第三章 网络最优化简介

运输问题，最短路问题，最大流问题，最小成本流问题等有关网络最优化通常应用于解决实际问题，并具有重要的经济意义。

就数学模型而言，它们是线性规划的几个重要特例。针对线性规划模型已有多项式时间算法，因为网络模型的特殊数学结构，利用其结构特性还可以设计出效率更高的求解算法。

3.1 运输模型

运输模型是一类特殊的线性规划，即从货源地（如生产厂家）装运货物到目的地（如经销商仓库），其目标是确定运输表使得总运输成本最小并满足供应和需求的限制条件。

运输模型可扩展应用于其他领域，包括投资控制，工作调度，人员指派等。

运输模型

经典的运输问题：

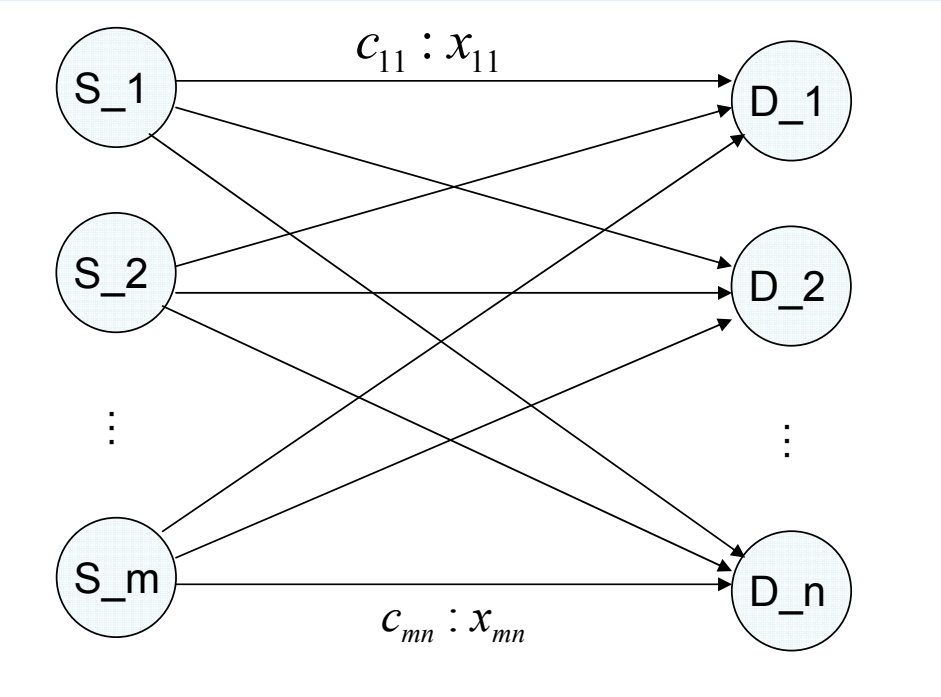
工厂i的货物量s_i, $i = 1, 2, \cdots, m$.

需求点j的需求量d_j, $j = 1, 2, \cdots, n$.

从工厂i到需求点j的单位货运费用c_{ij}及其发货量x_{ij}.

选取一个能使运输总费用达到最小的路径规划。

运输模型



运输模型

运输问题的数学描述

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \le s_i, i = 1, \cdots, m \\
& \sum_{i=1}^{m} x_{ij} \ge d_j, j = 1, \cdots, n \\
& x_{ij} \ge 0, \quad i = 1, \cdots, m \quad j = 1, \cdots, n.
\end{aligned} \tag{40}
$$

运输模型

Table: Mileage Chart

	Denver	Miami
Los Angeles	1000	2690
Detroit	1250	1350
New Orleans	1275	850

Table: Transportation Cost per Car

	Denver($j=1$)	Miami($j=2$)
Los Angeles($i=1$)	\$80	\$215
Detroit($i=2$)	\$100	\$108
New Orleans($i=3$)	\$102	\$68

运输模型

Table: MG Auto Transportation Model

	Denver	Miami	**Supply**
Los Angeles	80 x_{11}	215 x_{12}	**1000**
Detroit	100 x_{21}	108 x_{22}	**1500**
New Orleans	102 x_{31}	68 x_{32}	**1200**
Demand	**2300**	**1400**	

运输模型

$$
\begin{aligned}
\min \quad & z = 80x_{11} + 215x_{12} + 100x_{21} + 108x_{22} + 102x_{31} + 68x_{32} \\
\text{s.t.} \quad & x_{11} + x_{12} \le 1000 \quad (\text{LosAngeles}) \\
& x_{21} + x_{22} \le 1500 \quad (\text{Detroit}) \\
& x_{31} + x_{32} \le 1200 \quad (\text{NewOreleans}) \\
& x_{11} + x_{21} + x_{31} \ge 2300 \quad (\text{Denver}) \\
& x_{12} + x_{22} + x_{32} \ge 1400 \quad (\text{Miami}) \\
& x_{ij} \ge 0, \ i = 1, 2, 3, \ j = 1, 2
\end{aligned} \tag{41}
$$

运输模型

考虑产销平衡的情形, 即 $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = s_i, i = 1, \cdots, m \\
& \sum_{i=1}^{m} x_{ij} = d_j, j = 1, \cdots, n \\
& x_{ij} \geq 0, \quad i = 1, \cdots, m \quad j = 1, \cdots, n.
\end{aligned} \tag{42}
$$

运输问题的矩阵A形如：

$$
\begin{array}{c}
\begin{array}{cccccccccccccc}
x_{11} & x_{12} & \cdots & x_{1n} & x_{21} & x_{22} & \cdots & x_{2n} & \cdots & x_{m1} & x_{m2} & \cdots & x_{mn}
\end{array} \\
\left[\begin{array}{cccccccccccccc}
1 & 1 & \cdots & 1 & & & & & & & & & & \\
 & & & & 1 & 1 & \cdots & 1 & & & & & & \\
 & & & & & & & & \vdots & & & & & \\
 & & & & & & & & & 1 & 1 & \cdots & 1 & \\
1 & & & & 1 & & & & & 1 & & & & \\
 & 1 & & & & 1 & & & \cdots & & 1 & & & \\
 & & \vdots & & & & \vdots & & & & & \vdots & & \\
 & & \vdots & & & & \vdots & & & & & \vdots & & \\
 & & & 1 & & & & 1 & & & & & & 1
\end{array}\right]
\end{array}
$$

注意，这个矩阵并非行满秩，秩为 $n + m - 1$, 故可行基解最多有 $n + m - 1$的大于0的分量。

运输模型

对于供大于求（或供低于求），我们可以通过添加冗余变量，变换为供需平衡。例如，若 $\sum_{i=1}^{m} s_i > \sum_{j=1}^{n} d_j$, 可以假设存在额外的需求方 $n+1$, 令 $c_{i,n+1} = 0, i = 1, 2, \ldots, m$, 以及 $d_{n+1} = \sum_{i=1}^{m} s_i - \sum_{j=1}^{n} d_j$，从而可以构造供需平衡运输模型。

定理

运输问题有可行解的充分必要条件是供需平衡，即$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$。

Proof.

必要性显然。充分性：令$x_{ij} = \frac{s_i d_j}{T}, T = \sum_{i=1}^{m} s_i$, 可以验证此为可行解。 □

直接应用第二章的单纯形法，当然可以求解运输规划问题。然而，变量维度是mn，单纯性表太大，不易操作。我们这里针对运输模型的特点，设计新的单纯形法。

运输模型-表格作业法

运输问题的对偶

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} s_i u_i + \sum_{j=1}^{n} d_j v_j \\ \text{s.t.} \quad & u_i + v_j \le c_{ij}, \quad i = 1, \cdots, m, \quad j = 1, \cdots, n \\ & u_i, v_j \quad \text{无限制} \end{aligned} \tag{43}$$

利用对偶关系： $c_{ij} - c_B^\top B^{-1} a_{ij} = c_{ij} - (u_i + v_j)$

对基变量x_{ij}而言，$u_i + v_j = c_{ij}$, 即$\sigma_{ij} = u_i + v_j - c_{ij} = 0$.

对非基变量x_{ij}而言，若$\sigma_{ij} = u_i + v_j - c_{ij} \le 0$, 已对偶可行（原问题最优）；若$\sigma_{ij} = u_i + v_j - c_{ij} > 0$, 非对偶可行（原问题非最优），则引进基。

运输模型

运输模型的单纯形法求解步骤：

1. 选取一组$m+n-1$个路径，作为初始可行基解（Northwest-Corner Starting Solution）。
2. 检验当前解是否可改进，如果可改进，则引进一个非基变量进行步3，否则停止。
3. 当把步2中挑选的变量引进时，确定哪个路径应当由基解中退出。
4. 调整其他基本路径的流量（满足可行性），返回到步2.

我们将以例子来说明运输模型的单纯形算法。

运输模型

Table: 某公司的运输表

	D(1)	D(2)	D(3)	D(4)	**Supply**
S(1)	10 x_{11}	2 x_{12}	20 x_{13}	11 x_{14}	**15**
S(2)	12 x_{21}	7 x_{22}	9 x_{23}	20 x_{24}	**25**
S(3)	4 x_{31}	14 x_{32}	16 x_{33}	18 x_{34}	**10**
Demand	**5**	**15**	**15**	**15**	

运输模型

Table: 算法迭代1:（步骤1:）初始可行基解 (Northwest-Corner Starting Solution) 从左上角出发，令$x_{11} = \min\{d_1, s_1\}$, 然后向下或者右，使得原问题达到所有等式成立。这样便得到一个初始可行基解（蓝色值为可行变量）。

	D(1)	D(2)	D(3)	D(4)	**Supply**
S(1)	$c_{11} = 10$ $x_{11} = \mathbf{5}$	2 **10**	20	11	**15**
S(2)	12	7 **5**	9 **15**	20 **5**	**25**
S(3)	4	14	16	18 **10**	**10**
Demand	**5**	**15**	**15**	**15**	

运输模型

Table: 算法迭代1:（步骤2:） . 令$u_1 = 0$. 然后根据可行基解的对偶关系，解出所有 $n + m - 1$组等式 $u_i + v_j = c_{ij}$, 这里ij属于可行基解下标。这样得到所有的对偶变量值。对非基变量，计算 $(u_i + v_j) - c_{ij}$ (红色方框中的值)，大于0的变量可以作为转轴，最大值作为入基变量。本例子中选取 x_{31}作为入基变量。

	$v_1 = 10$	$v_2 = 2$	$v_3 = 4$	$v_4 = 15$	**Supply**
$u_1 \equiv 0$	$c_{11} = 10$ $x_{11} = 5$	2 10	20 [-16]	11 [4]	**15**
$u_2 = 5$	12 $u_2 + v_1 - c_{21} = [3]$	7 5	9 15	20 5	**25**
$u_3 = 3$	4 [9]	14 [-9]	16 [-9]	18 10	**10**
Demand	**5**	**15**	**15**	**15**	

运输模型

Table: 算法迭代1:（步骤2:）（找回路） 从入基变量出发，寻找包含可行基解的回路。本例中，回路为： $x_{31}, x_{34}, x_{24}, x_{22}, x_{12}, x_{11}, x_{31}$. 在该回路中，令入基变量由 0 增加到值$\theta$, 并相应的更改回路中的其他值，使得原问题等式成立。

	$v_1 = 10$	$v_2 = 2$	$v_3 = 4$	$v_4 = 15$	**Supply**
$u_1 \equiv 0$	10 $5 - \theta$	2 $10 + \theta$	20 [-16]	11 [4]	**15**
$u_2 = 5$	12 [3]	7 $5 - \theta$	9 15	20 $5 + \theta$	**25**
$u_3 = 3$	4 θ [9]	14 [-9]	16 [-9]	18 $10 - \theta$	**10**
Demand	**5**	**15**	**15**	**15**	

定义 (回路)

我们将表中 $x_{i_1 j_1}, x_{i_1 j_2}, x_{i_2 j_2}, x_{i_2, j_3}, \ldots, x_{i_s, j_s}, x_{i_s, j_1}$, ($i_1, i_2, \ldots, i_s$互不相同，且 $1 \le i_k \le m$, $j_1, j_2, \ldots, j_s$互不相同，且 $1 \le j_k \le m$, $1 \le k \le s$) 形成的集合成为一个回路。

运输模型

Table: 算法迭代1:（步骤3:）确定出基变量，令 θ为使得不等式 $x_{ij} \geq 0$的最大值。本例中， $x_1 1$ 和 x_{22}处， $\theta = 5$相等，对应退化解。我们任选其中一个，让x_{11}出基。接着，更新对偶变量。

	$v_1 = 1$	$v_2 = 2$	$v_3 = 4$	$v_4 = 15$	**Supply**
$u_1 \equiv 0$	10 [-9]	2 15	20 [-16]	11 [4]	**15**
$u_2 = 5$	12 [-6]	7 0	9 15	20 10	**25**
$u_3 = 3$	4 5	14 [-9]	16 [-9]	18 5	**10**
Demand	**5**	**15**	**15**	**15**	

运输模型

Table: 算法迭代2:（步骤2-3:）重复单纯形法第2，3 步（确定入基变量，以及找回路）

	$v_1 = 1$	$v_2 = 2$	$v_3 = 4$	$v_4 = 15$	**Supply**
$u_1 \equiv 0$	10 [-9]	2 $15 - \theta$	20 [-16]	11 θ [4]	**15**
$u_2 = 5$	12 [-6]	7 $0 + \theta$	9 15	20 $10 - \theta$	**25**
$u_3 = 3$	4 5	14 [-9]	16 [-9]	18 5	**10**
Demand	**5**	**15**	**15**	**15**	

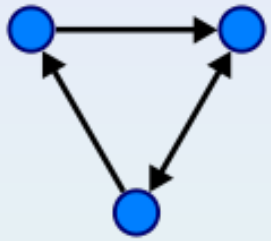
运输模型

Table: 算法迭代3：（步骤2-3）所有 $u_i + v_j - c_{ij}$为负数，得到最优解

	$v_1 = -3$	$v_2 = 2$	$v_3 = 4$	$v_4 = 11$	**Supply**
$u_1 \equiv 0$	10 [-13]	2 5	20 [-16]	11 10	**15**
$u_2 = 5$	12 [-10]	7 10	9 15	20 [-4]	**25**
$u_3 = 7$	4 5	14 [-5]	16 [-5]	18 5	**10**
Demand	**5**	**15**	**15**	**15**	

网络模型

设$G = (V, E)$为有向图，其中V是节点的集合，E是边的集合。



有时把某个节点作为初始点s, 另一个作为终点t而特殊对待。

各边$e \in E$上赋有成本$c(e)$以及容量$u(e)$, 且都取实数值。

由它们组成的$|E|$维相应向量分别记为

$$\mathbf{c} = (c(e)|e \in E), \quad \mathbf{u} = (u(e)|e \in E).$$

统括这些元素的$\mathcal{N} = (G, s, t, \mathbf{c}, \mathbf{u})$称为网络。

3.2 最短路径问题

最短路径问题中，成本$c(e)$解释为边e的长度。最短路径问题的典型形式是:

在网络$\mathcal{N} = (G, s, \mathbf{c})$中求出始点$s$到其它各点$v \in V$的最短路径及其长度。

另外，定义$w \in V$到$v \in V$的路径(path)

$$\pi = v_{i_0}(= w), v_{i_1}, \cdots, v_{i_k}(= v)$$

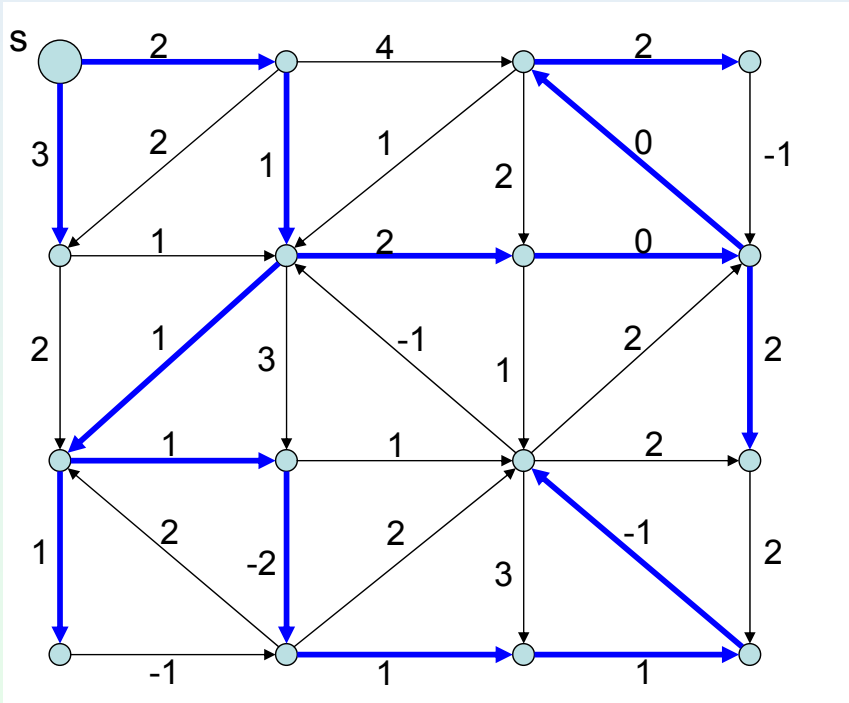
长度为

$$c(\pi) = \sum_{j=0}^{k-1} c(v_{i_j}, v_{i_{j+1}}).$$

最短路径问题

当$\mathcal{N}$中没有长度为负的回路时，存在有从s到所有点v的最短路径，它们可用以s为根的**最短路径树**来表示。

求s到v的最短路径，只要沿着s到v的最短路径树的边走下去就行。



如何求得该最短路径树？最短路径问题的模型算法？

最短路径问题

最短路径问题的线性规划描述：

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{s,(s,j)\in E} x_{sj} = 1 \\
& \sum_{k,(k,j)\in E} x_{kj} - \sum_{k,(i,k)\in E} x_{ik} = 0, \forall k \in V - \{s,t\} \qquad (44) \\
& - \sum_{t,(i,t)\in E} x_{it} = -1 \\
& x_{ij} \geq 0, \forall (i,j) \in E
\end{aligned}
$$

最短路径问题

最短路径问题的对偶为：

$$\begin{array}{ll} \max & (y_s - y_t) \\ \text{s.t.} & y_i - y_j \leq c_{ij}, \quad \forall (i,j) \in E \end{array} \tag{45}$$

最短路径问题的算法

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。

暴力穷举法：由于可能的路径为指数多个，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：

Dijkstra's algorithm. 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。

最短路径问题的算法

算法 DIJKSTRA

输入 有向图$G=(V,E)$, 各边长度$c: E \to \mathbb{R}_+$, 始点$s \in V$.

输出 从始点到所有节点$v \in V$的最短路径及其长度$c^*(v)$.

步一 初始化：令$d(s) := 0, d(v) := \infty (v \in V - \{s\})$, 以及$X := \emptyset$.

步二 迭代：选取一个满足$d(v^*) = \min\{d(v) \mid v \in V - X\}$的节点$v^* \in V - X$.

步三 更新： $X := X \cup \{v^*\}$.
进一步对$w \in V - X$的各边$e = (v^*, w) \in E$作如下更新：

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

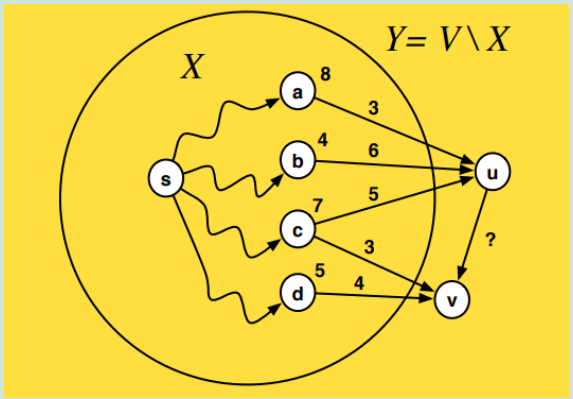
步四 结束判定：如果$P = V$则结束计算；否则回到步二。

最短路径问题的算法

- 步一：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。
- 步二详解：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 $\min_{x \in X,\ v \in V-X} (d(x) + w(x, v))$的解。

例

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 $p(v) = d(d) + w(d, v) = 9$.



- 步三：更新已访问集合 $X = X \cup \{v^*\}$, 更新临时距离 $d(w), w \in V - X$.
- 步四：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。