算法设计
——网络流

陈卫东

chenwd@scnu.edu.cn

华南师范大学 计算机学院

2021-10

网络流技术

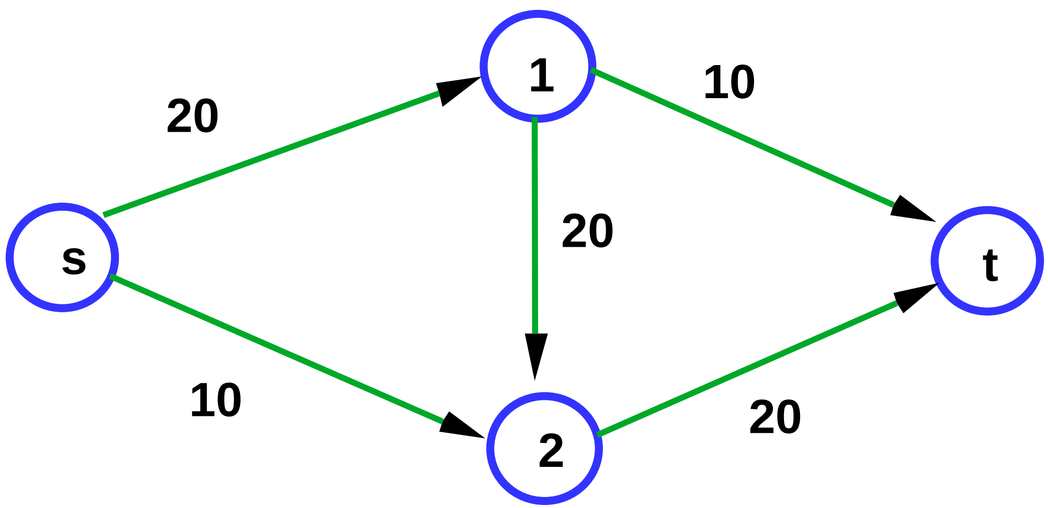
- 网络与网络流
- 典型算法
- 应用举例
- 问题的扩展
- 其他应用实例（见教材）

网络与网络流

- **网络**

　　给定有向图$G=(V,E)$，对于图G的每一条边$(u,v)\in E$，都有一个非负的容量$c(u,v)\geq 0$。如果$(u,v)\notin E$，则$c(u,v)=0$。图中有两个特殊的顶点s和t，其中顶点s只有出边，t只有入边，且图中至少存在一条从s到t的路径。这样的有向图我们称为**流网络**（简称为**网络**），记作(G,s,t,c)，其中顶点s称为网络的**源点**，t称为网络的**汇点**，c称为网络的**容量函数**。

网络与网络流



流网络示例

网络与网络流

- **网络流**

对于网络(G,s,t,c)，若实值函数f: $V\times V\to R$ 满足下列性质:

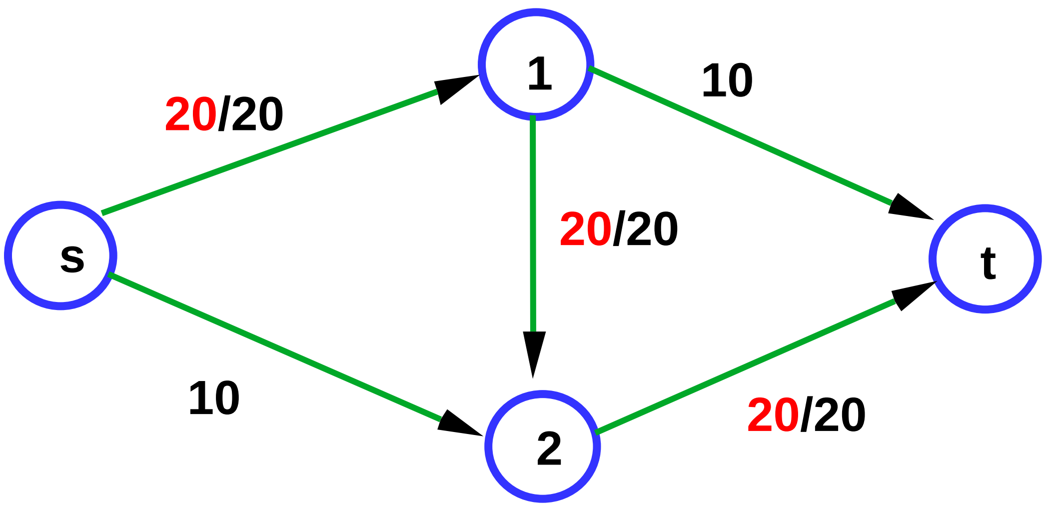
- 容量约束性质，即任取$u,v\in V$，有$f(u,v)\le c(u,v)$;
- 反对称性质，即任取$u,v\in V$，有$f(u,v)=-f(v,u)$;
- 流守恒性质，即任取$w\in V-\{s,t\}$，有 $\sum_{(u,w)\in E} f(u,w)=\sum_{(w,v)\in E} f(w,v)$

则称f为网络的流，其中$f(u,v)$表示从顶点u到顶点v的流量。

流 f 的大小定义如下: $|f|=\sum_{v\in V} f(s,v)$

$|f|$ 表示了从源点流出的流量大小。

网络与网络流



网络流示例

网络与网络流

- **最大流问题**

在网络(G,s,t,c)中给每条边(u,v) 赋予一个流值 $f(u,v)$ (flow) 。最大流问题的数学模型描述如下:

$$
\begin{aligned}
&(1)\ \max \sum_{v\in V} f(s,v) \\
&s.t. \\
&(2)\ f(u,v) \le c(u,v) \quad u,v\in V \\
&(3)\ f(u,v) = -f(v,u) \quad u,v\in V \\
&(4) \sum_{(u,w)\in E} f(u,w) = \sum_{(w,v)\in E} f(w,v), \quad w\in V-\{s,t\}
\end{aligned}
$$

容量约束性质

反对称性质

流守恒性质

$$(1)\ \max \sum_{v \in V} f(s, v)$$

$$s.t.$$

$$(2)\ f(u, v) \le c(u, v) \quad u, v \in V$$

$$(3)\ f(u, v) = -f(v, u) \quad u, v \in V$$

$$(4) \sum_{(u,w) \in E} f(u, w) = \sum_{(w,v) \in E} f(w, v), \quad w \in V - \{s, t\}$$

其中:

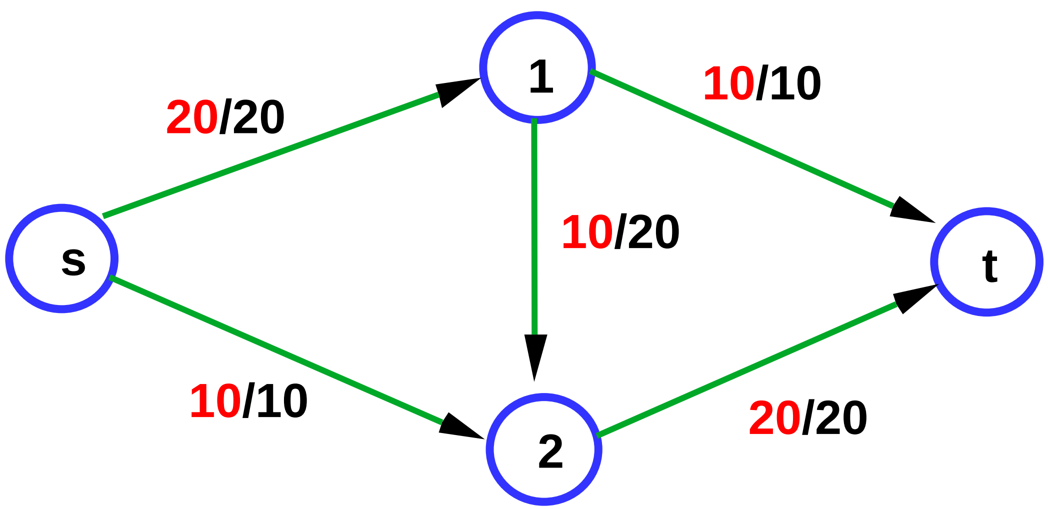
(1) 表示要求从源点流出的流量要最大。

(2) 容量限制条件：边的流量不超过边上的容量。

(3) 规定反向边的流量为正向边的流量的相反数。

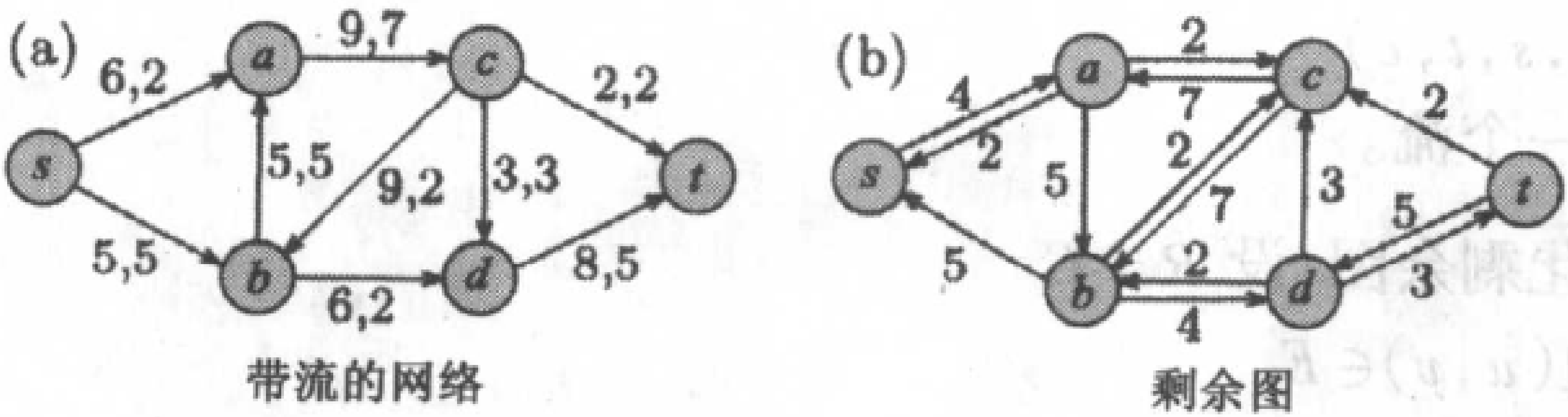
(4)流量守恒条件, 表示除源点s和汇点t外每个点的流入等于流出。

网络与网络流



最大流示例

▪ 带流网络的剩余图



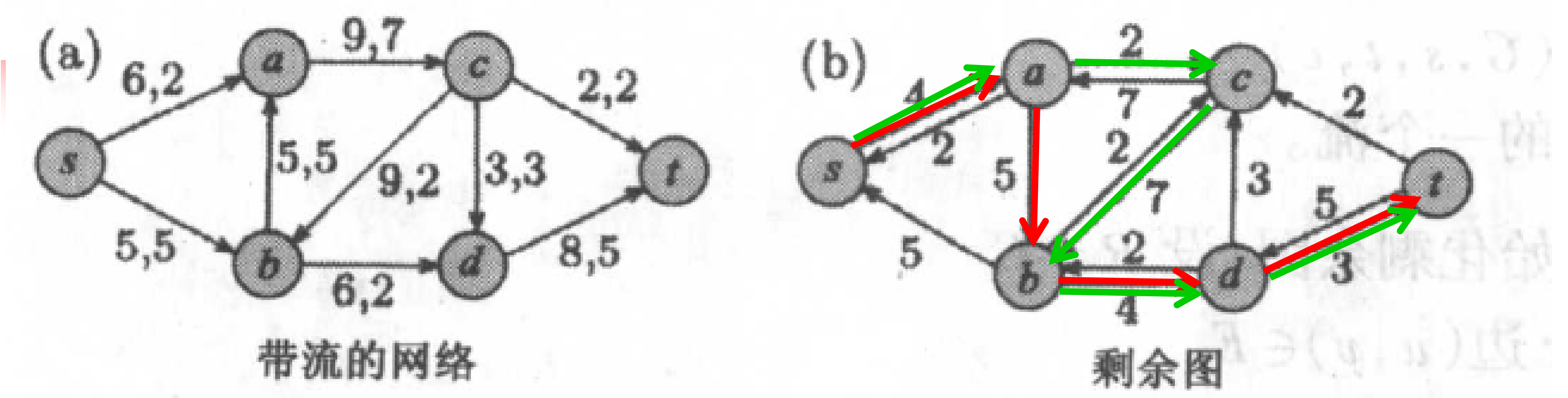
具有流的网络和它的剩余图

定义 给出一个 G 上的流 f 及它的容量函数 c，顶点对上 f 的剩余容量函数定义如下：对于每一对顶点 $u, v \in V$，$r(u,v) = c(u,v) - f(u,v)$，流 f 的剩余图是一个具有容量 r 的有向图 $R = (V, E_f)$，其中

$$E_f = \{(u,v) \mid r(u,v) > 0\}$$

注：对于一条边(v,u)来说，如果$c(v,u)=0$，则原图中不允许从v到u有流，但剩余图中可能从v到u有流：如果$f(v, u)>0$且$c(v, u)=0$，则$r(v, u)= c(v, u)-f(v, u)=f(u, v)>0$.

■ 剩余图中增广路径



剩余图中增广路径示意图

- 红色增广路径的瓶颈容量为3，沿着该路径可增加3个流量。
- 绿色增广路径的容量为2，沿着该路径可增加2个流量。

在剩余图中如何找增广路径?

——最短路径增值法、最大容量增值法、…

（可通过图的遍历方法、Dijkstra算法来实现）

网络与网络流

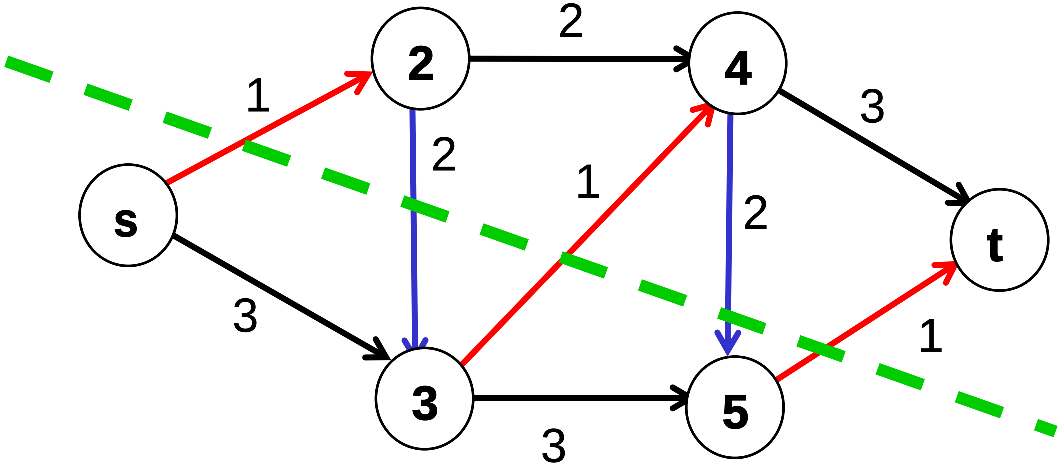
■ 最小割

网络(G,s,t,c)的割（cut）将点集V划分为和S和T两个部分使得源点$s\in S$且汇点$t\in T$，记作(S,T)。

割(S,T)的容量（capacity）： $$c(S,T)=\sum_{u\in S,v\in T}c(u,v)$$

最小割（minimum cut）： 网络中容量最小的割。

最小割示例



S={s,3,5}， T={2,4,t}

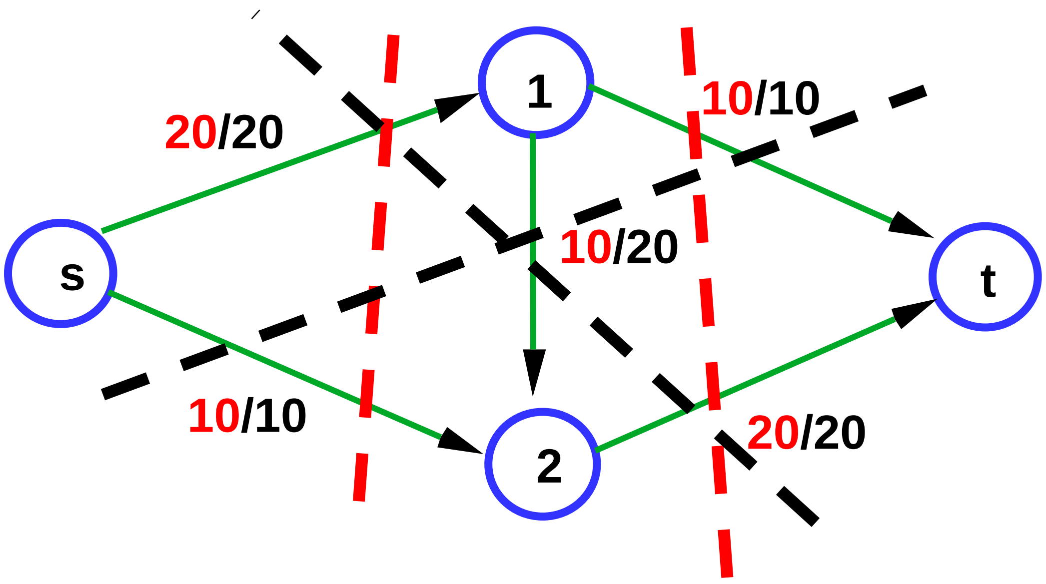
割 (S,T)的容量为 $c(S,T)=3$

割 (T,S)的容量为 $c(T,S)=4$

【最大流最小割定理】

网络$G=(V,E,c)$ 中一个流为 f， 一个割为$\{S,T\}$， 则

f是最大流且$\{S,T\}$是最小割 $\Leftrightarrow$ $c(S,T) = |f|$。



注: 红色线标注的割是最小割，等于最大流

定理 （最大流最小割定理）设(G,s,t,c)为一个网络，f为G中的流，下面的三个语句是等价的。

(a) 存在一个割$\{S,T\}$，$c(S,T)=|f|$；

(b) f是G中的最大流；

(c) 对f不存在增广路径。

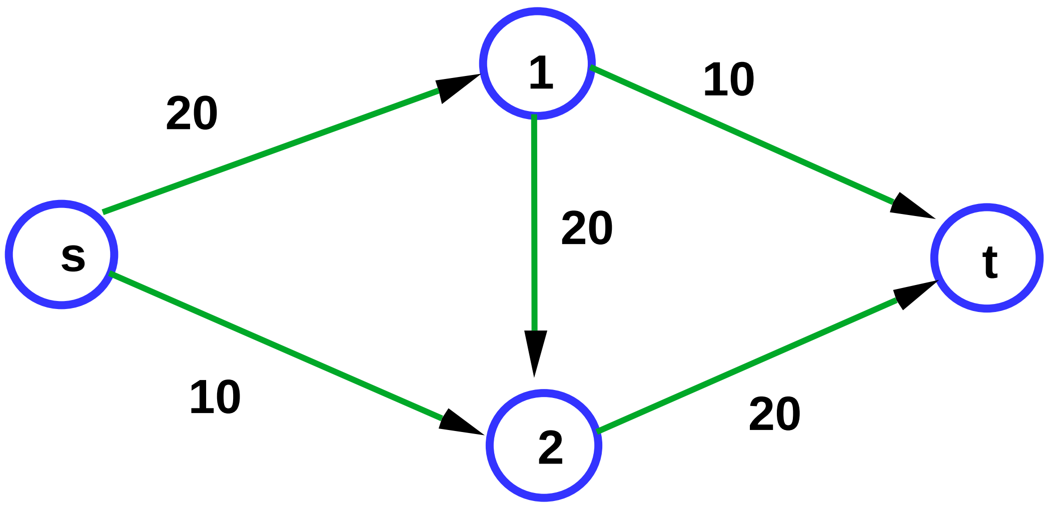
证明：(a)→(b)：因为对任意割$\{A,B\}$，均有$|f|\leq c(A,B)$，而$c(S,T)=|f|$蕴含着f是最大流，同时说明 $c(S,T)$ 是最小割，即最大流等于最小割。

(b)→(c)：如果在G中有一条增广路径p，那么$|f|$可以通过沿着p的流而增加，也就是说，f不是最大流。

(c)→(a)：假定没有f的增广路径。设S是在剩余图R中从s开始能到达的顶点集合，令$T=V-S$，则R中不包含从S到T的边，这样G中所有从S到T的边是饱和的，就有$c(S,T)=|f|$。

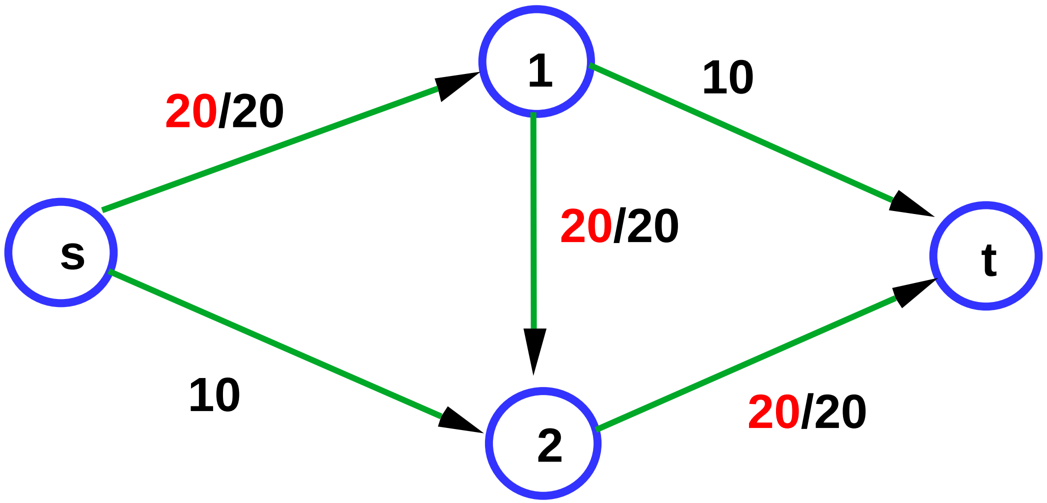
上述(c)→(a)的证明给出一个在给定网络中找到最小割的算法。

问题：如何求网路中的最大流?



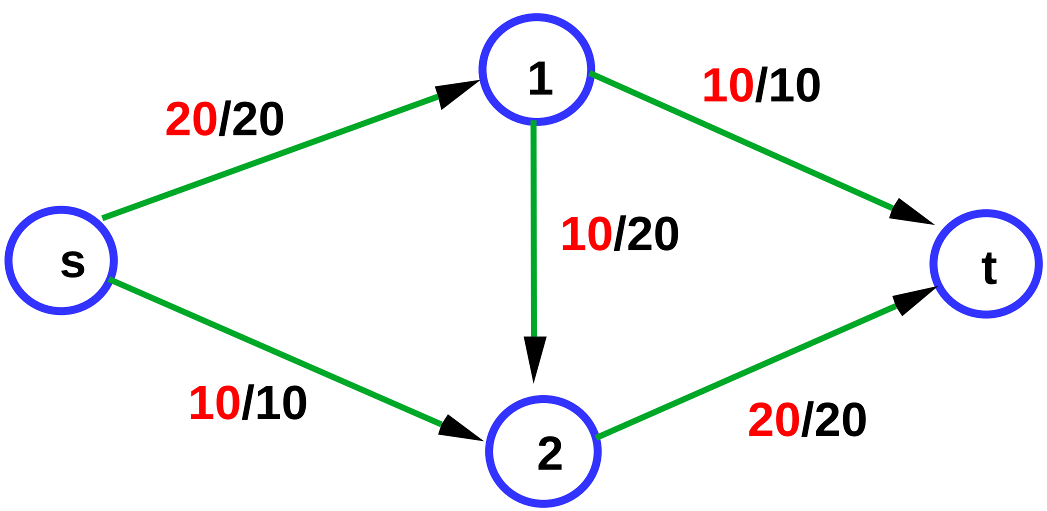
方法：迭代改进方法

（在当前流 f 的剩余图中寻找增广路径来扩充 f 得到更大流 f'）



初始流

如何求得？



一个最大流

典型算法

- FordFulkerson算法 ——$O(m|f^*|)$
- 最大容量增值算法 ——$O(mn^2\log c^*)$
- 最短路径增值算法
  - 简单最短路径增值算法(MPLA) ——$O(nm^2)$
  - Dinic 算法 ——$O(mn^2)$
  - MPM 算法 —— $O(n^3)$

思路： $f_0 \to f_1 \to f_2 \to f_3 \to \dots \to f^*$

$$|f_0| < |f_1| < |f_2| < |f_3| < \dots < |f^*|$$

FordFulkerson算法

通过找一条任意的增广路径来扩充流，可使用图的遍历方法实现。

算法 FORD-FULKERSON

输入： 网络(G,s,t,c)。

输出： G 中的一个流。

```
初始化剩余图，设 R = G
for 边(u,v)∈E
    f(u,v) ← 0
end for
while 在 R 中有一条增广路径 p = s,…,t
    设 Δ 为 p 的瓶颈容量
    for p 中的每条边(u,v)
        f(u,v) ← f(u,v) + Δ
    end for
    更新剩余图 R
end while
```

在m条边的网络中,若容量均为整数, 则算法时间复杂度为$O(m|f^*|)$, 其中$|f^*|$是最大流值.

FordFulkerson算法

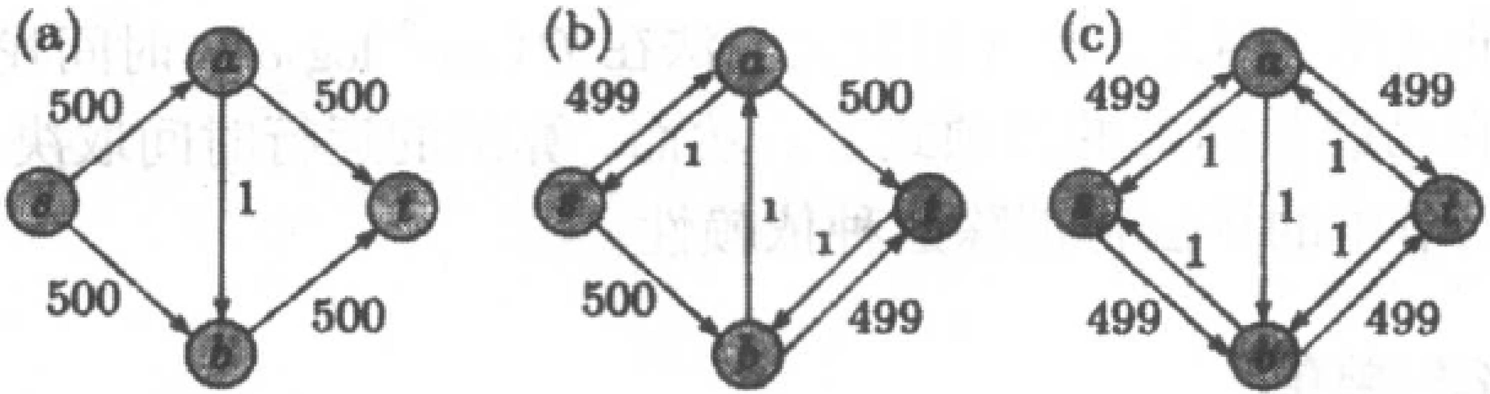


图　　在一个图上 Ford-Fulkerson 方法执行得差的例子

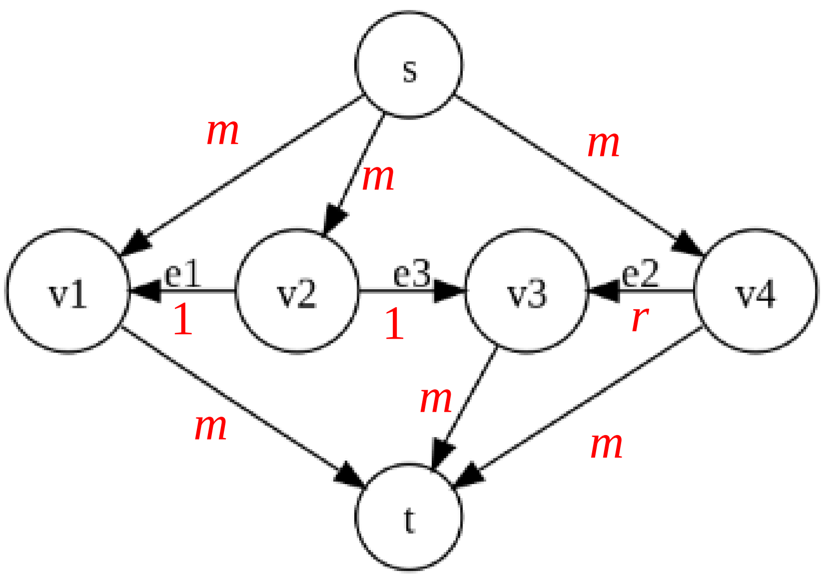
如果容量是无理数，则 Ford-Fulkerson 方法可能不会终止。然而如果流收敛，它可能收敛到一个不一定最大的值。如果容量是整数，因为每次扩张流至少增加 1，因此这种方法至多 $|f^*|$ 步就可以计算出最大流 f^* 了。由于每条增广路径可以在 $O(m)$ 时间中找到(比如，用深度优先搜索)，此方法的整个时间复杂性是 $O(m|f^*|)$(当输入的容量都是整数时)。注意到时间复杂性取决于输入值。作为一个例子，考虑图　　(a)所示的网络，如果算法中交替选择增广路径 s,a,b,t 和 s,b,a,t，则增值步数是 1000。前两个剩余图如图　　(b)和图　　(c)所示。

FordFulkerson算法

Ford-Fulkerson算法不会终止的例子。

(参见Wikipedia, the free encyclopedia： **Ford–Fulkerson algorithm**)

下面网络中边e_1, e_2, e_3容量分别为 1, r,1, 其它边的容量为 m>2. 注意r满足$r+r^2=1$，即 $r=(\sqrt{5}-1)/2$，根据下表来选用增广路径(augmenting path)，其中$p_1=\{s,v_4,v_3,v_2,v_1,t\}$，$p_2=\{s,v_2,v_3,v_4,t\}$，$p_3=\{s,v_1,v_2,v_3, t\}$.

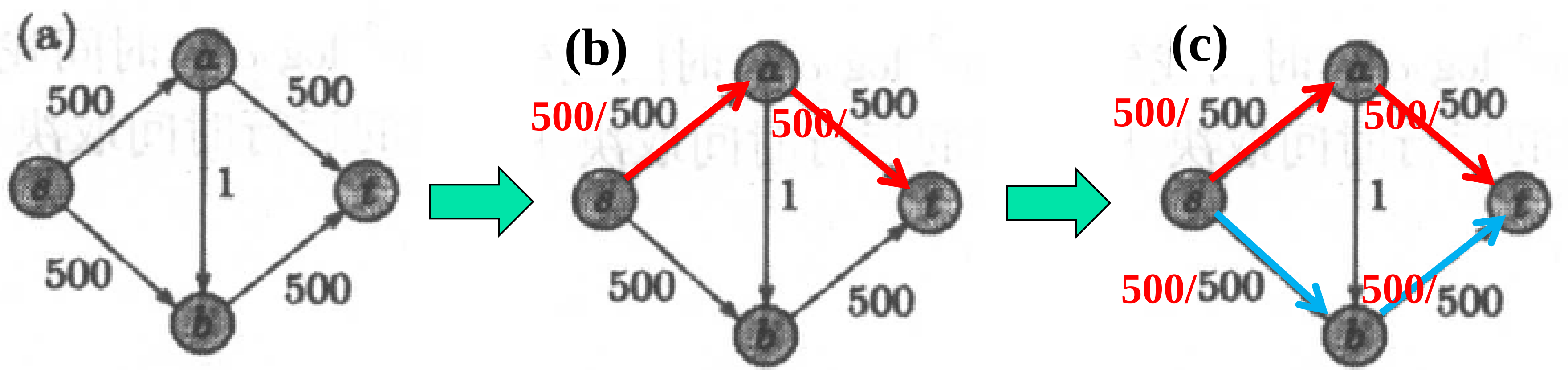


Step	Augmenting path	Sent flow	Residual capacities		
			e_1	e_2	e_3
0			$r^0=1$	r	1
1	$\{s,v_2,v_3,t\}$	1	r^0	r^1	0
2	p_1	r^1	r^2	0	r^1
3	p_2	r^1	r^2	r^1	0
4	p_1	r^2	0	r^3	r^2
5	p_3	r^2	r^2	r^3	0

由表可见，n次重复执行Step2-5后剩余图中e_1,e_2,e_3容量为 r^n，r^{n+1}，0. 在Step5后总流量为 $1+2(r+r^2)$, 因此反复执行Step2-5总流量将收敛到 $1+2(r+r^2+r^3+\ldots) = 3+2r$, 而最大流为 $2m+1$. 在该实例上算法永远不停止.

最大容量增值算法（MCA）

通过找一条最大瓶颈的增广路径来扩充流，可使用Dijkstra算法实现。



定理 如果边容量都是整数，那么 MCA 在 $O(m \log c^*)$ 增值步内构造一个最大流，其中 c^* 是最大边容量。

应用一个修改过的单源最短路径问题的 Dijkstra 算法，可以在 $O(n^2)$ 时间内找到一条最大瓶颈容量路径。因此，MCA 启发式方法在 $O(mn^2 \log c^*)$ 时间找到一个最大流。

时间复杂性现在已经是输入的多项式了。然而，算法的运行时间取决于输入的值这将不令人满意

简单最短路径增值算法 (MPLA)

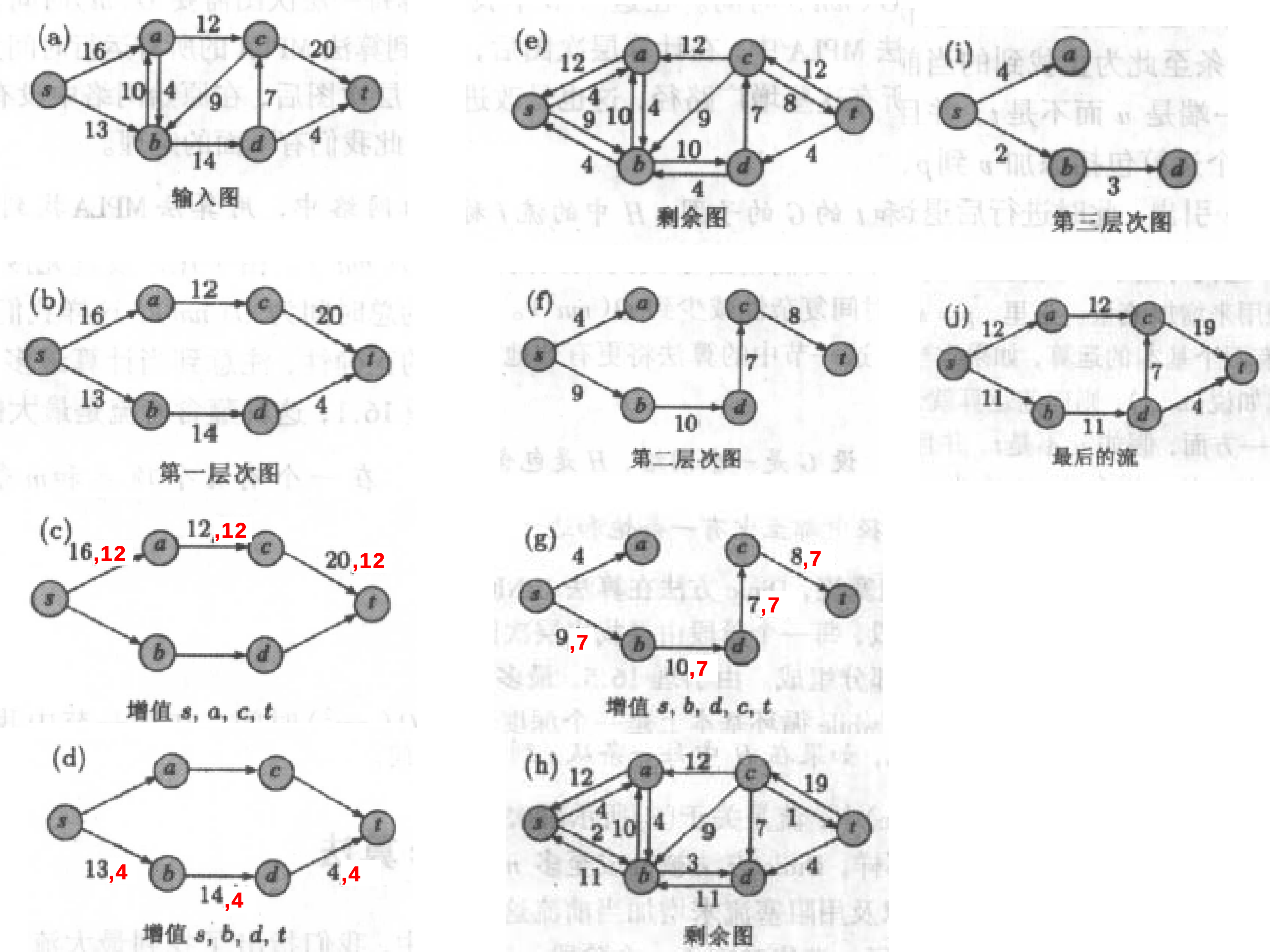
算法　　MPLA

输入：网络(G, s, t, c)。

输出：G 中的最大流。

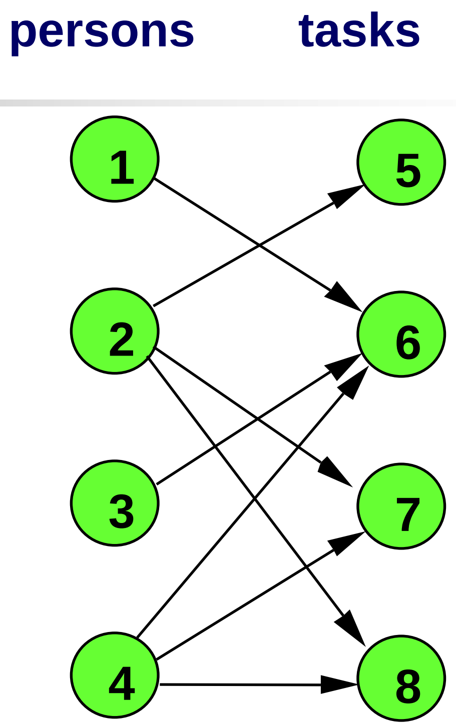
```
for 每条边(u, v)∈E
    f(u, v) ← 0
end for
初始化剩余图，设 R = G
查找 R 的层次图 L
while t 为 L 中的顶点
    while t 在 L 中能从 s 到达
        设 p 为 L 中从 s 到 t 的一条路径
        设 Δ 为 p 的瓶颈容量
        用 Δ 增值当前流 f
        沿着路径 p 更新 L 和 R
    end while
    用剩余图 R 计算新的层次图 L
end while
```

定理　在一个n个点和m条边的网络中，算法MPLA找到最大流需要时间$O(nm^2)$.


(a)
16
12
20
10
4
9
7
13
14
4
输入图
(e)
12
12
12
4
9
10
4
9
7
8
10
4
4
4
剩余图
(i)
4
2
3
第三层次图
(b)
16
12
20
13
14
4
第一层次图
(f)
4
8
7
9
10
第二层次图
(j)
12
12
19
7
11
11
4
最后的流
(c)
16,12
12,12
20,12
增值 s, a, c, t
(g)
4
8,7
7,7
9,7
10,7
增值 s, b, d, c, t
(d)
13,4
14,4
4,4
增值 s, b, d, t
(h)
12
12
19
4
2
10
4
9
7
1
3
11
11
4
剩余图

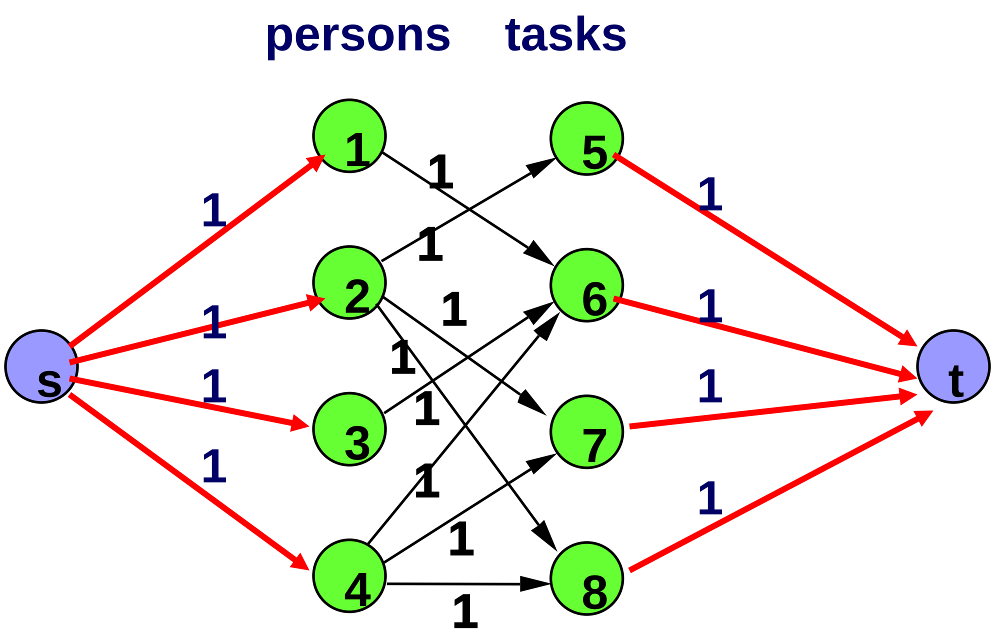
应用举例

- 匹配问题



Is there a way of assigning persons to tasks so that each person is assigned a task, and each task has a person assigned to it?

应用举例



Does the maximum flow from s to t have 4 units?

应用举例

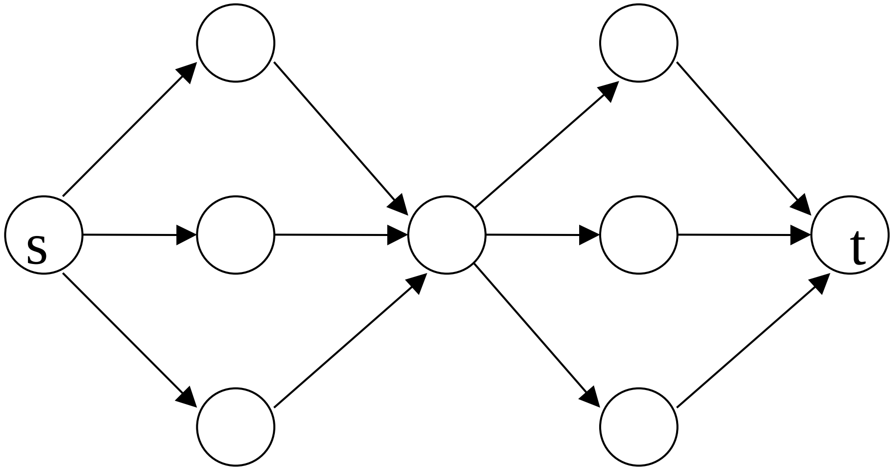
- ## 边不相交路径问题

互连网络中某些链或节点难免会失效，因此需要设计容错路由算法。网络可用图来建模，有一种容错路由方案就是构造图中两点间最多数目的边不相交路径。

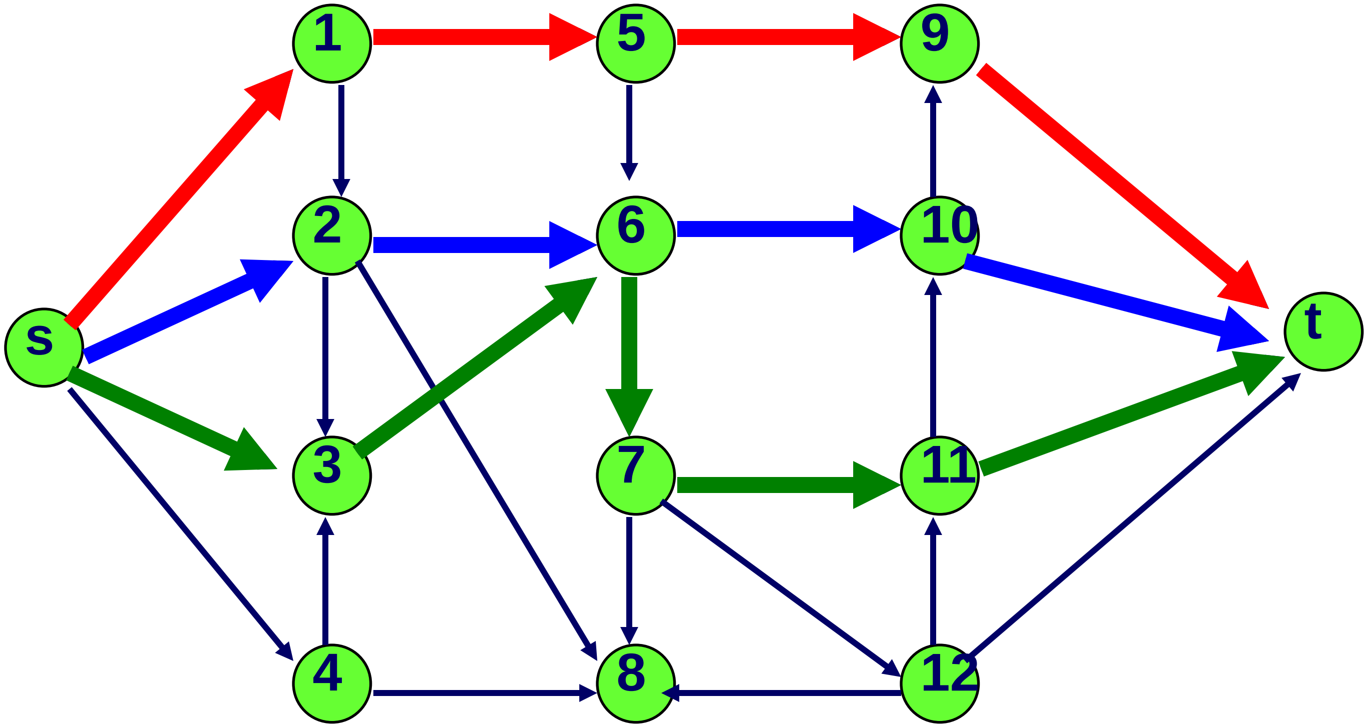
给定一个n点有向图，其邻接矩阵为$A[1\ldots n][1\ldots n]$。请设计一个算法能求得该图中从第1个结点到第n个结点间的最多条数目的边不相交路径。

Network Reliability

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



There are 3 arc-disjoint s-t paths



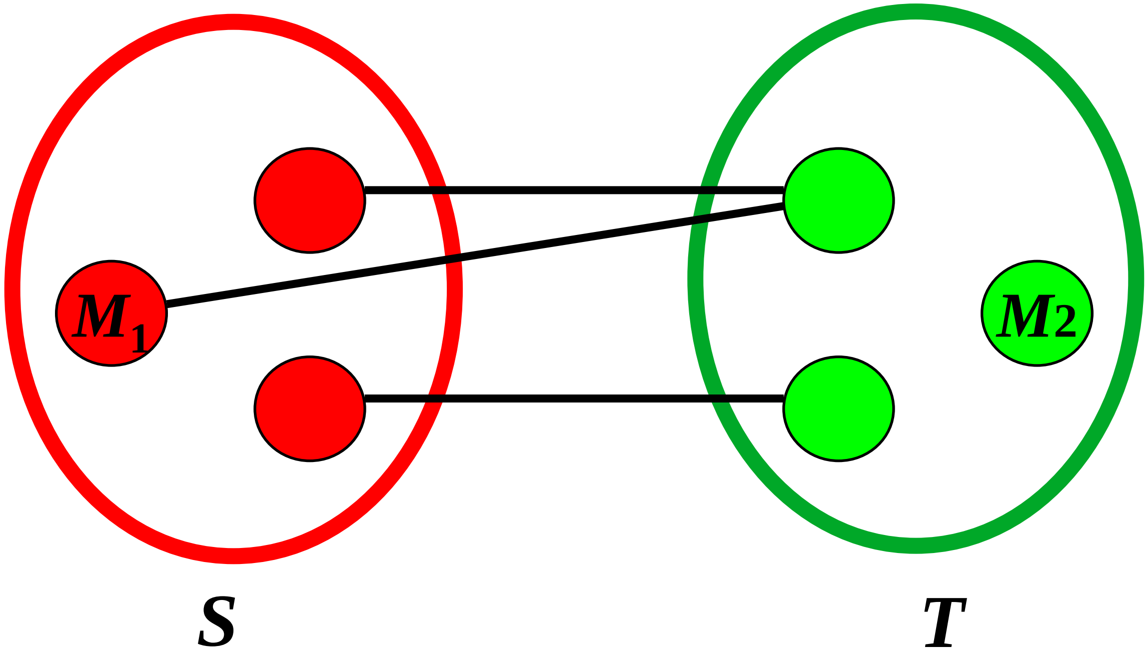
应用举例

- ## 系统划分问题

　　一个软件系统由多个模块组成，模块与模块之间往往有调用关系。为了分析软件系统，用无向图给系统建模：每个模块看作一个点，模块与模块之间有调用则相应点之间连边。假设M_1和M_2是系统中的两个模块，现在需要将整个系统分解为两个子系统使得M_1在一子系统、M_2在另一子系统且满足位于不同子系统的模块之间的调用次数最少。

　　给定一个有n个点的无向图（即软件系统的模块调用图），其邻接矩阵为$A[1...n][1...n]$。请设计一个算法将该图划分为两个子图分别包含节点1和节点n，且这两子图之间的交互边数最少。

应用举例

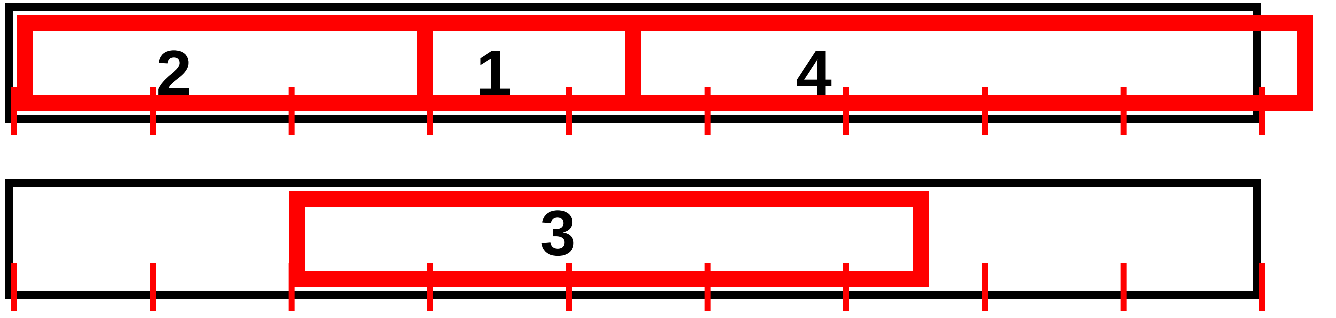


应用举例

- **Scheduling on Uniform Parallel Machines (并行机调度)**

Job(j)	1	2	3	4
Processing Time	1.5	3	4.5	5
Release Time	2	0	2	4
Due Date	5	4	7	9

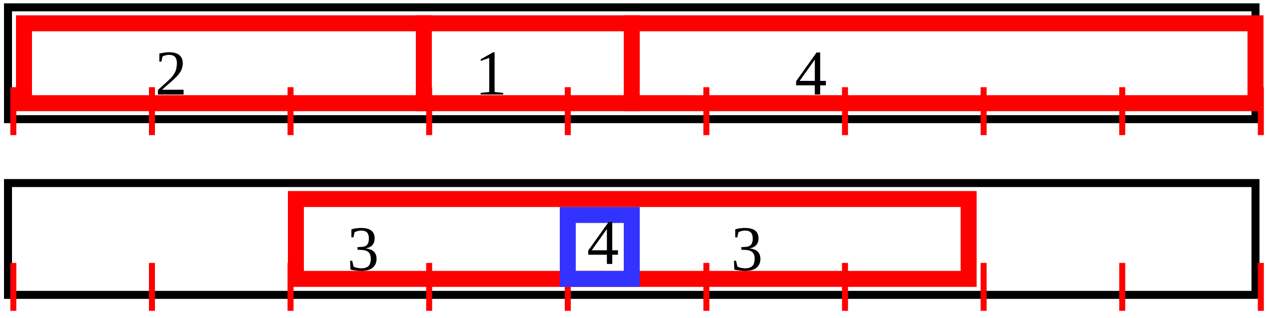
Suppose there are 2 parallel machines



No schedule is possible unless preemption is allowed

Job(j)	1	2	3	4
Processing Time	1.5	3	4.5	5
Release Time	2	0	2	4
Due Date	5	4	7	9

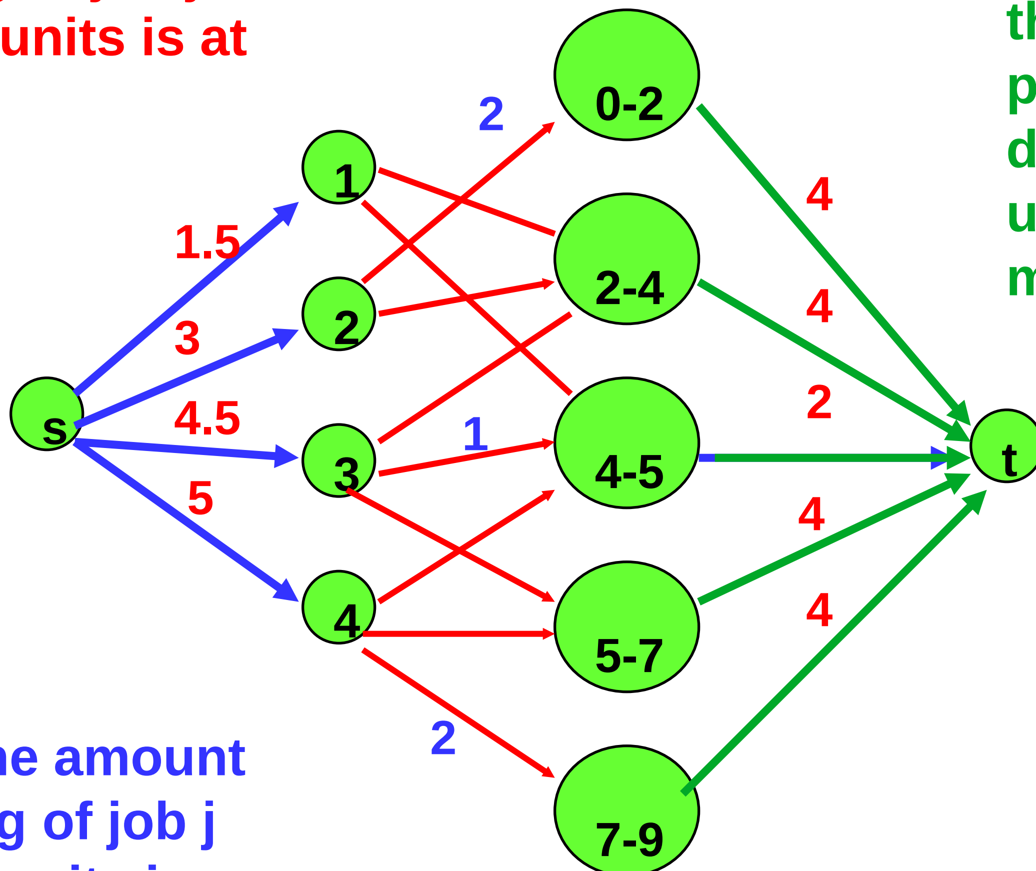
Suppose there are 2 parallel machines



A feasible schedule with preemption allowed

Transformation into a maximum flow problem

red arcs: the amount of processing of job j during t time units is at most t.

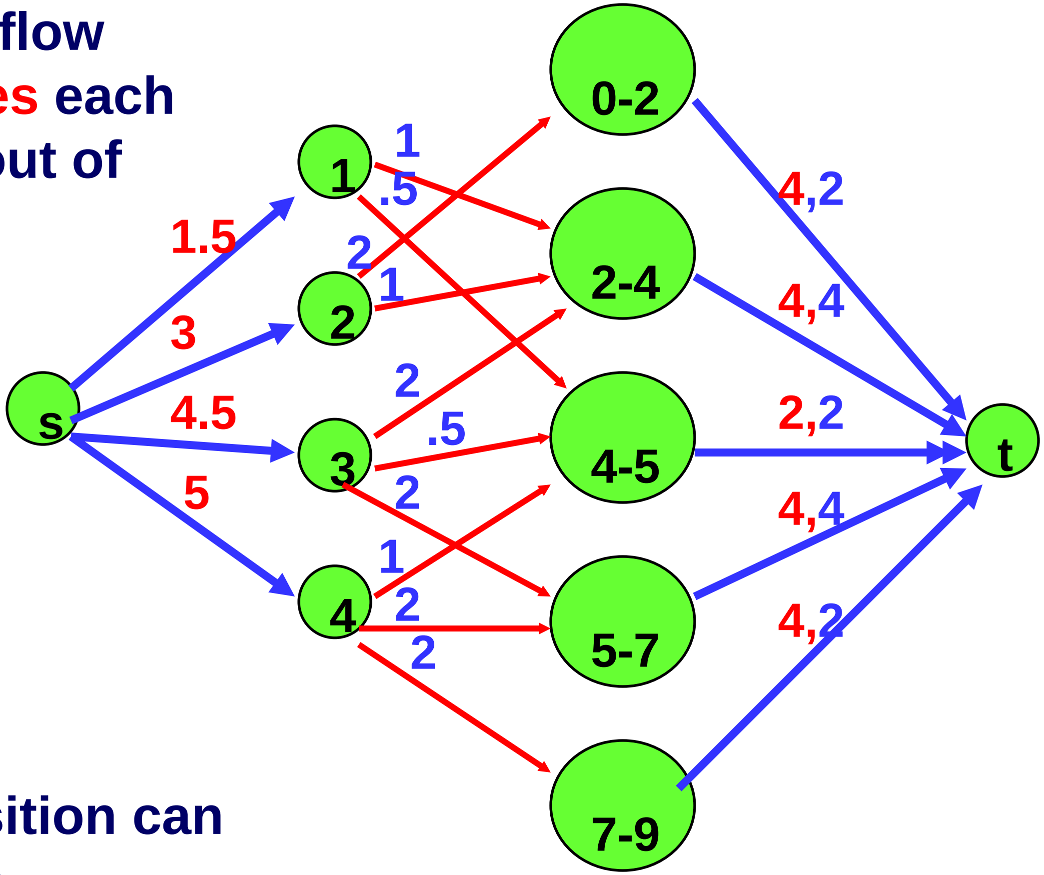


green arcs: the amount of processing during t time units is at most Mt.

blue arcs: the amount of processing of job j over all time units is p_j.

A maximum flow

There is a feasible schedule if there is a maximum flow that saturates each of the arcs out of s.



Flow decomposition can be used to transform flows into schedules

问题的扩展

- 最小费用流问题
- 带有上下界的流问题

问题的扩展

- **最小费用流问题(Minimum Cost Flow)**

在网络$G=(V,E)$上除给定容量函数c外，还增加如下的非负权函数w：$V\times V\rightarrow R$。任给$(u,v)\in E$，对应权值记作$w(u,v)$，表示边的单位流量的费用或成本，即通过该边单位流所需要费用。

一个流 f 的总费用定义为 $w(f)=\sum_{(u,v)\in E} w(u,v)f(u,v)$

如果流量为$|f|$的所有流中f具有最小的费用，则称流 f为最小费用流。

问题的扩展

- 最小费用流问题(Minimum Cost Flow)

最小费用流问题通常有两种形式：

- 给定流量值d，在网络中计算流量值为d的最小费用流。
- 不给定流量值，计算流量值最大的最小费用流，此问题常称最小费用最大流问题。

注：前面的最大流问题实际上是最小费用流问题的特例。

问题的扩展

- **最小费用流问题(Minimum Cost Flow)**

算法1——消除回路算法（Canceling Cycle）

- **时间复杂度:**

  给定最大流 f, **Canceling Cycle**算法可在时间 $O(|V||E|w')$ 内找到流量值为 $|f|$ 的最小费用流。其中w'是初始最大流 f 所具有的费用。

问题的扩展

算法2——最小费用路算法（Minimum Cost Path）

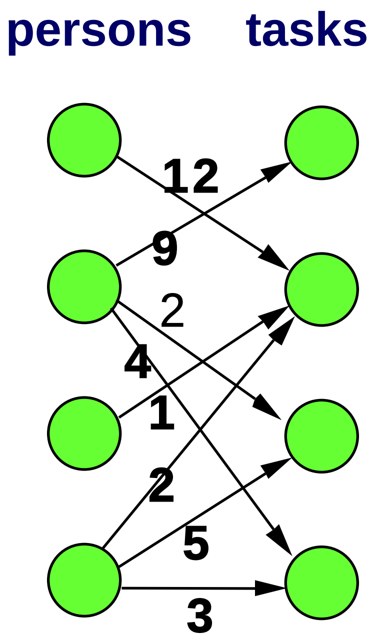
- **时间复杂度:**

  Minimum Cost Path算法可在时间 $O(|f^*||V||E|)$ 内找到最小费用流的最大流 f^*。

问题的扩展

- **最小费用流问题(Minimum Cost Flow)**

【例】指派问题



问题的扩展

- **有上下界的流问题**

给定一个加权的有向图 $G=(V, E, b, c)$，满足：

(1)容量限制条件： $b(u,v) \le f(u,v) \le c(u,v)$

(2)流量平衡条件： $\sum_{(u,w)\in E} f(u,w) = \sum_{(w,v)\in E} f(w,v)$

求解思路： 通过添加点、边，转化为最大流问题求解

练习题

- 编程实现求解最大流问题的最短路径增广算法、Dinic算法以及MPM算法，并用实验分析方法比较三种算法的效率

- XOJ题目：1089

- POJ 题目：1087,1149,1273,1274,1325,
  1459,1637,1698,1719,2195,
  2239,2516,2771,3020,3041.