典型优化问题的模型与算法

(Models and Algorithms for Typical Optimization Problems)

主讲人：朱晗
东北财经大学 管理科学与工程学院
Email：hanzhu@dufe.edu.cn

感谢东北大学系统工程研究所课程组

第 11 章 最大流与最小费用流问题与算法

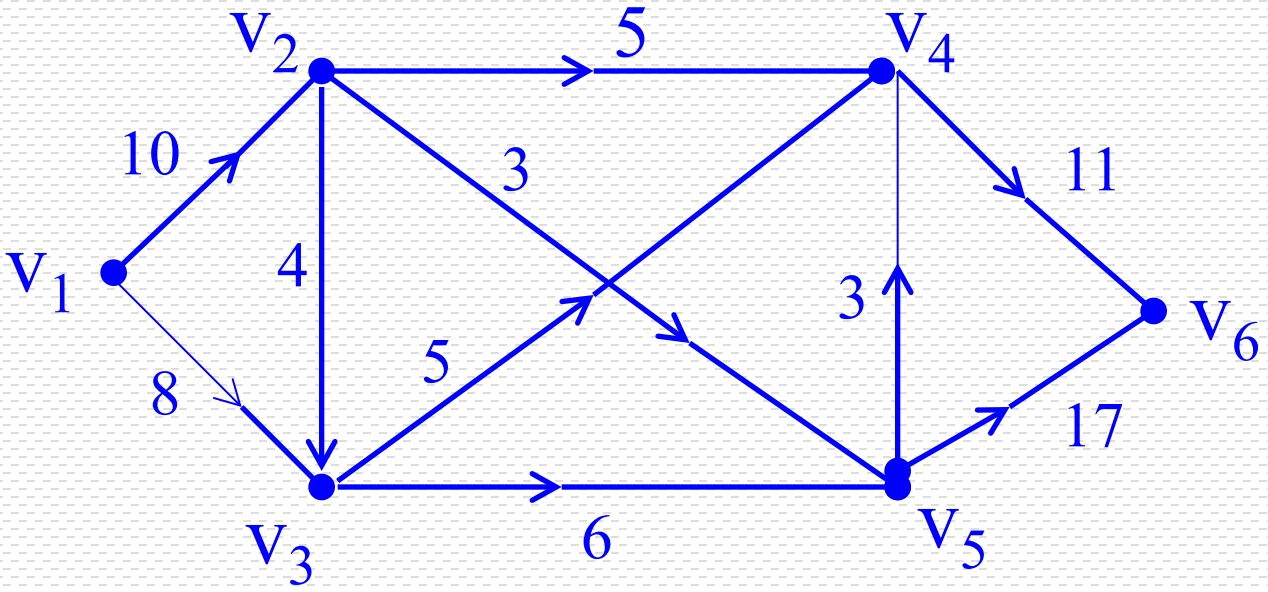
- 网络与网络流
- 什么是最大流问题
- 最大流问题的数学描述
- 求解最大流问题的算法
- 最小费用流问题的特征与变种
- 求解最小费用最大流的算法

网络与网络流

- 现实中的实例： 管道网络、通信网络、交通网络、电力网络等，这里的流量，如车流，人流、物流、信息流、现金流。水流、电流。。。。
- 特征：可以描述为一个网络图，边， 顶点。。。
- 网络中每条边的最大通过能力（容量）是有限的，实际流量不超过容量。
- 最大流问题(maximum flow problem)，组合最优化问题，即如何充分利用装置的能力，使得运输的流量最大，以取得最好的效果。

网络与网络流

连接某产品产地v_1和销地v_6的交通网如下：



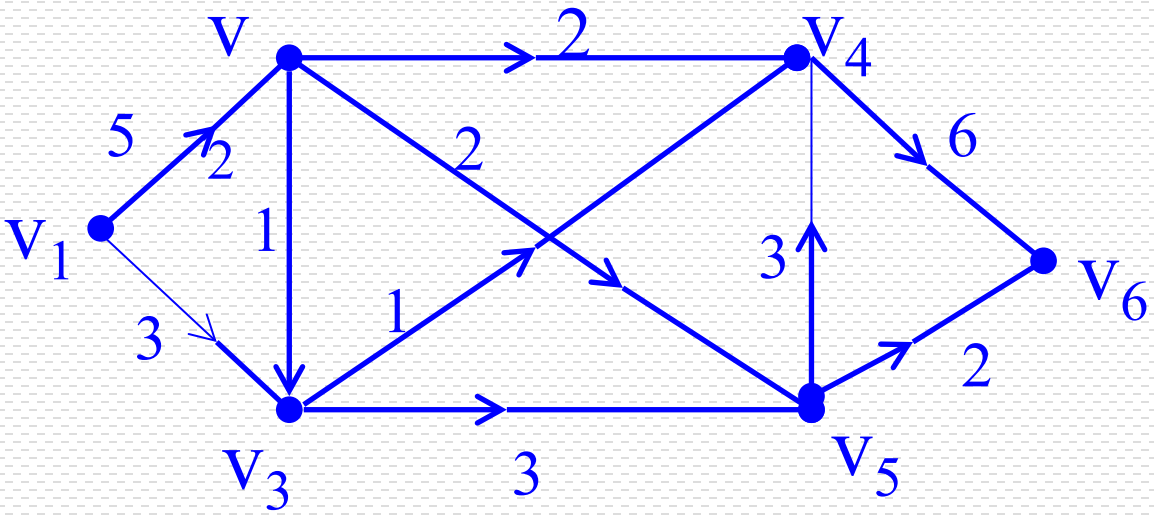
（a）

弧（v_i，v_j）：从v_i到v_j的运输线，

弧旁数字：这条运输线的最大通过能力—容量。

网络与网络流

制定一个运输方案，使从v_1运到v_6的产品数量最多。



（b）

弧旁数字：运输数量—流量。

问题：这个运输网络中，从v_1到v_6的最大输送量是多少？

网络与网络流

基本概念

1、网络与流

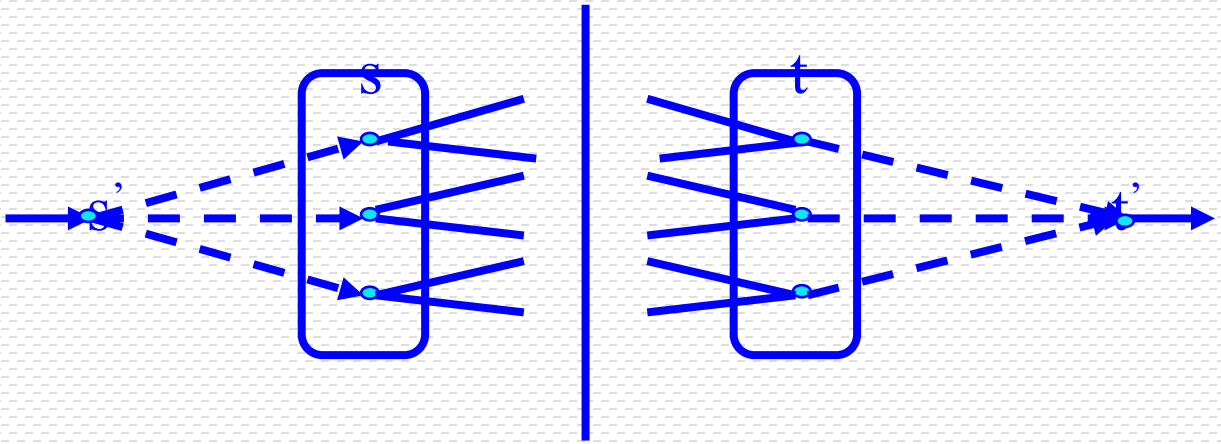
设一个赋权有向图$D=(V,A)$，
在V中指定一个**发点**v_s和一个**收点**v_t，
其它的点叫做**中间点**。
对于D中的每一个弧$(v_i, v_j) \in A$，
都有一个非负数c_{ij}，叫做**弧的容量**。
我们把这样的图D叫做一个**容量网络**，
简称**网络**，记做$D=(V, A, C)$。

弧的容量：对网络上的每条弧(v_i,v_j)都给出一个最大的通过能力，记为$c(v_i,v_j)$或简写为c_{ij}。

网络与网络流

基本概念

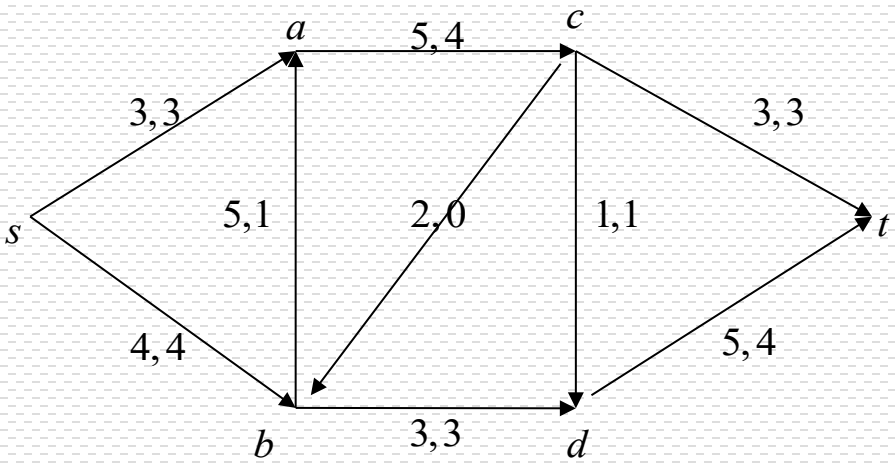
对有多个发点和多个收点的网络，
可以另外虚设一个总发点和一个总收点，
并将其分别与各发点、收点连起来（见图），
就可以转换为只含一个发点和一个收点的网络。

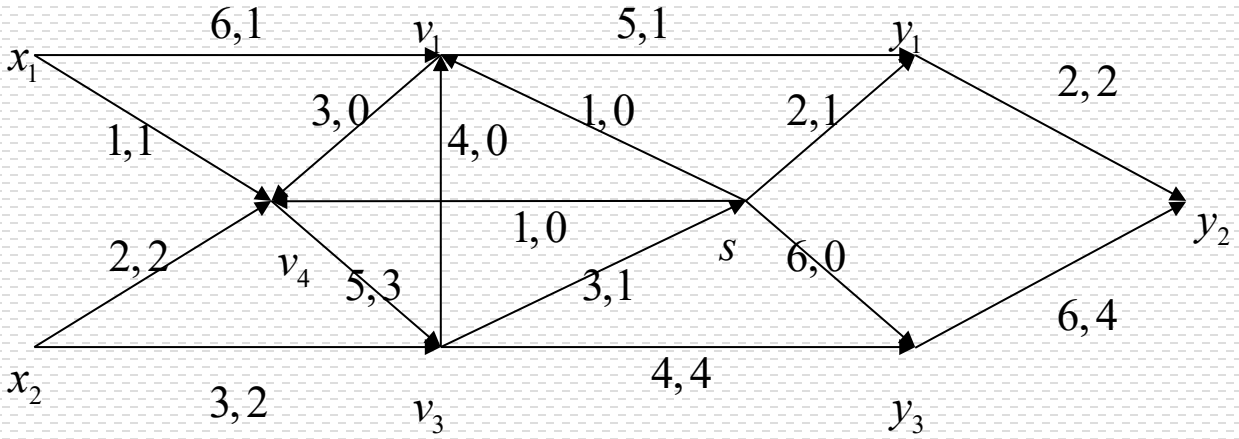


网络与网络流

基本概念

单源单汇网络和多元多汇网络





网络与网络流

定义 2 设 N 为一个网络，f 是 E 上的非负函数，如果：

（1）容量限制条件：$0 \le f(e) \le c(e)$，$\forall e \in E$；

（2）流量守恒条件：$\sum_{e \in N^+(v)} f(e) = \sum_{e \in N^-(v)} f(e), \quad v \in I,$

其中：$N^+(v)$ 表示 v 的所有出弧的集，$N^-(v)$ 表示 v 的所有入弧的集。则称 f 是网络 N 的一个**流**，$f(e)$ 是边 e 的**流量**。

注 1：（1）容量约束表示通过边的流量不能超过该边的容量；守恒条件表示在每个中间点，流进与流出该点的总流量相等，即保持中间点的流量平衡。

（2）任一网络至少存在一个流，如零流 $(f(e) = 0, e \in V)$。

网络与网络流

定义 3 设 f 是网络 N 的一个流，$A \subseteq V$，则称 $\sum_{e \in N^+(A)} f(e) - \sum_{e \in N^-(A)} f(e)$ 为**流出 A 的净流量**，称 $\sum_{e \in N^-(A)} f(e) - \sum_{e \in N^+(A)} f(e)$ 为**流入 A 的净流量**。

注 2:（1）流入、流出任何中间点的净流量为 0；

（2）流出发点集 X 的净流量等于流入收点集 Y 的净流量。

定义 4 设 f 是网络 N 的一个流，则 f 的**流的价值** $\mathrm{Val}f$ 定义为

$$\mathrm{Val}f = \sum_{e \in N^+(X)} f(e) = \sum_{e \in N^-(Y)} f(e)$$

即流的价值是发点集的流出量，也是收点集的流入量。

注 3：任何一个多源多汇网络 $N=(V,E,c,X,Y)$ 都等价与一个单源单汇网络 $N'=(V',E',c',X',Y')$。在解决实际问题时，常把多源多汇网络转化为单源单汇网络。

（1）$V'=V\cup\{s,t\}$，s,t 分别是 N' 的发点与收点；

（2）$E'=E\cup\{(s,x)\,|\,x\in X\}\cup\{(y,t)\,|\,y\in Y\}$；

（3）$c'=c(e),e\in E$；$c'(s,x)=\infty,x\in X$，$c'(y,t)=\infty,y\in Y$。

图 1 所示网络等价于图 2 所示的单源单汇网络。

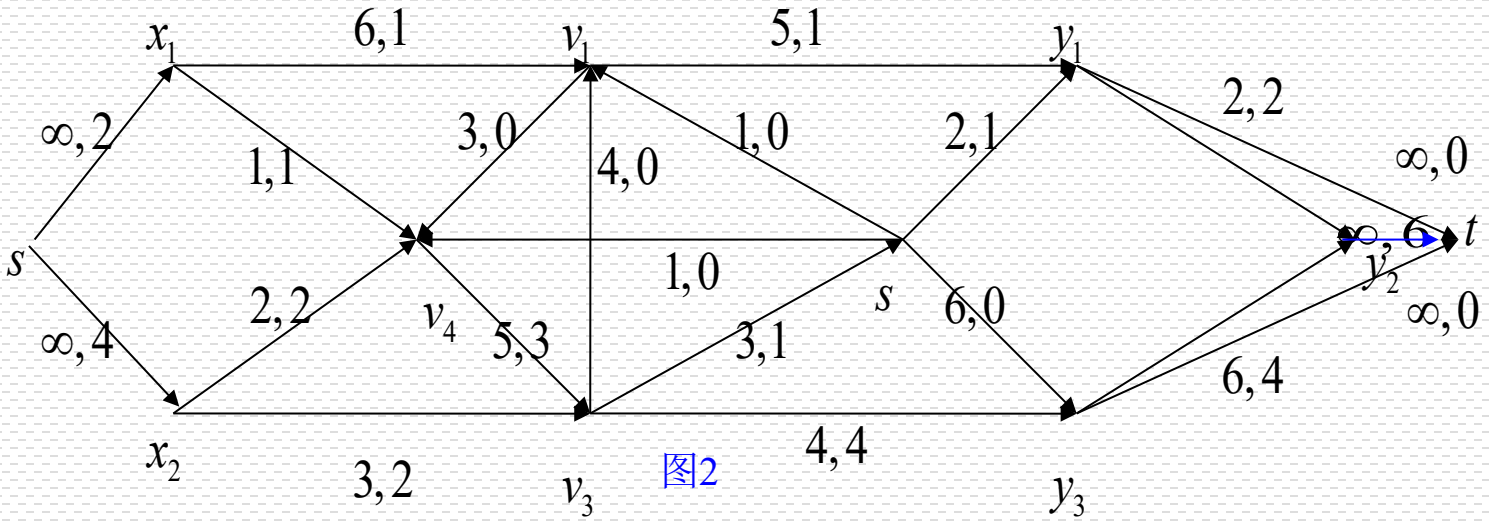


图2

网络与网络流

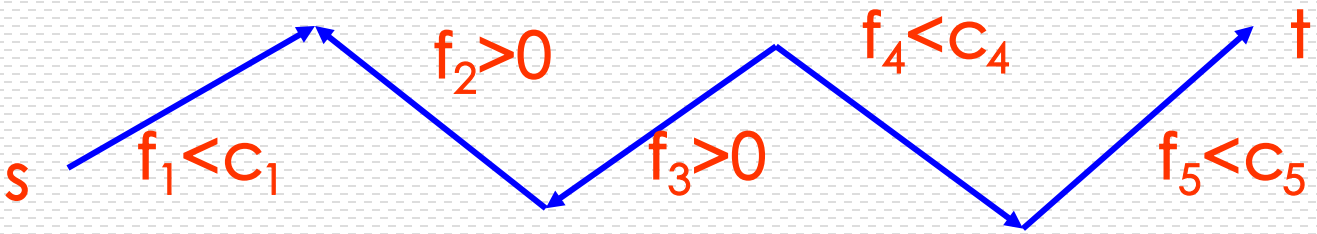
增广链：f是一个可行流，如果满足：

$$\begin{cases} 0 \le f_{ij} < c_{ij} & (v_i, v_j) \in \mu^+ \\ 0 < f_{ij} \le c_{ij} & (v_i, v_j) \in \mu^- \end{cases}$$

即 μ^+ 中的每一条弧都是非饱和弧

即 μ^- 中的每一条弧都是非零流弧

则称 μ 为从 v_s 到 v_t 的关于 f 的一条增广链。



最大流问题

1.什么是最大流问题

最大流问题(maximum flow problem)是一种**组合优化**问题，即讨论如何充分利用装置的能力，使得**运输的流量最大**以取得最好的效果的问题。

在具体描述**最大流问题**前，我们先介绍几个**网络流问题**中常见的定义：

G = (V,E)　　表示整个图.

V　　表示整个图中的所有结点的集合.

E　　表示整个图中所有边的集合.

s　　表示网络的源点

t　　表示网络的汇点.

对于每条边(u,v),　　有一个容量c(u,v)　(c(u,v)>=0)

（如果c(u,v)=0，则表示(u,v)不存在在网络中。相反，如果原网络中不存在边(u,v)，则令c(u,v)=0.）

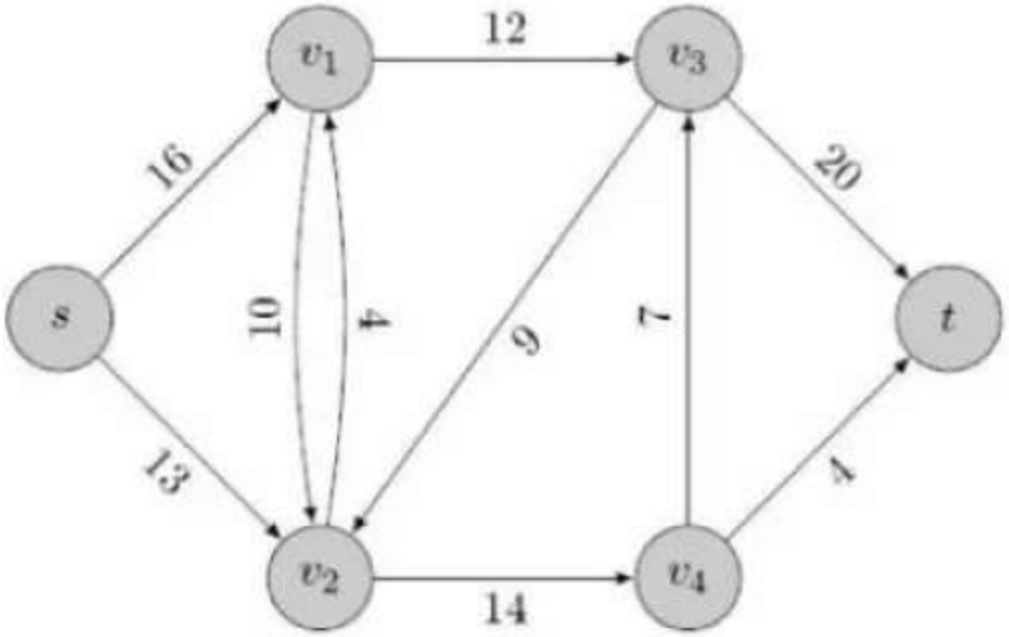
对于每条边(u,v)　　有一个流量f(u,v).

最大流问题的描述：图模型

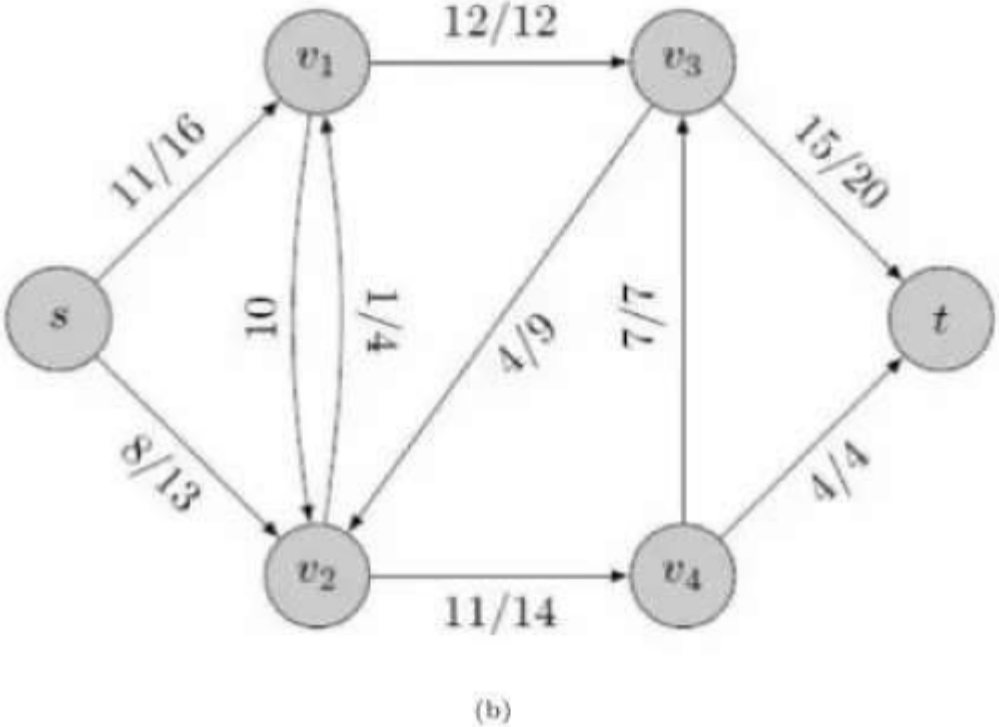
0.1 最大流问题

如图1所示，假设需要把一些物品从结点s（称为源点）运送到结点t（称为汇点），可以从其他结点中转。图1(a)中各条有向边的权表示最多能有多少个物品从这条边的起点直接运送到终点。例如，最多可以有9个单位的物品从结点v_3运送到v_2。

图1(b)展示了一种可能的方案，其中每条边中的第一个数字表示实际运送的物品数目，而第二个数字就是题目中的上限。



最大流问题的描述：线性规划模型



(b)

图 1: 物资运送问题

最大流问题的描述：线性规划模型

求解从源点可到达汇点的流量(也就是图1(b)的第一个数字)最大的问题称为最大流问题（Maximum-Flow Problem)。对于一条边(u,v)，它的物品上限称为容量（capacity)，记为$c(u,v)$（对于不存在的边(u,v)，$c(u,v)=0$)；实际运送的物品称为流量（flow)，记为$f(u,v)$。注意，“把3个物品从u运送到v，又把5个物品从v运送到u”没什么意义，因为它等价于把两个物品从v运送到u。这样，就可以规定$f(u,v)$和$f(v,u)$最多只有一个正数（可以均为0)，并且$f(u,v)=-f(v,u)$。这样规定就好比“把3个物品从u运送到v”等价于“把-3个物品从v运送到u”一样。

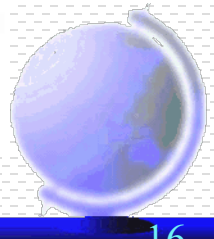
在最大流问题中，容量c和流量f满足

容量限制 即$f(u,v)\le c(u,v)$.

斜对称性 $f(u,v)=-f(v,u)$.

流量平衡 对于除了结点s和t外的任意结点u, $\sum_{(u,v)\in E} f(u,v)=0$.

目标 最大化$|f|=\sum_{(s,v)\in E} f(s,v)=\sum_{(u,t)\in E} f(u,t)$，即从$s$点流出的净流量（它也等于流入$t$点的净流量，其中$E$为边集)。



最大流问题的描述：线性规划模型

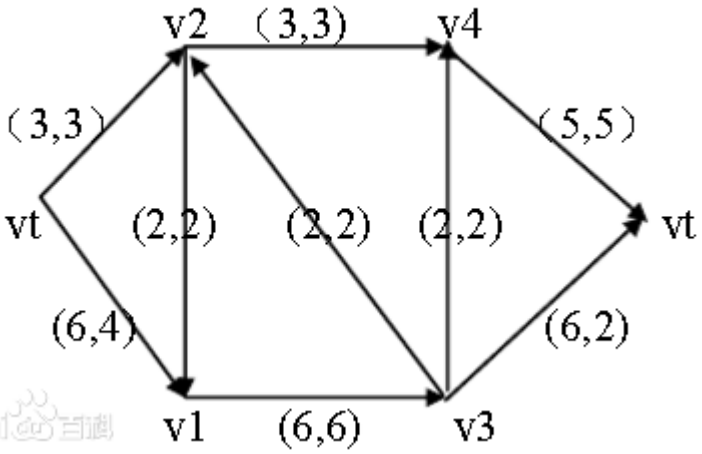
● 可行流

称满足下列两个条件的流为可行流：

1. 容量限制条件：对G中的每条边(v_i, v_j)，有$0 \leqslant f_{ij} \leqslant c_{ij}$；即每条边上的流量非负而且最大也只能达到容量的限制。

2. 平衡条件：对中间点v_i，物资的输入量和输出量相等。对发、收点v_s, v_t，，f_{ij}为网络流的总流量。

一个流$f=\{f_{ij}\}$，当$f_{ij}=c_{ij}$，则称流f对边(vi,vj)是饱和的，否则称f对(v_i, v_j)不饱和。



$$\min V = f^*$$

$$s.t\begin{cases} \sum f_{ij} - f_{ji} = 0 & (i \neq s,t) \\ \sum f_{sj} - f_{js} = v(f) & (i = s) \\ \sum f_{ij} - f_{ji} = -v(f) & (i = t) \end{cases}$$

最大流问题的性质

● 割集与最小割定理

容量网路G=(V,E,C),v_s,v_t为发、收点，若有边集E'为E的子集，将G为两个子图G_1,G_2，即点集V被剖分其为两个顶点集合分别记$S,\overline{S}$，必有$S\cup\overline{S}=V,S\cap\overline{S}=\varnothing,v_s\in S,v_t\in\overline{S}$。若有边集E'为E的子集，满足下列两个的性质，则称E'为G的割集（也称截集），记$E'=(S,\overline{S})$。

1.若把整个截集的弧从网络G=(V,E,C)中丢去，则不存在从v_s和vt的有向路，即图(V,E-E')不连通。

2.只要没把整个截集删去，就存在从v_s和vt的有向路，即当E''为E的真子集，图G(V,E-E'')仍连通。

由此可知，截集是从起点v_s到终点v_t的必经之路。[1]

割集$(S,\overline{S})$中所有始点在S，终点在$\overline{S}$的边的容量之和，称为$(S,\overline{S})$的割集容量，记为$C(S,\overline{S})$。容量网络G的割集有很多个，其中割集容量最小这成为网络G的最小割集容量（简称最小割）。

最大流问题的性质

- 割集与最小割定理

定理一

设f为网络G=(V,E,C)的任一可行流，流量为v(f)，$(S,\overline{S})$是分离v_s,v_t的任一割集，则有 $f(v)\le C(S,\bar{S})$。

定理二

由定理一可知，最大流的流量v(f)和某一割集K的容量相等，而且最大流的流量本身也不带任一割集的容量，因此割集一定是最小的割集。[2]

任一网络G中，从v_s到v_t的最大流的流量等于分离v_s、v_t的最小割的容量（最小的割集的容量）。

最大流问题的性质

最大流与最小割的关系：

定理 1 设 f 是 N 的流，$(A,\overline{A})$ 是一个割，则：

（1）$\mathrm{Val}\, f = \sum_{e\in N^+(A)} f(e) - \sum_{e\in N^-(A)} f(e)$

（2）$\mathrm{Val}\, f \le C(A,\overline{A})$。

式（1）表明运输网络的一个自源 s 到汇 t 的流值，等于任何分离 s 和 t 的割中流的净值，即割的自 A 到 $\overline{A}$ 的弧中的流减去自 $\overline{A}$ 到 A 的流的总体。

定理 2（最大流最小割定理）

（1）设 f 是流，K 是割，若 $\mathrm{Val}\, f = C(K)$，则 f 是最大流，K 是最小割。

（2）网络 N 的最大流的价值等于最小割的容量。

上述定理是图论的重要核心，关于图的许多结果，在适当的选择网络后，应用这个定理往往能够轻易地获得解决。从这个定理的证明中，可以引出求网络最大流的一个算法。但这种方法实际做起来有困难，因为没解决如何寻找增广链的方法。

最大流问题的性质

增广链及最大流算法

定义 7　若 f 是网络 N 的一个流，对 $e \in E$，

（1）若 $f(e)=c(e)$，则称 e 为 f 的**饱和弧**；

（2）若 $f(e)<c(e)$，则称 e 为 f 的**不饱和弧**；

（3）若 $f(e)>0$，则称 e 为 f 的**正弧**；

（4）若 $f(e)=0$，则称 e 为 f 的**零弧**；

定义 8　若 P 是网络 N 中从源 s 到汇 t 的一条**初等链**（点、边不重复的有向路），定义链的方向为从 s 到 t，则链上的弧（有向边）分为两类：

正向弧：弧的方向与链的方向一致，正向弧的全体记作 P^+；

反向弧：弧的方向与链的方向相反，反向弧的全体记作 P^-。

最大流问题的性质

- 可增广链
  - f是一个可行流，f_{ij}表示由i点指向j点的流量；如果满足前向弧的流量非负且小于容量，或后向弧的流量大于0且不超过容量，则称μ为从v_s到v_t的关于f的可增广链

$$\begin{cases} 0 \le f_{ij} < c_{ij} & (v_i, v_j) \in \mu^+ \\ c_{ij} \ge f_{ij} > 0 & (v_i, v_j) \in u^- \end{cases}$$

  - 可增广链的实际意义是：沿着这条从v_s到v_t输送的流，仍有潜力可挖，只要前向弧的流量增加或后向弧的流量减少，就可以将截集的流量提高。调整后向流，在各点仍满足平衡条件及容量限制，仍为可行流。
  - 从另一个角度来说，可以提高流量的可行流也不是最大流；
  - 可行流f是最大流的充要条件是：不存在从v_s到v_t的可增广链。

求解最大流问题的算法

2.求解最大流问题的算法

0.2 求解最大流的算法

求解最大流算法多种多样，例如Dinic，Edmonds-Karp，HLPP，SAP，ISAP……但是总体来说，求解最大流有两种思路，一种是增广路算法，一种是预流推进算法。在这里，我们着重讲解增广路算法，因为预流推进算法较之复杂，虽然从时间复杂度上面或许有一些优势，但是需要一些数据结构（优先队列）的基础，**并且在运筹学的教科书上通常是介绍增广路算法**，因此有兴趣的读者可以查阅相关资料了解预流推进算法，在此便不做赘述。在这里我们着重介绍两种经典的增广路算法（Ford-Fulkerson Method），分别是Edmonds-Karp算法和Dinic算法。

有时称为标号算法

求解最大流问题的算法-标号算法

算法思路：第一步是标号过程，通过标号来寻找可增广链；第二步是调整过程，沿可增广连调整f以增加流量

1.标号过程

（1）给发点（始点）以标号$(\triangle,+\infty)$或$(0,+\infty)$。

（2）选择一个已标号的顶点v_i，对于v_i的所有未给标号的邻接点v_j按下列规则处理：

(a)若后向边$(v_j,v_i)\in E$，且$f_{ji}>0$时，则令$\delta_j=\min(f_{ji},\delta_i)$，并给$v_j$以标号$(-v_i,\delta_j)$，这表明$v_j$点的$v_i$点的边最多可以减少$\delta_j$的流量以提高整个网络的流量。

(b)若前向边$(v_i,v_j)\in E$，且$f_{ij}<c_{ij}$时，令$\delta_j=\min(c_{ij}-f_{ij},\delta_i)$，并给$v_j$以标号$(+v_i,\delta_j)$，这表明$v_i$点到$v_j$点的边最多可以增加$\delta_j$的流量以提高整个网络的流量。

括弧内的第一个数字表示这个节点得到的得到标号前的第一个结点的代号，第二个数字表示从上个标号节点到这个标号节点允许的最大调整量δ，假定发点的调整量不限，所以标记为+∞。

（3）重复（2）直到收点v_t被标号或不再有顶点可被标号为止。

若v_t没有得到标号，说明标号过程已无法进行，可行流f已是最大流。若v_t得到标号，说明存在一条可增广链，转入调整过程。标号若有多条增广链时，不用刻意考虑哪种调整更适合，只需一条一条地转入调整过程，不用全盘考虑。

求解最大流问题的算法-标号算法

2.调整过程

（1）令这条被找到的增广链中所有的前向弧全部加上δ_j的流量，所有的后向弧全部减去δ_j的流量，至于不在增广链之中的边的流量则不需要调整。

$$f'_{ij}=\begin{cases} f_{ij}+\delta_{ij} & (v_i,v_j)\in \mu^+ \\ f_{ij}-\delta_{ij} & (v_i,v_j)\in u^- \\ f_{ij} & (v_i,v_j)\notin u^- \end{cases}$$

（2）去掉所有标号，回到第1步，对可行流f'重新标号。

求解最大流问题的算法

3.两种增广路算法

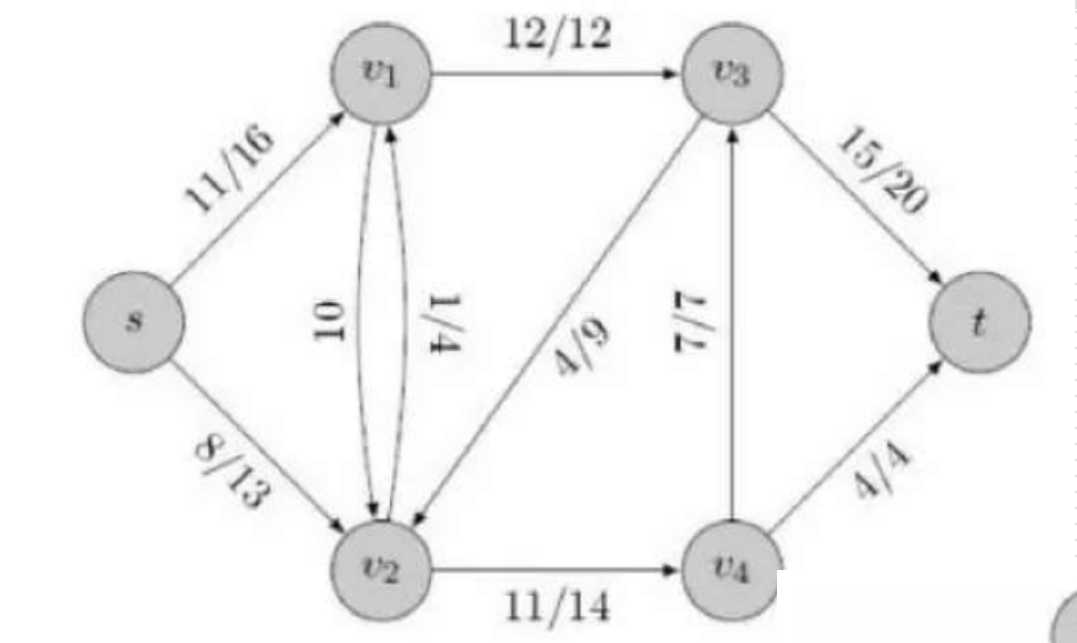
—增广路算法—

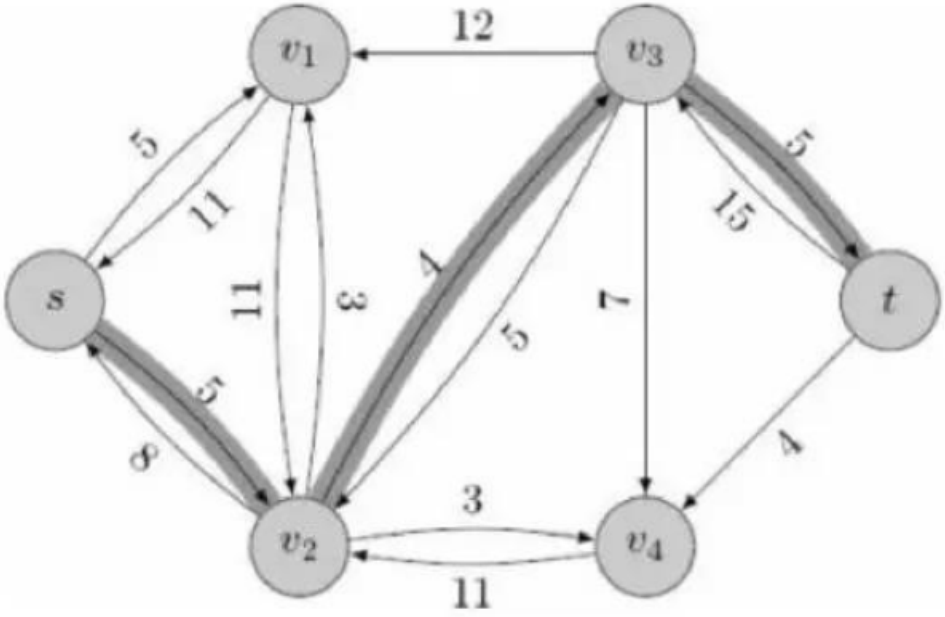
0.3 增广路算法

增广路算法思想很简单，从零流（所有边的流量均为0）开始不断增加流量，保持每次增加流量后都满足容量限制、斜对称性和流量平衡3个条件。

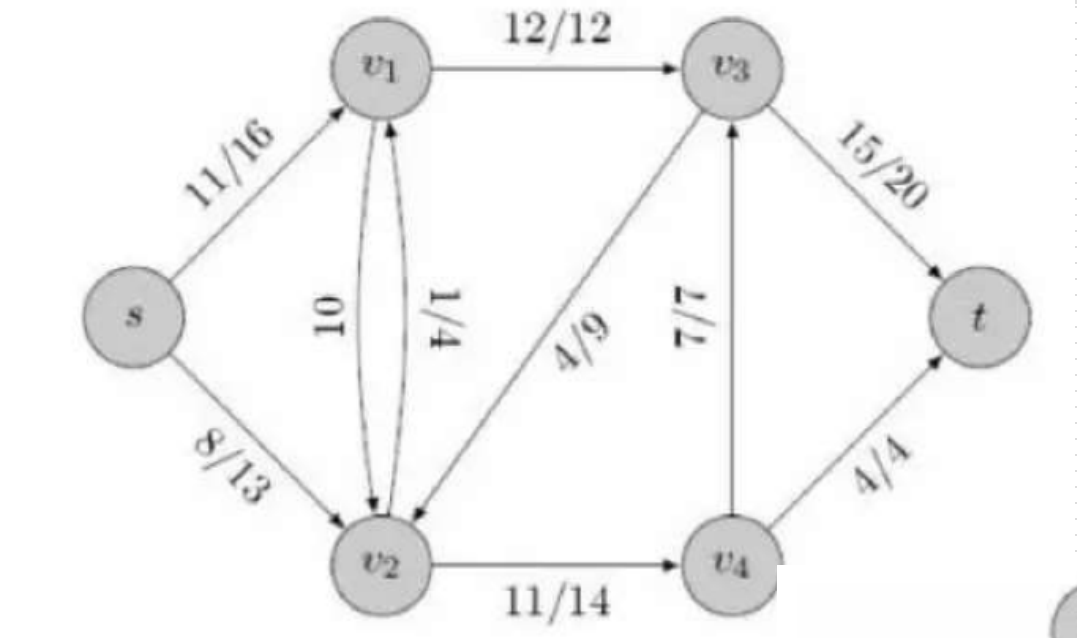
计算出图2(a)中的每条边上容量与流量之差（称为残余容量或者残余流量，简称残量），得到图2(b)中的残量网络（residual network，即Dinic章节中介绍的残留网络）。同理，由图2(c)可以得到图2(d)。注意残量网络中的边数可能达到原图中边数的两倍，如原图中边(s, v_1)，$c(s, v_1)$=16，$f(s, v_1)$=11的边在残量网络中对应正反两条边，残量分别为16-11=5和0-(-11)=11。该算法基于这样一个事实：残量网络中任何一条从s到t的有向道路都对应一条原图中的增广路（augmenting path）——只要求出该道路中所有残量的最小值d，把对应的所有边上的流量增加d即可，这个过程称为增广（augmenting）。不难验证，如果增广前的流量满足3个条件，增广后仍然满足。显然，只要残量网络中存在增广路，流量就可以增大。可以证明它的逆命题也成立：如果残量网络中不存在增广路，则当前流就是最大流。这就是著名的增广路定理。

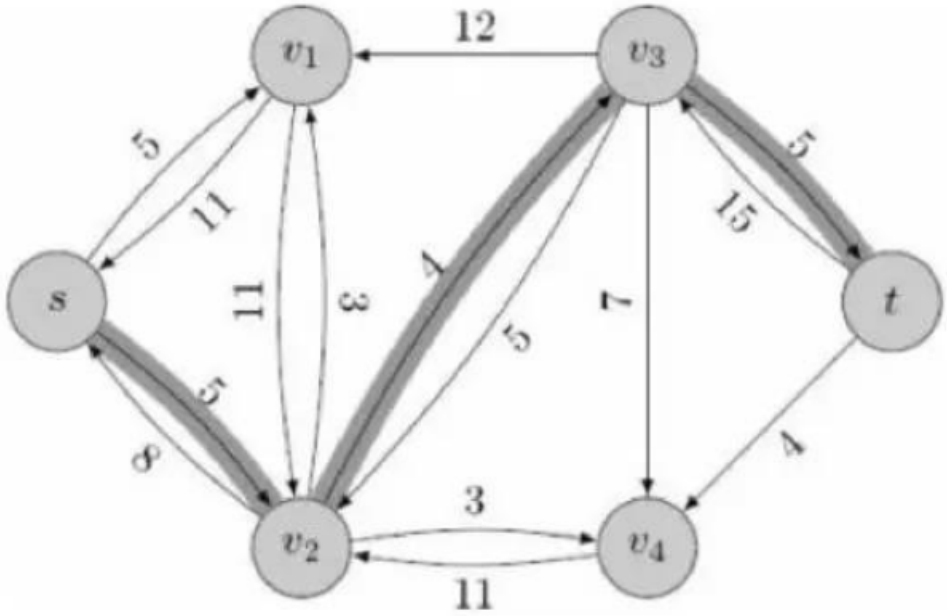
求解最大流问题的算法-增广路算法



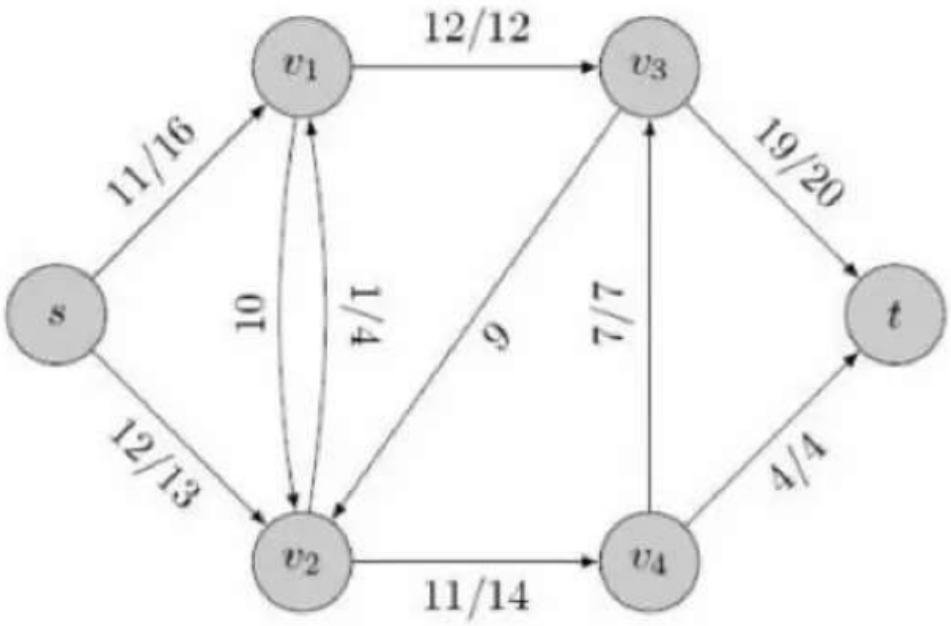


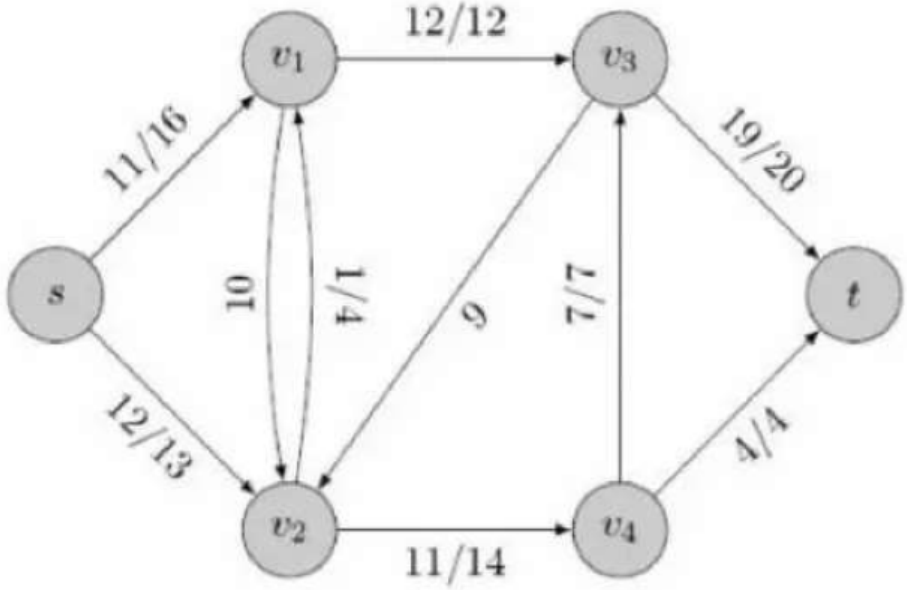
求解最大流问题的算法-增广路算法





求解最大流问题的算法-增广路算法





求解最大流问题的算法-增广路算法

0.3.1 Edmonds-Karp算法

正如上一个章节中提到，我们需要去找一条任意的增广路，“找任意路径”最简单的办法无疑是用深度优先搜索（Deep First Search，DFS），但很容易找出让它很慢的例子。一个稍微好一些的方法是使用广度优先搜索（Breadth First Search，BFS），它足以应对数据不刁钻的网络流题目。这就是Edmonds-Karp(EK)算法。简单来说，EK算法的原理就是上述的增广路算法（Ford–Fulkerson Method），准确来说Ford–Fulkerson不是一种算法，而是一种方法，而EK则是一种属于Ford-Fulkerson方法的算法。而相应的EK的算法步骤也就很明了了，如下所示：

1.初始化数据

2.BFS找到一条增广路

3.找到这条增广路的限制边（也就是残余流量最小的边），记录其残余流量Min，增广路所有正向边残余流量减去Min, 同时反向边残余流量加上Min

4.重复步骤2，直到没有增广路存在

EK算法的时间效率是$O(m^2n)$，其中m是边的数量，n是点的数量，但是大多数情况下，EK算法的表现要优于$O(m^2n)$。

两种求解算法-DINIC算法

0.3.2 Dinic算法

Dinic算法的思想也是分阶段地在层次网络中增广。它与EK算法不同之处是：EK每个阶段执行完一次BFS增广后，要重新启动BFS从源点V_s开始寻找另一条增广路;而在Dinic算法中，只需一次DFS过程就可以实现多次增广，这是Dinic算法的巧妙之处。在介绍Dinic算法之前，首先详细的介绍一些概念：

残留网络 ：设有容量网络$G(V,E)$及其上的网络流f，G关于f的残留网络即为$G'(V',E')$，其中G'的顶点集V'和G的顶点集V相同，即$V'=V$，对于G中任何一条弧(u,v),如果$f(u,v)<c(u,v)$，那么在G'中有一条弧$(u,v)\in E'$，其容量为$c'(u,v)=c(u,v)-f(u,v)$，如果$f(u,v)>0$,则在G'中有一条弧$<v,u>\in E'$，其容量为$c'(v,u)=f(u,v)$.

从残留网络的定义来看，原容量网络中的每条弧在残留网络中都化为一条或者两条弧。在残留网络中，从源点到汇点的任意一条简单路径都对应一条**增广路**，路径上每条弧容量的**最小值**即为能够一次增广的最大流量。

两种求解算法-DINIC算法(续)

顶点的层次 ：在残留网络中，把从源点到顶点u的最短路径长度，称为顶点u的层次。源点V_s的层次为0.例如图3就是一个分层的过程。 注意：

（1）对残留网络进行分层后，弧可能有3种可能的情况。

1、从第i层顶点指向第$i+1$层顶点。

2、从第i层顶点指向第i层顶点。

3、从第i层顶点指向第j层顶点（$j<i$）。

（2）不存在从第i层顶点指向第$i+k$层顶点的弧($k \geq 2$)。

（3）并非所有的网络都能分层。如下图：

有了上述概念做支撑，我们来介绍一下Dinic算法的具体步骤：

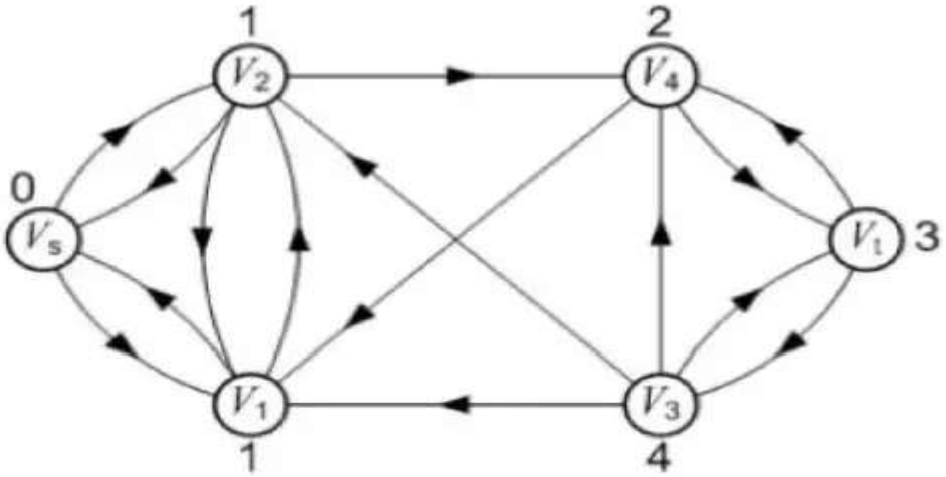
1. 用BFS建立分层图注意:分层图是以当前残留网络为基础建立的,所以要重复建立分层图

2. 用DFS的方法寻找一条由源点到汇点的路径,获得这条路径的流量I（路径上残量最小的边的残量）根据这条路径修改整个图,将所经之处正向边流量减少I,反向边流量增加I,注意I是非负数

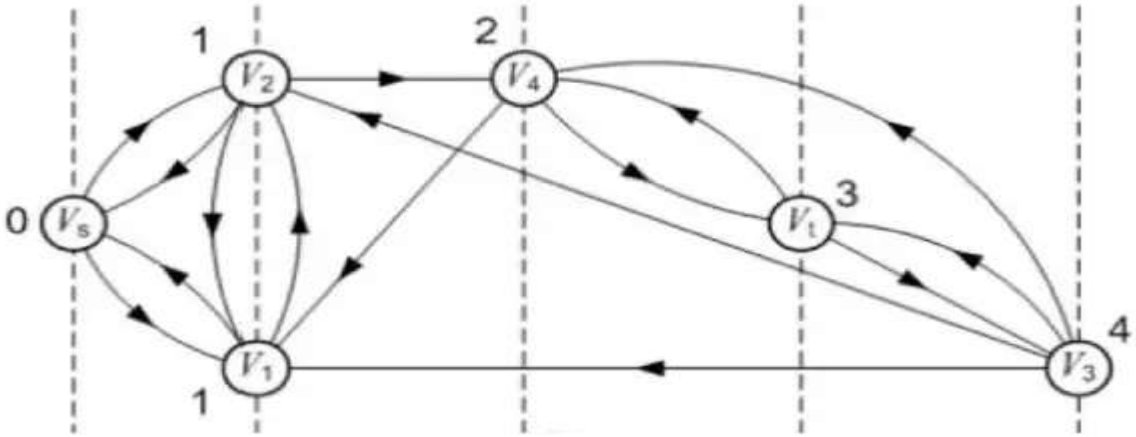
3. 重复步骤2,直到DFS找不到新的路径时,重复步骤1，直到图无法分层

Dinic算法的时间复杂度为$O(n^2m)$, 其中n为点数，m为边数，可以看出，通常来说$n<m$,所以Dinic算法的时间复杂度是优于EK算法的。而且在实际的表现中，Dinic通常会有更好的表现。

两种求解算法-DINIC算法(续)



(a)



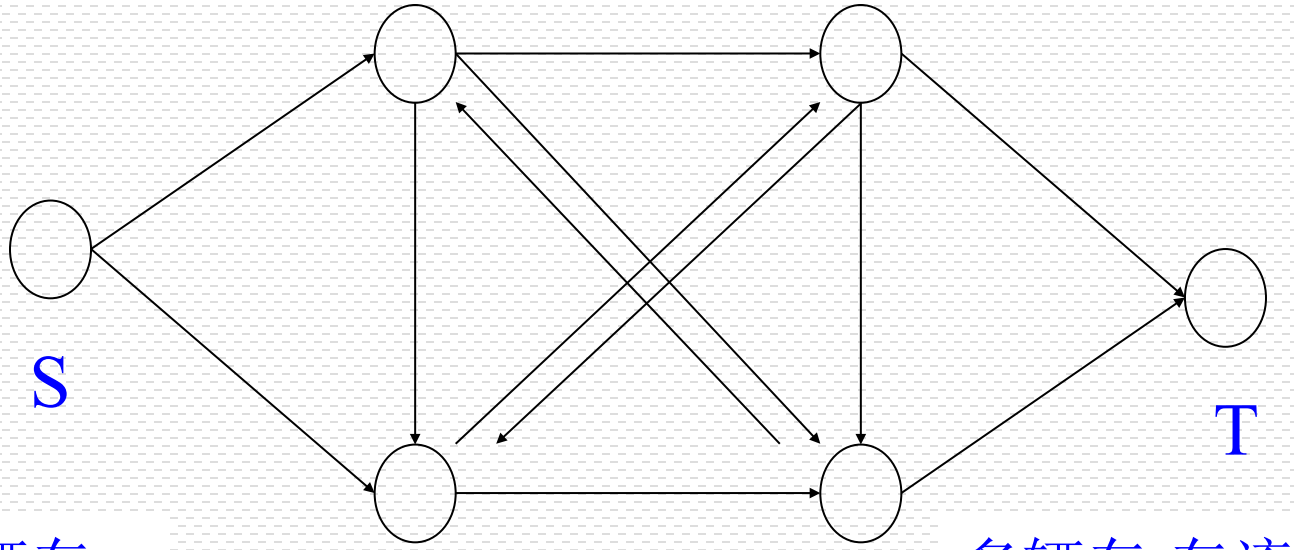
(b)

图 3: 分层

最小费用流问题

最小费用流问题的例子

公路交通网络：车辆路线确定



1辆车

最短路问题

多辆车:车流

最小费用流问题

✓ 许多实际问题都可以转化为最小费用流问题

最小费用流问题

定义 在流网络$N=(V,A，L,U,D)$ 上增加如下的权函数：
C：$A\to R$为弧上的权函数，弧（i，j）对应的权 C（i，j）记为c_{ij}，称为弧（i，j）的单位流量的成本或费用（cost）；
此时网络可称为容量-费用网络 (一般仍可简称为网络),记为$N=(V,A,C，L,U,D)$.

$d_i>0$:供应点(supply node)或源(source)、起始点或发货点
$d_i<0$:需求点(demand node)或汇(sink) 、终止点或吸收点
$d_i=0$:转运点(transshipment node)或平衡点、中间点

可以把$L\neq 0$的网络转化为$L=0$的网络进行研究(思考？)
除非特别说明, 假设$L=0$,网络简记为$N=(V,A,C，U,D)$.

最小费用流问题

定义 （容量-费用网络中的流(flow) 的定义同前一章）
流x的（总）费用定义为 $$c(x)=\sum_{(i,j)\in A} c_{ij}x_{ij}$$ 线性费用网络

最小费用流问题就是在网络中寻找总费用最小的可行流.

$$\min c(x)=\sum_{(i,j)\in A} c_{ij}x_{ij} \qquad s.t. \sum_{j:(i,j)\in A} x_{ij}-\sum_{j:(j,i)\in A} x_{ji}=d_i, \qquad \forall i\in V \qquad (7.1)$$

$$0\le x_{ij}\le u_{ij}, \qquad \forall (i,j)\in A \qquad (7.2)$$

通常又称为转运问题(transshipment problem)或容量受限的转运问题(capacitated transshipment problem).

引理 最小费用流问题存在可行流的必要条件 $$\sum_{i\in V} d_i=0.$$

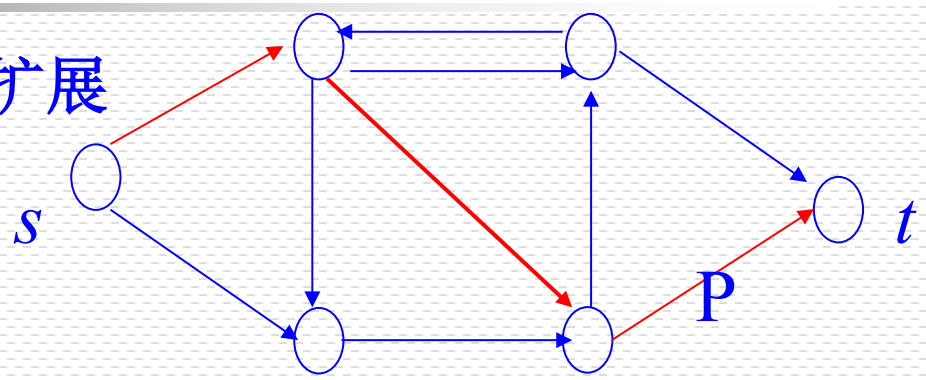
经典的最小费用流问题：单源单汇(起点s，终点t)，寻找从s流到t的给定流量(或最大流量、最小流量等)的最小费用流.

$$d_s=v, d_t=-v \qquad d_i=0 (i\ne s,t)$$

思考： 经典问题与一般问题有什么关系？是否等价？

最小费用流问题

- 最小费用流模型的特例及扩展



例 最短路问题

在有向网络中，令所有弧上容量下界为0，容量（上界）为1，并令图中节点s的供需量为1，节点t的供需量为-1，则：

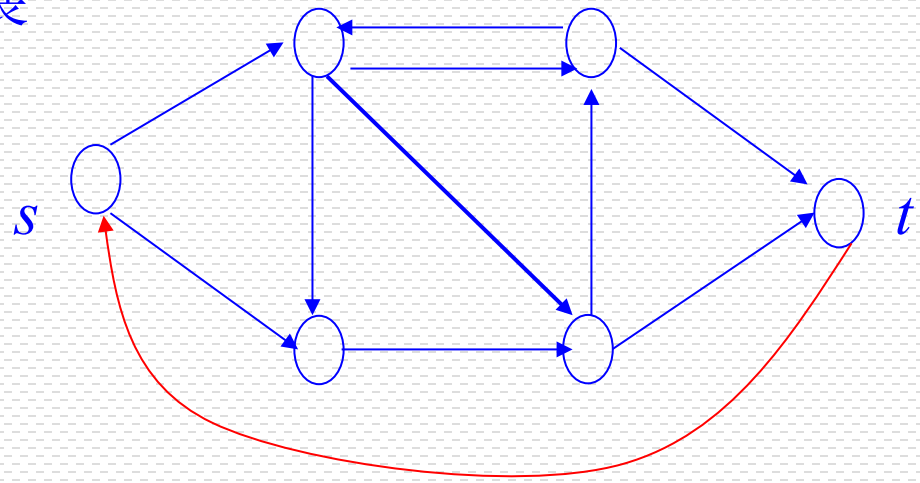
从s到t的一条有向路正好对应于网络中的一个可行流x

（弧(i, j)位于s-t路上：$x_{ij}=1$；弧(i, j)不在s-t路上：$x_{ij}=0$）.

如果再令所有弧上的(单位流量)的费用为“弧长”，则此时的最小费用流问题就是第五章讨论过的最短路问题.

- 最小费用流模型的特例及扩展

例 - 最大流问题



设s为起点,t为终点,增加弧(t,s), 令

$$c_{ts} = -1, u_{ts} = +\infty$$

而令所有其他弧上的费用为0,
所有顶点上的供需量(外部流量)全为0.

最小费用流问题

- 最小费用流模型的特例及扩展

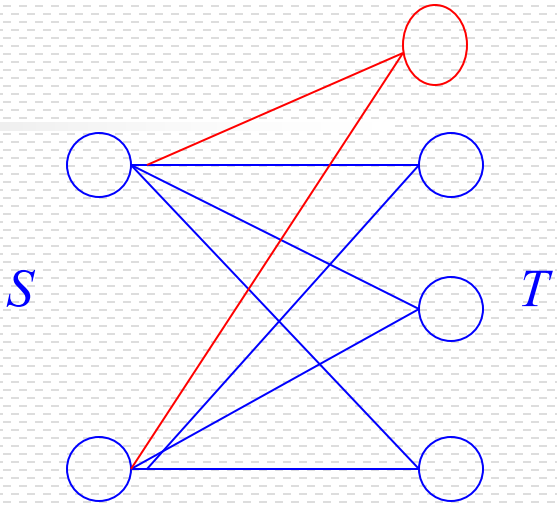
例 -运输问题(transportation Problem)
又称Hitchcock问题（Hitchcock,1941年）

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad (7.5)$$

$$s.t. \quad \sum_{j:(i,j)\in A} x_{ij} \le a_i, \quad i \in S, \qquad (7.6)$$

$$\sum_{i:(i,j)\in A} x_{ij} \ge b_j, \quad j \in T, \qquad (7.7)$$

$$x_{ij} \ge 0, \quad (i,j) \in A. \qquad (7.8)$$



完全二部图; 只有源和汇, 没有中间转运点

如果每条弧上还有容量(上下限)的限制，则称为容量受限的运输问题(capacitated transportation problem).

有解的必要条件 $\sum_{i\in S} a_i \ge \sum_{j\in T} b_j$

可以不失一般性 $\sum_{i\in S} a_i = \sum_{j\in T} b_j$

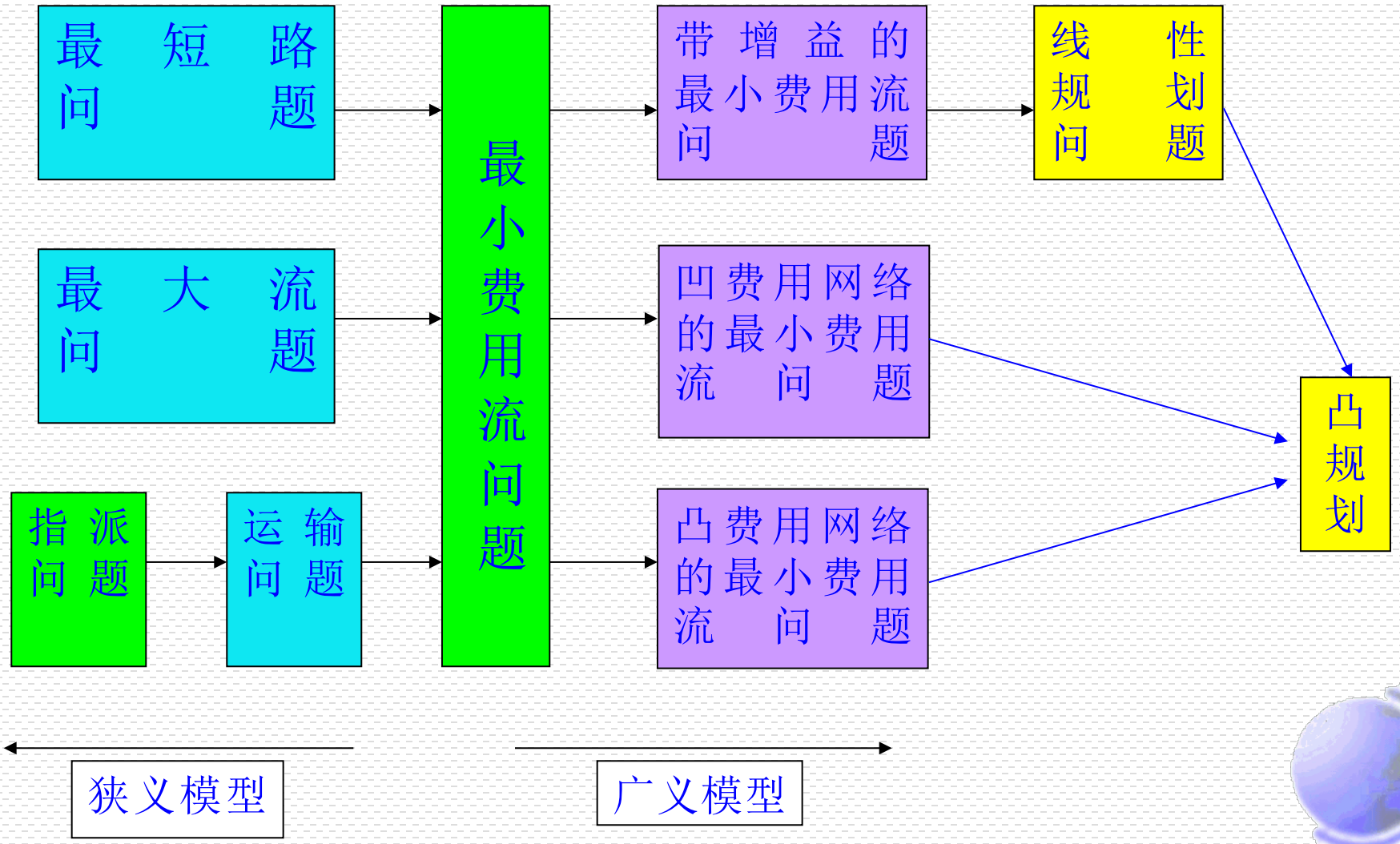
指派问题(assignment problem)

$a_i = b_j \equiv 1, |S| = |T|$

39

最小费用流问题

- 最小费用流模型的特例及扩展



求解最小费用流问题的算法

- **消圈算法与最小费用路算法**

➢单源单汇网络

➢可行流x的流量（或流值）为$v=v(x)=d_s=-d_t$

➢如果我们并不给定d_s和d_t，则网络一般记为$N=(s,t,V,A,C,U)$

➢容量可行且转运点流量守恒的流称为s-t可行流，有时为了方便也称为可行流.

➢*最小费用流问题就是在网络$N=(s,t,V,A,C,U)$中计算流值为v的最小费用流x*

➢*或者当不给定流值时，计算流值最大的最小费用流x (此时流x称为最小费用最大流).*

- **瑕疵算法、最小费用路算法、松弛算法**

End