最大流的值 $|f*|-|f|$

如果将标号算法应用到具有流 f 的网络 N 上，从 s 出发向外扩张标号，则必定通过前向弧后向弧。因为 $N(f)$ 恰好由这些前向弧后向弧组成，所以：

在网络 $N=(s,t,V,A,b)$ 中关于可行流 f 的标号过程，相当于从 s 出发对 $N(f)$ 执行搜索算法。

由于 N 中关于可行流 f 的最短增广路，对应于 $N(f)$ 中自 s 到 t 的最短路（弧数最少的路）。

所以，沿最短增广路增加流值，相当于沿 $N(f)$ 中自 s 到 t 的最短路增加流值。

求最短路的搜索算法－广探法，即按先标号先检查的方法搜索节点。

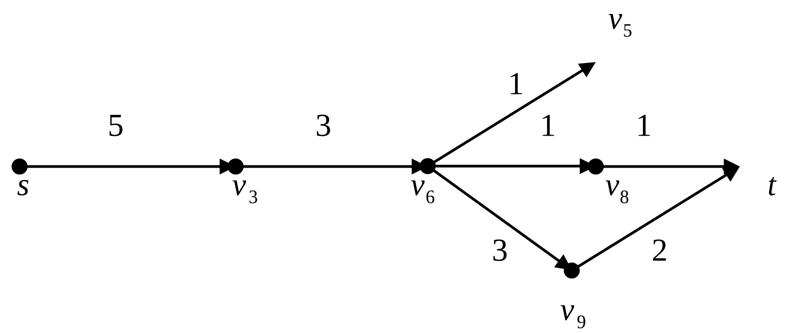
广探法将 $N(f)$ 中的节点，依 s 到它们的距离（即最短路的弧数），划分为许多互不相交的层数（第 0 层、第 1 层、第 2 层、等等）：

由于我们感兴趣的只是 $N(f)$ 中自 s 到 t 的最短路，因而我们能够对算法进一步改进和简化：

1） 最短的 $s-t$ 路不能通过层次和 t 的层次相同及比 t 的层次还高的节点，去掉这样的节点及其与之关联的弧，仍然能保持本阶段的一切有用的信息；
2） 任何一条最短的 $s-t$ 路，都是从 0 层节点（s）出发，然后经过第 1 层、第 2 层、等等；即，任何一条最短的 $s-t$ 路仅包含 j 层到 $j+1$ 层的弧，因此可以去掉从高层到低层的弧和同一层节点之间的弧，也能保持本阶段的一切有用的信息。

去掉 1）和 2）中的节点和弧后，就得到$N(f)$的一个子图，记为$AN(f)$，称之为网络N关于可行流f的辅助网络。

如下图，就是某一个网络N关于可行流f的辅助网络：



网络N关于可行流f的辅助网络有下述的特殊结构——分层网络。

定义 6.2: 一个分层网络$L=(s,t,U,A,b)$是具有顶点集合为U的网络，其中U是互不相交的集合$U_0,U_1,\cdots,U_d$的并，使得

$$U_0=\{s\},U_d=\{t\} \text{ 并且 } A\subseteq\bigcup_{j=1}^{d}(U_{j-1},U_j)$$

显然，$AN(f)$是一个分层网络。

注：

（1）$AN(f)$很容易构造：当对$N(f)$执行广探法时，只要把这样一些弧保留下来即可，这些弧是引导一个新的节点得到标号且其层次低于t所在的层次。因此，构造辅助网络的计算复杂度为$O(|A(f)|)=)O(|A|)$。

（2）应用辅助网络，很容易找出关于当前可行流的最短增广路。进而，实现改进的标号算法的第二阶段——在同一次迭代中实现流的最多次增广。

定义 6.3： 设$N=(s,t,V,A,b)$是一个分层网络，N中关于某一流g的一条增广路，如果它不包含后向弧，则称之为**前向增广路**。如果N不存在关于流g的前向增广路，则称g为极大流（当然不一定是最大流）。

设计最大流多项式算法的核心：

在分层网络$AN(f)$中求极大流$\Leftrightarrow$在$AN(f)$中求尽可能多的最短路。