

# 第9讲 网络流问题 (下)

罗国杰

[gluo@pku.edu.cn](mailto:gluo@pku.edu.cn)

2024年春季学期

# 网络流的应用

- 二部图的最大匹配
- 赋权二部图的匹配
- 带需求的流通
- 边不相交和点不相交路径
- 项目选择（最大权闭包）
- 经典图像分割算法
- 最小密度子图

## 二部图匹配

**定义** 设简单二部图 $G=\langle A, B, E \rangle$ ,  $M \subseteq E$ , 如果 $M$ 中任意两条边都不相邻, 则称 $M$ 是 $G$ 的**匹配**. 边数最多的匹配称作**最大匹配**. 当 $|A|=|B|=n$ 时, 边数为 $n$ 的匹配称作**完美匹配**.

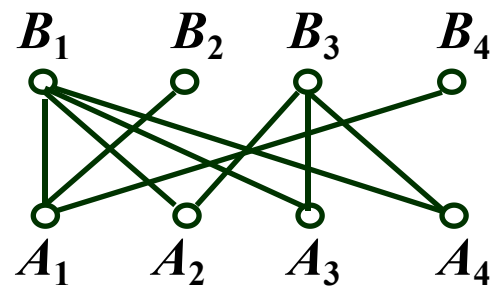
**例5** 有4名新入学的硕士生和4位硕导,  $A_1$ 申请 $B_1, B_2$ 或 $B_4$ 作导师,  $A_2, A_3$ 和 $A_4$ 都申请 $B_1$ 或 $B_3$ 作导师. 每名硕士生有一位导师, 每位硕导只收一名新生. 如何分配才能尽可能满足学生要求.

作二部图 $G=\langle A, B, E \rangle$ , 其中

$$A = \{ A_i \mid 1 \leq i \leq 4 \}$$

$$B = \{ B_j \mid 1 \leq j \leq 4 \}$$

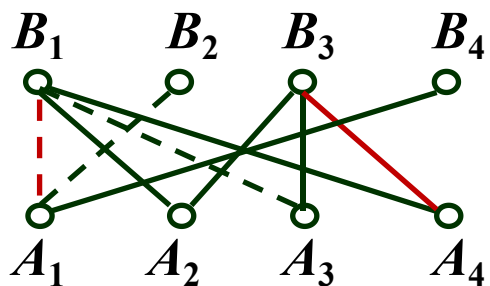
$$E = \{ (A_i, B_j) \mid A_i \text{ 申请 } B_j \text{ 作导师, } 1 \leq i \leq 4, 1 \leq j \leq 4 \}$$



# 增广交错路径

**定义** 设  $M$  是二部图  $G$  的匹配.

- 称  $M$  中的边为**匹配边**
- 不属于  $M$  的边为**非匹配边**
- 与匹配边关联的顶点为**饱和点**
- 不与匹配边关联的顶点为**非饱和点**
- $G$  中由匹配边和非匹配边交替构成的路径称为**交错路径**
- 起点和终点都是非饱和点的交错路径称为**增广交错路径**



$M$  与  $P$

$$M = \{ (B_1, A_1), (B_3, A_4) \}$$

饱和点:  $B_1, B_3, A_1, A_4$

非饱和点:  $A_2, A_3, B_2, B_4$

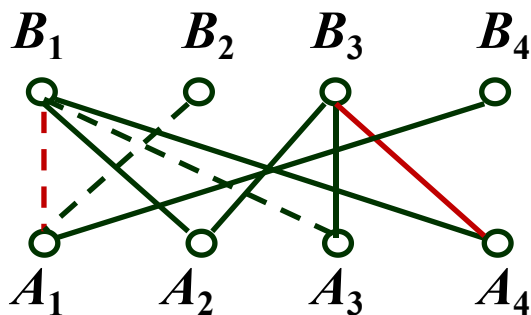
增广交错路径  $P$ :  $A_3 B_1 A_1 B_2$

# 最大匹配的条件

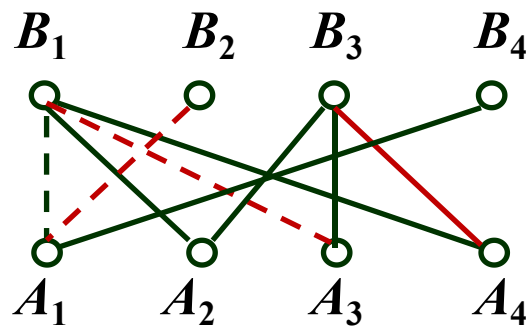
**引理14** 设  $M$  是二部图  $G$  的一个匹配,  $P$  是一条关于  $M$  的增广交错路径, 则

$$M' = M \oplus E(P)$$

是一个匹配 且  $|M'| = |M| + 1$ .



$M$  与  $E(P)$

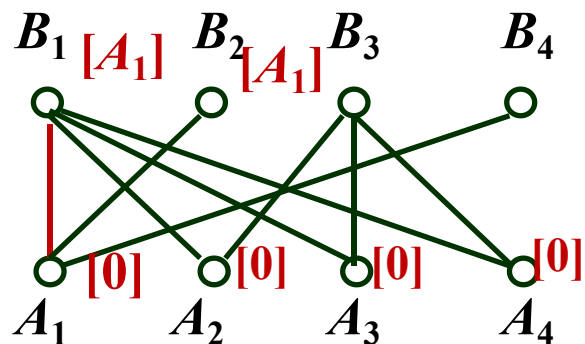


$M'$

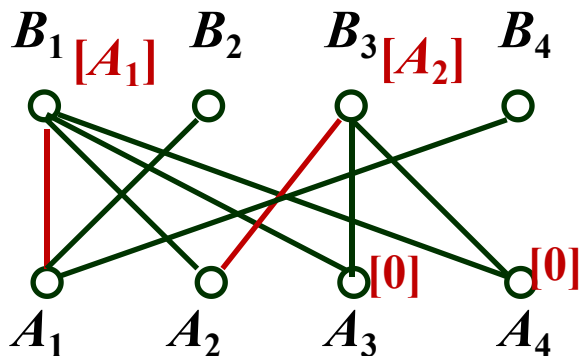
**定理8** 二部图的匹配是最大匹配当且仅当不存在关于它的增广交错路径.

# 匈牙利算法

**匈牙利算法** 从一个初始匹配  $M$  开始, 每次找一条增广交错路径  $P$ , 令  $M \leftarrow M \oplus E(P)$ , 直到不存在增广交错路径为止.

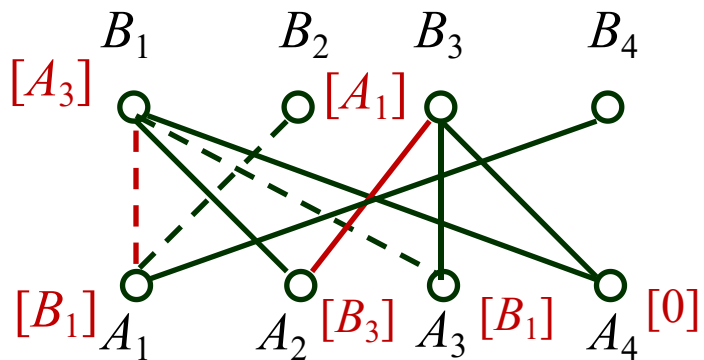


增广交错路径:  $A_1B_1$   
 $M = \{(A_1, B_1)\}$

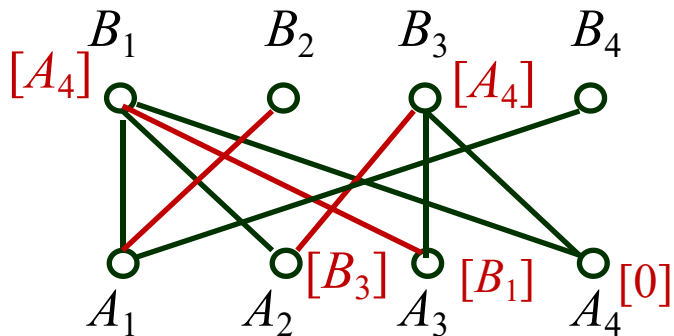


增广交错路径:  $A_2B_3$   
 $M = \{(A_1, B_1), (A_2, B_3)\}$

# 匈牙利算法 (续)



增广交错路径:  $B_2A_1B_1A_3$



匹配  $M \oplus E(P)$   
 $= \{(A_1, B_2), (A_2, B_3), (A_3, B_1)\}$

算法时间:  $O(\min\{|A|, |B|\} \cdot |E|)$

阶段数:  $\min\{|A|, |B|\} + 1$ , 每个阶段检查时间:  $O(|E|)$

## 二部图最大匹配与最大流

### 最大匹配转化成最大流

设  $G=\langle A, B, E \rangle$ , 作容量网络  $N_G = \langle V, E', c, s, t \rangle$ , 其中  $c \equiv 1$ ,

$$V = \{s, t\} \cup A \cup B$$

$$E' = \{ \langle s, A_i \rangle \mid A_i \in A \} \cup \{ \langle B_j, t \rangle \mid B_j \in B \} \cup \{ \langle A_i, B_j \rangle \mid (A_i, B_j) \in E \}$$

$G$  的匹配  $\Leftrightarrow N_G$  上的 0-1可行流

前向边是非饱和边, 后向边是饱和边, 增广链  $\Leftrightarrow$  增广交错路径.  
匈牙利算法是FF算法的应用.

### 算法设计思想:

1. 构造对应容量网络  $N_G$
2. 对  $N_G$  应用 Dinic 算法求最大流  $f$
3. 再把  $f$  转化为  $G$  的匹配  $M$

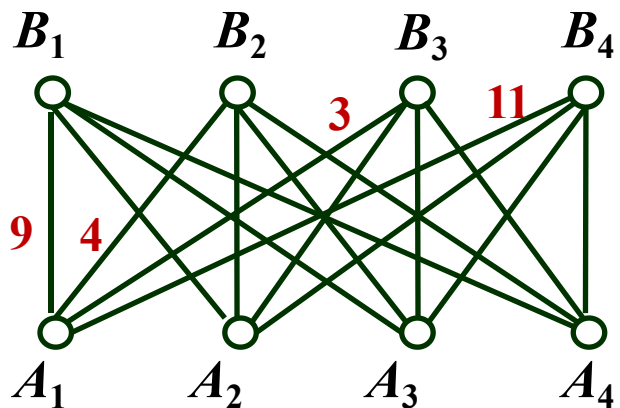
时间:  $O(n^{1/2}m)$



# 赋权二部图完美匹配

**指派问题** 给定赋权完全二部图  $G=\langle A,B,E,w\rangle$ , 其中  $|A|=|B|=n$ ,  $w: E\rightarrow R$ , 求  $G$  的权最小的完美匹配  $M$ .

**例6** 有4项任务由4个人完成, 每人完成一项. 预计每个人完成每一项任务的时间如下表所示. 如何分配才能是完成任务的总时间最少?



	任 务			
	$B_1$	$B_2$	$B_3$	$B_4$
人 $A_1$	9	4	3	11
$A_2$	8	5	8	4
$A_3$	3	8	3	1
$A_4$	10	6	9	6

使用匈牙利算法, 时间  $O(n^3)$

# 指派问题与线性规划

指派问题可表成 线性规划(P)

$$x_{ij} = \begin{cases} 1, & (A_i, B_j) \in M, \\ 0, & \text{否则,} \end{cases}$$

$$\min \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_{ij}$$

$$s.t. \quad \sum_{j=1}^n x_{ij} = 1, \quad 1 \leq i \leq n$$

$$\sum_{i=1}^n x_{ij} = 1, \quad 1 \leq j \leq n$$

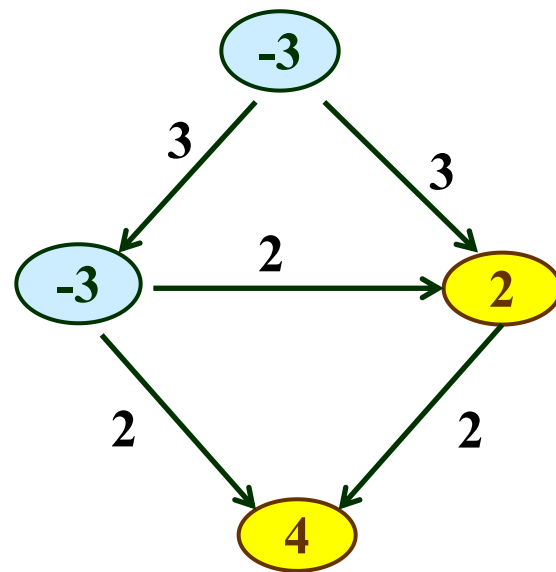
$$x_{ij} \geq 0, \quad 1 \leq i \leq n, 1 \leq j \leq n$$

# 带需求的流通

给定带需求的流通图  $N = \langle V, E, c, S, T \rangle$ ,  $\forall v \in V$  存在需求  $d_v$ , 所有发点集合  $S$ , 所有收点集合  $T$ .

- 收点:  $d_v > 0$ , 表示  $v$  对流有  $d_v$  的需求
- 发点:  $d_v < 0$ , 表示  $v$  有  $-d_v$  的供给
- 结点:  $d_v = 0$ ,  $v$  不是发点和收点

所有容量和需求都是整数



## 带需求的流通

问是否存在可行流通? 即存在函数  $f: E \rightarrow \mathbb{R}^*$ , 满足

- (1) 容量条件:  $\forall e \in E, 0 \leq f(e) \leq c_e$
- (2) 需求条件:  $\forall v \in V, f^{\text{in}}(v) - f^{\text{out}}(v) = d_v$

## 带需求的流通：必要条件

**命题1** 如果存在一个带需求 $\{d_v\}$ 的可行流通, 那么  $\sum_v d_v = 0$

证 假设存在可行流通  $f$ . 那么

$$\sum_v d_v = \sum_v (f^{\text{in}}(v) - f^{\text{out}}(v))$$

对于每条边  $e = \langle u, v \rangle$ , 值  $f(e)$  恰好在  $f^{\text{in}}$  和  $f^{\text{out}}$  中各出现1次.  
两项抵消; 总和是0.

根据命题1, 有

$$\sum_{v:d_v > 0} d_v = \sum_{v:d_v < 0} -d_v$$

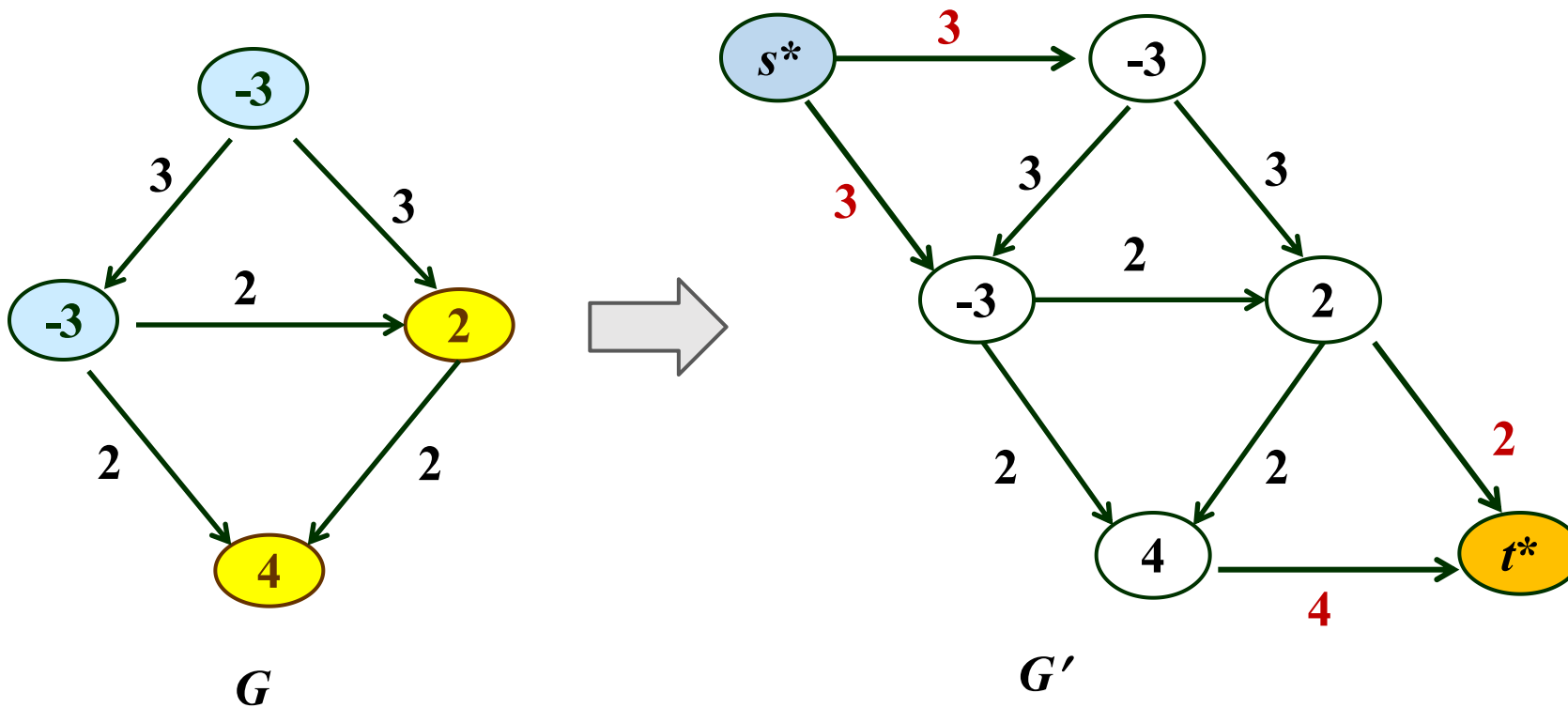
将该和记作  $D$ .

# 带需求的流通：用最大流建模

加超结点  $s^*$  和  $t^*$ ，构造单发点单收点的容量网络  $G'$

从  $s^*$  到每个  $v \in S$  加边  $e = \langle s^*, v \rangle$ ，令  $c_e = -d_v$

从每个  $v \in T$  加边  $e = \langle v, t^* \rangle$ ，令  $c_e = d_v$



# 带需求的流通：算法与拓展

**命题2**  $G'$  没有大于  $D$  的  $s^*-t^*$  流, 因为  $c(\{s^*\}, V \cup \{t^*\}) = D$ , 其中

$$D = \sum_{v:d_v>0} d_v = \sum_{v:d_v<0} -d_v$$

**定理**  $G$  中存在一个带需求  $\{d_v\}$  的可行流通, 当且仅当  $G'$  的最大  $s^*-t^*$  流有值  $D$ .

## 算法

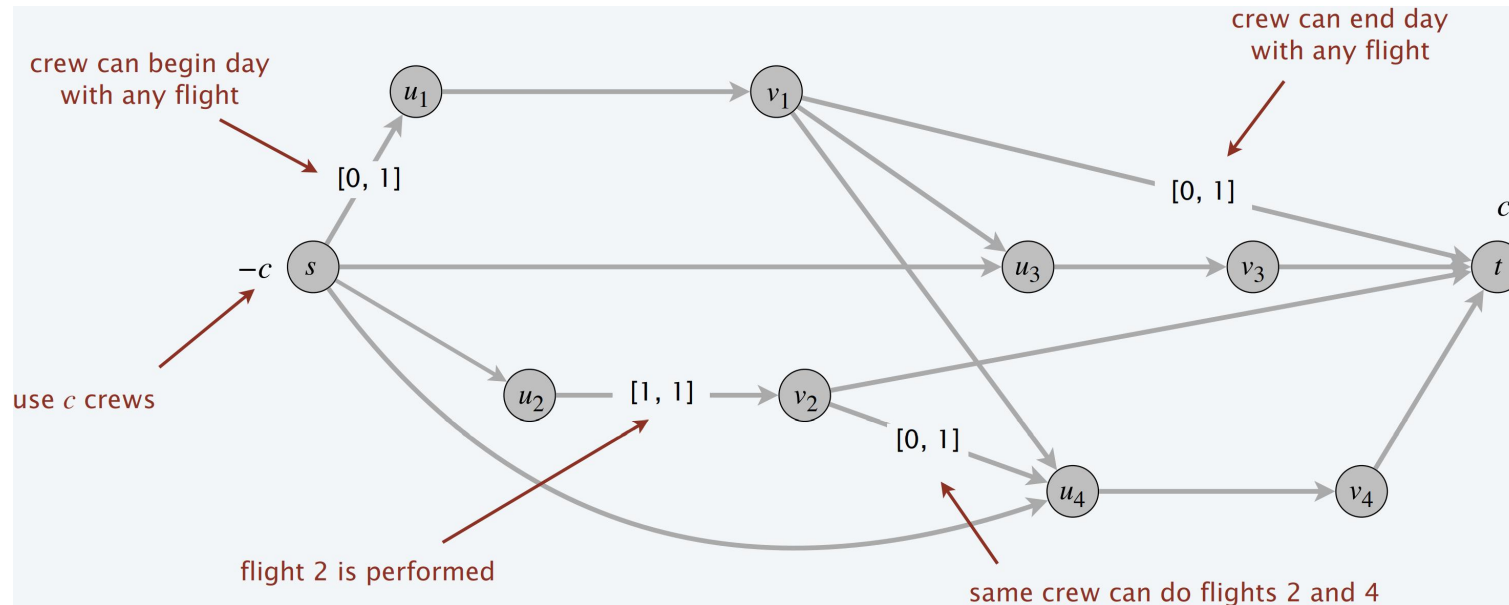
1. 将流通图  $G$  转换为对应的容量网络  $G'$
2. 对  $G'$  运行最大流算法找到  $s^*-t^*$  最大流  $f$
3. 如果  $v(f) = D$ , 则  $G$  存在带需求  $\{d_v\}$  的可行流通  
否则在该需求下不存在可行流通

**拓展** 带需求和下界的可行流通. 每条边增加下界  $l_e$ ,  $0 \leq l_e \leq c_e$ . 修改容量条件, 对每个  $e \in E$ ,  $l_e \leq f(e) \leq c_e$ . 需求条件不变.

# Airline Scheduling (1/2)

## ► “Toy problem.”

- Manage flight crews by reusing them over multiple flights.
- Input: set of  $k$  flights for a given day.
- Flight  $i$  leaves origin  $o_i$  at time  $s_i$  and arrives at destination  $d_i$  at time  $f_i$
- Minimize number of flight crews.



# Airline Scheduling (2/2)

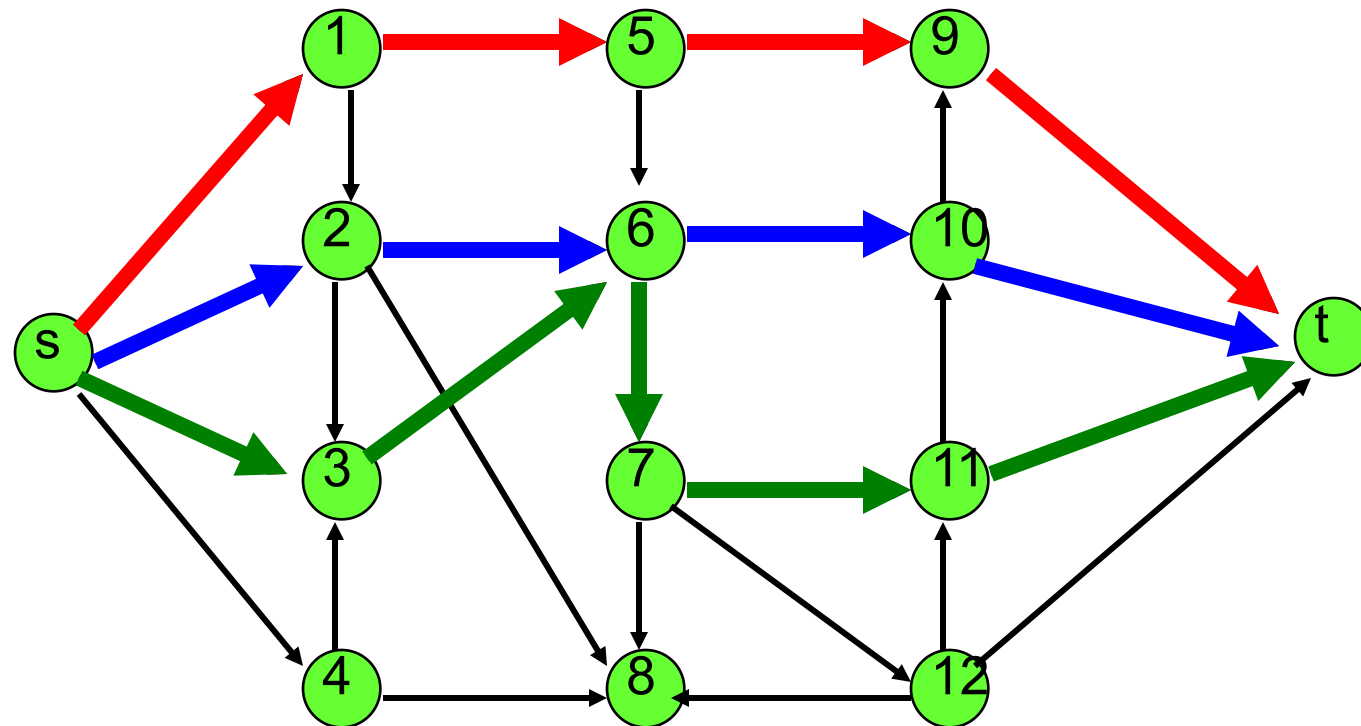
- Real-world problem models countless other factors:
  - ▶ Union regulations: e.g., flight crews can fly only a certain number of hours in a given time window.
  - ▶ Need optimal schedule over planning horizon, not just one day.
  - ▶ Deadheading has a cost.
  - ▶ Flights don't always leave or arrive on schedule.
  - ▶ Simultaneously optimize both flight schedule and fare structure.
- Message
  - ▶ Our solution is a generally useful technique for efficient reuse of limited resources but trivializes real airline scheduling problem.
  - ▶ Flow techniques useful for solving airline scheduling problem



# Network Reliability

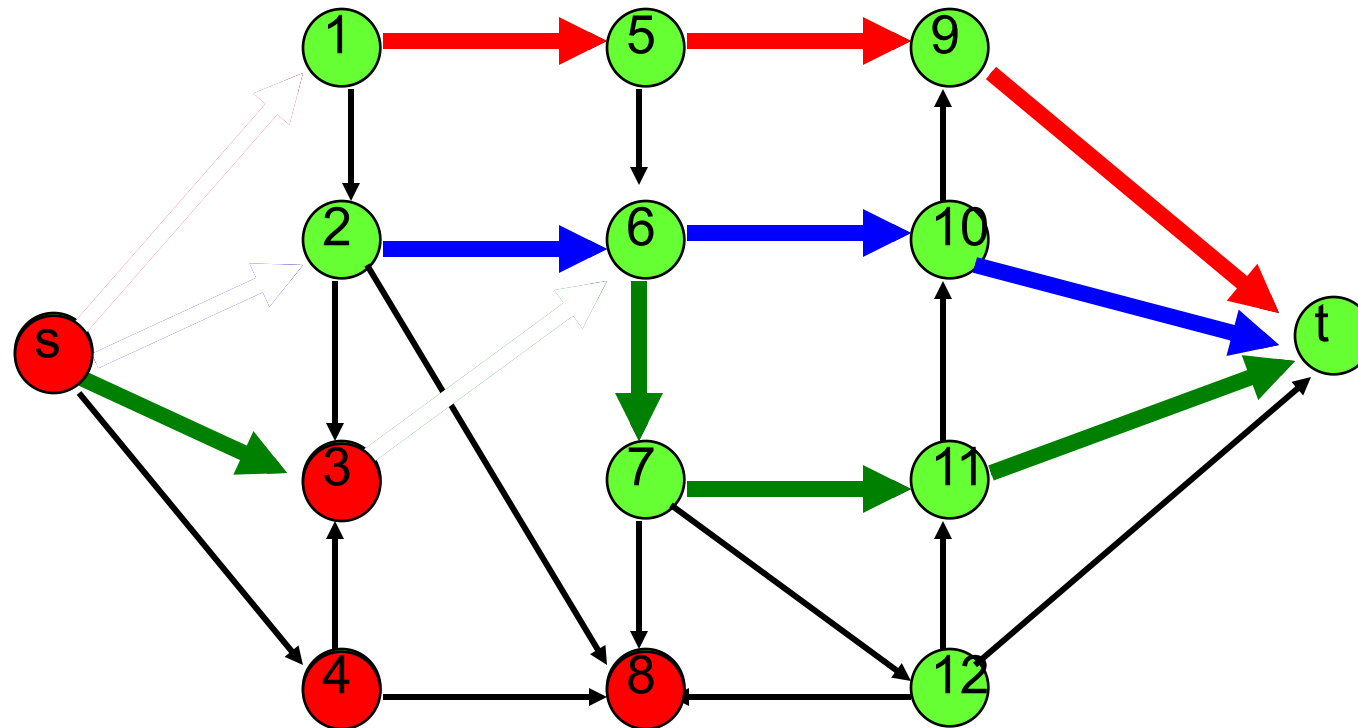
- Communication Network
- What is the maximum number of edge-disjoint paths from  $s$  to  $t$ ?
  - ▶ How can we determine this number?

There are 3 edge-disjoint s-t paths



# Deleting 3 edges disconnects s and t

- Theorem [Menger 1927]. The max number of edge-disjoint s-t paths equals the min number of edges whose removal disconnects t from s.

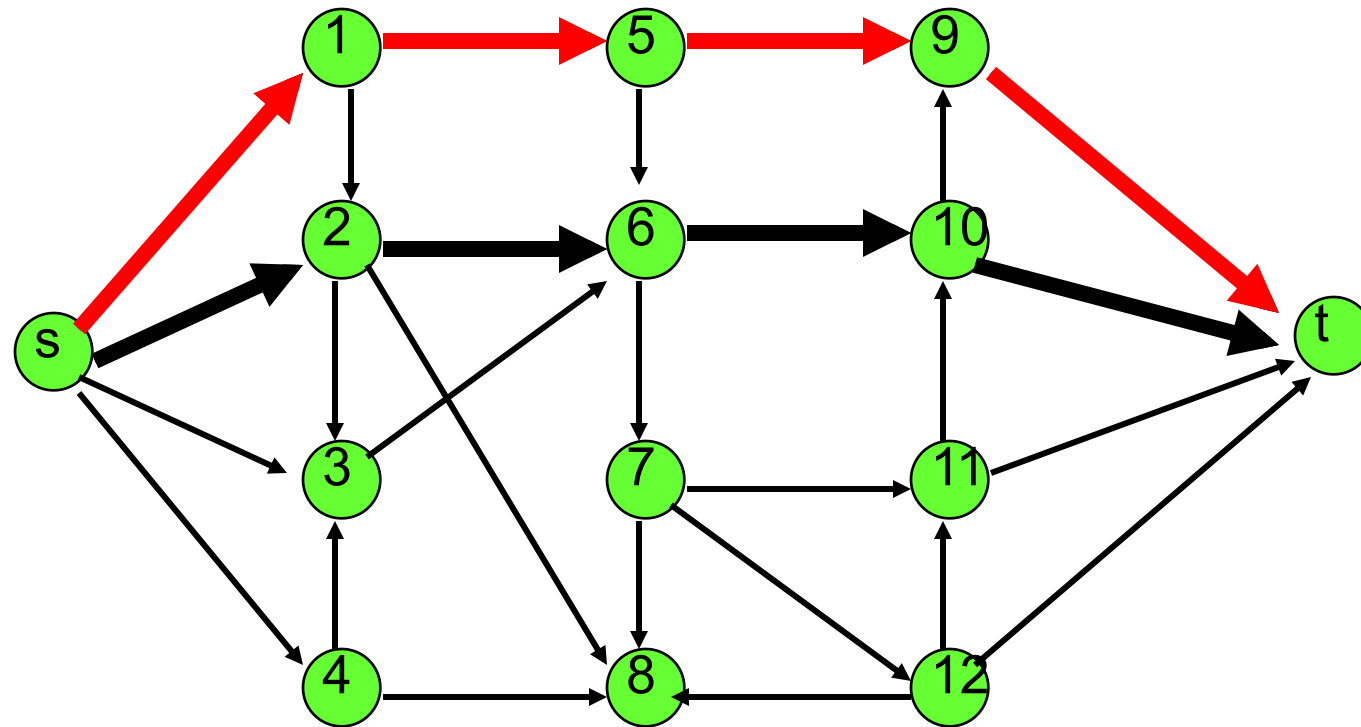


Let  $S = \{s, 3, 4, 8\}$ . The only arcs from  $S$  to  $T = N \setminus S$  are the 3 deleted arcs.

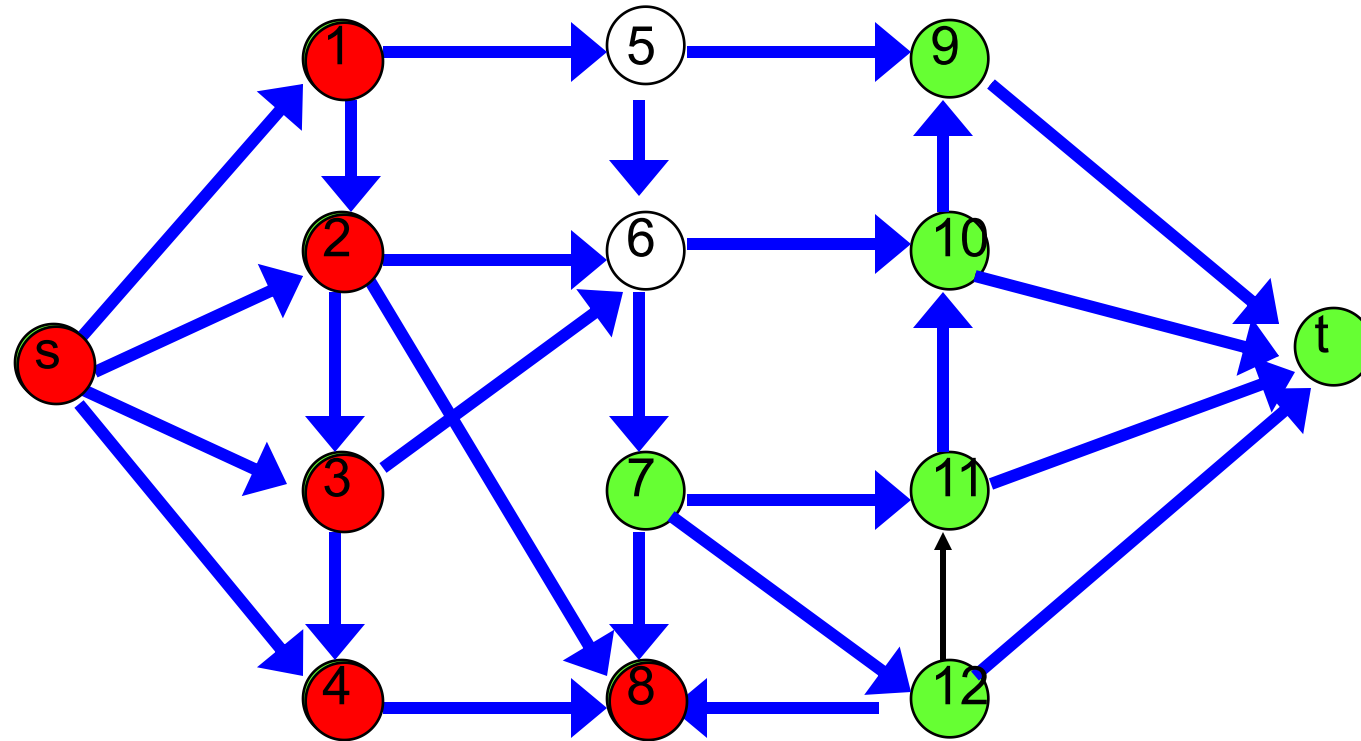
## Node-disjoint paths

- Two  $s$ - $t$  paths  $P$  and  $P'$  are said to be **node-disjoint** iff the only nodes in common to  $P$  and  $P'$  are  $s$  and  $t$ ).
- How can one determine the maximum number of node disjoint  $s$ - $t$  paths?
- **Answer:** node splitting
- **Theorem.** Let  $G = (N, A)$  be a network with no arc from  $s$  to  $t$ . The maximum number of node-disjoint paths from  $s$  to  $t$  equals the minimum number of nodes in  $N \setminus \{s, t\}$  whose removal from  $G$  disconnects all paths from nodes  $s$  to node  $t$ .

There are 2 node-disjoint s-t paths.



Deleting 5 and 6 disconnects  $t$  from  $s$ .



Let  $S = \{s, 1, 2, 3, 4, 8\}$

Let  $T = \{7, 9, 10, 11, 12, t\}$

There is no arc directed from  $S$  to  $T$ .

# Project Selection (Maximum Weight Closure)

► 最大化项目收益  $\Leftrightarrow$  最小化  $\text{cap}(A,B) = \sum_{v \in B: p_v > 0} p_v + \sum_{v \in A: p_v < 0} (-p_v) = \sum_{v: p_v > 0} p_v - \sum_{v \in A} p_v$

► 容量网络构造

► 虚拟源  $s$  和汇  $t$

► 项目  $v$  价值  $p_v > 0$

•  $(s,v)$  容量  $p_v$  的边

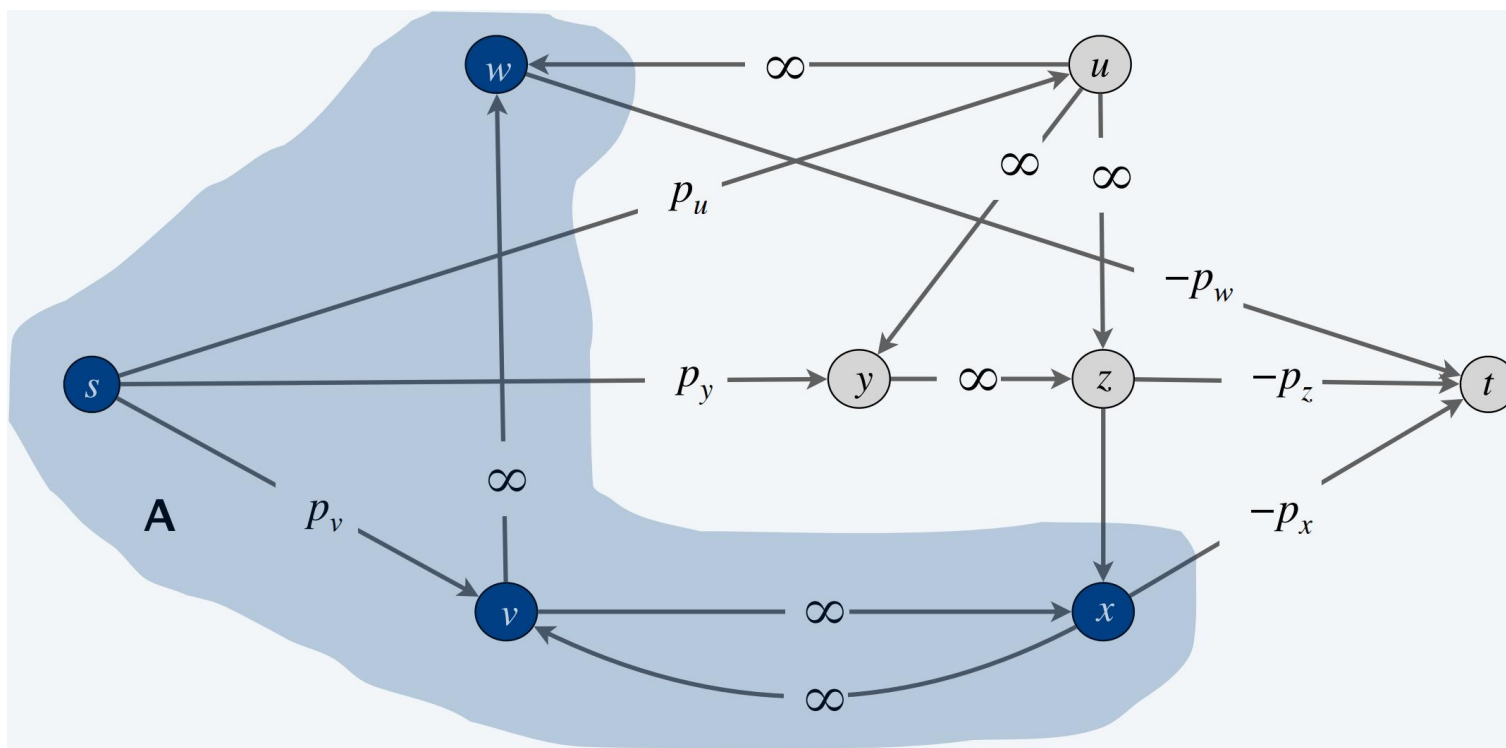
► 项目  $v$  价值  $p_v < 0$

•  $(v,t)$  容量  $-p_v$  的边

► 项目  $v$  依赖  $w$

•  $(v,w)$  容量  $\infty$  的边

► 最大流求解最小  $\text{cap}(A,B)$



picture from J. Kleinberg and É. Tardos, "Algorithm Design", Pearson 2006.

Jean-Claude Picard. 1976. "Maximal Closure of a Graph and Applications to Combinatorial Problems." Manage. Sci. 22, 11 (July 1976), 1268–1272.

# 图像分割

- Separate foreground from background





# 图像分割

- ➡ 图像分割：将一幅图片的前景与背景分离  
对每个像素进行“前景 / 背景”的标记



图片



分割后

注：有些 DNN 生成像素级的前景背景预测+阈值化以足够精确；网络流可应用在不太精确的 DNN 或其他预测方法的场合

# 对图像分割的建模

- 设  $V$  是基本图像中的像素集合. 用  $E$  表示所有相邻像素对的集合, 构成无向图  $G=\langle V,E\rangle$ .
- 像素  $i$  属于前景的可能性:  $a_i$ , 属于背景的可能性:  $b_i$
- 如果  $a_i > b_i$ , 像素  $i$  标为前景, 否则标为背景. 标记为前景的像素构成集合  $A$ , 标记为背景的像素构成集合  $B$ .
- 如果像素  $i$  的邻居都标为“背景”, 那么倾向于将  $i$  标为背景, 使标记更“光滑”. 对于每对邻居像素  $(i, j)$ , 定义分离罚分  $p_{ij} > 0$  来惩罚  $i$  和  $j$  标记不同.
- 图像分割问题: 将像素点集划分为  $A$  和  $B$ , 使得划分的权值  $q(A,B)$  最大化 (即得到最优标记). 其中

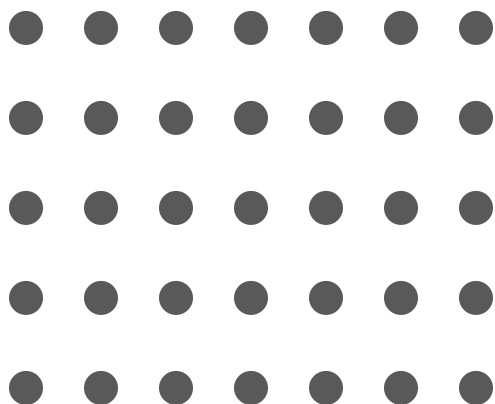
$$q(A,B) = \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

# 图像分割：最优标记与最小割

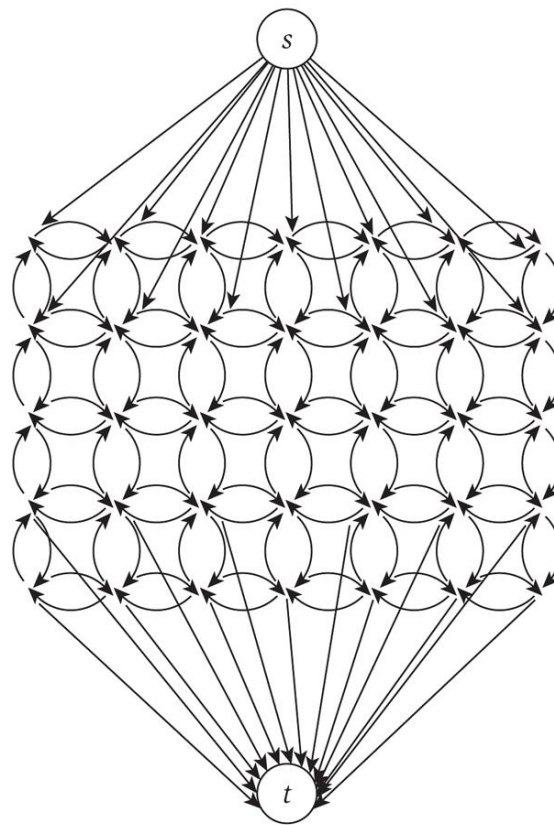
- 两个问题的相似性
  - ▶ 都涉及图的点割集(A,B)
- 区别：
  - ▶ 图像最优标记的图是无向图，流网络最小割是有发点  $s$ 、收点  $t$  的有向图
  - ▶ 最优标记是最大化问题，最小割是最小化问题
  - ▶ 最优划分问题的结点有参数  $a_i$  和  $b_i$ ，最小割问题边有参数
- 思路：
  - ▶ 将最优标记的邻居图转化为一个容量网络

## 图像分割：转化为容量网络 $G'$

- 设 $G=\langle V, E \rangle$ 是图像最优标记问题的无向图，将每条边 $(u, v) \in E$ 转化成一对有向边 $\langle u, v \rangle$ 和 $\langle v, u \rangle$ ，构成有向图
- 超源点  $s$  (前景)，超汇点  $t$  (背景)
  - 加边  $\langle s, v \rangle$  和  $\langle v, t \rangle$ ,  $\forall v \in V$ .



图像点阵列



对应  
容量  
网络  
 $G'$  (部  
分边)

# 图像分割：G的最大化与G'的最小化

► 令  $Q = \sum_i (a_i + b_i)$

$$\Rightarrow \sum_{i \in A} a_i + \sum_{j \in B} b_j = Q - \sum_{i \in A} b_i - \sum_{j \in B} a_j$$

►  $q(A, B)$  的最大化转变为  $q'(A, B)$  的最小化

$$q(A, B) = Q - \sum_{i \in A} b_i - \sum_{j \in B} a_j - \sum_{\substack{(i, j) \in E \\ |A \cap \{i, j\}| = 1}} p_{ij}$$

$$q'(A, B) = \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{\substack{(i, j) \in E \\ |A \cap \{i, j\}| = 1}} p_{ij}$$

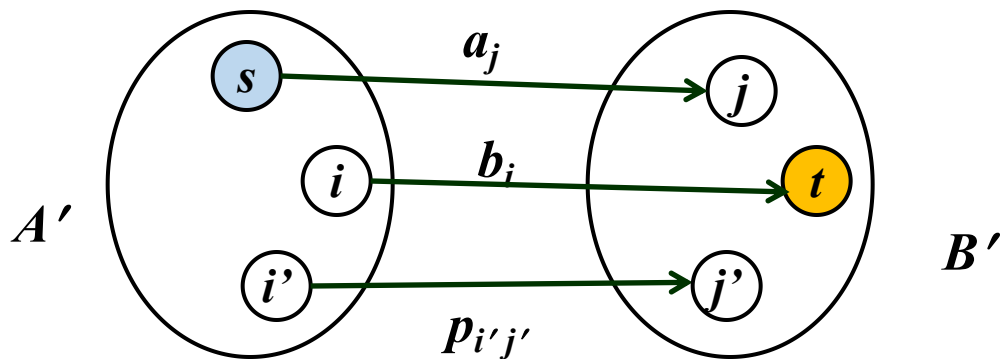
# 图像分割：容量分配

## 边容量

- ▶ 对边  $e = \langle s, i \rangle$ , 令  $c_e = a_i$ ; 对边  $e = \langle i, t \rangle$ , 令  $c_e = b_i$ .
- ▶ 对邻居边  $e = \langle j, i \rangle$  和  $e' = \langle i, j \rangle$ , 令  $c_e = c_{e'} = p_{ij}$ .

## 穿过割 $(A', B')$ 的边分成三类:

- ▶ 边  $\langle s, j \rangle$ ,  $j \in B'$ : 为割的容量贡献  $a_j$ .
- ▶ 边  $\langle i, t \rangle$ ,  $i \in A'$ : 为割的容量贡献  $b_i$ .
- ▶ 边  $\langle i', j' \rangle$ ,  $i' \in A', j' \in B'$ : 为割的容量贡献  $p_{i'j'}$



## 图像分割：割的容量

- 所有边  $\langle s, j \rangle$ ,  $j \in B'$ , 对割容量贡献  $\sum_{j \in B} a_j$
- 所有边  $\langle i, t \rangle$ ,  $i \in A'$ , 对割容量贡献  $\sum_{i \in A} b_i$
- 所有边  $\langle i, j \rangle$ ,  $i \in A'$ ,  $j \in B'$ ; 对割容量贡献  $\sum_{i \in A, j \in B} p_{ij}$

$$c(A', B')$$

$$= \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}|=1}} p_{ij}$$

$$= q'(A, B)$$

# 图像分割算法

## ► 主要步骤

- 建立像素邻居图  $G$
- 将  $G$  转化成容量网络  $G'$
- 求解  $G'$  的最小割  $(A', B')$
- 删除  $s$  和  $t$  可以得到划分  $(A, B)$
- 将  $A$  中像素标记为前景,  $B$  中像素标记为背景

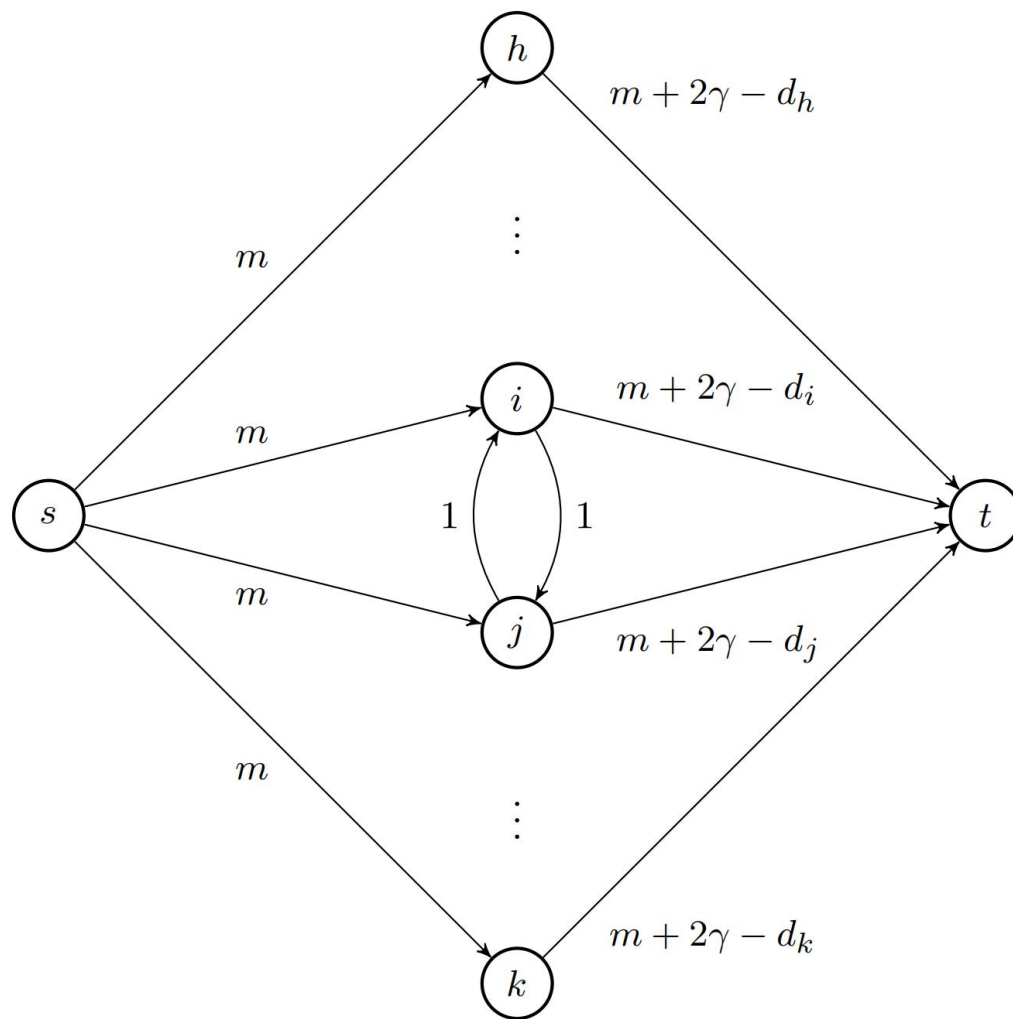
## ► 时间复杂度 $O(nm^2)$



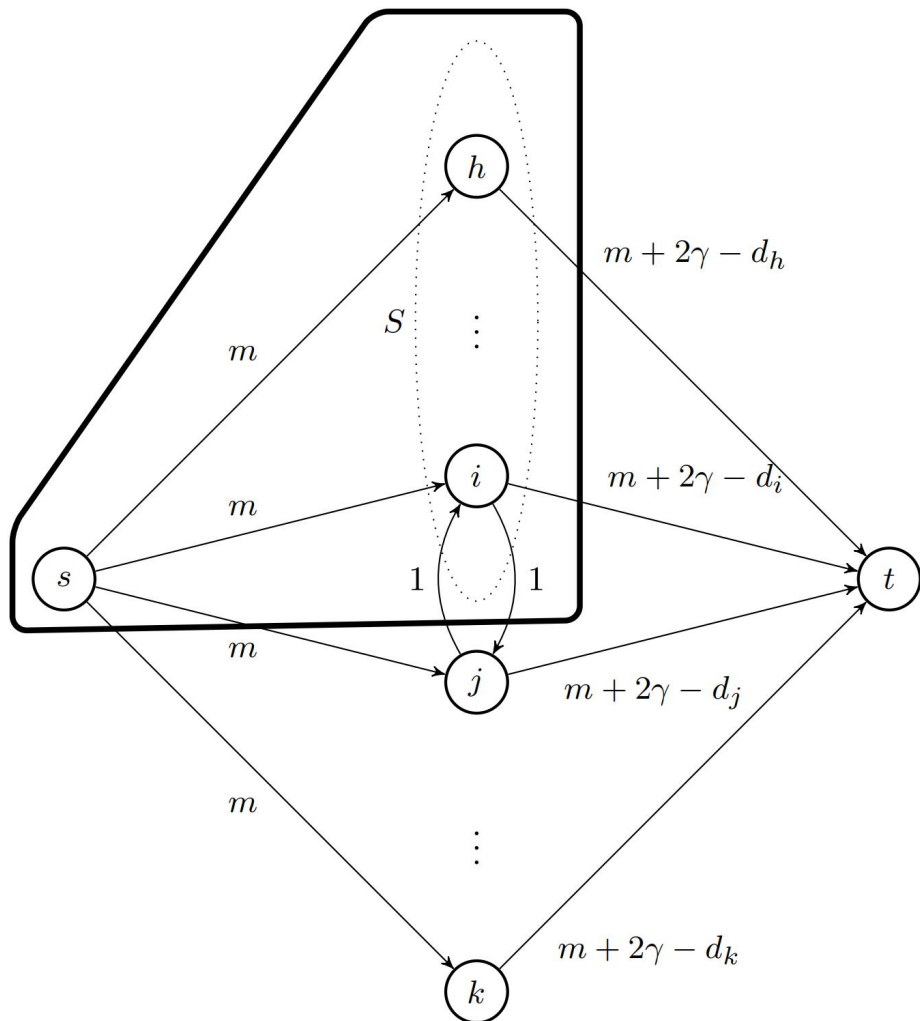
# 最大密度子图

- 给定无向图  $G = (V, E)$
- 非空顶点子集  $S \neq \emptyset$  导出子图  $G(S) = (S, E(S))$ , 其中  $E(S) = \{(i, j) \in E: i, j \in S\}$
- 子图密度  $D(S) = |E(S)| / |S|$
- 求密度最大的子图
  
- 思路
  - ▶ 构造判断  $D^* \leq \gamma$  的最大流问题
  - ▶ 二分密度最大值  $\gamma$ , 求  $O(\log n)$  次最大流

# 最大密度子图：最大流模型



# 最大密度子图： $\{s\} \cup S$ 的割大小



记  $\delta(S)$  为恰有一个顶点在  $S$  的边集

$\{s\} \cup S$  的割大小

$$\begin{aligned}
 & m |V-S| + |\delta(S)| + \sum_{i \in S} (m + 2\gamma - d_i) \\
 &= m (n - |S|) + |\delta(S)| + (m |S| + 2\gamma |S| - \sum_{i \in S} d_i) \\
 &= mn + |\delta(S)| + 2\gamma |S| - \sum_{i \in S} d_i \\
 &= mn + |\delta(S)| + 2\gamma |S| - (2|E(S)| + |\delta(S)|) \\
 &= mn + 2|S|(\gamma - |E(S)|/|S|)
 \end{aligned}$$

引理：  $\gamma < D^*$  当且仅当 最大流  $< mn$

## 最大密度子图：若干性质 (1/2)

- $\{s\} \cup S$  的割大小:  $= mn + 2|S|(\gamma - |E(S)|/|S|)$
- 引理:  $\gamma < D^*$  当且仅当 最大流  $< mn$ 
  - ▶ 必要性:  $\gamma < D^*$  时能找到  $< mn$  的割集, 得最大流  $< mn$ 。
  - ▶ 充分性: 最大流  $< mn$  时, 对应割集大小  $< mn$ , 得  $\gamma < |E(S)|/|S| \leq D^*$
- 推论: 假设  $D'$  是第二大的子图密度, 由  $D' \leq \gamma < D^*$  最大流问题对应的最小割集  $\{s\} \cup X$  的  $X$  组成最大密度子图
  - ▶ 密度  $\leq \gamma$  顶点子集  $S$  组成的  $\{s\} \cup S$  不可能是最小割
  - ▶ 只有密度  $> \gamma$  的顶点子集  $X$  才能组成最小割  $\{s\} \cup X$

## 最大密度子图：若干性质 (2/2)

- $\{s\} \cup S$  的割大小:  $= mn + 2|S|(\gamma - |E(S)|/|S|)$
- 引理:  $\gamma < D^*$  当且仅当 最大流  $< mn$
- 推论: 假设  $D'$  是第二大的子图密度, 由  $D' \leq \gamma < D^*$  最大流问题对应的最小割集  $\{s\} \cup X$  的  $X$  组成最大密度子图
- 算法
  - ▶ 置  $[l, u]$  初值为  $(0, m]$ 。当  $\gamma = (u+l)/2$  的最大流为  $mn$  时, 置  $u = (u+l)/2$ ; 否则, 置  $l = (u+l)/2$ 。
  - ▶  $u - l < 1/n^2$  时终止
- 引理: 当  $u - l < 1/n^2$  时, 令  $\gamma = l$ , 有  $D' \leq \gamma < D^*$ 
  - ▶ 两个不同子图的密度差异  $\Delta = |m_1/n_1 - m_2/n_2| = |(m_1n_2 - m_2n_1)/(n_1n_2)| \geq 1/n^2$
- 定理: 上述算法调用  $O(\log n)$  次最大流求解器即能求出最大密度子图

# 整数流定理

- 如果网络流问题的流入、流出、边容量都是整数，目标函数也是线性函数，那么对应的线性规划所有的基本可行解都是整数解。因此，单纯形法能求出其中一个整数最优解
- 直观理由：单纯形法表格中，约束矩阵相关的值始终在  $\{-1, 0, +1\}$  范围内
- 一般地，约束矩阵是 Totally Unimodular Matrix 的线性规划存在整数最优解（略）

# 本讲小结

- 二部图的最大匹配
- 赋权二部图的匹配
- 带需求的流通
- 边不相交和点不相交路径
- 项目选择（最大权闭包）
- 经典图像分割算法
- 最小密度子图
  
- 整数流定理（略）