图与网络分析第3节
最短有向路、最大流

西安交通大学电信学院系统工程研究所
翟桥柱、吴江

最短有向路：定义

最短路问题的一种特殊形式，可与DP部分做对比

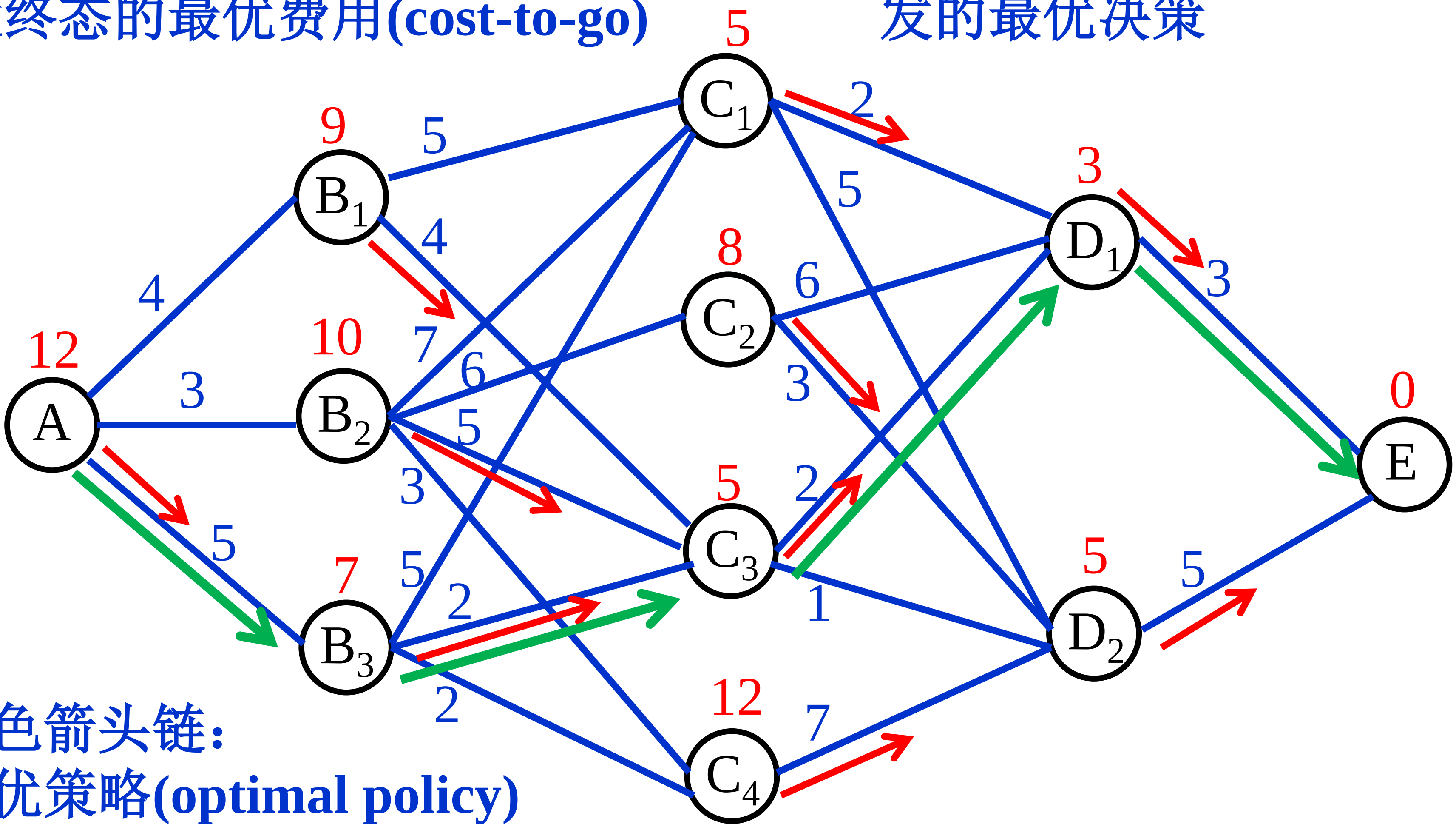
最短有向路： 设$G=(V, A, W)$是有向网络，P是v_i到v_j的一条有向路，则称$W(P)=\sum_{a\in P} w(a)$为路P的权。v_i到v_j的所有有向路中权最小的有向路称为v_i到v_j的最短有向路。

怎样求最短有向路？函数空间迭代法的图论表述

- 假定求v_1到其余所有顶点的最短有向路
- 假定不含权为负或零的(初级)有向回路
- 只考虑简单有向图，$w_{i,j}=+\infty$表示无v_i到v_j的直达弧

红色数字：从每个状态出发到达终态的最优费用(cost-to-go)

红色箭头：从每个状态出发的最优决策



绿色箭头链：

最优策略(optimal policy)

最短有向路：基本方程？函数方程？

设$G=(V,A,W)$是有向网络，$|V|=n$，$|A|=m$

记 u_j 为v_1 到 v_j 的最短有向路权，则由DP最优性原理：

$$\begin{cases} u_1 = 0 \\ u_j = \min\limits_{k \neq j}\left\{u_k + w_{k,j}\right\};\ j = 2,3,\cdots,n \end{cases}$$

最优解必定要经过指向它的弧

该方程表明在确定从1到j的**最短距离**时，需要选择j的最优的**前趋节点**k，使得从1到k的最短距离与边(k,j)的长度相加**之和最小**

定理5.5.1：设有向网络G中不含非正有向回路(权为负或零的回路)，且从v_1 到其余顶点均有有限长度的有向路，则基本方程的解存在且唯一。

最短有向路：基本方程？函数方程？

证明： 解存在性，显然。唯一性？

- 若u_j是解，则其一定对应着从v_1到v_j的一条有向路。
- 归纳法可证u_j是一定是最短路权
- 方程组求解并不容易：“**递归**”定义?**怎样化简？**

$$\begin{cases} u_1 = 0 \\ u_j = \min\limits_{k \neq j} \{u_k + w_{k,j}\};\ j = 2,3,\cdots,n \end{cases}$$

P212定理5.5.2： 化递归为递推的充分条件。

$$\begin{cases} u_1 = 0 \\ u_j = \min\limits_{k < j} \{u_k + w_{k,j}\};\ j = 2,3,\cdots,n \end{cases}$$

定理5.5.1： 设有向网络G中不含非正有向回路(权为负或零的回路)，且从v_1到其余顶点均有有限长度的有向路，则基本方程的解存在且唯一。

最短有向路：基本方程？函数方程？

定理5.5.3： 当有向网络中所有弧的权值均为正时，基本方程的简化形式成立。(距v_1第k+1个最近点的最短有向路中，最后一段弧一定是从前k个最近点中某个顶点发出的)

若要寻找到节点j的最短路，从j开始沿着子图的边退回节点1即可。所有节点的最短路组成了**最短支撑树**。**根节点？**

P212定理5.5.2： 化递归为递推的充分条件。

$$\begin{cases} u_1 = 0 \\ u_j = \min\limits_{k<j} \{u_k + w_{k,j}\};\ j = 2,3,\cdots,n \end{cases}$$

定理5.5.1： 设有向网络G中不含非正有向回路(权为负或零的回路)，且从v_1到其余顶点均有有限长度的有向路，则基本方程的解存在且唯一。

最短有向路：Dijkstra算法(标号法) 1959

设$G=(V,E,W)$是连通的无向网络，$|V|=n$，$|E|=m$

Step 1. 置 $u_1=0, R=\{2,3,\cdots,n\}$

$u_j=w_{1,j}, p_j=1; (2\le j\le n)$

Step 2. 求 $u_k=\min\limits_{j\in R} u_j$ 置 $R=R\setminus\{k\}$

Step 3. 若 $R=\phi$，停；否则

对 $j\in R$, 若 $u_k+w_{k,j}<u_j$ 置

$$u_j=u_k+w_{k,j}, p_j=k$$

转Step2；.

R:尚未确定最短有向路的顶点

u_j:当前已探索到的到v_j的最短有向路权

p_j:指针，当前已探索到的到v_j的最短有向路最后一段弧的尾

算法说明：

- 假定G为简单有向图
- 弧权值全为正
- 三个数据集合：

  R、u_j、p_j
- 三个计算操作：

  找出距离起点最小点；

  更新其它点到达起点距离；

  修正指针
- 计算复杂性$O(n^2)$
- 正确性：基于定理5.5.2和5.5.3

最短有向路：Dijkstra算法(标号法)

例子:P213图5.5.1

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array}
\begin{array}{c} p_j \quad u_j \\ \begin{pmatrix} 1 & 0 \\ 1 & 5 \\ 1 & +\infty \\ 1 & 3 \\ 1 & +\infty \\ 1 & 4 \end{pmatrix} \end{array}
\xrightarrow{k=4}
\begin{array}{c} p_j \quad u_j \\ \begin{pmatrix} 1 & 0 \\ 1 & 5 \\ 4 & 10 \\ 1 & 3 \\ 4 & 8 \\ 1 & 4 \end{pmatrix} \end{array}
\xrightarrow{k=6}
\begin{array}{c} p_j \quad u_j \\ \begin{pmatrix} 1 & 0 \\ 1 & 5 \\ 4 & 10 \\ 1 & 3 \\ 4 & 8 \\ 1 & 4 \end{pmatrix} \end{array}
\xrightarrow{k=2}
\begin{array}{c} p_j \quad u_j \\ \begin{pmatrix} 1 & 0 \\ 1 & 5 \\ 2 & 8 \\ 1 & 3 \\ 4 & 8 \\ 1 & 4 \end{pmatrix} \end{array}
\xrightarrow{k=3}
\begin{array}{c} p_j \quad u_j \\ \begin{pmatrix} 1 & 0 \\ 1 & 5 \\ 2 & 8 \\ 1 & 3 \\ 4 & 8 \\ 1 & 4 \end{pmatrix} \end{array}
\xrightarrow{k=5}
\begin{array}{c} p_j \quad u_j \\ \begin{pmatrix} 1 & 0 \\ 1 & 5 \\ 2 & 8 \\ 1 & 3 \\ 4 & 8 \\ 1 & 4 \end{pmatrix} \end{array}
$$

思考：需n-1次迭代

和函数空间迭代法的对比

即便相邻若干次p, u都不变化，也不能终止，必须迭代n-1次

最短有向路：补充内容

其他情况下的最短路算法:

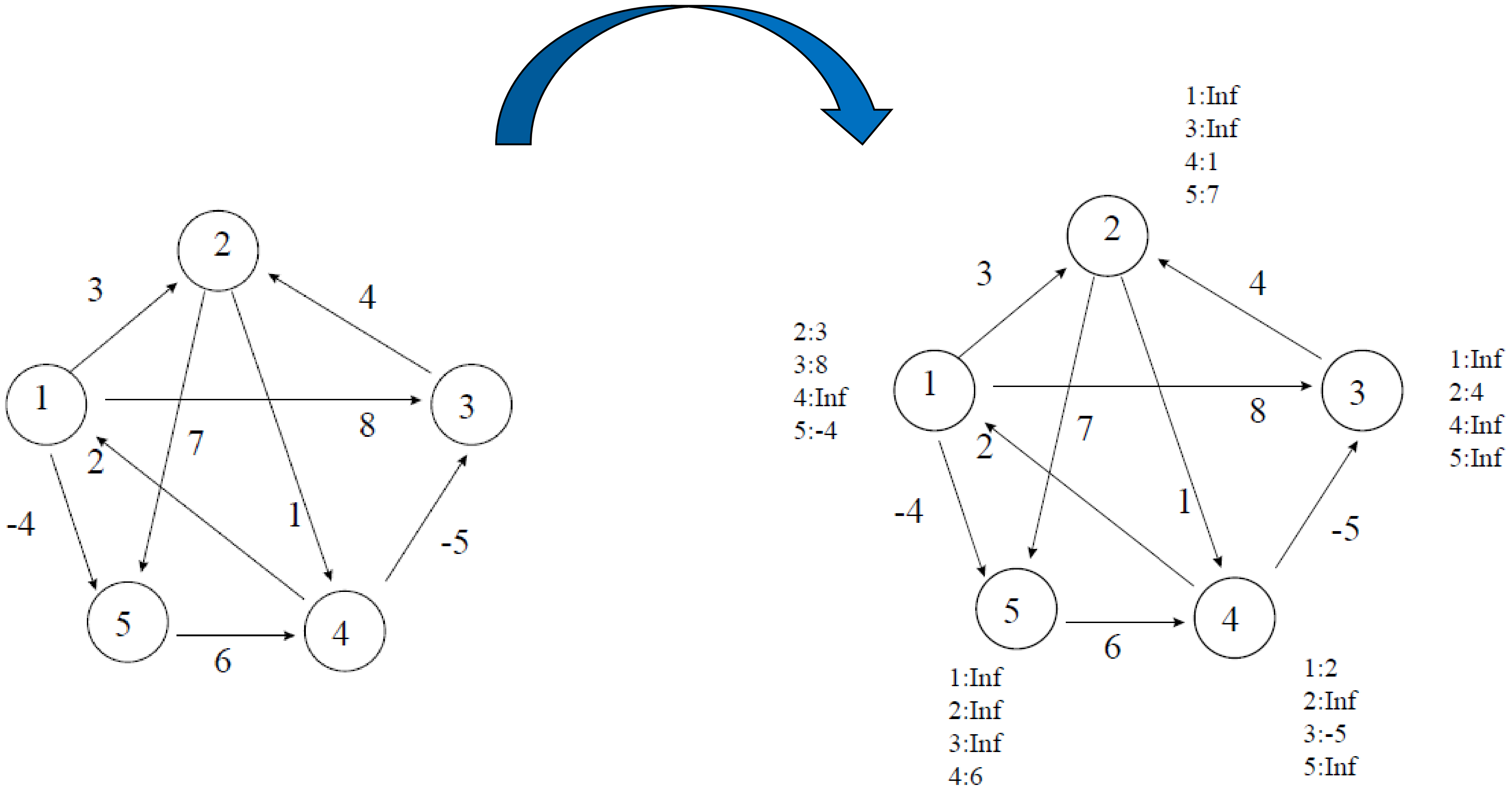
修正标记法：Bellman-Ford, 1958/1956

有负权弧：Ford,1964

任意两顶点之间最短路：Floyd, 1962

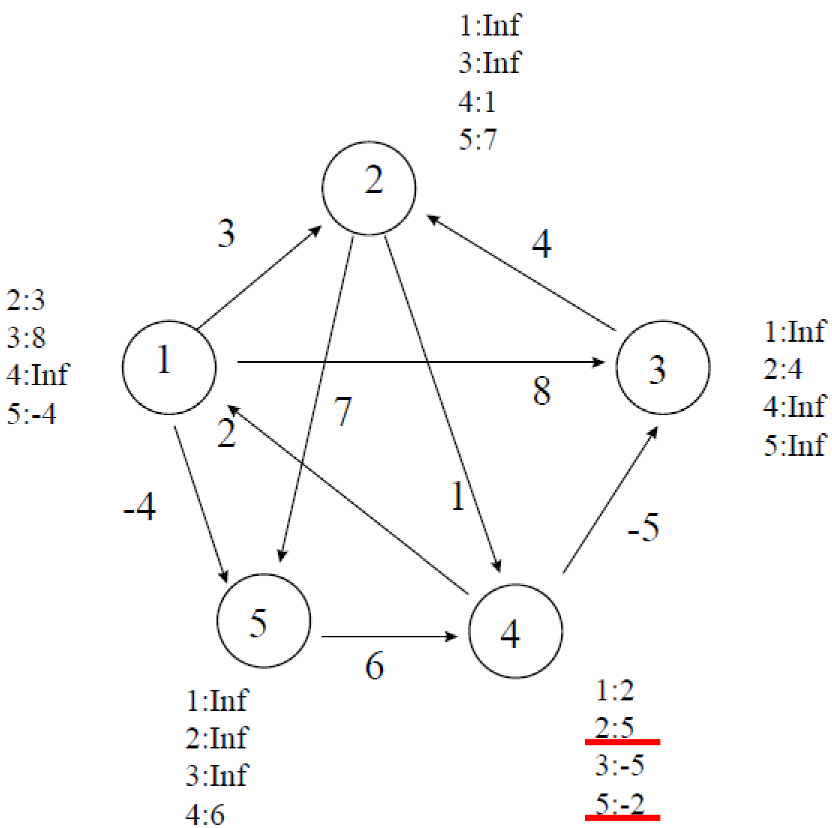
第k条最短路：Dantzig, 1967

***k*=0**

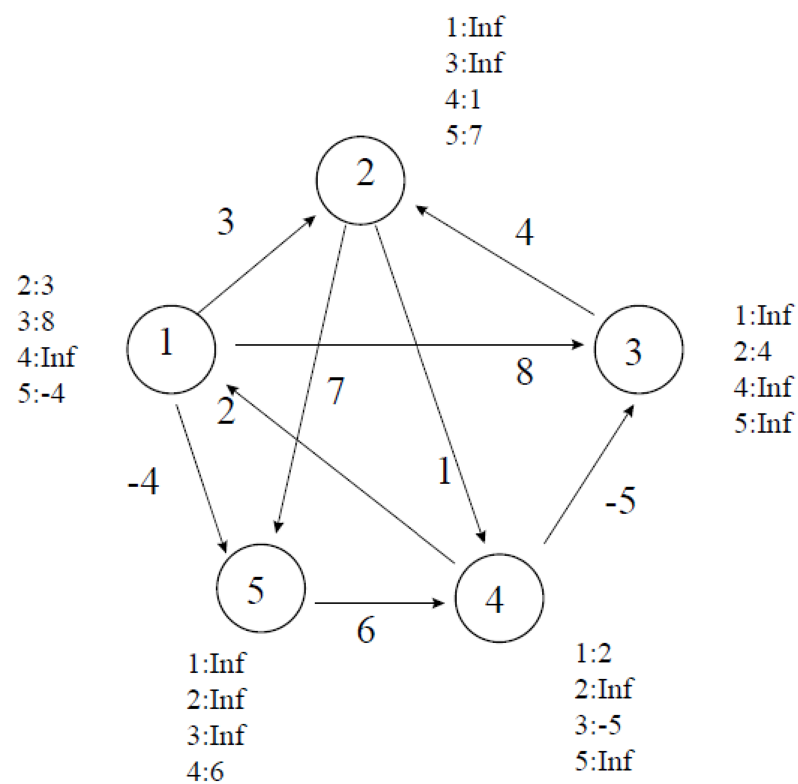

1:Inf
3:Inf
4:1
5:7
2:3
3:8
4:Inf
5:-4
1:Inf
2:4
4:Inf
5:Inf
1:Inf
2:Inf
3:Inf
4:6
1:2
2:Inf
3:-5
5:Inf

最短有向路： Floyd-Warshall-All-Pairs-Shortest

***k*=1**

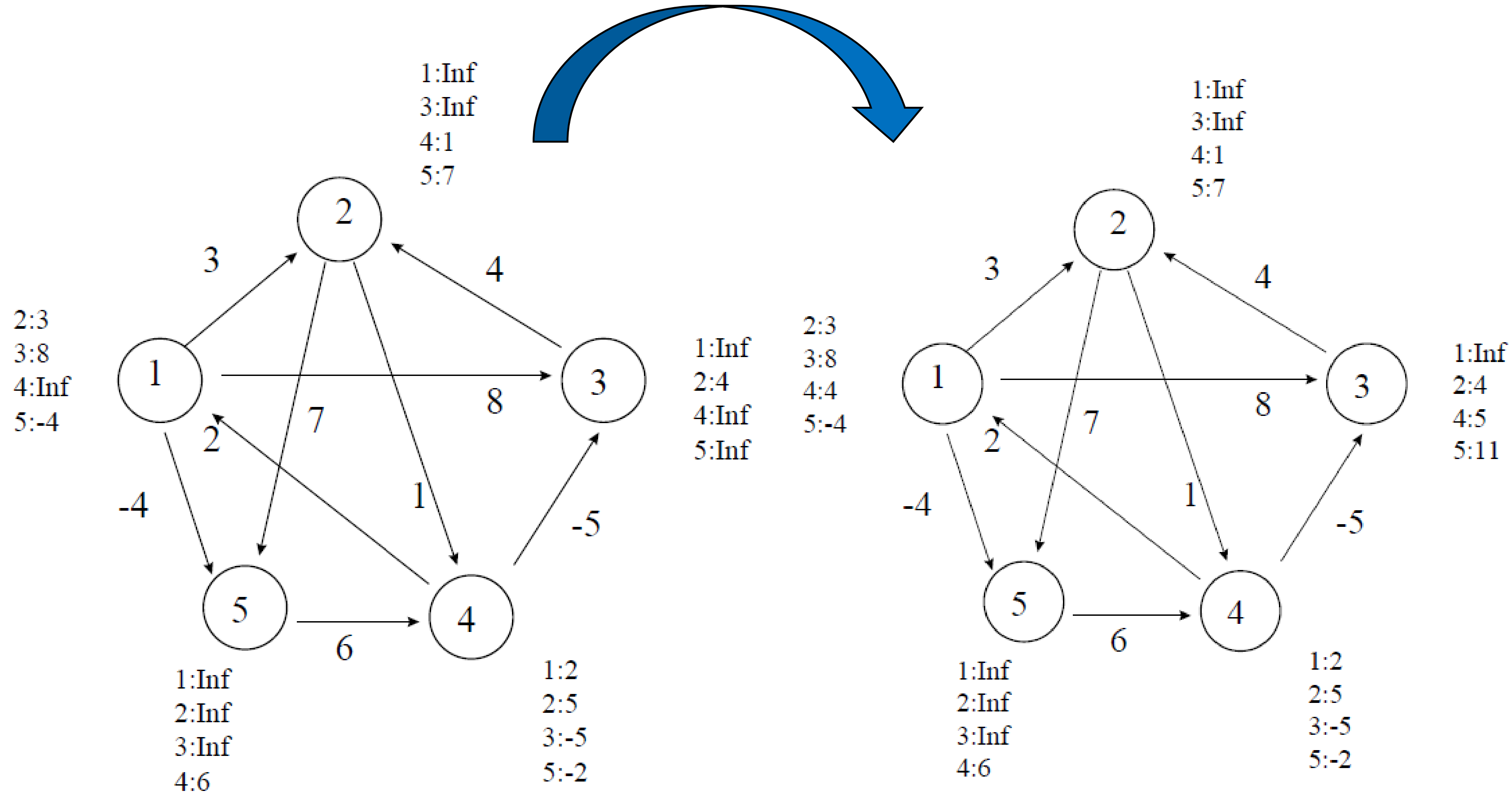


***k*=0**



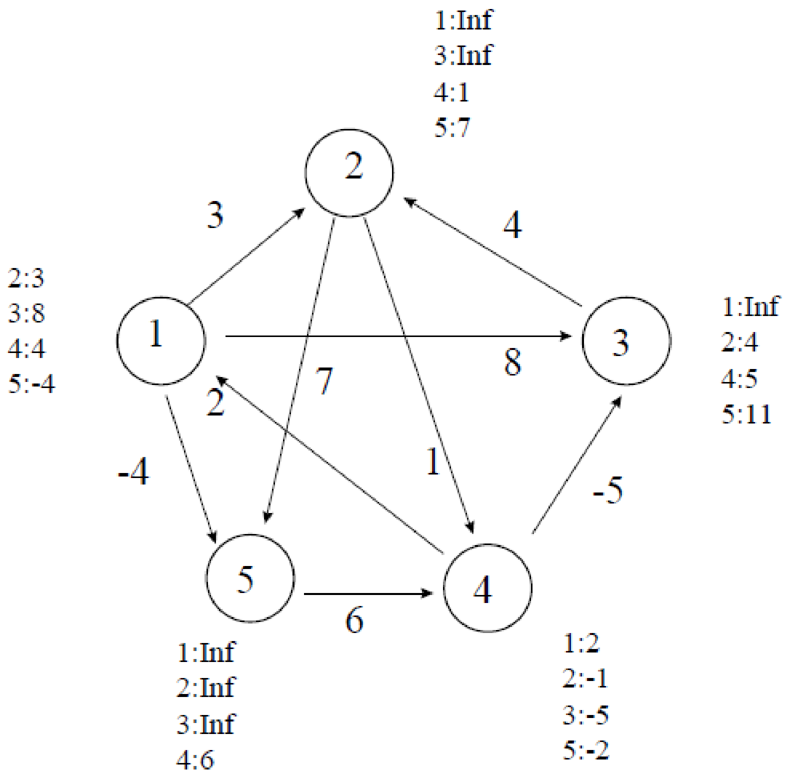
最短有向路： Floyd-Warshall-All-Pairs-Shortest

***k*=1** ***k*=2**

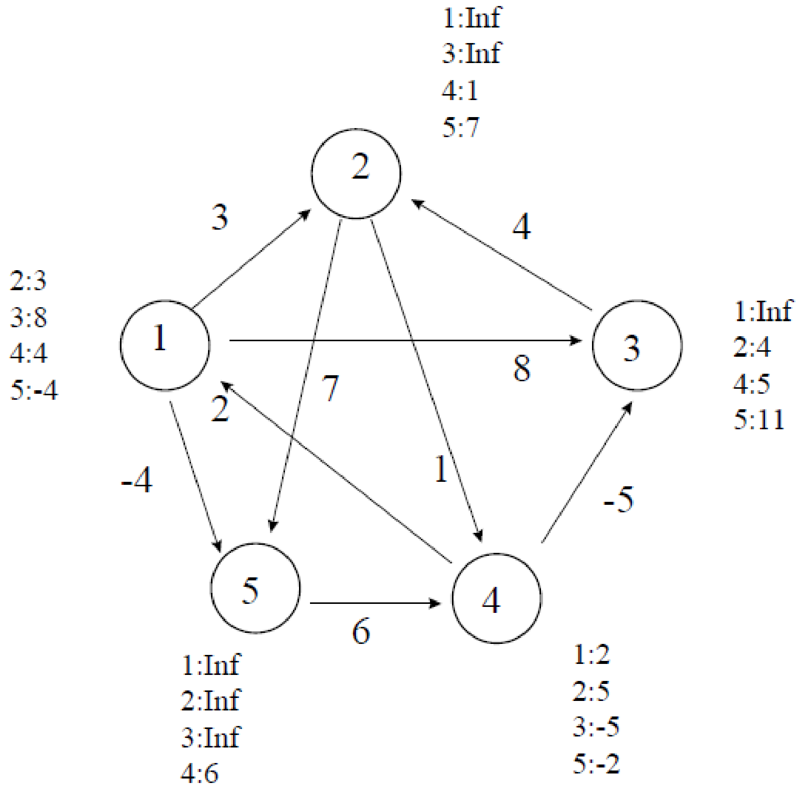


最短有向路： Floyd-Warshall-All-Pairs-Shortest

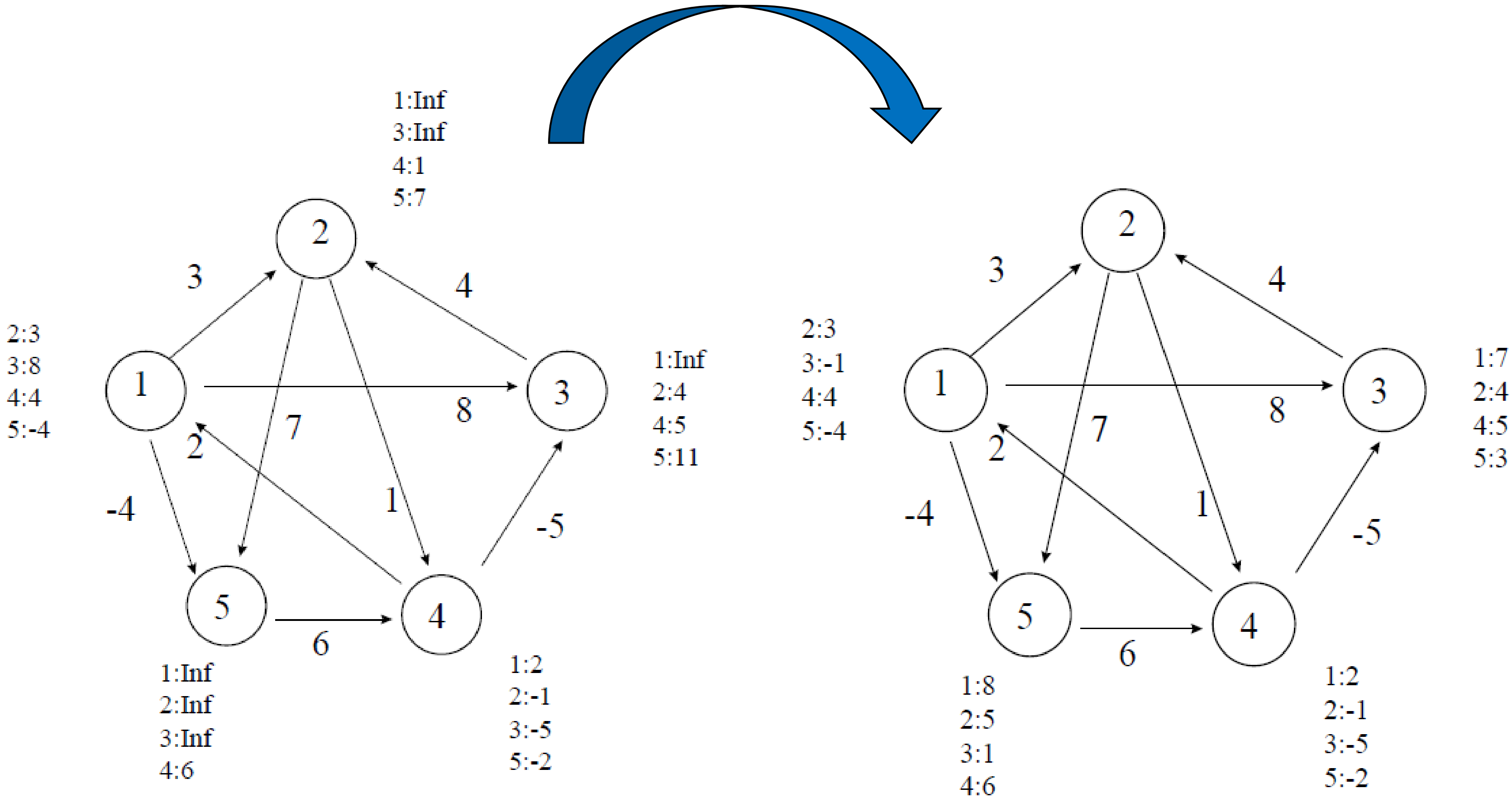
***k*=3**



***k*=2**

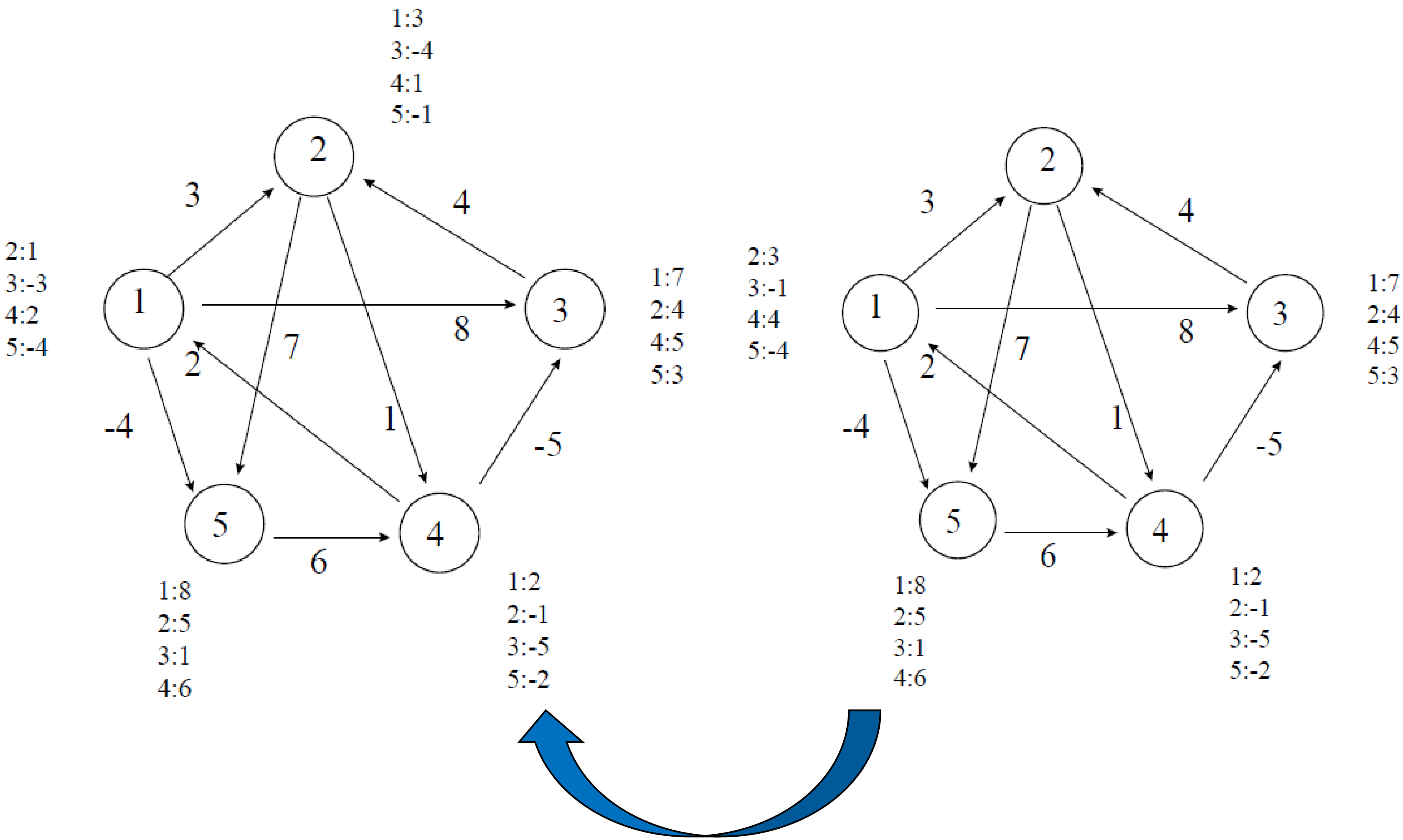


***k*=3** **_k_=4**


1:Inf
3:Inf
4:1
5:7
2:3
3:8
4:4
5:-4
1:Inf
2:4
4:5
5:11
1:Inf
2:Inf
3:Inf
4:6
1:2
2:-1
3:-5
5:-2
2:3
3:-1
4:4
5:-4
1:7
2:4
4:5
5:3
1:8
2:5
3:1
4:6
1:2
2:-1
3:-5
5:-2

最短有向路： Floyd-Warshall-All-Pairs-Shortest

***k*=5** **_k_=4**

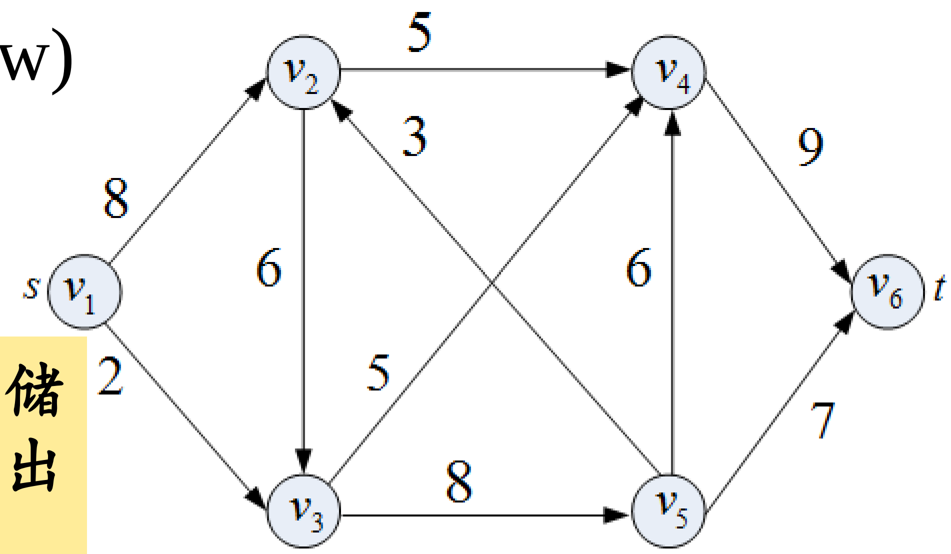


最大流：定义与模型

交通网、通信网……：**流(flow)**

弧权值：**容量(capacity)**

源(s)、汇(t)、最大流



基本假设： 单向流动、源(无限储藏)、汇(无限接纳)、中间顶点出入平衡(无储存功能)

- 设$G = (V, A, C)$是有向网络，$c_{i,j}$表示弧(i, j)的容量
- **最大流问题**：LP问题
- **最大流**：LP问题的最优解、最优值

$$\max_{v,\, x_{i,j}} v$$

$$s.t.\quad 0 \le x_{i,j} \le c_{i,j}\ ;\ (i, j) \in A$$

$$\sum_{j,\,(i,j)\in A} x_{i,j} - \sum_{k,\,(k,i)\in A} x_{k,i} = \begin{cases} v, & i = s \\ -v, & i = t \\ 0, & i \ne s,t \end{cases}$$

结合几何直观的图论方法比通用线性规划算法更有效

最大流：相关概念

可行流： LP问题的可行解，可在弧上标注

s-t无向路： LP有向网络G的基本图中从s到t的一条路

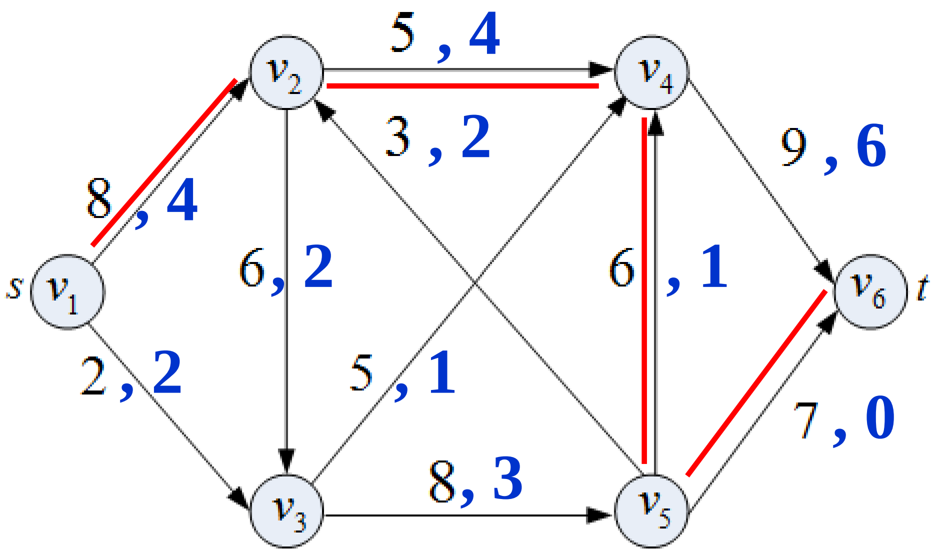
前向弧： s-t无向路中与s到t方向一致的弧

反向弧： s-t无向路中与s到t方向相反的弧

增广路： 对一个可行流，若一条s-t无向路中前向弧流量均严格小于其容量，反向弧流量均严格大于0，则称为一条增广路

增广路最大可增容量：

所有前向弧的剩余容量、所有反向弧流量中的最小值



若存在增广路，就不是最大流

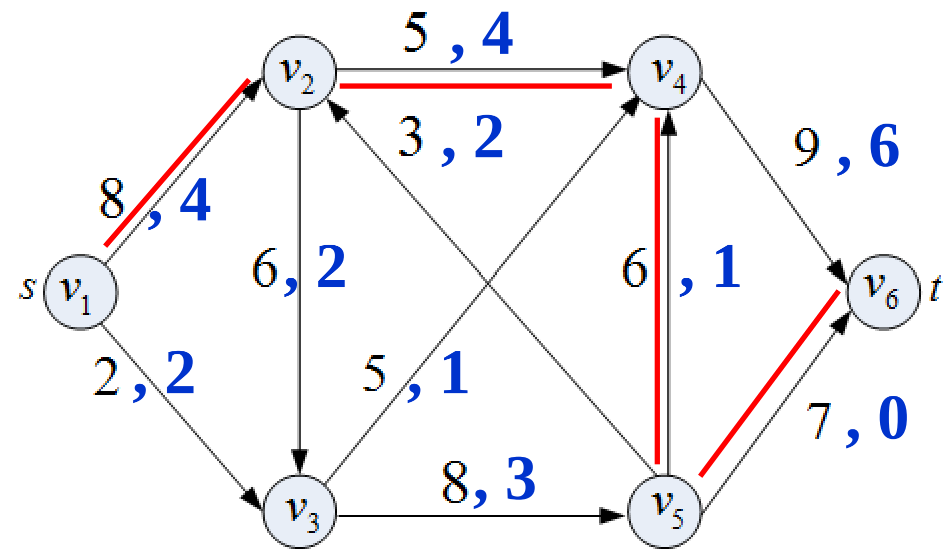
$$\max_{v,\, x_{i,j}} v$$

$$s.t.\quad 0 \le x_{i,j} \le c_{i,j}\ ;(i,j)\in A$$

$$\sum_{j,\,(i,j)\in A} x_{i,j} - \sum_{k,\,(k,i)\in A} x_{k,i} = \begin{cases} v, & i = s \\ -v, & i = t \\ 0, & i \ne s,t \end{cases}$$

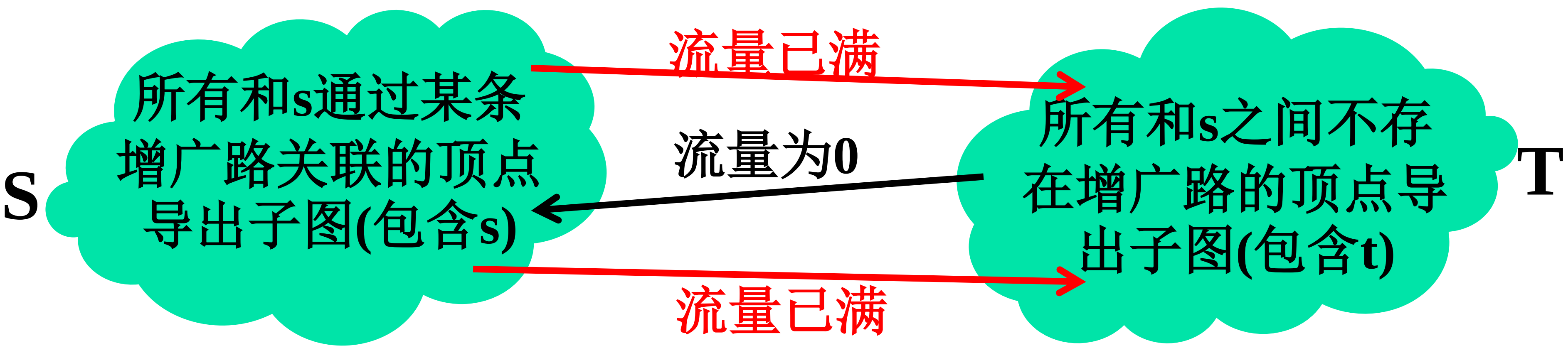
最大流：性质与判定

- 增广路的概念可推广：**s-j增广路**
- 增广路的任意前部片段仍为增广路
- 给定一个可行流，所有s-j增广路关联的顶点和弧对应一个子有向图，其基本图为连通图



若存在增广路，就不是最大流

不存在从s到t 的增广路意味着什么？



最大流：性质与判定

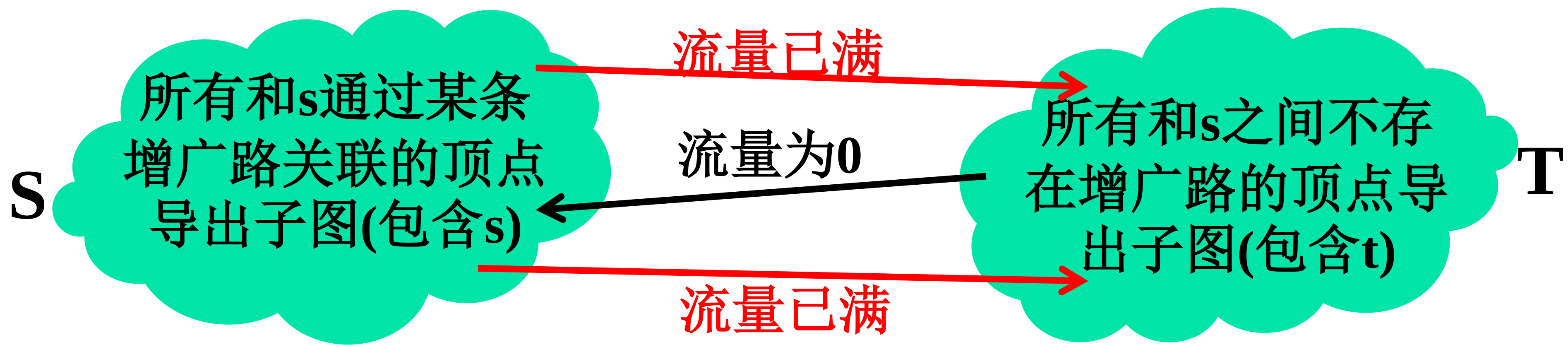
定理5.6.1： 一个可行流是最大流当且仅当不存在关于它的从 s 到 t 的增广路。

(s, t)-割： 若$S,T \subset V, S \cap T = \phi, S \cup T = V, s \in S, t \in T$ 则所有尾在S中，头在T中的弧构成的集合称为一个 (s, t)-割。

- **(s, t)-割不是弧割**
- **(s, t)-割的容量**定义为其包含的所有弧的容量之和

命题： 任何一个可行流的(s, t)-流量不超过任何一个(s, t)-割的容量。

不存在从s到t 的增广路意味着什么？



最大流：性质与判定

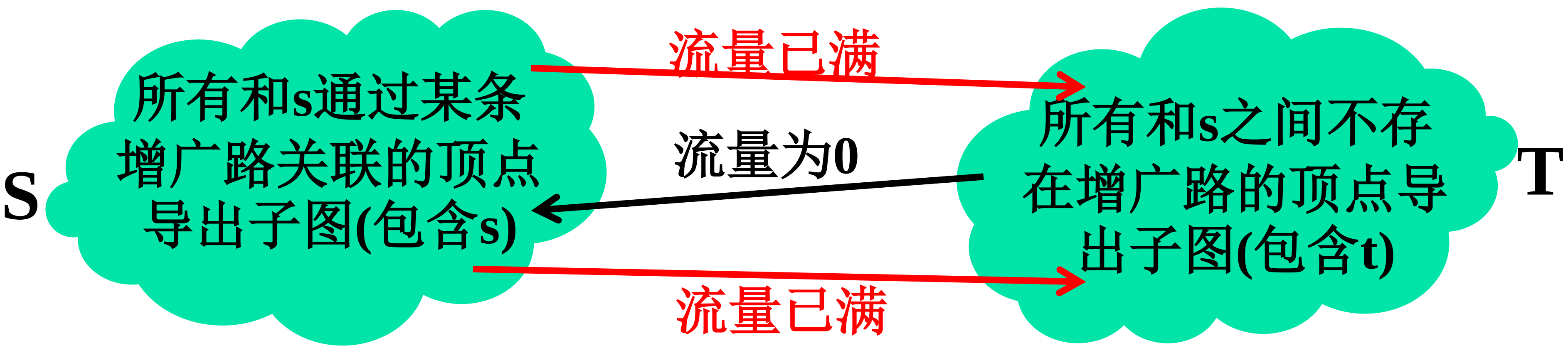
定理5.6.1： 一个可行流是最大流当且仅当不存在关于它的从 s 到 t 的增广路。

定理5.6.3(最大流最小割定理)： 一个(s, t) 流的最大值等于(s, t)-割的最小容量。

- **(s, t)-割不是弧割**
- **(s, t)-割的容量**定义为其包含的所有弧的容量之和

命题： 任何一个可行流的(s, t)-流量不超过任何一个(s, t)-割的容量。

不存在从s到t 的增广路意味着什么？



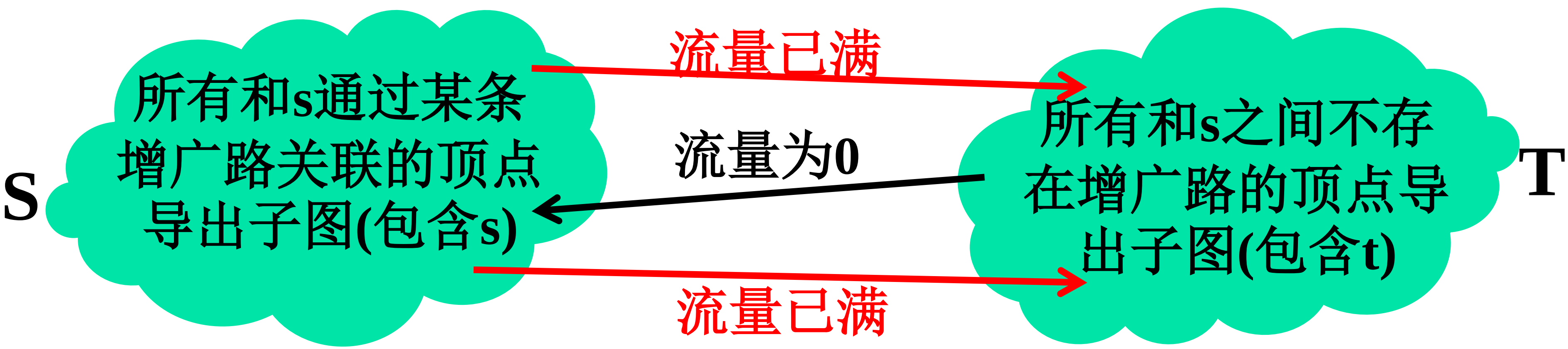
最大流：算法思想

定理5.6.1： 一个可行流是最大流当且仅当不存在关于它的从 s 到 t 的增广路。

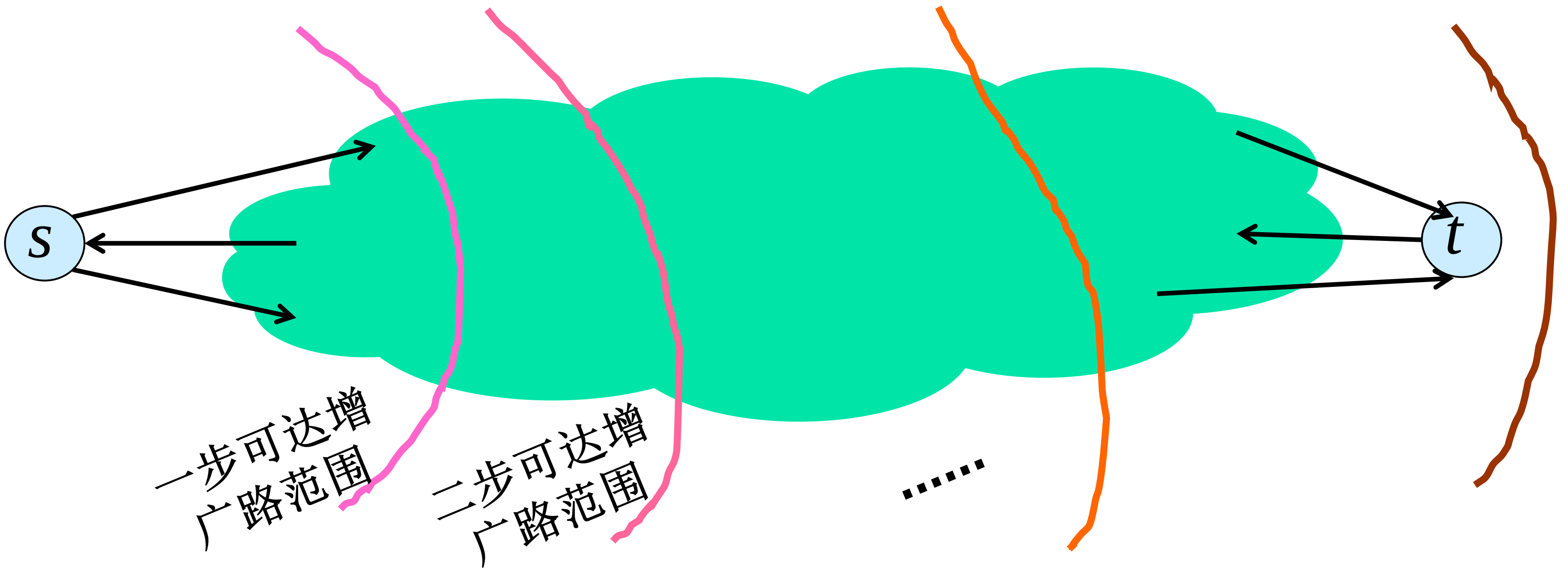
定理5.6.3(最大流最小割定理)： 一个(s, t) 流的最大值等于(s, t)-割的最小容量。

求最大流： 不直接求解线性规划(初始解？两阶段？)，而是不断寻找增广路(初始解-0流)，改进，直至无(s,t)增广路

不存在从s到t 的增广路意味着什么？



最大流：增广路算法



最大流：增广路算法

双标号： $(p(k),\delta(k))$，$\delta(k)$表示从 s 到 k 的增广路可增加多少流量

$p(k)=i$ 表示该增广路中最后一条弧是(i, k)，正向弧

$p(k)=-i$ 表示该增广路中最后一条弧是(k, i)，反向弧

Step 1. 赋顶点 s 永久标号：$p(s)=*, \delta(s)=+\infty$ 置 $S=\{s\}, R=V\setminus\{s\}$

Step 2. 若 $t\in S$ 已找到一条增广路，停；否则继续

Step 3. 对顶点分别位于S, R 中的所有弧(i, j) 依次进行检查：

$$\begin{cases} \text{若 } i\in S, j\in R, x_{i,j}<c_{i,j} \text{ 置 } p(j)=i, \delta(j)=\min\left\{\delta(i), c_{i,j}-x_{i,j}\right\} \\ \qquad S=S\bigcup\{j\}, R=R\setminus\{j\} \\ \text{若 } j\in S, i\in R, x_{i,j}>0 \text{ 置 } p(i)=-j, \delta(i)=\min\left\{\delta(j), x_{i,j}\right\} \\ \qquad S=S\bigcup\{i\}, R=R\setminus\{i\} \end{cases}$$

转Step2；若未发现这样的弧，停，此时不存在 (s, t) 增广路

最大流：算法(Ford-Fulkerson, 1962)

Step 1. 设置初始可行流(如0流),
赋顶点 s 永久标号： $p(s)=*, \delta(s)=+\infty$

Step 2. 执行增广路算法，寻找一条(s, t) 增广路,
若无增广路，转Step4；否则继续

Step 3. 获得的增广路中，每条前向弧增加流量$\delta(t)$,
每条反向弧减少流量$\delta(t)$；保留源顶点标号,
删除其余顶点标号，转Step2

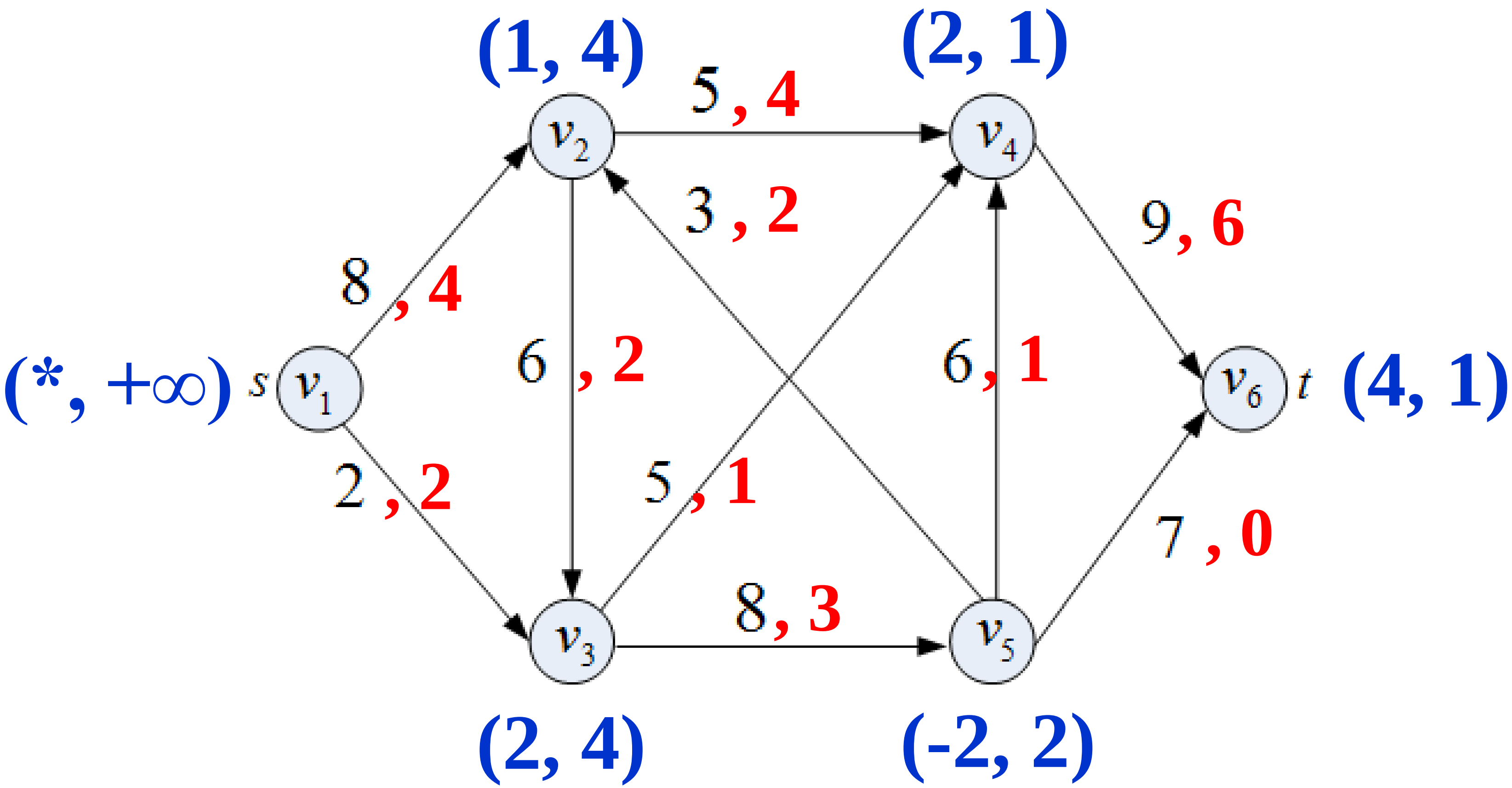
Step 4. 输出最优解，若有必要，输出最小(s, t)-割

算法说明:

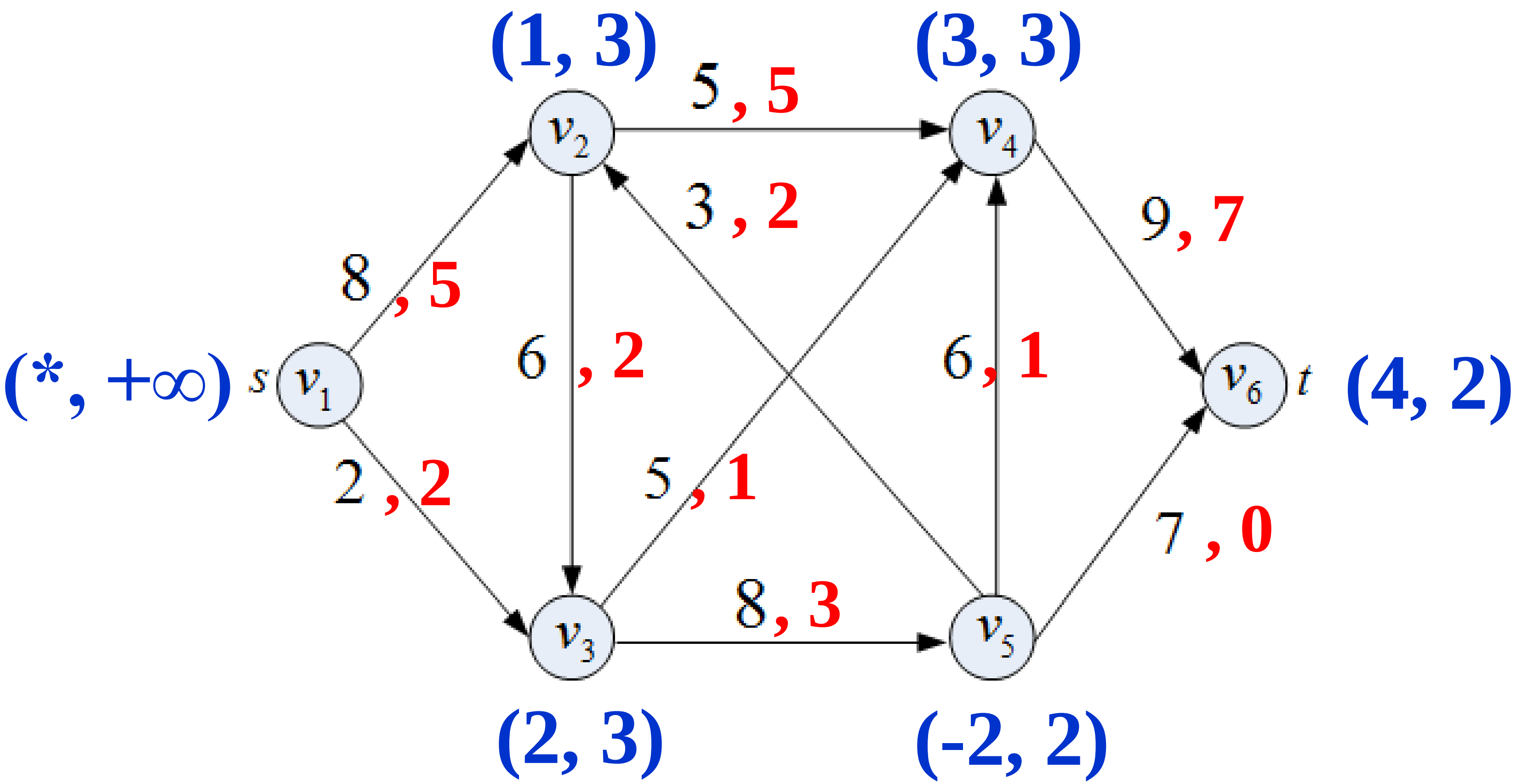
- 正确性：定理5.6.1；若存在增广路，增广路算法一定会发现
- 收敛性：**可能需要无穷多次迭代，未必收敛到最大流(1962)**
- **定理5.6.2**：若所有弧容量均为整数，则有限步收敛至最大流
- 各种修正算法(保证有限步收敛)
- 计算复杂性：P218

最大流： 算法(Ford-Fulkerson, 1956)

例子:P218图5.6.5

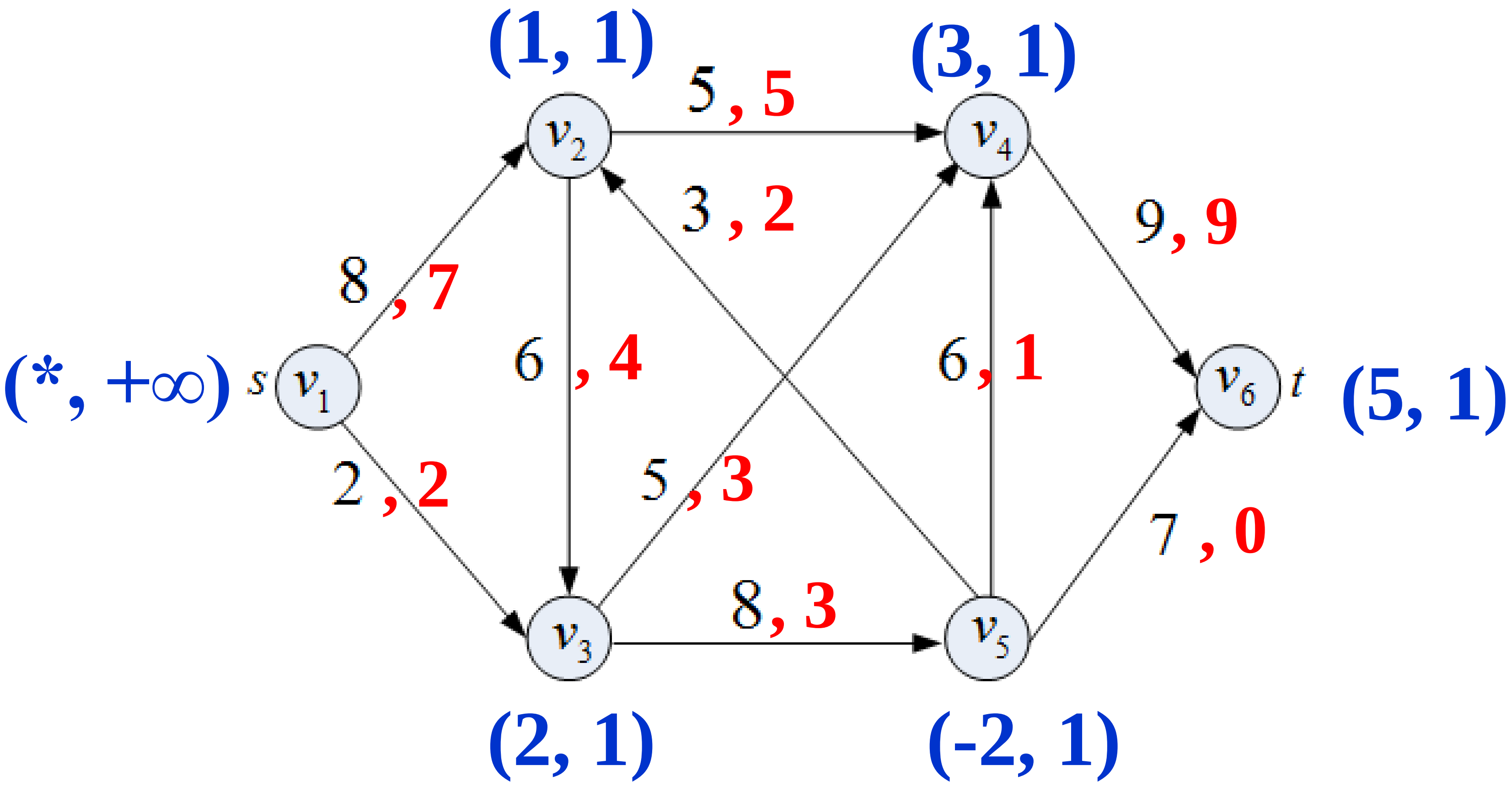


例子:P218图5.6.5



最大流： 算法(Ford-Fulkerson, 1956)

例子:P218图5.6.5

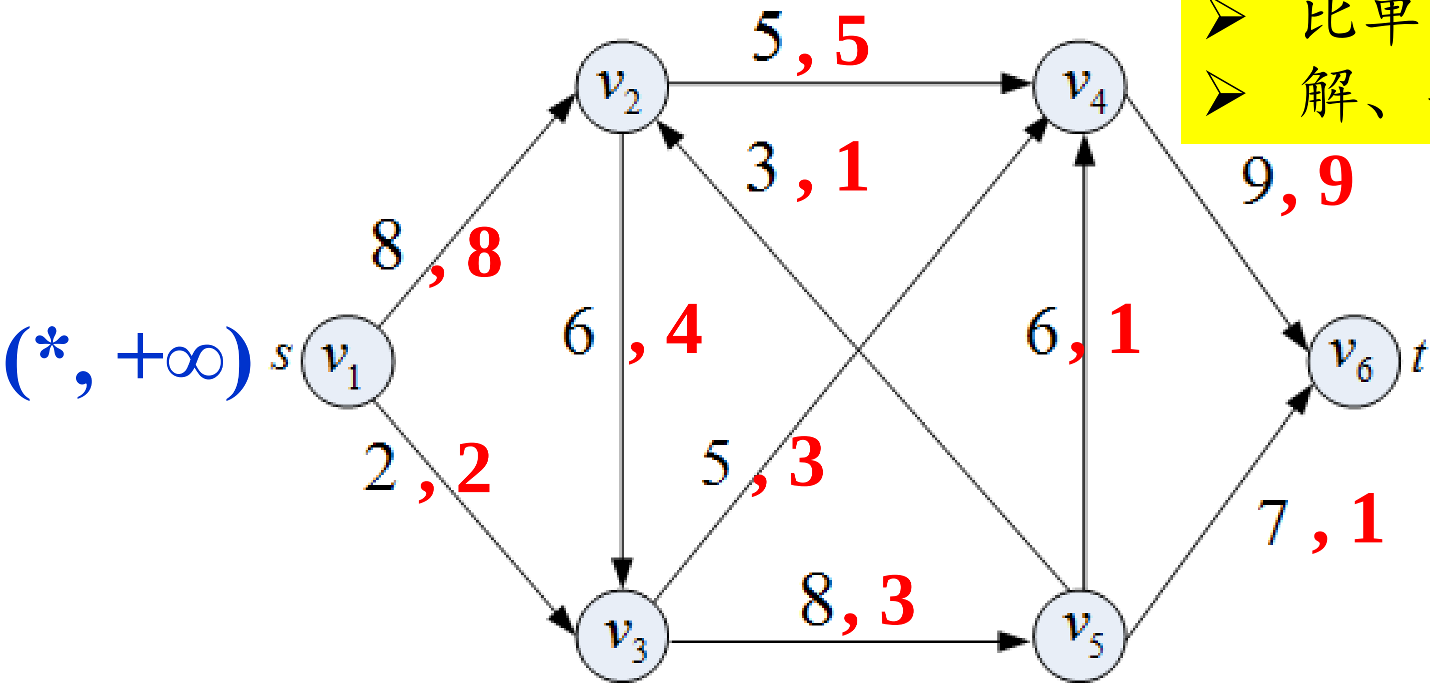


最大流： 算法(Ford-Fulkerson, 1956)

例子:P218图5.6.5

启示:

- 算法中有较大任意性，可加入启发式规则
- 比单纯形法效率高
- 解、割均不唯一



无增广路，已得最大流，已得最小割

谢谢！