组合最优化

（第 5 章-1）

中国科学院研究生院
数学科学学院

郭田德

电话：88256412
Email: tdguo@ucas.ac.cn

第五章 组合优化问题的原始-对偶算法-1

5.1 最优树和最优路

给定连通图$G=(V,E)$，记$|V|=n,|E|=m$。$\forall e\in E$，给定费用c_e。在所有复杂性估计中，总是假设$n=O(m)$且$m=O(n^2)$。（在我们所考虑的问题中，不满足这个假定的情况是平凡的）

记号：对图$G=(V,E)$和$A\subseteq V$，**记两个与顶点集合A有关的边的集合**：

$\delta(A)=\{e\in E: e$的一个端点在A中，而另一个端点在$V\setminus A$中$\}$；

$\gamma(A)=\{e\in E: e$的两个端点都在A中$\}$。

对于$A\subseteq V$且$A\neq\phi$，称$\delta(A)$为图$G=(V,E)$的一个**割集**。

最小生成树问题（MST）：找到$G=(V,E)$的一棵最小费用的生成树。

MST 算法：

1. 基本思路：用“贪婪”的思想来构造算法

   (1) Kruskal 算法的基本思路：保持$G=(V,E)$的一个**生成森林**$H=(V,F)$，初始化$F=\phi$。在每一步往F中加一条最小费用边$e\notin F$，并保持$H=(V,F)$是森林，当$H=(V,F)$是生成树为止。

   (2) Prim 算法的基本思路：保持**一棵树**$H=(V(H),T)$，对某个$r\in V$，初始化$V(H)=\{r\}$，$T=\phi$。在每一步往T中加一条不在T中的最小费用边$e\notin F$，并保持$H=(V(H),T)$是一棵树，当$H=(V(H),T)$是生成树为止。

2. MST 算法的正确性

定理 5.1 对具有任意边费用c的任何连通图$G=(V,E)$，Kruskal 算法和 Prim 算法都可找到一棵最小生成树。

3. MST 算法的具体实施

（1） 在计算机上存储图$G=(V,E)$的标准方法：对每个$v\in V$，保存一个列表L_v，它包含与v关联的所有边，以及每条边的另一个端点，并保存这条边的费用。

注意：由于每条边都有两个端点，所以每条边和费用在两个不同的列表中被保存了两次。

（2） Prim 算法：

将$H=(V(H),T)$初始化为$\{\{r\},\phi\}$；

While H不是生成树

从$\delta(V(H))$中选一条最小费用边添加到T中。

具体实施：

将$V(H)$保存为一个描述端点状态的特征向量$\{x_v : v\in V\}$，其中

$$x_u=\begin{cases}1, & u\in V(H)\\ 0, & u\notin V(H)\end{cases},$$

在每一步遍历E，对给定的$f=uv\in E$，如果$x_u\neq x_v$，则端点u或v有且只有一个在$V(H)$中，从而$f\in\delta(V(H))$，将c_f与当前遇到的最小值比较，遍历完E后，将最小费用边$f^*=u^*v^*$添加到T中，然后更新特征向量$\{x_v : v\in V\}$，如果$x_{u^*}=0$，则取$x_{u^*}=1$，否则，取$x_{v^*}=1$。所以一步中需要比较$O(m)$次，总共需要n步。所以，算法的计算复杂度为$O(nm)$。

思考题：如何将 Prim 算法改进，使得算法的计算复杂度为$O(n^2)$？（Tarjan, 1983）。

（3） Kruskal 算法：

将E按费用由小到大排序为$\{e_1,e_2,\cdots,e_m\}$；

将$H=(V,F)$初始化为(V,ϕ)；

For $i=1$ to m

If e_i的端点在H中的不同分支中

把e_i添加到F中。

算法计算复杂度：

第一步骤排序：算法复杂度为$O(m\log m)$；

第二步骤的计算复杂度为$O(m\log n)$。

所以 Kruskal 算法的计算复杂度为$O(m\log m)$。

思考题：给出第二步骤的计算复杂度为$O(m\log n)$具体实施过程。

4. 最小生成树的线性规划模型

对任意集合 A，向量 $p\in R^{A}$ 及任意 $B\subseteq A$，记：$p(B)=\sum_{j\in B}p_j$。

$$\begin{aligned}&\min c^T x\\&s.t.\ x(\gamma(S))\le |S|-1, \text{对所有} S,\phi\ne S\subset V\\&\quad x(E)=|V|-1\\&\quad x_e\ge 0, \text{对所有} e\in E\end{aligned}\qquad (5.1)$$

令 $\phi\ne S\subset V$，T 是一棵生成树的边集，x^0 是 T 的特征向量，则 $x^0(\gamma(S))=|T\cap\gamma(S)|$，且由于 T 是一棵树，不含圈，它至多为 $|S|-1$。又有 $x^0\ge 0, x^0(E)=|V|-1$，所以 x^0 是（5.1）的可行解，且 $c^T x^0=c(T)$，即可行解的目标函数值与对应的生成树的费用相等。（5.1）的最优值是 MST 的费用的下界。可以证明二者相等。

定理 5.2 设 x^0 是关于费用 c_e 的一个 MST 的特征向量，则 x^0 是（5.1）的最优解。

最短路问题（Shortest Path Problem）：

输入：有向图 $G=(V,E)$，顶点 $r\in V$，以及实费用向量 $\{c_e:e\in E\}$；

目标：对每个 $v\in V$，找到一条从 r 到 v 的最小费用有向路（如果存在的话）。

可以修正给定的图，使得对每个 $v\in V$，都有一条从 r 到 v 的有向路，加一条费用足够大的弧 rv 即可。

算法的基本思路（所有解决最短路问题的方法都以此为基础）：假设已知对每个 $v\in V$ 存在一条费用为 y_v 的从 r 到 v 的有向路，并且我们找到一条满足 $y_v+c_{vw}<y_w$ 的弧 $vw\in E$。由于把 vw 附加到从 r 到 v 的有向路上，可以到的一条从 r 到 w 的有向路，且费用比 y_w 更便宜。特别地，如果 $y=(y_v,v\in V)$ 是 r 到 v 的有向路的最小费用向量，那么 y 必然满足：

$$y_v+c_{vw}\ge y_w, \text{对所有 } vw\in E \qquad (5.2)$$

如果 $y=(y_v,v\in V)$ 满足（3.2）且 $y_r=0$，则称 $y=(y_v,v\in V)$ 是一个可行势。

注：（5.2）是基本要求，如果 $y_r\ne 0$，可以令 $y_v:=y_v-y_r$。

命题：令 y 是可行势，P 是从 r 到 v 的有向路，则 $c(P)\ge y_v$。

证明：假设 $P=\{r=v_0,e_1,v_1,e_2,\cdots,e_k,v_k=v\}$, 那么

$$c(P)=\sum_{i=1}^{k}c_{e_i}\ge\sum_{i=1}^{k}(y_{v_i}-y_{v_{i-1}})=y_{v_k}-y_{v_0}=y_v。$$

可行势与线性规划

定理 5.3 令 $G=(V,E)$ 是有向图，$r,s\in V$ 且 $c\in R^E$，如果对每个 $v\in V$，存在一条从 r 到 s 的最小费用有向路，那么

$\min\{c(P):P$ 是从 r 到 s 的有向路$\}=\max\{y_s:y$ 是可行势$\}$。

为了方便，去掉 $y_r=0$ 的要求，并且将线性规划问题写成

$$\begin{aligned}&\max y_s-y_r\\&s.t.\quad y_w-y_v\le c_{vw},\text{对所有}vw\in E\end{aligned}\qquad(5.3)$$

(5.3)的对偶规划为：

$$\begin{aligned}&\min\sum_{e\in E}c_e x_e\\&s.t.\quad \sum(x_{wv}:w\in V,wv\in E)-\sum(x_{vw}:w\in V,vw\in E)=b_v,\text{对所有}v\in V\\&\qquad x_{vw}\ge 0,\text{对所有}vw\in E\end{aligned}\qquad(5.4)$$

其中 $b_v=\begin{cases}1,v=s\\-1,v=r\\0,\text{其它}\end{cases}$。

当最短路存在时，(5.4)有最优解，它是一条简单有向路的特征向量。(5.4)的可行解多面体的顶点是简单有向路的特征向量。

附注：路的长度与割量

1. 带非负长度的最短路

设 $D=(V,A)$ 是一个有向图，且 $s,t\in V$。一个**迹**是一个序列 $P=(v_0,a_1,\cdots,v_m,a_m)$，其中 a_i 是从顶点 v_{i-1} 指向 v_i 的弧 $(i=1,2,\cdots,m)$。如果 $v_0,\cdots,v_m$ 全不相同，则称 P 为一条路。

如果 $s=v_0$ 且 $t=v_m$，分别 P 是的起点和终点，称 P 为一条 $s-t$ 迹，如果 P 为一条路，则称为一条 $s-t$ 路。P 的长度为 m。s 到 t 的距离为所有 $s-t$ 路中最短路径的长度。（如果不存在 $s-t$ 路，则 s 到 t 的距离定义为 ∞。）

不难确定出 s 到 t 的距离：置 V_i 表示有向图 $D=(V,A)$ 中从 s 出发距离为 i 的顶点集合。注意到对每一个 i：

（1）　V_{i+1} 等于顶点集合 $v\in V\setminus(V_0\bigcup V_1\bigcup V_2\bigcup\cdots V_i)$ 且存在 $u\in V_i$ 使得 $(u,v)\in A$。

这给出了一个确定集合 V_i 的直接算法：置 $V_0:=\{s\}$ 且下一步按照规则（1）顺序确定

出$V_1, V_2, \cdots$，直到$V_{i+1} = \phi$为止。

事实上，这给出了一个线性时间算法：

定理 1.1. 上述算法的运行时间为$O(|A|)$。

证明：直接从上述叙述中可以得到。

事实上，上述算法找到了从s到所有能到达顶点的距离。更进一步，它还给出了最短路径。这些路径可以由一棵根树（有向）$T = (V', A')$描述出来，其中树根为s，V'是从s可以到达的D中所有顶点，，并且满足对任给的$u, v \in V'$，每一个在T中的有向$u - v$路都是在D中的最短$u - v$路。

注：一棵树根为s根树（有向）是一个有向图，满足其对应的无向图是一棵树且对每一个顶点$t \neq s$，其入度都为 1。因此，每一个可以由s的顶点t，有唯一一条有向$s - t$路。

当然，当我们在算法中到达顶点t时，我们存储到达t的弧。那么在算法的最后，所有存储的弧形成一棵满足上述性质的根树。

我们也可以用一个平凡的最小—最大松弛来描述一条$s - t$路。我们称A的一个子集A'是一个$s - t$割，如果$A' = \delta^{\text{out}}(U)$对某个$V$的子集$U$满足$s \in U$且$t \notin U$。

注：$\delta^{\text{out}}(U)$和$\delta^{\text{in}}(U)$分别表示离开或进入顶点集合U的弧的集合。

显然：任给一条$s - t$路$P = \{e_1, e_2 \cdots, e_k\}$，存在唯一的一列不相交的割$C_1, \cdots, C_k$，使得$C_i \cap P = \{e_i\} (i = 1, 2, \cdots, k)$。

那么，下述结果由 Robacker[1956]给出：

定理 1.2. 一条$s - t$路的最短长度等于不相交的$s - t$割对的最大数目。

证明：对一条$s - t$路，假设l表示$s - t$路的最短长度，N表示不相交的$s - t$割对的最大数目。显然，$l \geq N$，因为每一条$s - t$路与每一个$s - t$割相交于一条弧。反过来，对$i = 0, 1, \cdots, d - 1$，考虑$s - t$割$\delta^{\text{out}}(U_i)$，其中d表示从s到t的距离，且U_i表示从s出发距离至多为i的顶点集合，从而得到$N \geq l$。所以$l = N$。

上述结果可以推广到弧长具有一个确定长度的情况。对任意“长度”函数$l : A \to Q_+$及任意迹$P = (v_0, a_1, \cdots, v_m, a_m)$，令$l(P)$表示$P$的长度，即：

（2） $$l(P) := \sum_{i=1}^{m} l(a_i).$$

那么现在从s到t的距离（用l表示）就等于所有$s - t$路中最短路径的长度。如果不存在$s - t$路，则s到t的距离定义为∞。

当然，根据 Dijkstra]1959]，对任给的顶点t，有一个简单的算法求出从$s - t$的最短路

径及其长度。开始置$U := V, f(s) := 0, \forall v \neq s, f(v) := \infty$，接下来应用下述迭代规则：

（3） 寻找$u \in U$在$u \in U$上最小化$f(u)$。对每一条弧$a = (u,v) \in A$，如果

$f(v) > f(u) + l(a)$，重置$f(v) := f(u) + l(a)$。重置$U := U \setminus \{u\}$。

当$U = \phi$时停止迭代。那么：

定理 1.2. 最后的函数f给出了从s出发的距离。

证明：对任意顶点v，令$\mathrm{dist}(v)$表示从s到v的距离，显然，在整个迭代过程中都有$\forall v \in V, f(v) \geq \mathrm{dist}(v)$。我们下面证明在整个迭代过程中都有$\forall v \in V \setminus U, f(v) = \mathrm{dist}(v)$。开始时，$U := V, f(s) := 0, \forall v \neq s, f(v) := \infty$，结论成立。

考虑（3）中的任何一次迭代。需要证明对选中的$u \in U$有$f(u) = \mathrm{dist}(u)$。假设$f(u) > \mathrm{dist}(u)$。令$s = v_0 v_1 \cdots v_k = u$是最短的$s-t$路。令$i$是$v_i \in U$中的最小下标。

那么，$f(v_i) = \mathrm{dist}(v_i)$。事实上，如果$i = 0$，则$f(v_i) = f(s) = 0 = \mathrm{dist}(s) = \mathrm{dist}(v_i)$。如果$i > 0$，因为$\forall v_{i-1} \in V \setminus U$，所以

（4） $f(v_i) \leq f(v_{i-1}) + l(v_{i-1}, v_i) = \mathrm{dist}(v_{i-1}) + l(v_{i-1}, v_i) = \mathrm{dist}(v_i)$。

这就意味着$f(v_i) \leq \mathrm{dist}(v_i) \leq \mathrm{dist}(u) < f(u)$，与$u$的选择矛盾。

显然，算法的迭代步数为$|V|$，每一步迭代需要的计算量为$O(|V|)$。因此算法的运行时间为$O(|V|^2)$。事实上，通过存储（3）中得到的最后弧的每一个顶点v，我们寻找的了一棵根树$T = (V', V')$，其中树根为s，V'是从s可以达到的所有顶点集合，满足如果$u, v \in V'$，T包含一条$u-v$有向路，那么这条路是D中的最短的$u-v$路。

因此，我们有：

定理 1.4. 给定一个有向图$D = (V, A)$，$s, t \in V$和一个长度函数$l: A \to Q_+$，可以在$O(|V|^2)$时间内找到一条最短的$s-t$路。

证明：见上。

算法的改进见 1.2 节。

定理 1.2 推广到带权重的情况如下。

割与割量：我们称 A 的一个子集 A' 是一个 $s-t$ 割，如果 $A'=\delta^{\text{out}}(U)$ 对某个 V 的子集 U 满足 $s\in U$ 且 $t\notin U$，其中 $\delta^{\text{out}}(U)$ 和 $\delta^{\text{in}}(U)$ 分别表示离开或进入顶点集合 U 的弧的集合。用 $C(\delta^{\text{out}}(U))$ 和 $C(\delta^{\text{in}}(U))$ 分别表示割量，即：

$C(\delta^{\text{out}}(U))=\min\{l(e):e\in\delta^{\text{out}}(U)\}, C(\delta^{\text{in}}(U))=\min\{l(e):e\in\delta^{\text{in}}(U)\}$。

路径割量：任给一条 $s-t$ 路 $P=\{e_1,e_2\cdots,e_k\}$，存在唯一一列不相交的割 $C_1,\cdots,C_k$，使得 $C_i\bigcap P=\{e_i\}(i=1,2,\cdots,k)$，称 $\sum_{i=1}^{k}C(\delta^{\text{out}}(C_i))$ 为 $s-t$ 路 P 的路径割量。

定理 1.5. 给定一个有向图 $D=(V,A)$，$s,t\in V$ 和一个长度函数 $l:A\to Z_+$，则 $s-t$ 路的最短路长度等于 $s-t$ 割 $C_1,\cdots,C_k$（可以重复）的最大数目 k，满足每一条弧 a 在割 C_i 至多为 $l(a)$。

证明：假设 $s-t$ 路的最短路长度为 l，$s-t$ 割 $C_1,\cdots,C_k$（可以重复）的最大数目 k，满足每一条弧 a 在割 C_i 至多为 $l(a)$，则显然 $l\geq k$，因为假设 P 是任意一条 $s-t$ 路，$C_1,\cdots,C_k$ 是任何一组满足上述要求的割，则

$$l(P)=\sum_{a\in P}l(a)\geq\sum_{a\in P}(a\in C_i\text{的}i\text{的数目})=\sum_{i=1}^{k}|C_i\bigcap P|\geq\sum_{i=1}^{k}1=k \tag{5}$$

至于相等性，令 d 表示 s 到 t 的距离，且 U_i 表示从 s 出发距离小于 i 的顶点集合，$i=1,2,\cdots,d$。置 $C_i=\delta^{\text{out}}(U_i)$，得到满足要求的 $s-t$ 割 $C_1,\cdots,C_k$。

定理 1.6. 给定一个有向图 $D=(V,A)$，$s,t\in V$ 和一个长度函数 $l:A\to Q_+$，则 $s-t$ 路的最短路长度等于最大 $s-t$ 路径割。

思考题：路径割量与可行势是什么关系？