MATH 3011 运筹学 **秋季 2023**

Lecture 6: 最短路和最大流问题

Lecturer: 陈士祥 *Scribes:* 陈士祥

1 最短路径问题

1.1 最短路径问题介绍

最短路径问题是图论和计算机科学中的一个经典问题，其目标是在一个加权图中找到两点之间的最短路径。这个问题在多种实际应用中都非常重要，比如在网络路由、地图导航、社交网络分析等领域。

最短路径问题的关键要素：

- 加权图：这个问题通常在一个加权图中提出，其中图的每条边都有一个权重，代表两个顶点之间的“距离”或“成本”。
- 起点和终点：最短路径问题通常涉及找到从一个指定的起点到一个指定的终点的最短路径。
- 路径长度：路径的长度是路径上所有边的权重之和，最短路径问题旨在最小化这个总和。

解决方法：最短路径问题可以通过多种算法来解决，其中最著名的包括：

- 迪杰斯特拉算法（Dijkstra's Algorithm）：适用于非负权重的图，可以找到从单一源点到所有其他顶点的最短路径。
- 贝尔曼-福特算法（Bellman-Ford Algorithm）：适用于含有负权重边的图，可以处理图中的负权边，同时也能检测图中的负权环。
- 弗洛伊德算法（Floyd-Warshall Algorithm）：用于计算所有顶点对之间的最短路径，适用于任何加权图。

应用实例：

- 地理信息系统：在地图应用中寻找两地之间的最短行车或步行路线。
- 网络通信：在数据包传输中找到最有效的路径，以最小化延迟和带宽消耗。
- 社交网络分析：计算社交网络中两个人之间的“距离”或“联系程度”。

最短路径问题不仅在理论研究中重要，而且在我们日常生活中的许多方面都有广泛应用。通过各种算法的应用，我们能够高效地解决这些问题，无论是在现实世界中的物理网络，还是在虚拟世界中的信息网络。我们之探讨非负权重的图，因此探讨最短路径的线性规划建模以及 Dijkstra 算法。

1.2 最短路径问题的线性规划建模

用 x_{ij} 表示顶点 i 到 j 的 “流量”。$x_{ij}=1$ 即为 i 到 j 的边在这条路径上，$x_{ij}=0$ 即表示不在路径上。故首先我们可以建模为如下 01 整数线性规划模型：

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{(s,j)\in E} x_{sj} - \sum_{(j,s)\in E} x_{js} = 1 \\
& \sum_{(k,j)\in E} x_{kj} - \sum_{(i,k)\in E} x_{ik} = 0, \quad \forall k \in V-\{s,t\} \\
& \sum_{(t,i)\in E} x_{ti} - \sum_{(i,t)\in E} x_{it} = -1 \\
& x_{ij} \in \{0,1\}, \forall (i,j)\in E
\end{aligned} \tag{6.1}$$

上述约束中，

- $\sum_{(s,j)\in E} x_{sj} - \sum_{(j,s)\in E} x_{js} = 1$ 表示从起点 s 流出的流量为 1.
- $\sum_{(k,j)\in E} x_{kj} - \sum_{(i,k)\in E} x_{ik} = 0, \quad \forall k \in V-\{s,t\}$ 表示除了起点 s 终点 t 的顶点，流入和流出守恒。
- $\sum_{(t,i)\in E} x_{ti} - \sum_{(i,t)\in E} x_{it} = -1$ 表示流出终点的流量为 1.

然而，带有形如 $x_{ij}\in\{0,1\}$ 的 01 约束问题难以求解。我们可以将其转化为**松弛问题：**

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{(s,j)\in E} x_{sj} - \sum_{(j,s)\in E} x_{js} = 1 \\
& \sum_{(k,j)\in E} x_{kj} - \sum_{(i,k)\in E} x_{ik} = 0, \quad \forall k \in V-\{s,t\} \\
& \sum_{(t,i)\in E} x_{ti} - \sum_{(i,t)\in E} x_{it} = -1 \\
& x_{ij} \geq 0, \forall (i,j)\in E
\end{aligned} \tag{6.2}$$

求解上述线性规划问题(6.2)将自动得到最短路的最优解。我们给出一些 high-level 的解释

- 问题的本质：在最短路径问题中，我们寻找的是从起点到终点的单一路径。这意味着任何给定的边要么是路径的一部分（在这种情况下，它的值是 1），要么不是（值为 0）。没有理由选择一条边超过一次，因为这不会形成有效的路径（会形成环路或重复路径），也不会有助于减少总成本。

- 流守恒约束：在最短路径问题的 LP 表述中，除了起点和终点外，每个顶点的进入流量和离开流量必须相等。这种约束保证了从起点到终点的连续路径。在这样的设置下，任何一条边最多只会被使用一次，因为重复使用会违反流守恒约束。
- 线性规划的最优解特性：线性规划问题的解在可行域的边界上。在最短路径问题中，由于边的选择变量只能对总成本产生非负贡献，因此在最优解中，这些变量要么取值 0（不选择这条边），要么取值 1（选择这条边）。不需要变量取大于 1 的值，因为这不会带来任何额外的优势。

最短路径问题的对偶为：

$$\begin{aligned} \max \quad & (y_s - y_t) \\ \text{s.t.} \quad & y_i - y_j \le c_{ij}, \quad \forall (i,j) \in E \end{aligned} \tag{6.3}$$

因此，一般网络的最短路径问题可以看成是一个线性规划模型（事实上是一个更特殊的运输模型），可依据对偶性构造其求解算法。

当然，也可通过暴力穷举法：由于可能的路径为指数多个，但显然不是一个好方法。
如果基于动态规划的思想，可给出最短路径问题的强多项式时间算法。以下仅说明具有代表性的算法之一的 **Dijkstra's algorithm**.

1.3 Dijkstra 算法

Dijkstra 的主要思想如下，我们显然有最短路径的性质：若 s 到 v 的一条最短路径经过某个顶点 c，即 $s \to c \to v$，那么这条路径上子路径 $s \to c$ 是 s 到 c 的最短路径。

Algorithm 1 Dijkstra 算法

1: [输入] 有向图或者无向图 $G = (V, E)$, 各边长度 $c : E \to \mathbb{R}_+$, 始点 $s \in V$.
2: [输出] 从始点到所有节点 $v \in V$ 的最短路径及其长度 $c^*(v)$.
3: 初始化：令 $d(s) := 0, d(v) := \infty (v \in V - \{s\})$, 以及 $X := \emptyset$
4: **while** $X \neq V$ **do**
5: 选取任一个满足 $d(v^*) = \min\{d(v) \mid v \in V - X\}$ 的顶点 $v^* \in V - X$
6: 更新：$X := X \cup \{v^*\}$.
7: 进一步对 $w \in V - X$ 的各边 $e = (v^*, w) \in E$ 作如下更新：

$$d(w) := \min\{d(w), d(v^*) + d(v^*, w)\}.$$

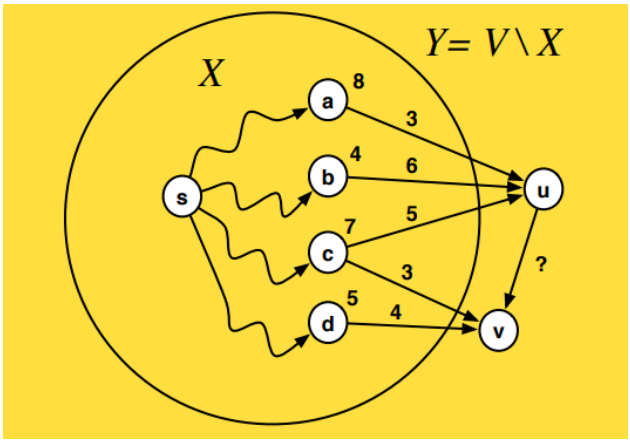
8: **end while**

算法1的各步解释如下

- 第 3 行：把图分为已访问节点 X 和未访问节点 $V-X$, 并记录 s 到所有点的（临时）最短路径为 $d(v)$. 对于已访问节点 $x \in X$，$d(x)$ 是 s 到 x 的最短路径；未访问节点 $y \in V-X$, $d(y)$ 叫做临时最短路径。

- 第 5 行详解：对于已访问节点集合 X，我们寻找 X 的所有邻居节点中，到 X 距离最小的顶点。即 $v^* \in V-X$ 是问题 $\min\limits_{x\in X,\ v\in V-X}(d(x)+w(x,v))$ 的解。

  Example 6.1 若我们有如下的图，已知 s 到所有 X 中所有顶点最短距离。对于 $u, v \in V-X$, 我们得到 $p(v)=d(d)+w(d,v)=9$.



- 第 6-7 行：更新已访问集合 $X=X\cup\{v^*\}$, 更新临时距离 $d(w), w\in V-X$.

- 第 4 行：当所有节点均访问过，即 $X=V$ 时，结束算法；否则继续更新集合 X。

Example 6.2 如图6.1，给定网络和各边上的路径长度，我们用表格6.1的方式展示 *Dijkstra* 算法。

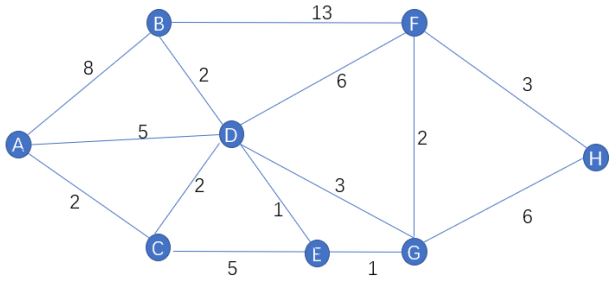


图 6.1: Dijkstra 算法示例

X	A	B	C	D	E	F	G	H
A	0	8_A	2_A	5_A	∞	∞	∞	∞
C		8_A	2_A	4_C	7_C	∞	∞	∞
D		6_D		4_C	5_D	10_D	7_D	∞
E		6_D			5_D	10_D	6_E	∞
B		6_D				10_D	6_E	∞
G						8_G	6_E	12_G
F						8_G		11_F
H								11_F

表 6.1: Dijkstra 算法表格

2 最大流问题

2.1 最大流问题概述

最大流问题是图论和网络优化中的一个经典问题，其核心目标是确定在一个网络（通常是一个有向图）中从一个特定的源点（source）到一个特定的汇点（sink）能够传输的最大流量。这个问题在诸如交通系统、通信网络、管道网络等许多实际应用场景中非常重要。

2.2 最大流问题的关键要素

- **有向图**：问题通常在一个有向图中提出，图中的每条边有一个非负容量，代表这条边可以承载的最大流量。
- **源点和汇点**：指定一个源点和一个汇点。流量从源点出发，流向汇点。
- **边容量限制**：图中每条边的流量不能超过其容量。
- **流守恒约束**：除了源点和汇点外，图中每个顶点的流入量必须等于流出量。

2.2.1 解决方法

最大流问题可以通过以下算法来解决：

1. **Ford-Fulkerson 方法**：通过不断寻找从源点到汇点的增广路径来增加网络的流量，直到无法找到更多的增广路径为止。
2. **Edmonds-Karp 算法**：Ford-Fulkerson 方法的一个特定实现，使用广度优先搜索来寻找增广路径，确保算法的多项式时间复杂度。

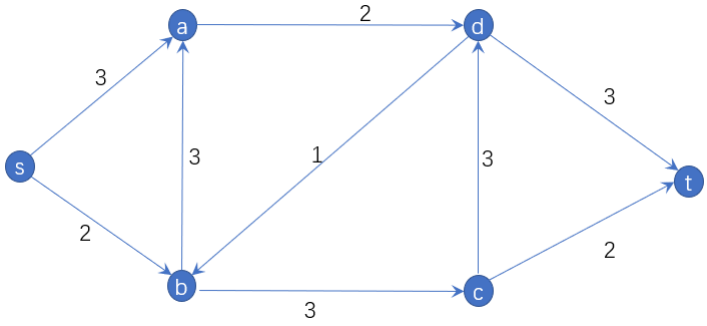


图 6.2: 有向网络图示，各边均指定了方向，每条边给定了最大流量。问题即为确定从 s 出发到达 t，网络中的最大流量。

3. **Dinic 算法**:通过构建层次图来实现更高效的搜索增广路径,通常比 Ford-Fulkerson 方法和 Edmonds-Karp 算法更快。

我们只介绍 Ford-Fulkerson 方法。

2.2.2 应用实例

- **交通系统**：计算道路或交通网络中的最大车流量。
- **通信网络**：确定数据在网络中从一个点到另一个点的最大传输速率。
- **管道网络**：在水管或石油管道网络中，确定最大的流体传输能力。

2.3 最大流算法

Definition 6.1 (Flow) *流量 (flow) 是有向图 $G = (V, E)$ 上的一个函数 $f : E \to R$, 满足如下条件:*

1. **容量约束:** $0 \le f(e) \le c(e)$.
2. **流量守恒:** *对于任意 $v \in V - \{s, t\}$, 有 $\sum_{e \in \mathrm{In}(v)} f(e) = \sum_{e \in \mathrm{Out}(v)} f(e)$, 这里 $\mathrm{Out}(v)$ 和 $\mathrm{In}(v)$ 分别表示 G 中流出和流入 v 节点的边集合。*

f 也可以写成顶点的函数，即 $f : V \times V \to R$. 相应地,

1. **容量约束:** 若 $(u, v) \in E$, 则 $0 \le f(u, v) \le c(u, v)$. (若 u, v 不直接相连, $f(u, v) = 0$.)
2. **流量守恒:** 对于任意 $u \in V - \{s, t\}$, 有 $\sum_{v \in V} f(u, v) = 0$.
3. **反对称:** $f(u, v) = -f(v, u)$.

Definition 6.2 从 s 的纯流出量，定义为流 f 的值，记作 $|f|$，

$$|f| \triangleq \sum_{v \in V} f(s, v) = f(s, V)$$

Lemma 6.1 有如下结论

- $f(X, X) = 0$,
- $f(X, Y) = -f(Y, X)$
- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$, *if* $X \cap Y = \emptyset$.
- $f(u, V) = 0$, $\forall u \in V - \{s, t\}$.

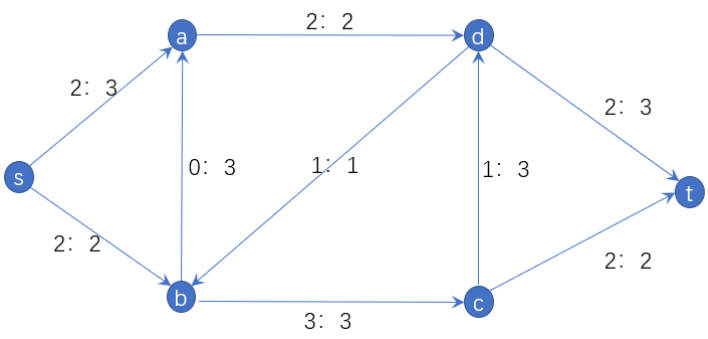
Theorem 6.1 流的值 $|f|$ 等于流出 t 的流的和：

$$|f| = \sum_{v \in V} f(v, t) = f(V, t).$$

证明：

$$\begin{aligned} |f| &= f(s, V) \\ &= f(V, V) - f(V - s, V) \\ &= f(V, V - s) \\ &= f(V, t) + f(V, V - s - t) \\ &= f(V, t). \end{aligned} \tag{6.4}$$

所谓最大流问题，就是求出使流值达最大的可行流问题，如下图，我们在图6.2中，给每条边添加满足条件的流量，得到 s 到 t 的流值为 4，等于从 s 的流出值和流入 t 的流值. 显然，若是改变某一边的流量，将会影响整个网络中的流量，故最大流问题是比较复杂的问题。



与最短路类似，我们也可也用线性规划可描述最大流问题：

$$\begin{aligned} \max \quad & f(s,V) = \textstyle\sum_{v:(s,v)\in E, v\in V} f(s,v) \\ \text{s.t.} \quad & \sum_{v\in V} f(v,u) = \textstyle\sum_{w\in V} f(u,w), \forall u \in V-s-t \\ & 0 \le f(u,v) \le c(u,v), (u,v) \in E. \end{aligned} \tag{6.5}$$

求解上述线性规划问题可以得到最大流的最优解。但是很多情况下，这并不是一种有效率的方法。下面我们将介绍 Ford-Fulkerson 方法。

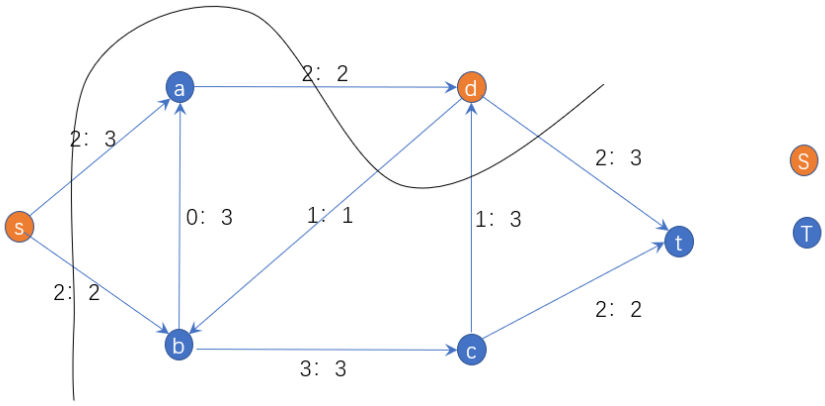
2.3.1 最小割问题 (Minimum Cut Problem)

Definition 6.3 流网络 $G=(V,E)$ 的一个割 (S,T) 是指，顶点集合 V 的一个分割，使得 $s\in S, t\in T$. 如果 f 是 G 的流量，那么一个割上的流量为 $f(S,T)$.

也就是说，割把一个流网络的顶点集划分成两个集合 S 和 T, 使得源点 $s\in S$ 且汇点 $t\in T$.

Definition 6.4 分割 (S,T) 的容量为

$$c(S,T) = \sum_{\substack{(u,v)\in E \\ u\in S,\ v\in T}} c(u,v).$$



如图，$S=\{s,d\}$, $T=\{a,b,c,t\}$. 分割的容量为 $c(S,T)=3+2+1+3=9$. 割流量为 $f(S,T)=2+2+(-2+1-1-2)=4$.

所谓最小割问题，即找到一种割法，使割的容量最小。也就是说，找到通过能力最弱的断面。

Lemma 6.2 对于任意割，我们有 $|f|=f(S,T)$. 另外 $|f|\le c(S,T)$.

Proof:

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) \\ &= f(s,V) + f(S-s,V) \\ &= f(s,V) \\ &= |f|. \end{aligned} \tag{6.6}$$

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u\in S}\sum_{v\in T} f(u,v) \\ &\le \sum_{u\in S}\sum_{v\in T} c(u,v) \\ &= c(S,T). \end{aligned} \tag{6.7}$$

■

从直观上看，割集 $(S,T) = \{(u,v) \in E, u \in S, v \in T\}$ 是从源点 s 到汇点 t 的必经之路，如果该路堵塞则流从 s 无法到达 t。于是我们可以得到下面的定理。

Theorem 6.2 最大流最小割定理：任意一个流网络的最大流量等于该网络的最小割的容量。

可以写出最大流线性规划问题(6.5)的对偶问题，并且其是最小割问题的对偶问题。 因此，由线性规划强对偶定理也可以证明上面的结论。

2.4 余网络（residual network）和流扩充路（augmenting path）

网络 $\mathcal{N} = (G, s, t, \mathbf{c})$ 中给定一个可行流 f，**余网络** 构造方式如下：

对于 $\mathcal{N}$ 的各边 $e = (u,v) \in E$，按照以下规则生成边 (u,v) 或 (v,u)，得到的有向边集合记为 E_f，同时确定其容量 c_f.

R-1 如果 $c(e) - f(e) > 0$, 则生成 $(u,v) \in E_f$, 并令其容量 $c_f(u,v) = c(u,v) - f(u,v)$.

R-2 如果 $f(u,v) > 0$, 则生成 $(v,u) \in E_f$, 并令其容量 $c_f(v,u) = f(u,v)$.

所得到的有向网中所有容量 $\bar{c} > 0$ 的边构成余网络 $\mathcal{N}_f = (G_f, s, t, c_f)$, 其中有向图 $G_f = (V, E_f)$, 容量 $c_f = \{c_f(e) > 0 \mid \forall e \in E_f\}$. 如图6.3展示了原网络和生成的余网络。余网络 $\mathcal{N}_f$ 中从 s 到 t 的路称为**流扩充路**。

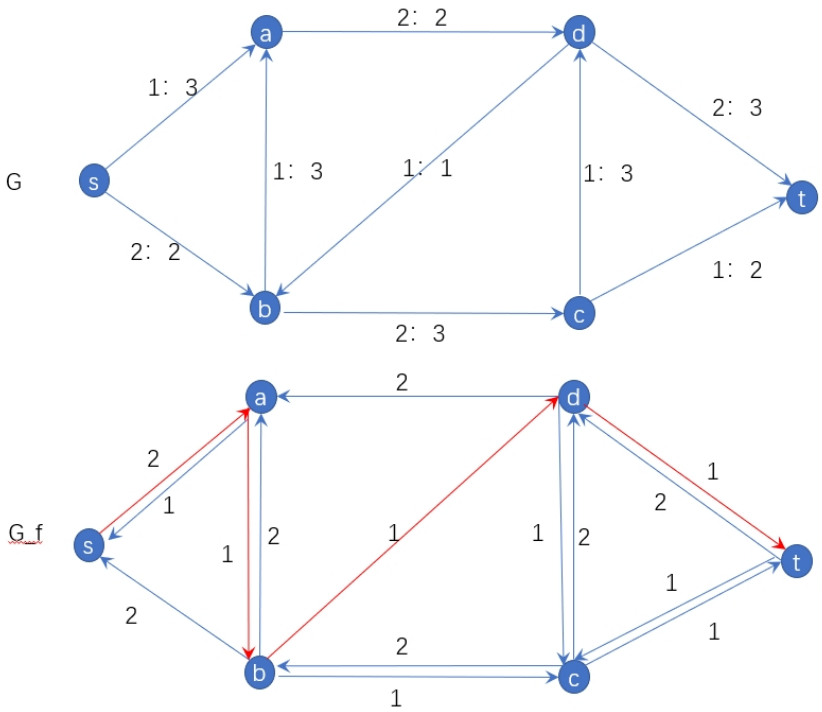


图 6.3: 原网络 (上) 和余网络 (下)。流扩充路为标红的边。

Lemma 6.3 (流扩充路) 给定网络 $\mathcal{N} = (G, s, t, c)$, 其可行流 f 为最大流的充要条件是, 余网络 $\mathcal{N}_f$ 中不存在有流扩充路。

如果存在有流扩充路 p, 则 f 可修正为流值更大的流, 通过沿着 p 对原网络上的相应路径更改流 f, 更新量为 $c_f(p) = \min_{(u,v)\in p}\{c_f(u, v)\}$.

算法 Ford-Fulkerson Algorithm 描述如下

输入 网络 $\mathcal{N} = (G, s, t, c)$, 其中有向图 $G = (V, E)$.

输出 从 s 到 t 的最大流值 $f_{\max}$.

步一 初始化: 令 $f(e) := 0(e \in E)$ 以及 $f_{\max} := 0$.

步二 余网络: 构造当前可行流 f 的余网络 $\mathcal{N}_f = (G_f, s, t, c_f)$.

步三 流的扩充: 如果 $\mathcal{N}_f$ 中没有流的扩充路则结束计算。相反, 如果存在有流的扩充路, 则选取其一 p, 通过沿着 p 对 f 增加 $c_f(p) = \min_{(u,v)\in p}\{c_f(u, v)\}$. (若 G_f 中的边与 G 中的有向边相反, 则减去相应的值。) 令 $f_{\max} = f + c_f(p)$. 回到 步二。

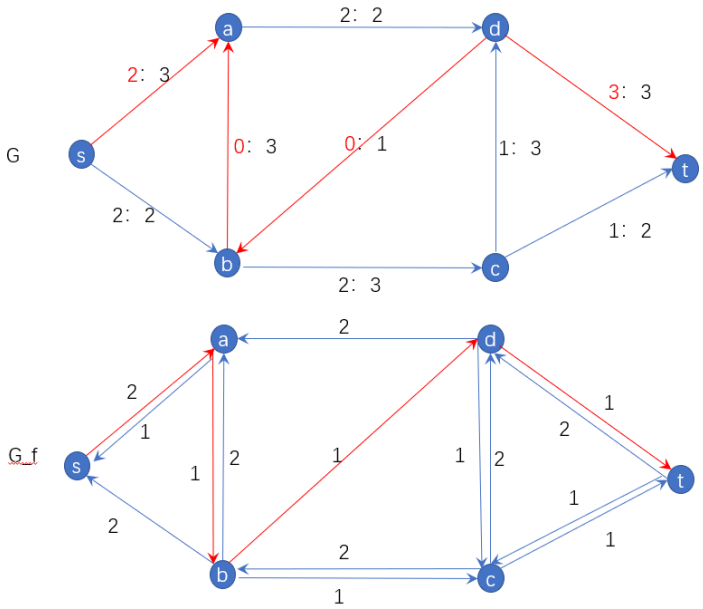


图 6.4: 构造余网络

3 最小成本流问题

考虑网络 $\mathcal{N} = (G, s, t, \mathbf{c}, w)$ 上的流 f, 其中 $w(u, v)$ 是边 (u, v) 的费用，其中流值固定为

$$|f| = f^*.$$

在流值 $|f| = f^*$ 不超过网络的最大流的条件下，求出使成本达最小的流就是所谓最小成本流问题。

Example 6.3 有足够多辆卡车要将数量无限的某种物品从一个地点运输到另外一个地点，现在有有限条单向行驶道路直接或者间接地连接了这两地。但是每一条道路都有运输通过总数量的限制，称为容量，同时携带物品通过该路段时，都会按照携带物品数量多少被收取一定的费用。如何合理地安排每辆车的行驶路线，使得在完成一定运输量的情况下，交付的总费用尽可能少？

注意，在此问题中总费用仅包括携带物品通过路段时被收取的费用，车辆和路线安排上没有限制，但通过某一路段的物品数量总和不得超过它的容量，收取的费用与携带物品的多少成正比。

最小成本流问题的线性规划模型：

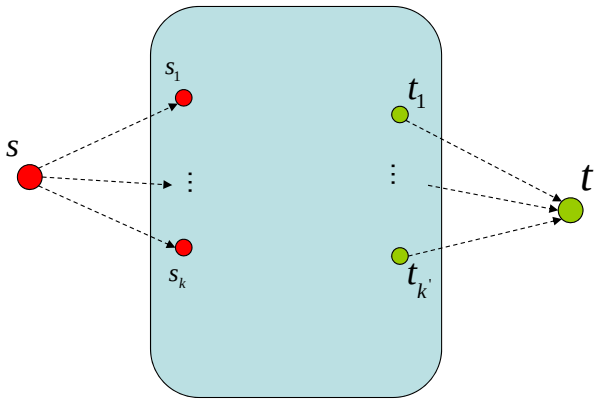
$$
\begin{aligned}
\min \quad & \sum_{e\in E} w(e)f(e) \\
\text{s.t.} \quad & \sum_{e\in \mathrm{Out}(v)} f(e) - \sum_{e\in \mathrm{In}(v)} f(e) = 0, v \in V - \{s,t\} \\
& \sum_{e\in \mathrm{Out}(s)} f(e) - \sum_{e\in \mathrm{In}(s)} f(e) = f^* \\
& \sum_{e\in \mathrm{Out}(t)} f(e) - \sum_{e\in \mathrm{In}(t)} f(e) = -f^* \\
& 0 \le f(e) \le c(e), e \in E.
\end{aligned} \tag{6.8}
$$

3.1 多个源点汇点的处理

在最小成本流问题中，可以有多个源点和汇点。如果最小成本流问题具有 k 个源点 $s_i, i = 1, \cdots, k$ 以及各自的流出量 $f(s_i) = f^*(s_i)$, 还有 k' 个汇点 $t_j, j = 1, \cdots, k'$ 以及各自的流出量 $f(t_j) = f^*(t_j)$, 则可以引入哑元源点 s 和哑元汇点 t, 并添加 $(k + k')$ 条边

$$(s, s_i): \ w(s, s_i) = 0, c(s, s_i) = f^*(s_i), i = 1, \cdots, k;$$

$$(t_j, t): \ w(t_j, t) = 0, c(t_j, t) = f^*(t_j), j = 1, \cdots, k'.$$



3.2 最短路问题转化为最小成本流问题

最小成本流问题具有很好的模型化能力：

(一) 考虑如下最短路径问题：由 s 出发，到任意点 $v \in V \setminus \{s\}$ 的最短路。
通过引入哑元终点 t', 加入从 $v \in V$ 出发的边 (v, t') 且满足

$$w(v, t') = 0, \quad c(v, t') = 1.$$

那么，最短路径问题成为求解从 s 到 t' 的具有流值 $f^* = n$ 的最小成本流问题。这里，n 是 V 的顶点个数, 并假定原网络各边的容量全为 ∞。

3.3 最大流问题转化为最小成本流问题

（二）最大流问题：通过引入哑元始点 s', 并构造如下两条边：

$$(s', s); \quad w(s', s) = 0, \quad c(s', s) = \infty$$

$$(s', t); \quad w(s', t) = 1, \quad c(s', t) = \infty$$

那么，最大流问题成为求解从 s' 到 t 的最小成本流问题。这里，设定原网络里 $w(e) = 0(e \in E)$, 并取最大流值的适当上界（比如 $\sum_{e \in E} c(e)$）为从 s' 出发的流值 f^*.

作业 6.1 通过构造说明最小成本流问题作为其特殊情况包含：最短路问题和最大流问题。

作业 6.2 考虑一个公司希望在员工与任务之间进行有效的分配。每个任务都需要特定的技能，并且每个员工都有一套技能。每个员工都有能处理的任务数的上限，每个任务需要一个员工去完成。

给定数据：

- 员工集合 $E = \{e_1, e_2, e_3\}$
- 任务集合 $T = \{t_1, t_2, t_3, t_4\}$
- 每个员工可以处理的任务数量为 $C = \{2, 1, 2\}$
- 任务分配矩阵 A 定义为：

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

其中 $A_{ij} = 1$ 表示员工 e_i 可以执行任务 t_j。

1. 描述上述数据的流网络图。
2. 使用 *Ford-Fulkerson* 算法或其他最大流算法，求出可以分配的最大任务数量。
3. 指出哪个员工应该分配哪个任务以达到最大流量。
4. 如果添加了一个新任务，它可以被 e_1 和 e_3 完成，应如何修改流网络？请指出修改后的最大流量分配。