原始-对偶方法基础

阳先毅

浙江大学计算机科学与技术学院

879946238@qq.com

2025 年 11 月 6 日

目录

运筹学中经典的原始-对偶线性规划形式

我们直接用一个非常经典的运筹学原始问题和对偶问题作为引入：

原始问题 (P):

$$
\begin{aligned}
&\min\ z = c'x \\
&Ax = b, \quad b \geq 0 \\
&x \geq 0
\end{aligned}
$$

对偶问题 (D):

$$
\begin{aligned}
&\max\ w = \pi' b \\
&\pi' A \leq c' \\
&\pi \text{ is free}
\end{aligned}
$$

Tips: $b \geq 0$ 是为方便分析而设的一般性假设。若某行 $b_i < 0$，可将其两边乘以 -1，同时翻转不等式方向，从而统一处理。

在前几节的学习中我们知道：

- 若原问题与对偶问题均有可行解，则最优值相等（强对偶定理）。
- 最优解存在的充分必要条件是满足 **互补松弛条件**：

$$
\begin{aligned}
\pi_i(a_i'x - b_i) = 0 \quad &\text{for all } i \\
(c_j - \pi' A_j)x_j = 0 \quad &\text{for all } j
\end{aligned}
$$

运筹学中经典的原始-对偶线性规划形式

如果我们能找到一个原始问题 P 的可行解 x，使得每当 $c_j - \pi' A_j > 0$ 时，都有 $x_j = 0$, 那么这个 x（以及该 π）就是最优解。
换句话说，我们可以引入一个 **允许指标集** (这里假设 A 是 mxn 矩阵)：

$$J = \{j \mid \pi' A_j = c_j\},$$

若我们能找到一组 x 满足

$$\begin{cases} \sum_{j \in J} a_{ij} x_j = b_i & i = 1, 2, ..., m \\ x_j \geq 0 & j \in J \\ x_j = 0 & j \notin J \end{cases}$$

的解, 那我们就同时找到了原问题 P 和对偶问题 D 的最优解。**原始-对偶方法的核心就在于给定某个 π 的时候，我们能否能找到这样的 x。**我们将维护 π 的可行性，利用一个辅助问题不断改进 π(和 x), 最终找到满足互补松弛的 π 和 x。

运筹学中经典的原始-对偶线性规划形式

原始对偶方法的整体框架如下图所示：

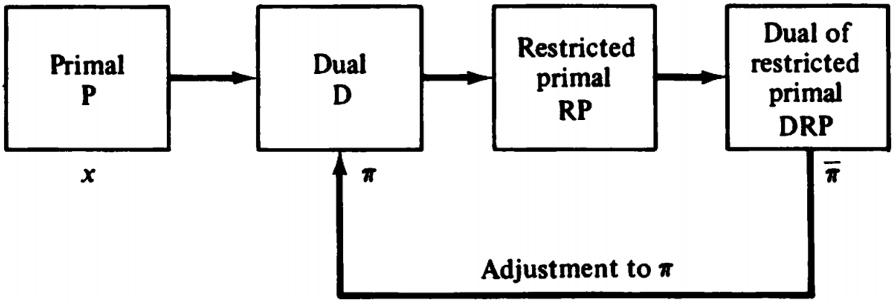


图 1: 原始-对偶框架

上图的 RP 和 DRP 就是前面提到的**辅助问题**。

运筹学中经典的原始-对偶线性规划形式

我们可以考虑稍微松弛一下 P4 的最优条件，构造新的问题：

$$
\begin{aligned}
\min\ \xi &= \sum_{i=1}^{m} x_i^a \\
\sum_{j\in J} a_{ij}x_j + x_i^a &= b_i, \quad i = 1,\dots,m \\
x_j &\ge 0, \quad j \in J \\
x_j &= 0, \quad j \notin J \\
x_i^a &\ge 0
\end{aligned} \tag{RP}
$$

显然如果 ξ_{opt}=0，我们就已经找到了原问题以及对偶问题的最优解，如果 $\xi_{opt} > 0$，那么对偶问题的解就可以被改进，我们用 RP 的对偶的最优解来改进：

$$
\begin{aligned}
&\max\ w = \pi' b \\
&\pi' A_j \le 0, \quad j \in J \\
&\pi_i \le 1, \quad i = 1,\dots,m \\
&\pi \text{ is free}
\end{aligned} \tag{DRP}
$$

运筹学中经典的原始-对偶线性规划形式

根据强对偶定理，显然后者最优解与前者一致，均大于等于 0，且在有改进空间时必然大于 0. **我们注意到它与对偶问题的目标何其一致，** 因而我们可以用它来改善对偶问题，假设 DRP 最优解为 $\bar{\pi}$

$$\pi^* = \pi + \theta\bar{\pi}$$

θ 是我们改善的倍率，由于 ${\pi^*}'b = \pi'b + \theta\bar{\pi}'b$ 且 $\bar{\pi}'b > 0$, 因而我们选择正的 θ 即可。类似单纯形法中的思路, 我们可以一直朝这个方向改进，直到对偶问题的某个可行约束变紧，约束如下

$$\pi^* A_j = \pi' A_j + \theta\bar{\pi}' A_j \leq c_j \quad \text{for all } j.$$

显然 $\bar{\pi}' A_j \leq 0$ 不会坏事，只有正的那些会使约束变紧，但如果所有 $\bar{\pi}' A_j \leq 0$ 呢？那么我们有:

定理 (Theorem 1)

If $\xi_{opt} > 0$ in (RP) and the optimal dual satisfies

$$\bar{\pi}' A_j \leq 0 \quad \text{for all } j \notin J,$$

then P is infeasible.

运筹学中经典的原始-对偶线性规划形式

上一页定理成立是显然的，对于 $j \in J$,DRP 中也保证非正。现在我们真正关心的是 θ 最大为多少，当然就是第一个紧的约束所限了：

定理 (Theorem 2)

When $\xi_{opt} > 0$ in (RP) and there exists a $j \notin J$ such that $\bar{\pi}' A_j > 0$, the largest θ that maintains feasibility of $\pi^* = \pi + \theta\bar{\pi}$ is

$$\theta_1 = \min_{\substack{j \notin J \\ \bar{\pi}' A_j > 0}} \left[\frac{c_j - \pi' A_j}{\bar{\pi}' A_j} \right]$$

The new cost is

$$w^* = \pi' b + \theta_1 \bar{\pi}' b = w + \theta_1 \bar{\pi}' b > w.$$

我在此稍作总结，原始对偶方法如下：

- 首先，找到一个对偶可行解。(大部分情况令 π=0 即可，少部分引入一个额外变量)
- 我们求解 RP 或 DRP，如果 $\xi > 0$，依次对 D 解进行优化
- 如果利用**单纯形法**解决 RP，我们将最终达到两种可能结果
  - 发现原问题 infeasible；
  - 达到无法优化的地步，解决！

原始对偶方法的有限性（总体不重要）

思考一下我们的求解过程，实际上就是一个基的迭代过程，我们在每一次都会加入一个新的可行列，并有可能会淘汰掉某些可行列。我们在 RP 问题中实际上就是不断从一个 bfs 过渡到另一个 bfs 。
bfs 是否会在迭代中重复是一个重要的问题，比如说基变量（也可以说基列）：

$$\{\pi_1, \pi_2\} \to \{\pi_3\} \to \{\pi_1, \pi_3\} \to \{\pi_1, \pi_2\} \ldots$$

类似于单纯形法，我们可以采用防循环规则（如 Bland’s rule）来避免基的重复。这样由于基的组合是有限的，我们可以在有限步内求解，否则有可能会陷入基的循环…… 我们在后面会看到。

原始-对偶最短路径算法

原始-对偶方法一个最简单的例子就是运用于最短路径算法。
考虑一个有向图 $G = (V, E)$，其中包含 m 个节点和 n 条弧。我们定义如下线性规划问题（原始问题）：

$$
\begin{aligned}
\min \quad & c^T f \\
\text{s.t.} \quad & Af = \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \leftarrow \text{对应源点} s \text{ 的行} \\
& f \geq 0
\end{aligned}
\tag{P}
$$

其中：

- A 是一个 $(m-1) \times n$ 的节点-弧关联矩阵，对应于去掉终点 t 的行（该行冗余）；
- $f \in \mathbb{R}^n$ 是流量向量，表示每条弧上的流（在此问题中取值为 0 或 1）；
- $c \in \mathbb{R}^n$ 是费用向量，表示每条弧的权重或距离。

该问题等价于在图中从源点 s 到终点 t 寻找一条最小费用路径。

原始-对偶最短路径算法

$$
\begin{aligned}
\max \quad & \pi_s \\
\text{s.t.} \quad & \pi_i - \pi_j \leq c_{ij} \qquad \forall (i,j) \in E \\
& \pi_i \geq 0 \qquad\qquad \forall i \\
& \pi_t = 0
\end{aligned} \tag{D}
$$

同样，我们记可行边为那些紧的边集：

$$J = \{\text{arcs } (i,j) : \pi_i - \pi_j = c_{ij}\}$$

在这些紧边构成的子图上，我们定义一个受限原始问题，即只允许在这些边上传输流量：

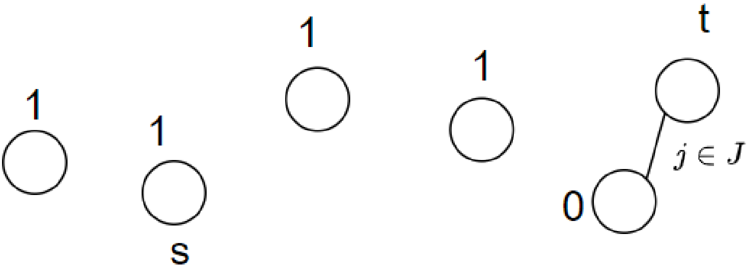
$$
\begin{aligned}
& \min \ \xi = \sum_{i=1}^{m-1} x_i^a \\
& \text{s.t.} \quad Af + x^a = \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \leftarrow \text{Row } s \\
& \qquad f_j \geq 0, \quad \forall j \\
& \qquad f_j = 0, \quad j \notin J \\
& \qquad x_i^a \geq 0
\end{aligned} \tag{RP}
$$

原始-对偶最短路径算法

接着我们写出受限原始问题的对偶：

$$
\begin{aligned}
& \max\ w = \pi_s \\
\text{s.t.}\quad & \pi_i - \pi_j \le 0 \quad && \forall (i,j) \in J \\
& \pi_i \le 1 && \forall i \\
& \pi_i \text{ is free} && \forall i
\end{aligned}
\tag{DRP}
$$

我们注意到 π_s, 必然小于等于 1，所以它只能取 0 或 1. 我们仅当取 1 时才能进行优化，那么我们尝试将 π_s 取为 1，显然如果没有一条仅仅使用 J 中的边构成的从 s 到 t 的通路的话，我们是一定能取到 1 的。如下所示：



原始-对偶最短路径算法

对 $\forall(i,j) \notin J$, 改进空间为 $c_{ij} - (\pi_i - \pi_j)$, 最先紧的必然会作为约束，即

$$\theta_1 = \min_{\substack{\text{arcs } (i,j)\notin J \\ \text{such that } \bar{\pi}_i - \bar{\pi}_j > 0}} \{c_{ij} - (\pi_i - \pi_j)\}$$

Tips: 分母实际上是 $\bar{\pi}_i - \bar{\pi}_j = 1$, 另外 $\bar{\pi}$ 本身也不是唯一的。

我们从某个对偶可行解开始，不断执行上述操作，直到我们求解的 $\xi_{opt} = 0$, 说明 s 和 t 此时仅用 $j \in J$ 的边就可以相连，这样我们就求解出了最优解。以下是一个所有边权都为正整数的例子：

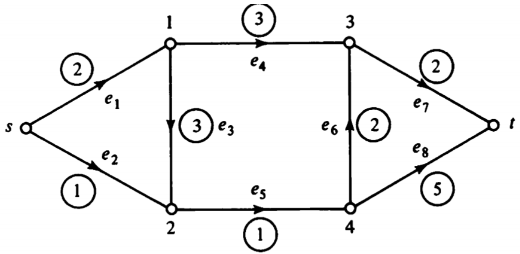


图 2: 最优解（不唯一）示意图

原始-对偶最短路径算法 (eg)

这个例子最好的一点是我们可以直接找到一个可行的对偶解，即令 $\pi = 0$，然后我们便可如下求解：

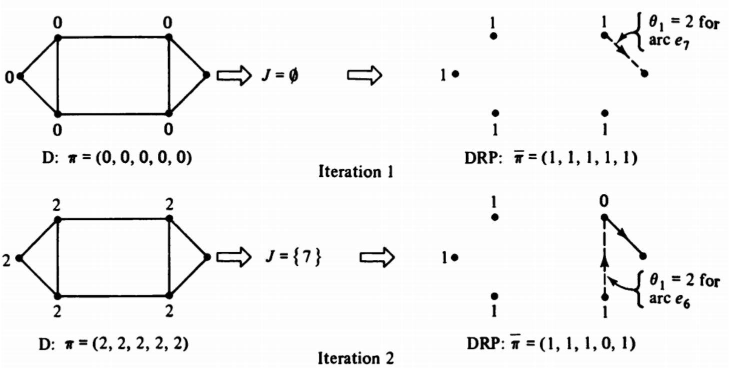


图 4: 简单例子

原始-对偶最短路径算法 (eg 续)

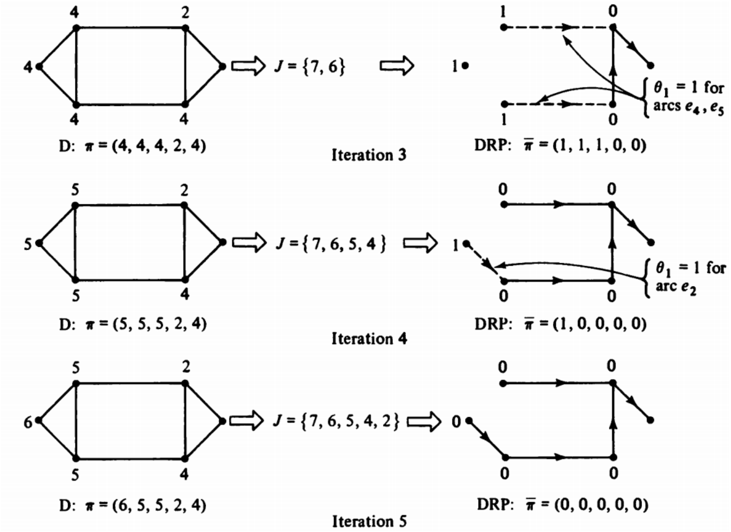


图 5: 简单例子

Dijkstra 算法

我们乍一看就会觉得原始对偶算法是从终点到起点的 dijkstra，但实际上应该这样理解：**Dijkstra 是在初始可行对偶解为 0 的情况下的原始对偶算法。**

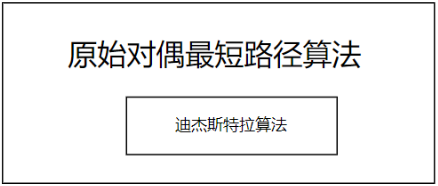


图 6: 简单例子

对于一个从起点执行 Dijkstra 算法的图 (非负权重)，我们可以将所有边的方向反转，保持边的 cost 不变，那么我们就可以从起点（反转后是终点）使用原始对偶方法，即 Dijkstra 算法。显然反转前后从 s->t 和从 t->s 的路径是一一对应的，因此我们就说明了 Dijkstra 算法是一种特殊条件下的原始对偶方法。

Floyd-Warshall 算法 (不重要，与原始-对偶无关且学过)

Dijkstra 算法可以认为是特殊情况下对原始对偶方法的简化，但是 Dijkstra 算法会在遇到负权的时候出错，为什么呢？因为 Dijkstra 已经默认是从 $\pi = 0$ 开始更新的了，但是在有负权边的情况下 $\pi = 0$ 也许并不是初始对偶可行解。

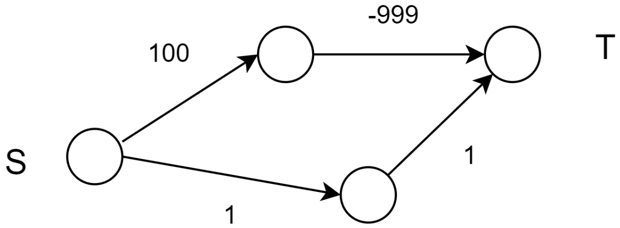


图 7: 另一个简单例子

显然如果令所有 $\pi = 0$ 的话，这里 cost 为-999 的边违反了对偶可行性。

Floyd-Warshall 算法 (不重要，与原始-对偶无关且学过)

Floyd-Warshall 算法的一个关键操作是 “三角操作”，即

$$d_{ik} = min\{d_{ik}, d_{ij0} + d_{j0k}\}$$

有以下定理：

定理 (Theorem 3)

如果我们对 $j = 1, 2, \ldots, n$ 依次执行三角操作，那么每个条目 d_{ik} 最终会等于从节点 i 到节点 k 的最短路径长度，假设所有边权 $c_{ij} \geq 0$。

FLOYD-WARSHALL ALGORITHM

Input: An $n \times n$ matrix $[c_{ij}]$ with nonnegative entries.

Output: An $n \times n$ matrix $[d_{ij}]$, where d_{ij} is the shortest distance from i to j under $[c_{ij}]$.

begin
  for all $i \neq j$ **do** $d_{ij} := c_{ij}$;
  for $i = 1, \ldots, n$ **do** $d_{ii} := \infty$;
  for $j = 1, \ldots, n$ **do**
    for $i = 1, \ldots, n, i \neq j$, **do**
      for $k = 1, \ldots, n, k \neq j$, **do**
        $d_{ik} := \min\{d_{ik}, d_{ij} + d_{jk}\}$
end

图 8: Floyd-Warshall 算法

最大流问题

我们考虑一个五元组 $N=(s,t,V,E,b)$, 其中 n=|V| 即顶点数量，m=|E| 即边数，b 是每条边的容量，我们可以将最大流问题写做：

$$\begin{aligned} \max \quad & v \\ \text{s.t.} \quad & Af+dv=0 \\ & f\le b \\ & f\ge 0 \end{aligned} \tag{D}$$

其中 $d\in\mathbb{R}^n$ 是定义如下的向量：

$$d_i=\begin{cases} -1, & i=s \\ +1, & i=t \\ 0, & \text{其他} \end{cases}$$

Tips:由于这个问题是 max 问题，所以本身就可以把这个问题看做标准原始-对偶框架下的一个 D 问题，接下来，我们尝试写出它的原始问题 P。

最大流问题

注意到等式约束有 |v|=n 个，所以我们引入 n 个关于顶点的对偶变量 $\pi(x)$；同理，我们引入 m 个关于边的对偶变量 $\gamma(y)$，可写出如下原问题

$$
\begin{aligned}
\min_{\pi,\gamma} \quad & \sum_{(x,y)\in E} \gamma(x,y)\cdot b(x,y) & \\
\text{s.t.} \quad & \pi(x)-\pi(y)+\gamma(x,y)\geq 0 & \forall (x,y)\in E \\
& -\pi(s)+\pi(t)\geq 1 & \\
& \pi(x) \text{ is free} & \forall x\in V \\
& \gamma(x,y)\geq 0 & \forall (x,y)\in E
\end{aligned}
$$

幸运的是，这个原问题其实有图上的含义：

定义 (Definition)

An s-t cut is a partition $(W,\bar{W})$ of the nodes of V into sets W and $\bar{W}$ such that $s\in W$ and $t\in \bar{W}$. The capacity of an s-t cut is

$$
C(W,\bar{W}) = \sum_{\substack{(i,j)\in E \\ i\in W,\ j\in \bar{W}}} b(i,j)
$$

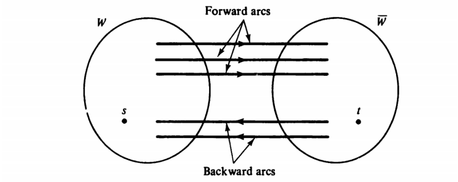


图 9: 一个割的示意图

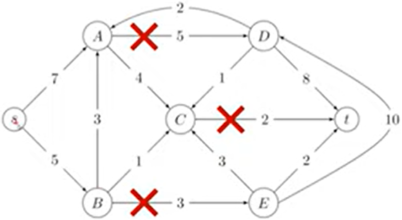


图 10: 上图的展示一个割，将所有顶点划分为 $W = \{s, A, B, C\}$ 与 $\bar{W} = \{D, E, t\}$,capacity=10

割与最大流的对偶（即上面的 P）的对应关系

我们发现每个割都对应一个 P 的可行解，有以下定理：

定理 (Theorem)

Every s-t cut determines a feasible solution with cost $C(W, \bar{W})$ to the dual of max-flow as follows:

$$\gamma(x, y) = \begin{cases} 1, & (x, y) \text{ such that } x \in W,\ y \in \bar{W} \\ 0, & \text{otherwise} \end{cases}$$

$$\pi(x) = \begin{cases} 0, & x \in W \\ 1, & x \in \bar{W} \end{cases}$$

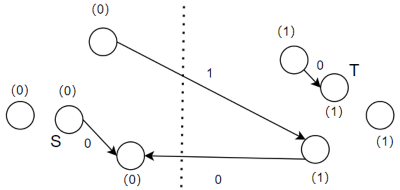


图 11: 我们可以一一验证问题 P 约束成立

最大流最小割定理

并且我们获得的流的大小确实是 $C(W, \bar{W})$

定理 (Theorem (Max-flow, min-cut))

The value v of any s-t flow is no greater than the capacity $C(W, \bar{W})$ of any s-t cut. Furthermore, the value of the maximum flow equals the capacity of the minimum cut, and a flow f and cut $(W, \bar{W})$ are jointly optimal if and only if

$$\begin{aligned} f(x,y) &= 0 && \text{for } (x,y) \in E \text{ such that } x \in \bar{W}, y \in W \\ f(x,y) &= b(x,y) && \text{for } (x,y) \in E \text{ such that } x \in W, y \in \bar{W} \end{aligned} \tag{6.6}$$

我们可以写出原问题和对偶问题的间隙，对 $(x, y) \in E$:

$$\begin{aligned} &\gamma(x,y)b(x,y) \\ &\geq (\pi(y) - \pi(x))\, b(x,y) \quad \text{找到 } \gamma \text{ 对应的那些 } \pi \\ &\geq (\pi(y) - \pi(x))\, f_{x,y} \end{aligned}$$

最大流最小割定理

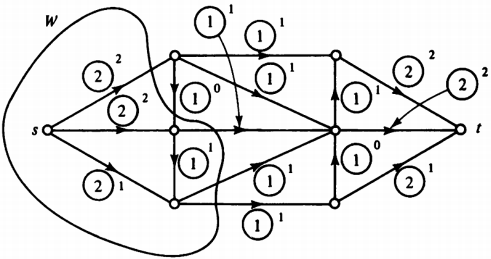
对于每条弧来说，我们希望

$$\gamma(x, y) b(x, y) = (\pi(y) - \pi(x)) f_{x,y}$$

对于完全在 W 或 $\bar{W}$ 内的弧显然成立

- 对于 $(x, y) \in E$ s.t. $x \in \bar{W}, y \in W, \gamma(x, y) = 0$, 要使等式成立必须使 $f_{x,y} = 0$
- 对于对于 $(x, y) \in E$ s.t. $x \in W, y \in \bar{W}, \gamma(x, y) = 1$，要使等式成立必须 $f_{x,y} = 1$。

以上实际上就是说明了互补松弛条件，下图是一个例子



Ford And Fulkerson 算法

总而言之，我们希望找到一种划分，如果这种划分上的流都满足前面定理所述，那么我们就找到了 maxflow 的最优解。FF 算法的优点就在于，我逐渐的去增加路径上的流，直到增加不了时，我就可以以**某种方式**对所有节点形成划分，这个划分上的流确实就满足定理了。(不过不能再增加流似乎更加直观地就告诉我们达到最大流了)
我们先定义一种路径（增广路径）用于增加流：

定义 (增广路径（Augmenting Path）)

给定一个可行流 f，一条从源点 s 到汇点 t 的 增广路径 是在忽略边方向的图中的一条路径，满足：

1. 对于每条正向经过的弧 (i, j)（称为 **前向弧**)，有
$$f(i, j) < b(i, j)$$
即该边未饱和，仍有剩余容量可用于增流。
2. 对于每条反向经过的弧 (j, i)（即原图中的 (i, j) 被逆向使用，称为 **后向弧**)，有
$$f(j, i) > 0$$
即该边已有正流量，可以“退流”以释放上游容量。

Ford And Fulkerson 算法

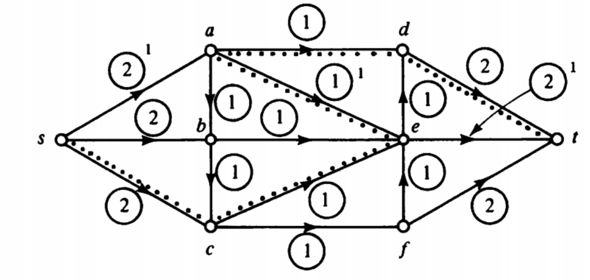


图 13: 增广路径的例子

首先让我们进一步将其 FF 算法思想与原始对偶框架联系一下：

- 我们从 flow=0 开始增加流: 这对应从一个对偶可行解开始（记住我们的原始问题就是对偶问题）。
- 我们通过增广路径对对偶可行解进行逐步优化（但我们并不像单纯形法一样用最优基优化）
- 我们最后到达互补松弛条件，意味着有最优解。

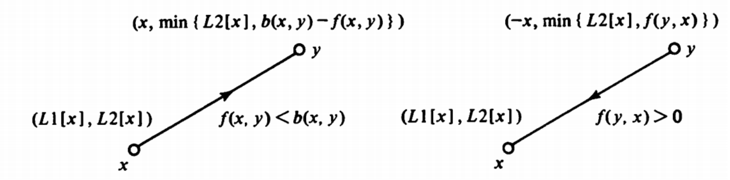
Ford And Fulkerson 算法

在一条增广路径上我们一次最多可以优化多少呢?

$$\delta = \min_{\text{arcs of } P} \begin{cases} b(i,j) - f(i,j), & \text{if } (i,j) \text{ is a forward arc} \\ f(j,i), & \text{if } (j,i) \text{ is a backward arc} \end{cases}$$

回到更基础的问题，我们怎么找增广路径呢？尤其是在图上已经有流的情况下找增广路径，那么 FF 算法采用了“标签传播”法：

- 我们每一轮从源点 s 开始传播。
- 每个点有标签 (L1,L2), 我们将标签向正向有剩余流量的边或者反向有剩余流量的边传播。(下图所示)
  - 如果可以传播到 t，我们就以该条路径作为增广路径。
  - 如果不能传播到 t, 我们可以据此构造一个割证明达到最优。



```
FORD AND FULKERSON ALGORITHM
Input:   A network N=(s,t,V,A,b)
Output: The maximum flow f of N.
begin
      f := 0; (comment: initialize flow)
again:  set all labels to 0, set LIST := {s}, set L2[s] := ∞;
        (comment: initialize for the search for new augmentation path)
        while LIST ≠ ∅ do
        begin
           let x be any node in LIST;
           remove x from LIST;
           scan x;
           if t is labeled then
              begin
                 augment flow f along augmentation path;
                 go to again
              end
        end
end

procedure scan
   begin
      label forward to all unlabeled nodes adjacent to x by arcs that
      are unsaturated, putting newly labeled nodes on LIST;
      label backward to all unlabeled nodes from which x is adjacent
      by arcs that have positive flows, putting newly labeled nodes
      on LIST;
   end
```

Ford And Fulkerson 算法

我们将证明当 Ford And Fulkerson 算法结束时一定找到了最优解。

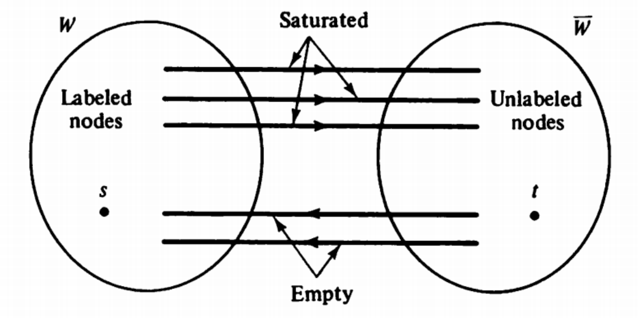


图 16: 算法结束时划分出来的顶点集合

我们记算法结束前最后一次传播过程中标记的节点集合为 W，未标记的节点记为 $\bar{W}$, 下面我将说明这确实构成割的定义：

- 前向弧必须满了，否则我们一定会通过前向弧标记 $\bar{W}$ 中的节点。
- 后向弧必须没有流，否则我们也会通过后向弧标记 $\bar{W}$ 中的节点。

Ford And Fulkerson 算法局限性 (不重要)

我们说这个算法结束时就可以得到最优解，但一个关键是算法能结束吗，使用原始-对偶方法能保证我们每步都能对 DUAL 目标函数严格提升：

- 如果提升都能保证提升大于某个常数 $c>0$, 自然能在有限步结束。
- 否则算法会陷入永远的循环，如下是一个示意。

下面是一个著名的 **非终止示例**，由 Ford 和 Fulkerson 提出，揭示了该算法在实数容量下的潜在问题。

构造一个递推数列：

$$a_{n+2} = a_n - a_{n+1}$$

初始条件为：

$$a_0 = 1, \quad a_1 = \sigma = \frac{\sqrt{5}-1}{2} < 1$$

可以通过数学归纳法证明：

$$a_i = \sigma^i \quad \text{for } i = 0, 1, 2, \ldots$$

Ford And Fulkerson 算法局限性续 (不重要)

如果我们会按照如下方式增广：

- 第 0 步：增广量为 $a_0 = 1$
- 第 n 步（$n \geq 1$）包含两次增广：
  - 增广 (a)：增加 a_{n+1}
  - 增广 (b)：增加 a_{n+2}

这导致的总流量会趋于极限：

$$a_0 + (a_2 + a_3) + (a_4 + a_5) + \cdots = a_0 + a_1 + a_2 + \cdots = \frac{1}{1-\sigma} = S$$

我在后一页会展示一张精心构造的图，它会满足以下性质：

- (x_1, y_1) 容量是 a_0,(x_2, y_2) 容量是 a_1,(x_3, y_3) 容量是 a_2,(x_4, y_4) 容量是 a_3
- 其他所有边的容量都是 S.

Ford And Fulkerson 算法局限性续 (不重要)

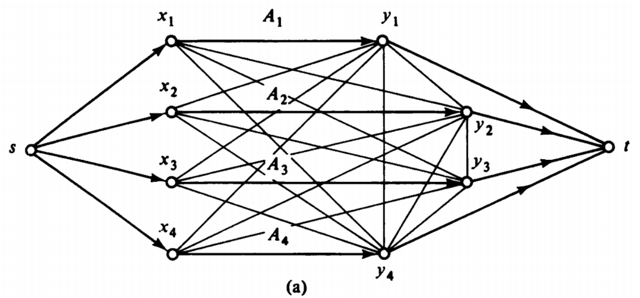


图 17: 特殊的图

我最开始增广 (x_1, y_1)，则 $\{(x_i, y_i)\}$ 这一组边的剩余容量是 $(0, a_1, a_2, a_2)$

我第 n 步的增广分为两个阶段，此时 $\{(x_i, y_i)\}$ 残余容量为 $(0, a_n, a_{n+1}, a_{n+1})$：
第一阶段： 我按照如下方式增广，$\{(x_i, y_i)\}$ 这组边的残余容量是 $(0, a_{n+2}, 0, a_{n+1})$

Ford And Fulkerson 算法局限性续 (不重要)

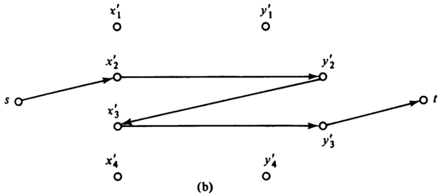


图 18: n 轮第一阶段增广

第二阶段： 我按照如下方式增广，$\{(x_i, y_i)\}$ 这组边的残余容量是 $(a_{n+2}, 0, a_{n+2}, a_{n+1})$

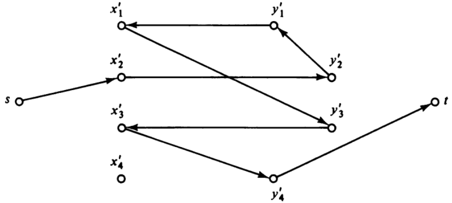


图 19: n 轮第二阶段增广

Ford And Fulkerson 算法局限性续 (不重要)

可见我们选择增广的路径终究会形成循环，我们并没有尝试更多更有效的路径，而这个问题的最优解显然是 4S，我们却只能收敛到了 $1/4$ OPT 的结果。

Tips: 可见经典的 FF 算法并没有发现一种解决路径循环的范式，进一步说，当 RP 是退化的时候，我们应该找到一种更好地增广方式 (而 Chapter9 才会说且与原始-对偶也无关了)。

最小费用流的构造

设 $N=(s,t,V,E,b)$ 是一个流网络，其底层有向图为 $G=(V,E)$，每条弧 $(i,j)\in E$ 上有一个代价权重 $c_{ij}\in\mathbb{R}^+$，给定目标流量值 $v_0\in\mathbb{R}^+$。

最小费用流问题: 求一个可行的 s-t 流 f，其总流量为 v_0，且总成本最小。同样可以表示成线性规划形式:

$$
\begin{aligned}
\min\quad & c'f & \\
\text{s.t.}\quad & Af=-v_0\,d & \text{每个节点}\\
& f\le b & \text{每条弧}\\
& f\ge 0 & \text{每条弧}
\end{aligned}
$$

其中:

$$
d_i=\begin{cases}-1, & i=s\\ +1, & i=t\\ 0, & \text{otherwise}\end{cases}
$$

最小费用流的构造

注意到原始对偶框架的核心是**我们需要找到一个简单的 DRP 可行解对 D 优化**，这个问题我们不必强求写出它的对偶问题并写出 DRP, 而是直接将其视作一个 DUAL（由于流守恒，我们可以改写等式）：

$$\begin{aligned} \max \quad & -c'f & & \\ \text{s.t.} \quad & Af \leq -v_0 d & & \text{每个节点} \\ & f \leq b & & \text{每条弧} \\ & -f \leq 0 & & \text{每条弧} \end{aligned} \tag{D}$$

接下来我们可以简洁地写出 DRP(我们同样可以改写不等式)：

$$\begin{aligned} \max \quad & -c'f & & \\ \text{s.t.} \quad & Af = 0 & & \text{每个节点} \\ & f \leq 0 & & \text{饱和弧} \\ & f \geq 0 & & \text{空弧} \\ & f \geq -1 & & \text{所有弧（因为} -c \leq 0) \end{aligned} \tag{DRP}$$

最小费用流的构造

我们注意到 DRP 的形式是极好的，$Af = 0$ 意味着一个圈，因而 DRP 的目标就在于找到一个圈，使得

- 对于满的边，我只能反向流。
- 对于正的边，我只能正向流。

定义 (增量网络)

给定一个带权网络 $N = (V, E)$ 和其上的一个可行流 f，我们定义该网络的增量网络 $N'(f) = (V, E')$ 如下：

- 每条弧 $e = (u, v) \in E$ 在 $N'(f)$ 中对应一条或两条弧：
  - 如果 $f(e) < c(e)$（即 e 未饱和），则在 E' 中包含前向弧 (u, v)，其容量为 $c(e) - f(e)$，成本为 $w(e)$。
  - 如果 $f(e) > 0$（即 e 非空），则在 E' 中包含后向弧 (v, u)，其容量为 $f(e)$，成本为 $-w(e)$。
- 所有容量为 0 的弧从 E' 中删除。

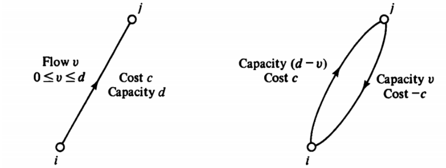


图 20: 这个前向弧和后向弧的定义实际上和 MAX FLOW 中是一样的

定义 (增广圈)

在增量网络 $N'(f)$ 中，如果存在一个圈 C，其所有边的总成本小于或等于 0，则称 C 是一个增广圈。通过沿这个圈调整流量，可以使整个网络的成本降低或保持不变。

因此 DRP 的意思就是说我们找一个增广圈，如果找得到 $\xi < 0$ 的增广圈，那么我们目标函数还能优化，否则就不能优化了。

最小费用流的构造——增广圈

因而我们可以发现一个最朴实无华的算法：

```
procedure cycle
begin
   use the max-flow algorithm to find a flow of value v₀;
   while there is a negative-cost cycle C in N′ do
         augment flow on C until N′ no longer contains C
end
```

图 21: 最小费用流的原始-对偶方法

当然，我们还可以从某种“最小 cost 增广路径”的方式成长式地来看待这个问题

定理（最优保持）

设f_1 是值为v 的最小费用流。

设f_2 是在$N'(f_1)$ 中沿一条s-t 增广路径P 上的单位流，且P 是成本最小的。

则$f_1 + f_2$ 是值为$v + 1$ 的最优流。

最小费用流的构造——增广圈

简单证明一下这个定理:

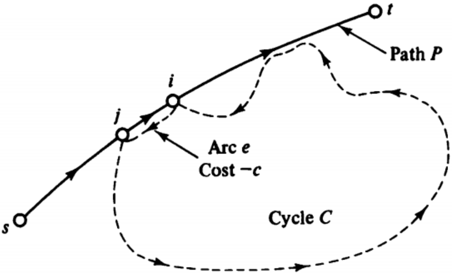


图 22: 从圈中构造路径证明问题

假如增量网络 $N'(f_1)$ 再加入了 f_2 形成了增量网络 $N'(f_1+f_2)$ 从而不是最优流了，那么必然存在增广圈，且这个增广圈某一段路径必然在 f_2 路径 P 上，那么由于圈 C 的 cost<0, 我们与其经过这段路径不如直接经过这个圈构造 f_2，使得费用更低, 因此 f_2 不是最优，矛盾。

最小费用流的构造——增广圈

那么这种算法就可以总结如下：

```
procedure buildup
begin
   while flow f < v0 do
      begin find a shortest path P from s to t in N';
               augment the flow along P until it
               reaches v0 or until P is no longer
               a least-cost augmentation path
      end
end
```

图 23: build-up 形式的最小费用流

Hitchcock 问题

Hitchcock 问题是一个非常经典的运筹学问题，我有 m 个厂家和 n 个需求点，但是我从厂家到需求点有不同的运输成本，应该怎么样才能最小化满足需求时的运输成本呢？

Tips: 供需都是 1 的时候就是指派问题。

我们可以把这个问题表示如下：

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} f_{ij} & \\
\text{s.t.} \quad & \sum_{j=1}^{n} f_{ij} = a_i, & \forall i = 1, \ldots, m \\
& \sum_{i=1}^{m} f_{ij} = b_j, & \forall j = 1, \ldots, n \\
& f_{ij} \geq 0, & \forall i, j
\end{aligned}$$

$$s.t. \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

Hitchcock 问题

我们可以在不损失问题表达力的情况下把等式改成不等式：

$$\sum_{j=1}^{n} f_{ij} \leq a_i, \quad \forall i = 1, \ldots, m$$

$$\sum_{i=1}^{m} f_{ij} \geq b_j, \quad \forall j = 1, \ldots, n$$

可以这样理解：

- (7.6) 表示：每个厂家 i 的总发货量不能超过其供应能力 a_i
- (7.7) 表示：每个需求点 j 的总收货量必须至少满足其需求 b_j

我们可以引入一个 **虚拟需求点**（fictitious terminal）来恢复平衡。

设总供应为 $A = \sum_{i=1}^{m} a_i$，总需求为 $B = \sum_{j=1}^{n} b_j$
若 $A > B$，则引入第 $n+1$ 个虚拟需求点，令：

$$b_{n+1} = A - B \quad \text{且} \quad c_{i,n+1} = 0 \quad \forall i = 1, \ldots, m$$

其实就是分配不掉的以 0 成本扔掉。

我们写出它的对偶问题如下：

$$
\begin{aligned}
\max \quad & w = \sum_{i=1}^{m} a_i\alpha_i + \sum_{j=1}^{n} b_j\beta_j \\
\text{s.t.} \quad & \alpha_i + \beta_j \le c_{ij}, && \forall i = 1, \ldots, m;\ j = 1, \ldots, n \\
& \alpha_i \ge 0, && \forall i = 1, \ldots, m \\
& \beta_j \ge 0, && \forall j = 1, \ldots, n
\end{aligned}
\tag{D}
$$

我们目前只需要知道 α 是关于源点约束的对偶变量，β 是关于需求点的对偶变量即可。很容易发现一个对偶问题的基础可行解，即 $\alpha_i = 0$,$\beta_j = \min_{1\le i\le m} c_{ij}$

定义可接受集合（Admissible Set）：

$$IJ = \{(i, j) : \alpha_i + \beta_j = c_{ij}\}$$

Hitchcock 问题

$$
\begin{aligned}
\min \quad & \xi = \sum_{t=1}^{m+n} x_t^\star \\
\text{s.t.} \quad & \sum_j f_{ij} + x_i^\star = a_i, && i = 1, \ldots, m \\
& \sum_i f_{ij} + x_{m+j}^\star = b_j, && j = 1, \ldots, n \\
& x_i^\star \geq 0, && i = 1, \ldots, m+n \\
& f_{ij} \geq 0, && (i,j) \in IJ \\
& f_{ij} = 0, && (i,j) \notin IJ
\end{aligned}
\tag{RP}
$$

注意到只有那些在 IJ 集合里的才有正的 f_{ij}，于是我们把目标函数写作：

$$
\xi = \sum_{i=1}^{m} a_i + \sum_{j=1}^{n} b_j - 2 \sum_{(i,j)\in IJ} f_{ij}
$$

Hitchcock 问题

上面问题的最优解其实等价于下面这个问题的最优解：

$$
\begin{aligned}
\max \quad & \sum_{(i,j)\in IJ} f_{ij} \\
\text{s.t.} \quad & \sum_j f_{ij} \le a_i, \quad i = 1, \ldots, m \\
& \sum_i f_{ij} \le b_j, \quad j = 1, \ldots, n \\
& f_{ij} \ge 0, \qquad (i,j) \in IJ \\
& f_{ij} = 0, \qquad (i,j) \notin IJ
\end{aligned}
\tag{RP'}
$$

非常奇妙的是，这个问题正是一个最大流问题，如后一页所示：

Hitchcock 问题

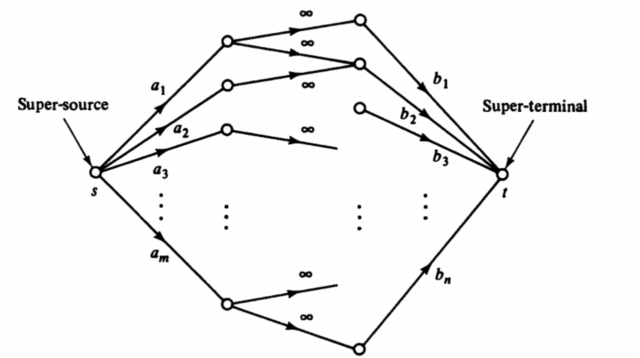


图 24: 从 RP 构造出来的最大流

- 引入了超级源点和超级终点。
- 用边的约束替代了源点和终点的约束。

Hitchcock 问题

定理 (Theorem)

在使用标签传播算法解决 *RP(*一个最大流问题*)* 后，我们可以利用标签构造出一个对应的 *DRP* 解（即一系列的 $\bar{\alpha}$,$\bar{\beta}$），构造方式略去，详见 *Combinatorial Optimisation,Peter Butkovic,P49*

总之，我们已经找到了求解 Hitchcock 问题的原始对偶方法：

- 首先找到 DUAL 的初始可行解 α_0, β_0
- 利用最大流算法求解 RP'，并构造对应 DRP 解，以此改进 D。
- 算法结束时我们就找到了最优解。

总结

原始对偶方法的思想是什么呢？其实就是抓住了某些 DRP 问题是好解的这一特征，于是我们我们可以多次求解这个结构更简单、更加离散的“小问题”，来使 D 逐渐逼近最优。

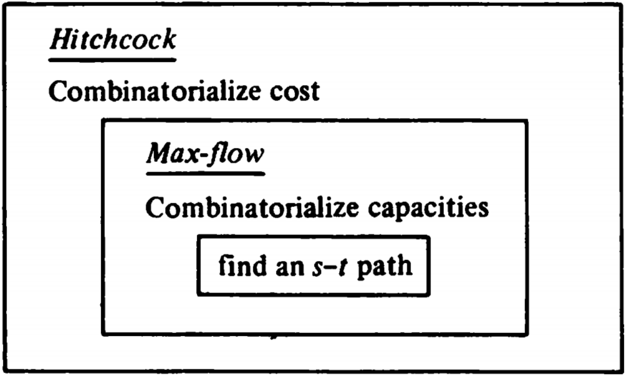


图 25: 最最最高屋建瓴的一张图，概括了原始-对偶核心思想以及其与组合优化的关系