

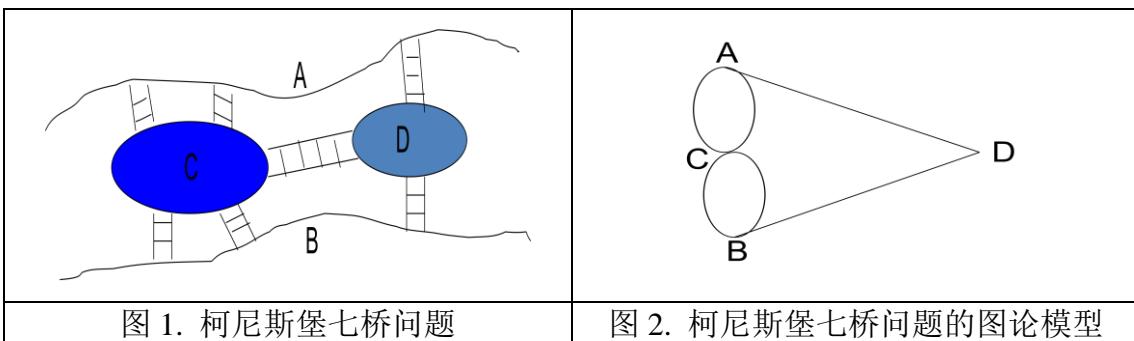
网络模型与优化

华东理工大学 王占全 摘抄 Web

1. 网络模型与优化概论

17、18 世纪，出现了一些有趣的问题，如迷宫问题、博弈问题、回路问题及棋盘上棋子的行走路线之类的有戏问题，这些问题吸引了许多学者。学者们将这些看起来无足轻重的问题抽象为数学问题，从而开辟了图论这门新学科的研究。从以下例子中例子初步了解图论这门学科的起源和发展。

图论起源于著名的柯尼斯堡七桥问题。在柯尼斯堡的普莱格尔河上有七座桥将河中的岛及岛与河岸联结起来，问题是要从这四块陆地中任何一块开始，通过每一座桥正好一次，再回到起点。然而无数次的尝试都没有成功。欧拉在 1736 年解决了这个问题，他用抽象分析法将这个问题化为第一个图论问题：即把每一块陆地用一个点来代替，将每一座桥用联接相应的两个点的一条线来代替，从而相当于得到一个“图”（如下图）。欧拉证明了这个问题没有解，并且推广了这个问题，给出了对于一个给定的图可以某种方式走遍的判定法则。这就是后来的欧拉路径和欧拉回路。这项工作使欧拉成为图论（及拓扑学）的创始人。



又如 1859 年，英国数学家汉密尔顿发明了一种游戏：用一个规则的实心十二面体，它的 20 个顶点标出世界著名的 20 个城市，要求游戏者找一条沿着各边通过每个顶点刚好一次的闭回路，即“绕行世界”。用图论的语言来说，游戏的目的是在十二面体的图中找出一个生成圈。这个生成圈后来被称为汉密尔顿回路。这个问题后来就叫做汉密尔顿问题。由于运筹学、计算机科学和编码理论中的很多问题都可以化为汉密尔顿问题，从而引起广泛的注意和研究。

发展至今，图论在计算机和离散数学的发展下取得了极大的发展，其应用范围很广，它不但能应用于自然科学，也能应用于社会科学。它非但广泛应用于电信网络、电力网络、运输能力、开关理论、编码理论、控制论、反馈理论、

随机过程、可靠性理论 W、化学化合物的辨认、计算机的程序设计、故障诊断、人工智能、印制电路板的设计、图案识别、地图着色、情报检索，也应用于诸如语言学、社会结构、经济学、运筹学、兵站学、遗传学等等方面。下面我们来讲述图与网络的一些基本概念。

1.1 图与网络基本概念

图论是数学的一个分支。它以图为研究对象。图论中的图是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。根据边是否有方向可以分为无向图和有向图。

1.1.1 无向图

设 V 是一个有 n 个顶点的非空集合， $V=\{v_1, v_2, \dots, v_n\}$ ， E 是一个有 m 条边的集合， $E=\{e_1, e_2, \dots, e_m\}$ ， E 中任一条边 e 是 V 的一个无序元素对 $[u, v]$ （这里 $u \neq v$ ），则 V 和 E 这两个集合组成了一个无向图，记作无向图 $G=(V, E)$ 。

若 $e=[u, v]$ 或 $[v, u]$ ，则称 u 与 v 为无向边 e 的顶点，边 e 与顶点 u 及 v 相关联， u 与 v 相邻。

对于 G ，有时为说明问题， V 与 E 也写成 $V(G)$ 及 $E(G)$ 。

给出图 $G=(V, E)$ ，就可以作出它的几何图。

例题 1. 给出无向图的点集和边集及关联关系，作出几何图。

$$V=\{v_1, v_2, v_3, v_4, v_5\}, E=\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$

关联关系如下表所示：

表 .1 节点与边列表

E	e_1	e_2	e_3	e_4	e_5	e_6	e_7
$e=[u, v]$	$[v_1, v_2]$	$[v_1, v_5]$	$[v_2, v_4]$	$[v_1, v_4]$	$[v_4, v_3]$	$[v_5, v_4]$	$[v_1, v_5]$

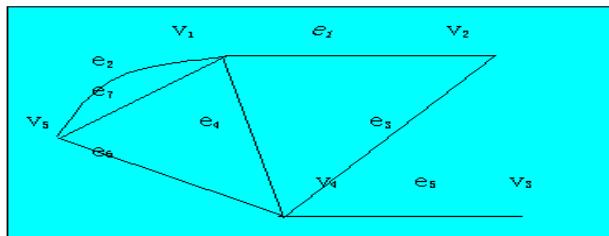


图 3 无向图

对于无向图，给出下面一些说明和概念：

平行边：若两条不同的边 e 和 e' 具有相同的顶点，则称 e 和 e' 为 G 的平行边。

简单图：若 G 无平行边，则称 G 为简单图

完备图：若图 G 中任两个顶点间恰有一条边相关联，则称图 G 为完备图。

1.1.2 有向图

设顶点的非空集合 $V=\{v_1, v_2, \dots, v_n\}$ ，边的集合 $E=\{e_1, e_2, \dots, e_m\}$ ， E 中任一条边 e 是 V 的一个有序元素对 $[u, v]$ （这里 $u \neq v$ ），则 V 和 E 这两个集合组成了一个有向图，记作有向图 $G=(V, E)$ 。

若 $e=[u, v]$ ，则称 u 为有向边 e 的起点， v 为有向边 e 的终点。

根据给定的顶点和边的集合以及关联关系，可以作出一个有向图，每一个边带有方向（方向与始点到终点的方向一致）。

类似于无向图，有向图有下列术语：

平行边：若两条不同的边 e 和 e' 的起点与终点都相同，则称 e 和 e' 为有向图 G 的平行边。

简单图：无平行边的有向图称为简单图。

完备图：若有向图中任两个顶点间，恰有两条边 $[u, v]$ 及 $[v, u]$ ，则称该有向图为完备图。

基本图：把有向图 G 的每条边出去定向后就得到一个无向图 G' 称 G' 为 G 的基本图。

对图（无论有向图还是无向图）来说，还有一个重要概念就是图的同构。

若图 $G=(V, E)$ 与 $G'=(V', E')$ 的顶点集合 V 和 V' 以及边的集合 E 与 E' 之间在保持关联关系的条件下一一对应，则称图 G 和 G' 为同构。

1.1.3 无向图有关的术语

无向图 G 中一个由顶点和边交错而成的非空有限序列

$$Q = v_{i_0} e_{j_1} v_{i_1} \dots v_{i_{k-1}} e_{j_k} v_{i_k}$$

且序列中边 e_{j_s} ($1 \leq s \leq k$) 的两个顶点恰为序列中顶点 $v_{i_{s-1}}$ 和 v_{i_s} ，则称 Q 为图 G 的链

在简单图中，链由它的顶点序列确定，所以简单图的链可用其顶点序列来表示： $Q = v_{i_0}v_{i_1} \dots v_{i_l}$ 若 $e \in E(Q)$ (链 Q 的边集)，也可简写为 $e \in Q$ 。

若 $k > 0$ ，且 $v_{i_0} = v_{i_k}$ (链的始点与终点相同)，则 Q 称为闭链；当 $v_{i_0} \neq v_{i_k}$ 时，称 Q 为开链。

若开链 Q 中顶点都不相同，则称 Q 为初等链。

若一个闭链 C ，除了第一个顶点和最后一个顶点相同外，没有相同的顶点和相同的边，则称闭链 C 为回路。

若图 G 中顶点 u 与 v 之间存在一条链，则称 u 与 v 在图 G 内是连通的。

若图 G 内任两点都连通，则称图 G 为连通图，否则成为分离图。

1.1.4 有向图有关的术语

在有向图 G 中，若 $Q = v_{i_0}e_{j_1}v_{i_1} \dots v_{i_{k-1}}e_{j_k}v_{i_k}$ 在 G 的基本图中是一条链（初等链），则 Q 就称为 G 的一条（链）初等链。

若 G 的链 $Q = v_{i_0}e_{j_1}v_{i_1} \dots v_{i_{k-1}}e_{j_k}v_{i_k}$ 中每条边 e_{j_s} 恰以 $v_{i_{s-1}}$ 为起点，以 v_{i_s} 为终点，则该链 Q 称为 v_{i_0} 至 v_{i_k} 的单向路，简称路。

在简单图中，可用其顶点序列来表示： $Q = v_{i_0}v_{i_1} \dots v_{i_l}$ 。

若 G 的路 Q 中每个顶点都不相同，则称 Q 为 v_{i_0} 至 v_{i_k} 的单向路径。简称路。

若单向路径的第一个顶点和最后一个顶点相同，则称为单向回路。

若图 G 中顶点 u 与 v ，从 u 到 v 之间存在单向路径，则称 u 可达 v 。

若图 G 内任两点之间相互可达，则称图 G 为强连通图。

1.1.5 图的顶点阶数

无向图 $G = (V, E)$ 中与顶点 v 关联的边数称为顶点的阶数，记作： $\delta(v)$ ， $\delta(v)$ 为偶数，称 v 为偶阶顶点； $\delta(v)$ 为奇数称为奇阶顶点。

两个常用定理：

定理 1：任一图中所有顶点的阶数之和为偶数。

定理 2：任一图中所有奇顶点的个数必为偶数。

1.2 图与网络的数据结构

在实际问题中遇到的图往往比较庞大。例如一个城市的公交系统，公交站点形成了图的顶点，公交线路形成了图的弧，这就形成一个庞大的有向图。人工分析此图多顶点之间，顶点与边之间的关联关系是不现实的，必须借助计算机等工具进行处理，那么将图的有关信息如何输入到计算机中或者说如何存储这些信息？可以实用关联举证、连接矩阵，弧表示方法，邻接表示方法等等。

1.2.1 邻接矩阵(Adjacency Matrix)表示法

$G=(V, A)$ 是一个简单有向图 $|V|=n$, $|A|=m$

图 $G=(V, A)$ 的邻接矩阵 C 是如下定义的： C 是一个 $n \times n$ 的 0-1 矩阵，即

$$C = (c_{ij})_{n \times n} \in \{0,1\}^{n \times n}, \quad c_{ij} = \begin{cases} 0, & (i, j) \notin A, \\ 1, & (i, j) \in A. \end{cases}$$

每行元素之和正好是对应顶点的出度，每列元素之和正好是对应顶点的入度。在邻接矩阵的所有元素中，只有 m 个为非零元。

	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table>	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1	1	0
0	1	1	0	0																						
0	0	0	1	0																						
0	1	0	0	0																						
0	0	1	0	1																						
0	0	1	1	0																						
图 4 有向图其矩阵																										

对于网络中的权，也可以用类似邻接矩阵的矩阵表示。只是此时一条弧所对应的元素不再是 1，而是相应的权而已。如果网络中每条弧赋有多种权，则可以用多个矩阵表示这些权。

1.2.2 关联矩阵(Incidence Matrix)表示法

图 $G=(V, A)$ 的邻接矩阵 C 是如下定义的： C 是一个 $n \times m$ 的 0-1 矩阵，即

$$B = (b_{ik})_{n \times m} \in \{ -1, 0, 1 \}^{n \times m}, \quad b_{ik} = \begin{cases} 1, & \exists j \in V, k = (i, j) \in A, \\ -1, & \exists j \in V, k = (j, i) \in A, \\ 0, & \text{其他。} \end{cases}$$

每行元素 1 的个数正好是对应顶点的出度，每行元素-1 的个数正好是对应顶点的入度。在关联矩阵的所有元素中，只有 $2m$ 个为非零元。

	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0																																		
1	0	1	1	0	0	0	0																																		
0	1	0	1	1	0	1	0																																		
0	0	1	0	1	1	0	1																																		
0	0	0	0	0	1	1	1																																		
图 5 有向赋权图其关联矩阵																																									

1.2.3 弧表 (Arc List) 表示法

所谓图的弧表，也就是图的弧集合中的所有有序对。弧表表示法直接列出所有弧的起点和终点。

特点：

共需 $2m$ 个存储单元，当网络比较稀疏时比较方便。

起点	1	1	2	3	4	4	5	5
终点	2	3	4	2	3	5	3	4
权(例)	8	9	6	4	0	3	6	7

图 6 弧表表示法

1.2.4 邻接表 (Adjacency Lists) 表示法

图的邻接表是图的所有节点的邻接表的集合；

而对每个节点，它的邻接表就是它的所有出弧

对于有向图 $G = (V, A)$ ，一般用 $A(i)$ 表示节点 i 的邻接表，即节点 i 的所有出弧构成的集合或链表（实际上只需要列出弧的另一个端点，即弧的头）。这种记法我们在本课程后面将会经常用到。

单向链表指针数组	1	2	8	—	3	9	0	A(1)={2,3}
	2	4	6	0				A(2)={4}
	3	2	4	0				A(3)={2}
	4	3	0	—	5	3	0	A(4)={3,5}
	5	3	6	—	4	7	0	A(5)={3,4}
								<small>31</small>

图 7 邻接表表示法

2. 网络模型

网络模型是指具有非常特殊结构的线性规划模型，应用该结构可以极大降低计算复杂度，提高效率，其在社会中用途非常广泛。其包括下面三个要素：

一是表征系统组成元素的节点。

二是体现各组成元素之间关系的箭线(有时是边)。

三是在网络中流动的流量，它一方面反映了元素间的量化关系，同时也决定着网络模型优化的目标与方向。

针对实际问题，有不同网络模型的问题，如最短路径问题，最小生成树问题，最大流问题，最小费用流问题，以及匹配和着色问题等等，在此我们仅介绍前面四种网络模型问题。在介绍这些模型之前，我们先介绍创建模型的一般过程。如图所示：

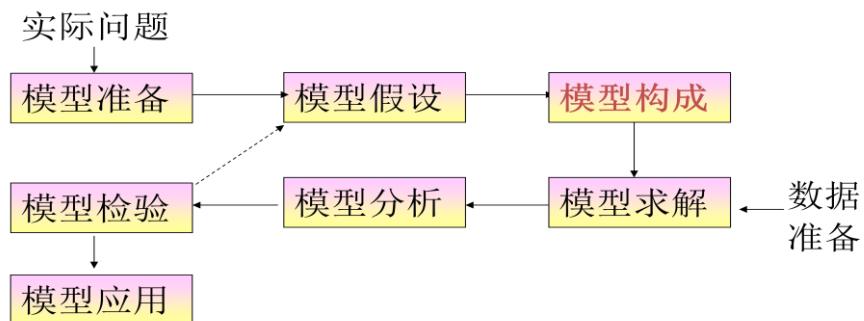


图 8 模型建模和求解

2.1 最短路

最短路问题可以用来解决许多生产问题。诸如各种管道铺设、线路安排、厂区布局、设备更新等等。另外像运价最小、运行时间最短、最可靠路等问题，都可以转化为最短路问题加以解决。针对最短路，其使用赋权图最多。下面介绍赋权图概念。

2.1.1 非负赋权图

现给有向图 $G = (V, E)$ ， $V = \{v_1, v_2, \dots, v_n\}$, $E = \{e_1, e_2, \dots, e_m\}$ 。设图 G 的每条边 $e = (v_i, v_j)$ 都与一个非负实数 W_{ij} 相对应， W_{ij} 称为边 e 的权， G 称为非负赋权图。 W_{ij} 也可表示为 $W(e)$ 或 $W(v_i, v_j)$ 。

这里所说的“权”，是指与边有关的数量指标。根据实际问题的需要，可以赋予不同的含义，例如表示距离、时间和费用等等。以后讨论的图都为赋权图，俗称为“网络”。

2.1.2 最短路径

若 P 为 G 中 u 至 v 的路径，称 $W(P) = \sum_{e \in P} W(e)$ 为 P 的长度。

若 P^* 为 G 中 u 至 v 的路径，且有： $W(P^*) = \min\{W(P) | P \text{ 为 } G \text{ 中 } u \text{ 至 } v \text{ 的路径}\}$ 则称 P^* 为 G 中 u 至 v 的最短路径。

非负赋权图中，最短路经具有下面的重要性质。

设想 v_s 到 v_t 的最短路径如下：

$v_s, \dots, v_j, \dots, v_k, \dots, v_t$

那么，从 v_s 沿 P 到 v_j 或 v_k 的路，就是 v_s 到 v_j 或 v_k 的最短路。也就是说， P 不仅是起点 v_s 到终点 v_t 的最短路，而且由 v_s 到 P 上任一中间点的最短路也在 P 上。

由此可以想到，为了求由 $v_s \rightarrow v_t$ 的最短路，可以先求 v_s 到中间点的路，然后再逐步扩展到终点 v_t 。

下面介绍求非负赋权图中某一顶点到另一顶点的最短路的算法，这个算法（Dijkstra algorithmn）是 Dijkstra 1959 年提出的。此外还有一些改进的算法，如 A* algorithmn, SPFA algorithmn, Bellman-Ford algorithmn, Floyd-Warshall algorithmn, Johnson algorithmn 等等。

2.1.3 Dijkstra 算法

在计算过程中，需将已经求出的到起点最短路的点与尚未求出到起点最短路的点分开，以正确执行迭代。为此将顶点分成两个集合 S 和 T ，已求出最短路的点置于 S 中，其它点置于 T 中。开始时 S 中仅含起点 v_s ，其它点全在 T 中，随着求最短路迭代工作的进行， S 中的点逐渐增多，当终点 v_t 也被纳入 S 中时，迭代结束。为了便于计算和区分各顶点是否已进入集合 S ，给已求出到起点最短路的点 v_k 赋以标号。这个标号由两部分组成，记为 $(d(v_s, v_k), i)$ ，其中 i 为 v_k 到起点最短路上的前点， $d(v_s, v_k)$ 为从起点 v_s 到 v_k 的最短路长。因每个标号含有两部分，故称为双标号法。

1. 算法步骤

(1) 给始点 v_s 赋以标号 $(0, s)$ ，并置 v_s 于 S 置，其它顶点于集合 T 中。

(2) 对图 G 里起点在 S 中终点在 T 中的边 e_{ij} ，计算： $d(v_s, v_k) = \min\{d(v_s, v_i) + \min_j[w_{ij}] \mid v_i \in S, v_j \in T\}$

并将 v_k 置于 S 中，同时赋给它标号 $(d(v_s, v_k), i)$

(3) 重复步骤(2)，当 $v_t \in S$ 时计算结束。 v_t 的第一个标号给出 $v_s \rightarrow v_t$ 的最短路长；利用第二个标号反向追踪，可得最短路径。

2. 实例演示算法求解的过程

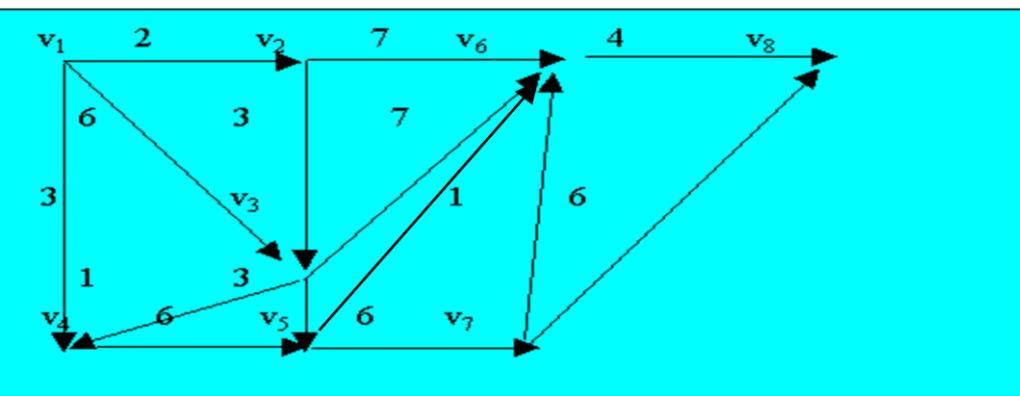


图 9 最短路径示意图

解：先给起点 v_1 赋标号 $(0, s)$ ，这时 $S=\{v_1\}$ 。考虑与点 v_1 相邻的顶点 v_2, v_3, v_4 ，计算：

$\min\{d(v_1, v_1) + \min[w_{12}, w_{13}, w_{14}]\} = 0 + \min\{2, 6, 3\} = 2 = d(v_1, v_2)$ ，这时，给 v_2 点赋标号 $(2, 1)$ ，此时 $S=\{v_1, v_2\}$ 。

考虑与 v_1, v_2 相邻的顶点 v_6, v_3, v_4 ，计算：

$\min\{d(v_1, v_1) + \min[w_{13}, w_{14}], d(v_1, v_2) + \min[w_{26}, w_{23}]\}$
 $= \min\{0 + \min[6, 3], 2 + \min[7, 3]\} = 3 = d(v_1, v_4)$ ，这时，给 v_4 点赋标号 $(3, 1)$ ，此时 $S=\{v_1, v_2, v_4\}$ 。

考虑与 v_1, v_2, v_4 相邻的顶点 v_6, v_3, v_5 ，计算：

$\min\{d(v_1, v_1) + w_{13}, d(v_1, v_2) + \min[w_{26}, w_{23}], d(v_1, v_4) + w_{45}\}$
 $= \min\{0 + 6, 2 + \min[7, 3], 3 + 6\} = 5 = d(v_1, v_3)$ ，这时，给 v_3 点赋标号 $(5, 2)$ ，此时 $S=\{v_1, v_2, v_4, v_3\}$ 。

考虑顶点 v_6, v_5 ，计算：

$\min\{d(v_1, v_2) + w_{26}, d(v_1, v_3) + \min[w_{36}, w_{35}], d(v_1, v_4) + w_{45}\}$

$=\min\{2+7, 5+\min[7,3], 3+6\}=8=d(v_1, v_5)$ ，这时，给 v_5 点赋标号 $(8, 3)$ ，此时 $S=\{v_1, v_2, v_4, v_3, v_5\}$ 。

考虑顶点 v_6 、 v_7 ，计算：

$$\min\{d(v_1, v_2)+w_{26}, d(v_1, v_3)+w_{36}, d(v_1, v_5)+\min[w_{56}, w_{57}]\}$$

$=\min\{2+7, 5+7, 8+\min[1,3]\}=9=d(v_1, v_6)$ ，这时，给 v_6 点赋标号 $(9, 5)$ 或 $(9, 2)$ ，此时 $S=\{v_1, v_2, v_4, v_3, v_5, v_6\}$ 。

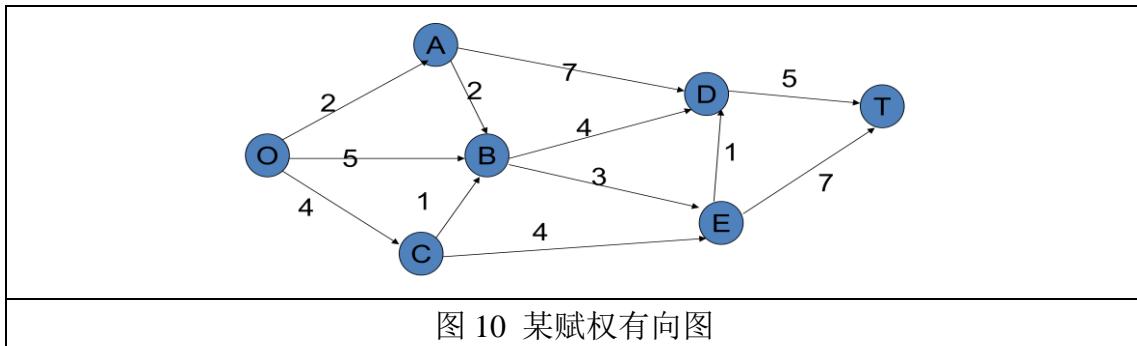
考虑顶点 v_7 、 v_8 ，计算：

$$\min\{d(v_1, v_5)+w_{57}, d(v_1, v_6)+w_{68}\}=\min\{8+6, 9+4\}=13=d(v_1, v_8)$$

这时，给 v_8 点赋标号 $(13, 6)$ ，此时 $S=\{v_1, v_2, v_4, v_3, v_5, v_6, v_8\}$ 。

因 $v_8 \in S$ ，计算结束。得到从 v_1 到 v_8 的最短路长为 13。由 v_8 点的第二个标号知最短路的前点为 v_6 点，找出其前点为 v_2 或 v_5 ，如此向前追踪，得到两条最短路径 $v_1 \rightarrow v_2 \rightarrow v_6 \rightarrow v_8$ 或 $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_8$ 。

下面给出另外一个例子，通过图例的形式给出，其图如下：



其步骤如下表：

表 2 步骤表

n	Solved nodes directly connected to unsolved nodes	Closest connected unsolved node	Total distance involved	nth nearest node	Minimum distance	Last connection
1	O	A	2	A	2	OA
2,3	O	C	4	C	4	OC
	A	B	2+2=4	B	4	AB
4	A	D	2+7=9			
	B	E	4+3=7	E	7	
	C	E	4+4=8			BE

5	A B E	D D D	2+7=9 4+4=8 7+1=8	D D	8 8	BD ED
6	D E	T T	8+5=13 7+7=14	T	13	DT

从中可以看出，从出发点到终点有两条最短路径，即为：
 $T \rightarrow D \rightarrow E \rightarrow B \rightarrow A \rightarrow O$ 或 $T \rightarrow D \rightarrow B \rightarrow A \rightarrow O$. 他们的距离都是 13 英里。

2.1.4 Floyd 算法

Floyd 算法又称为弗洛伊德算法，插点法，是一种用于寻找给定的加权图中顶点间最短路径的算法。

核心思路：通过一个图的权值矩阵求出它的每两点间的最短路径矩阵。

从图的带权邻接矩阵 $A = [a(i,j)] n \times n$ 开始，递归地进行 n 次更新，即由矩阵 $D(0)=A$ ，按一个公式，构造出矩阵 $D(1)$ ；又用同样地公式由 $D(1)$ 构造出 $D(2)$ ；……；最后又用同样的公式由 $D(n-1)$ 构造出矩阵 $D(n)$ 。矩阵 $D(n)$ 的 i 行 j 列元素便是 i 号顶点到 j 号顶点的最短路径长度，称 $D(n)$ 为图的距离矩阵，同时还可引入一个后继节点矩阵 $path$ 来记录两点间的最短路径。

采用的是松弛技术，对在 i 和 j 之间的所有其他点进行一次松弛。所以时间复杂度为 $O(n^3)$ ；

算法描述

a) 初始化： $D[u,v]=A[u,v]$

b) For $k:=1$ to n

For $i:=1$ to n

For $j:=1$ to n

If $D[i,j] > D[i,k] + D[k,j]$ Then

$D[i,j]:=D[i,k] + D[k,j];$

c) 算法结束： D 即为所有点对的最短路径矩阵。

算法过程如下：

把图用邻接矩阵 G 表示出来，如果从 V_i 到 V_j 有路可达，则 $G[i,j]=d$ ， d 表示该路的长度；否则 $G[i,j]=\text{空值}$ 。

定义一个矩阵 D 用来记录所插入点的信息， $D[i,j]$ 表示从 V_i 到 V_j 需要经过的点，初始化 $D[i,j]=j$ 。

把各个顶点插入图中，比较插点后的距离与原来的距离， $G[i,j] = \min(G[i,j], G[i,k]+G[k,j])$ ，如果 $G[i,j]$ 的值变小，则 $D[i,j]=k$ 。

在 G 中包含有两点之间最短道路的信息，而在 D 中则包含了最短通路径的信息。

比如，要寻找从 V_5 到 V_1 的路径。根据 D ，假如 $D(5,1)=3$ 则说明从 V_5 到 V_1 经过 V_3 ，路径为 $\{V_5, V_3, V_1\}$ ，如果 $D(5,3)=3$ ，说明 V_5 与 V_3 直接相连，如果 $D(3,1)=1$ ，说明 V_3 与 V_1 直接相连。

时间复杂度

$O(n^3)$

优缺点分析

Floyd 算法适用于 APSP(All Pairs Shortest Paths), 是一种动态规划算法, 稠密图效果最佳, 边权可正可负。此算法简单有效, 由于三重循环结构紧凑, 对于稠密图, 效率要高于执行 $|V|$ 次 Dijkstra 算法。

优点: 容易理解, 可以算出任意两个节点之间的最短距离, 代码编写简单;
缺点: 时间复杂度比较高, 不适合计算大量数据。

该算法详细步骤举例:

如下图: 为一个图

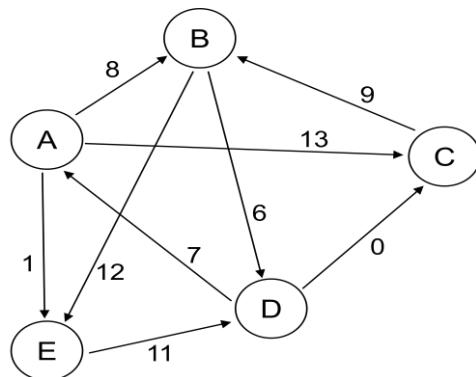


图 11 图

其邻接矩阵如下: (另外一种方法计算, 和课件的不同)

$$D^0 = \begin{pmatrix} & A & B & C & D & E \\ A & 0 & 8 & 13 & - & 1 \\ B & - & 0 & - & 6 & 12 \\ C & - & 9 & 0 & - & - \\ D & 7 & - & 0 & 0 & - \\ E & - & - & - & 11 & 0 \end{pmatrix}$$

D^0

$$D^1 = D^0 * D^0 = \begin{pmatrix} & A & B & C & D & E \\ A & 0 & 8 & 13 & - & 1 \\ B & - & 0 & - & 6 & 12 \\ C & - & 9 & 0 & - & - \\ D & 7 & 9 & 0 & 0 & 8 \\ E & - & - & - & 11 & 0 \end{pmatrix}$$

$D^1 = D^0 * D^0$

$$A \begin{pmatrix} 0 & 8 & 13 & \boxed{14} & 1 \\ B & - & 0 & - & 6 & 12 \\ C & - & 9 & 0 & \boxed{15} & 21 \\ D & 7 & 15 & 0 & 0 & 8 \\ E & - & - & - & 11 & 0 \end{pmatrix} D^2 = D^1 * D^1$$

$$A \begin{pmatrix} 0 & 8 & 13 & 14 & 1 \\ \boxed{13} & 0 & \boxed{6} & 6 & 12 \\ C & 22 & 9 & 0 & 15 & 21 \\ D & 7 & 9 & 0 & 0 & 8 \\ E & 18 & 20 & 11 & 11 & 0 \end{pmatrix} D^4 = D^3 * D^3$$

$$A \begin{pmatrix} 0 & 8 & 13 & 14 & 1 \\ B & - & 0 & - & 6 & 12 \\ C & - & 9 & 0 & 15 & 21 \\ D & 7 & 9 & 0 & 0 & 8 \\ E & - & - & - & 11 & 0 \end{pmatrix} D^3 = D^2 * D^2$$

$$A \begin{pmatrix} 0 & 8 & \boxed{12} & \boxed{12} & 1 \\ B & 13 & 0 & 6 & 6 & 12 \\ C & 22 & 9 & 0 & 15 & 21 \\ D & 7 & 9 & 0 & 0 & 8 \\ E & 18 & 20 & 11 & 11 & 0 \end{pmatrix} D^5 = D^4 * D^4$$

矩阵 D^5 既是所求的结果。

例题 2:

举例如下：如图 12，假设现状要在某村建一个商店和小学，试问：

- 商店应该建在何处，能使各村都离它较近？
- 已知各村小学生人数见下表，则小学应该建在何处，才能使各村小学生走的总路程最短？

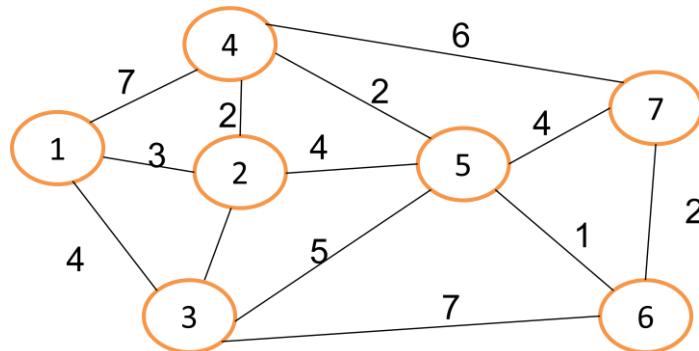


图 12 城镇分布图

表 3

Town	1	2	3	4	5	6	7
num	40	25	45	30	20	35	50

通过设置其邻近矩阵和上面的权重，通过矩阵乘法，求得各点之间最短路径，和相应两个问题的结果。两题结果分别是建设在第四个城镇，和学校建设在第五个城镇。

The shortest path distance

Max(dij)

	1	2	3	4	5	6	7	
1	0	3	4	5	7	8	10	10
2	3	0	3	2	4	5	7	7
3	4	3	0	5	5	6	8	8
4	5	2	5	0	2	3	5	5
5	7	4	5	2	0	1	3	7
6	8	5	6	3	1	0	2	8
7	10	7	8	5	3	2	0	10
dis	1325	920	1095	870	850	925	1215	

2.2 最小生成树

2.2.1 树的一些概念

无回路并且连通的无向图称为树，树中的边称为枝。

若 T 是图 G 的生成图，且 T 又是树，则称 T 是 G 的生成树。一个无向图可有若干棵生成树。

树 T 的六个等价定义：

1. T 连通且无回路。
2. T 无回路且只有 $n-1$ 条边。
3. T 连通且有 $n-1$ 条边
4. T 无回路，但不相邻的两个顶点之间联一条边，恰得到一个回路。
5. T 连通，但去掉任一条边后就不连通了。
6. T 的任意两个顶点之间恰有一条初等链。

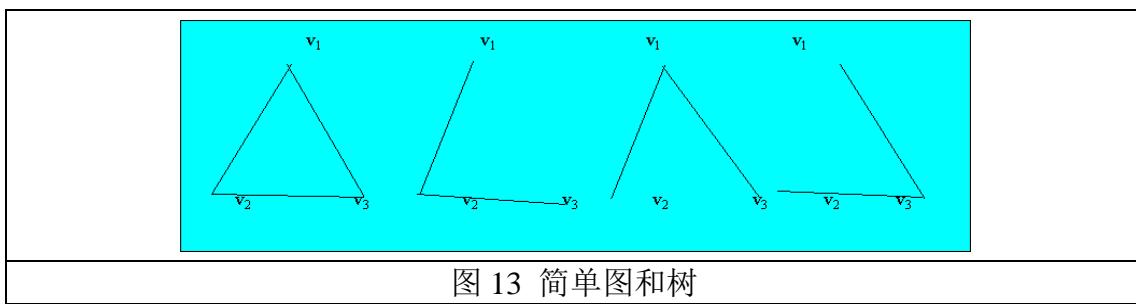
若给有向图 G, G 中任一顶点都可由 G 中顶点 u 到达, 则称 u 为 G 的根。

若有向图 G 有根 u, 且 G 的基本图是一棵树, 则称 G 为以 u 为根的有向树或根树。

2.2.2 最小生成树

如果图 T 是图 G 的一个生成子图而且又是一棵树, 则称树 T 是图 G 的一个生成树(支撑树)。

一个图可以有若干棵生成树, 如很简单的图(三角形形状的三棵生成树)



$$W(T) = \sum_{e_j \in T} W_j$$

若生成树 T^* 的总权 $W(T^*) = \min \{W(T) | T \text{ 是 } G \text{ 的生成树}\}$, 则称 T^* 为 G 的最小生成树, 简称最小树。

常需以最短线路连接若干个固定点, 如规划交通路网, 铺设煤气管道, 架设通讯线路等, 实质上都涉及到最小生成树问题。

2.2.3 最小生成树的两种方法

1. 加边法(避圈法, Kruskal 方法)

该方法的作法如下, 从所有边中选出一条权最小的边, 并把它纳入树中; 在余下的边中再选择一条权最小且与选进树中的边不构成圈的边, 将其纳入树中, 如此反复直到树中含有 $n-1$ 条边为止。

其算法主要步骤为:

- (1) 将图 G 的 m 条边按权的递增顺序排列; $w(e_{i1}) \leq w(e_{i2}) \leq \dots \leq w(e_{im})$
- (2) 令 $l(v_j) = j$, $j=1, 2, \dots, n$, $E_1 = \emptyset$. 循环变量赋值 $k=1$;
- (3) 设 $e_{ik} = [u, v]$, 若 $l(u) = l(v)$, 转步骤 (6), 否则令 $E_1 = E_1 \cup \{e_{ik}\}$;
- (4) 对满足 $l(v_j) = \max\{l(u), l(v)\}$ 的 v_j , 令 $l(v_j) = \min\{l(u), l(v)\}$;

- (5) 若 $|E_1|$ 中的边数 $|E_1| = n - 1$, 算法终止, 否则转步骤 (6);
 (6) 若 $|E_1| = m < n - 1$, 终止。图 G 为非连通图, 无支撑树, 否则, 令 $k = k + 1$, 转步骤 (3)

2. 破回路法（破圈法或丢边法）

在连通无向图 G 中, 任取一个圈, 将该圈中权最大的一条边 (如果含有两条, 则任取一条) 取掉; 在余下的圈中重复这一步骤, 直到不含圈为止, 就得到最小生成树。

应用举例, 某校区建设铺设网络, 规定网络铺设必须沿道架设, 应如何架设才最合理? 见图 14, 边上为建设成本费用。

使用 Kruskal 方法, 求最小生成树即为最合理的结果。其生成过程如下图所示:

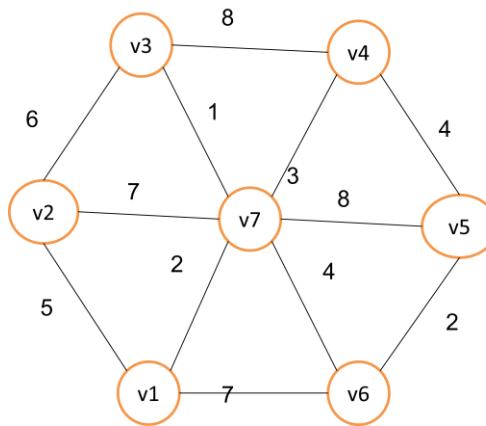
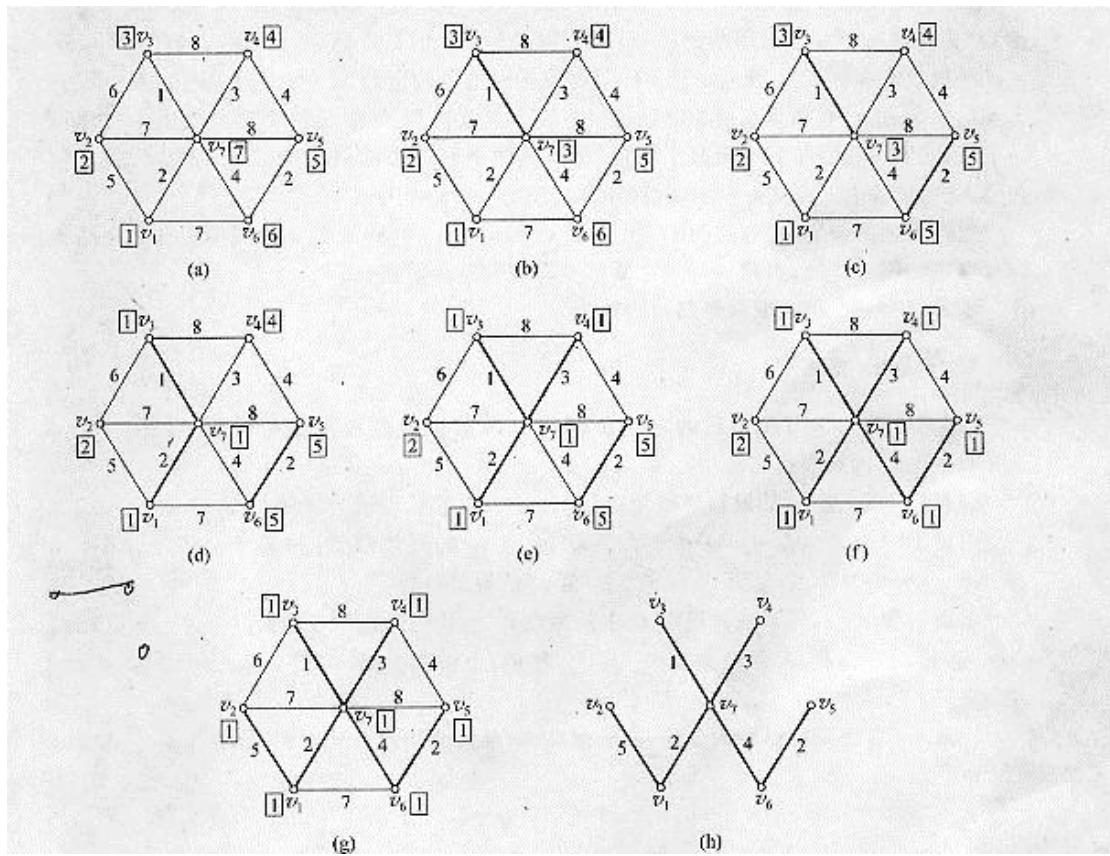


图 14 简单图和树

其图解如下:



2.3 最大流

研究网络的流量是很有意义的，例如交通系统中的车流、金融系统中的现金流、控制系统中的信息流、供水系统中的水流等。人们常常需要知道在既定网络中能通过的最大流量，判断设备的充分利用程度，这就提出了最大流量问题。

2.3.1 运输网络的概念

给有向图 $G = (V, E)$ ，对任一边 $e \in E$ ，有相应的一个非负整数 $C(e)$ ，且已取定 V 的两个非空子集 X 及 Y ， $X \cap Y = \emptyset$ ，则称 $G = (V, E, C, X, Y)$ 为一个运输网络，称 $C(e)$ 为边 e 的容量， X 中的顶点 x 为 G 的源， Y 中的顶点 y 为 G 的汇， $I = V - (X \cup Y)$ 中的顶点为 G 的中间点。

设 $f(e)$ 或 f_{ij} 或 $f(v_i, v_j)$ 为一个以 E 为定义域的非负整数函数，它满足下面几个条件。

1. 网络中弧的容量和流量

网络中某条弧 (v_i, v_j) 的最大通过能力称为它的容量，记作 c_{ij} ， $c_{ij} \geq 0$ ；而其流量是指通过它的某种流的实际流量，记为 f_{ij} 。

2. 可行流与最大流

在网络中满足下述条件的流称为可行流。

(1) 容量限制：对每一条弧 $a_{ij} = (v_i, v_j) \in A$, 均应有

$$0 \leq f_{ij} \leq c_{ij} \quad (1)$$

(2) 平衡条件：对每一个顶点来说，流入的流量之和应等于流出的流量之和。

若令所有弧的流量 $f_{ij}=0$, 就得到一个流量等于零的可行流（称为零流）。由此可见，可行流总是存在的。

最大流是网络总流量 f 为最大的可行流。求最大流是指寻求满足约束条件(1)的网络最大流量及其在各弧中的分配。

2.3.3 最大流与最小割

1. 割：若 S 为 V 一个子集， $x \in S$, $\bar{S}=V-S$, $y \in \bar{S}$, 记 K 为起点在 S 中，终点在 \bar{S} 中的全体有向边的集合，即：

$$K=\{(u, v) \mid u \in S, v \in \bar{S}\}$$

我们称边集 $K=(S, \bar{S})$ 为网络 G 的一个割，称 Cap 为 K 的容量，记为 Cap_K (“Cap”是英文 Capacity“容量”的简写）

例 . 给出运输网络图如下，试写出所有割及相应的的容量

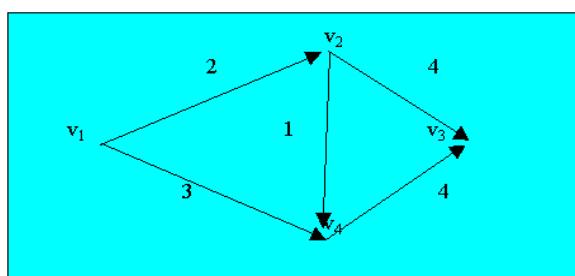


图 15 运输网络图

编号	S		K	Cap_K
1	{v ₁ }	{v ₂ , v ₃ , v ₄ }	{(v ₁ , v ₂), (v ₁ , v ₃)}	5
2	{v ₁ , v ₂ }	{v ₃ , v ₄ }	{(v ₁ , v ₃), (v ₂ , v ₃), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₃ , v ₄)} {(v ₂ , v ₄), (v ₃ , v ₄)} {(v ₁ , v ₃), (v ₂ , v ₄), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₂ , v ₃), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₁ , v ₃), (v ₂ , v ₄), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₁ , v ₃), (v ₂ , v ₃), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₁ , v ₃), (v ₂ , v ₄), (v ₃ , v ₄)}	8
3	{v ₁ , v ₃ }	{v ₂ , v ₄ }	{(v ₁ , v ₂), (v ₃ , v ₄)} {(v ₂ , v ₄), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₃ , v ₄)}	6
4	{v ₁ , v ₂ , v ₃ }	{v ₄ }	{(v ₂ , v ₄), (v ₃ , v ₄)} {(v ₁ , v ₂), (v ₃ , v ₄)}	8

可见，若把割 K 的边全从 G 中移去，图 $G-K$ 不一定分离成两部分，如割 K_3 ($G-K$ 仍为一个连通图)，但是它一定把 G 的全部自源到汇的路断开，也就是说此时的流不能在 G 上发生。故从直观上不难理解， G 的任一流 f 的流值 $\text{Val } f$ (“Val”是英文 Value“价值”的简写) 不能超过任一割的容量，这一点从上图中也可看出，如网络最大流值为 5，而最小割的容量也为 5，其它的容量都比 5 大。

2. 最大流与最小割

若 f^* 是满足下列条件的流：

$\text{Val } f^* = \max\{\text{Val } f \mid f \text{ 为 } G \text{ 的一个流}\}$ ，则称 f^* 为 G 的最大流。

若 K^* 是满足下列条件的割：

$\text{Cap } K^* = \min\{\text{Cap } K \mid K \text{ 为 } G \text{ 的一个割}\}$ ，则称 K^* 为 G 的最小割。

若 f 为 G 上的一个流，对任 $e \in E$ ，若 $f(e) = C(e)$ ，称边 e 为 f 饱和边；若 $f(e) < C(e)$ ，称边 e 为 f 不饱和边；若 $f(e) > 0$ ，称边 e 为 f 正边；若 $f(e) = 0$ ，称边 e 为 f 零边。

3. 增流链

设 $Q = x, v_1, \dots, u, v, \dots, t$ 为 f 的一条初等链（链上对应的边没有方向），若 G 中有 u 到 v 的有向边 (u, v) ，称边 (u, v) 为 Q 的前向边；若 G 中有 v 到 u 的有向边 (v, u) ，称边 (v, u) 为 Q 的后向边。

在例 1 中，初等链 $Q = v_1 v_4 v_2 v_3$

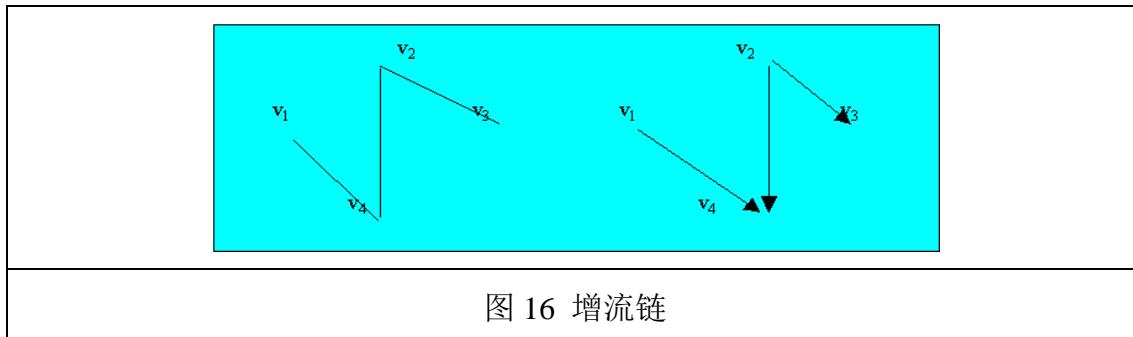


图 16 增流链

则边 (v_1, v_4) 、 (v_2, v_3) 是前向边， (v_4, v_3) 是后向边。

若 f 为上的流，对 $e \in E$ ，令：

$$l(e) = \begin{cases} C(e) - f(e) & \text{当 } e \text{ 是 } Q \text{ 的一条前向边} \\ f(e) & \text{当 } e \text{ 是 } Q \text{ 的一条后向边} \end{cases} \quad (2)$$

$$l(Q) = \min l(e) \quad e \in Q$$

也可以表示为：

$$\begin{cases} 0 \leq f_{ij} < c_{ij} (v_i, v_j) \in Q^+ \\ 0 < f_{ij} \leq c_{ij} (v_i, v_j) \in Q^- \end{cases}$$

当 $l(Q) = 0$ 时，称 Q 为 f 饱和链；当 $l(Q) > 0$ 时，称 Q 为 f 不饱和链。

故而，若 Q 为 f 饱和链，则链中至少有一条前向边为 f 饱和边（流量和容量相等），或至少有一条后向边为 f 零边（后向边最大可能减少为零）；若 Q 为 f 不饱和链，则 Q 的前向边都为 f 不饱和边（流量小于容量），后向边全为 f 正边（还有减少到零的量）。

一条从源 x 到汇 y 的 f 不饱和链就是一条 f 增流链。

若网络 G 中存在一条 f 增流链 ($l(Q) > 0$)，我们就可以得到 G 上一个新流

$$f'(e) = \begin{cases} f(e) + l(Q) & \text{当 } e \text{ 是 } Q \text{ 的一条前向边} \\ f(e) - l(Q) & \text{当 } e \text{ 是 } Q \text{ 的一条后向边} \\ f(e) & \text{其它} \end{cases} \quad (3)$$

此时的流值 $Val' = Val_f + l(Q)$ 。我们称 f' 为 f 基于 Q 的修改流修改后，对于 f 而言，链 Q 中至少有一条前向边为饱和边（流量和容量相等），或至少有一条后向边为 f 零边，即 Q 为 饱和链。

4. 最大流最小割集定理

前面已经讲述，割集是有 v_s (起点)到 v_T (终点) 的必经之路，无论拿掉哪个割集， v_s (起点)到 v_T (终点) 都将不再连通，所以，任何一个可行流的流量都不会超过任何一个割集的容量，因此有下面的网络最大流与最小割集的定理。

重要内容：设有 f 为网络 $D=(V,E,C)$ 的任何一个可行流，流量为 W , (S, \bar{S}) 为分离 v_s (起点)到 v_T (终点) 的一个割集，则有 $W \leq K(S, \bar{S})$.

从而给出了最大流—最小割集定理：任意一个容量网络 $D=(V,E,C)$ 中，从源 v_s 到汇 v_T 的最大流量等于分离 v_s 到 v_T 的最小割集容量。

由此给出此定理的推论：可行流 f 的最大流的充要条件是不存在从 v_s 到 v_T 关于 f 的可增广链。

2.3.3 最大流算法（标号法）

上述定理为我们提供了求网络最大流的一个方法，若给网络 G 上的一个初始流 f ，我们根据定理判断一下 G 中有无 f 的增流链，若无 f 增流链，则 f 即为最大流；若有 f 增流链 Q ，用上述对其上的流进行修改，可得到 f 基于 Q 的修改

f 流，再将 f 视为 f ，继续迭代，直到没有增流链为止。但是，对一个节点较多的运输网络，要通过观察的方法找出所有的增流链确实不易，下面的算法第一部分的标号法可以较容易的解决这个问题。

1. 求最大流的标号法

此方法由 Ford 和 Fulkerson 于 1956 年提出的，具体如下。

先给网络一个初始流（可行流，一般取平凡流，但也可以用节点流量守衡条件通过观察法给出一个可行流）。然后交替进行一下两个过程。

(1) 标号过程。这一过程是通过给某些顶点 v_j 标号，以寻找增流链，并进一步得出此链上的流量调整量。顶点的标号包括三部分。第一个标号表明该标号的来源，即填入其先行顶点的编号；第二个标号是“+”或“-”，“+”表示先行顶点 v_i 与要标号的顶点 v_j 组成的边是链中的前向边，“-”表示先行顶点 v_i 与要标号的顶点 v_j 组成的边是链中的后向边；第三个是该顶点流量的可增值，这样计算：

$$\Delta_{ij} = \min\{\Delta_i, \Delta_{ij}\}$$

其中： Δ_i 为先行顶点 v_i 的流量可增值；

$\Delta_{ij} = C(e) - f_{ij}$ ，若边 (v_i, v_j) 为前向边，

或 $\Delta_{ij} = f_{ji}$ ，若边 (v_j, v_i) 为后向边。

例如：边上的第一个数字表示容量，第二个表示流量，

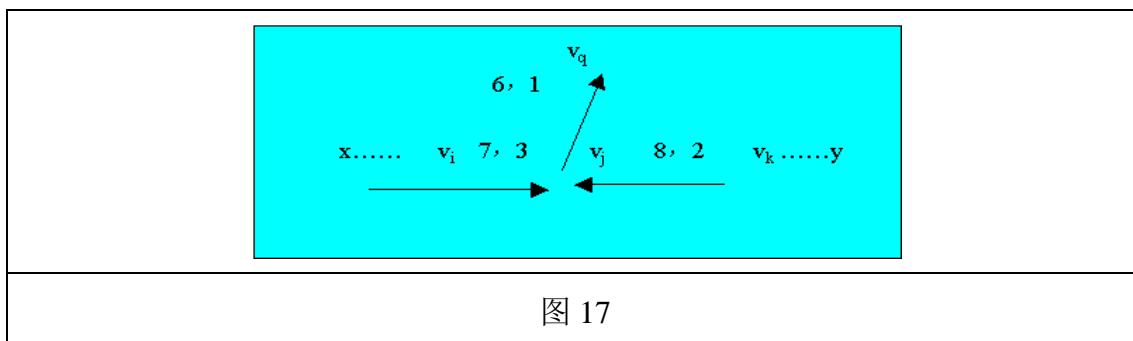


图 17

已知 v_i 的标号为 $(v_{i-1}, +, 2)$ ，那么 v_j 的标号为：

$(v_i, +, \Delta_j)$ 而 $\Delta_j = \min(2, 7-3) = 2$ ，即标号为： $(v_i, +, 2)$

v_k 的标号为： $(v_j, -, \Delta_k)$ 而 $\Delta_k = \min(2, 8-2) = 2$ ，即标号为： $(v_j, -, 2)$

开始标号时，先给起点 x 以标号 $(0, +, \infty)$ ，此处无穷大表示网络的最大流不受起点的限制。

设从 x 开始已使顶点 v_i 获得了标号，记已获得标号的点的集合为 V' ，没有获得标号的点的集合为 V'' ，任取一个已获标号的点 $v_i \in V'$ 查视与它相关联的其它未获标号的顶点 $v_j \in V''$ ，按照上面所述(例子)给出标号。但要注意，若 $\Delta_{ij}=0$ ，即 $C(e)=f_{ij}$ ，此时边 (v_i, v_j) 为前向边；或 $\Delta_{ij}=f_{ji}$ ，此时边 (v_j, v_i) 为

后向边，那么就不给 v_j 标号了，因为此时边 (v_i, v_j) 或 (v_j, v_i) 已是饱和边了。重复以上布置，直到汇点 $y \in V'$ 。

还要注意，在一次标号过程中，并不是把所有的顶点都要标号，而是只要找出一条增流链即可（并不要求一次把所有的增流链全找完）。

(2)按汇点 y 的第一个标号反向追踪其先行顶点，再反向追踪下一个先行顶点，如此继续，直至源点 x ，这样就得到了一条由源到汇的增量链，及整个链上流的修改量 Δ_y ，给增流链上所有前向边的原流量上加上修改量 Δ_y ，所有后向边的原流量减去修改量 Δ_y ，其它不在增流链上边的流量值保持不变。调整完后，这个链就成了饱和链。然后再重新进入标号过程和调整过程，直至不存在增流链为止，这时就得到了最大流。

2. 应用

下面给出一个例子，请大家仔细体会一下算法的应用过程。

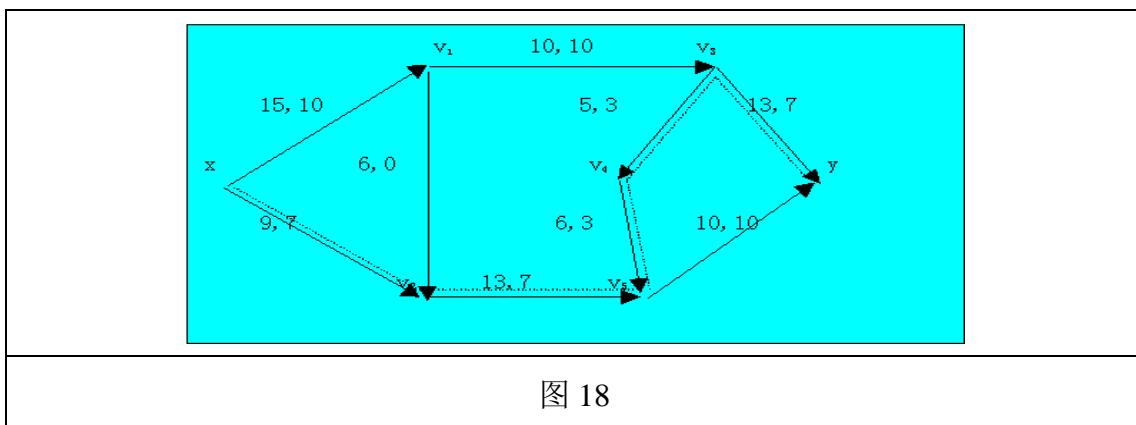


图 18

标号过程列入下表：

查视		X	X	v_1	V_2	v_5	v_4	v_3
标记	x	v_1	v_2	-	V_5	v_4	v_3	y
标号	$+\infty$	+5	+2		+2	-2	-2	+2

查视的点就是要标号的点的前行点。

如： v_1 的标号为 $(x, +, 5)$ ，其中 $5 = \min\{\infty, 15-10\}$

v_5 的标号为 $(v_2, +, 2)$ ，其中 $2 = \min\{2, 13-7\}$

v_4 的标号为 $(v_5, -, 2)$ ，其中 $2 = \min\{2, 6-3\}$ “-”表示 (v_4, v_5) 是后向边

修改量为 $\Delta = 2$ ，修改后如下图：

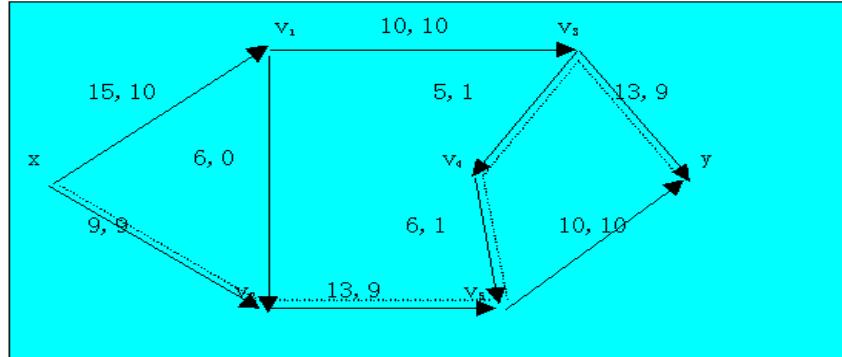


图 19

可以看出，链中有一条边 (x, v_2) 已是饱和边，所以该链就是一条饱和链。
再寻找其它的增流链如下：

标号过程列入下表：

查视		x	x	v ₁	v ₂	v ₅	v ₄	v ₃
标记	x	v ₁	v ₂	v ₂	v ₅	v ₄	v ₃	y
标号	$+\infty$	+5	-	+5	+4	-1	-1	+1

修改量为 $\Delta = 1$ ，修改后如下图：

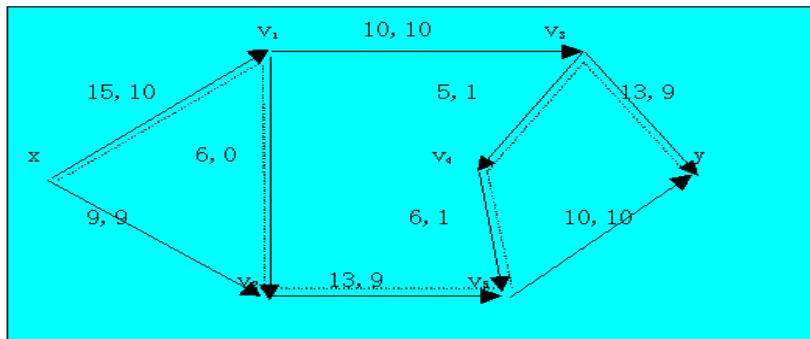


图 20

修改后，该链即为饱和链。

继续标号如下：

查视		x	x	v ₁	v ₂	v ₅	v ₅
标记	x	v ₁	v ₂	v ₂	v ₅	v ₄	y
标号	$+\infty$	+4	-	+4	+3	-	-

从标号过程可以看出，已经不存在由 x 到 y 的增流链了，求解过程结束。注意上面的这一步必不可少。

2.3.4 另外一种标号法（刘满凤书）

第一步标号过程

- (1) 对于源 v_s , 标号 $(-, +\infty)$;
- (2) 选择每个已经标号的顶点 v_i , 对于 v_i 的所有未标号的邻接点 v_j , 做如下处理:
 - 若 $(v_j, v_i) \in E$, 且 $f_{ij} > 0$, 则令 $\delta_j = \min(f_{ji}, \delta_i)$, 并且将 v_j 标号为 $(-v_i, \delta_j)$ 。
 $-v_i$ 表示 v_i 是 v_j 的后续点。
 - 若 $(v_i, v_j) \in E$, 且 $f_{ij} > c_{ij}$, 则令 $\delta_j = \min(c_{ij} - f_{ji}, \delta_i)$, 并且将 v_j 标号为 $(+v_i, \delta_j)$ 。
 $+v_i$ 表示 v_i 是 v_j 的前列点。

重复步骤 (2), 知道汇 v_T 不再有顶点可被标号。若 v_T 被标号, 则得到一条可增广链, 继续调整过程。若根据步骤 (2), v_T 不能获得标号, 则说明 f 已经是最大流。

第二步 调整过程

调整一般反向进行, 重新计算各条弧的流量。用 f'_{ij} 表示调整后的弧 (v_i, v_j) 的流量, 令

$$f'_{ij} = \begin{cases} f_{ij} + \delta_T, & (v_i, v_j) \text{ 为可增广链上的前向弧} \\ f_{ij} - \delta_T, & (v_i, v_j) \text{ 为可增广链上的后向弧} \\ f_{ij}, & (v_i, v_j) \text{ 不在可增广链上} \end{cases}$$

将调整后的流 f' 的标号全部去除, 并重复第一步和第二步的标号和调整过程, 对 f' 重新进行标号, 直到不能再进行标号为止。

举例如下: 在下图中, 该企业应如何安排运输, 才可获得从 v_s 到 v_t 的最大运输能力? 边上数字为最大流量, 和目前流量。

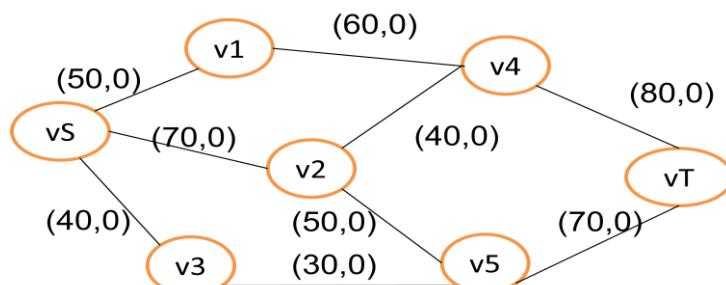
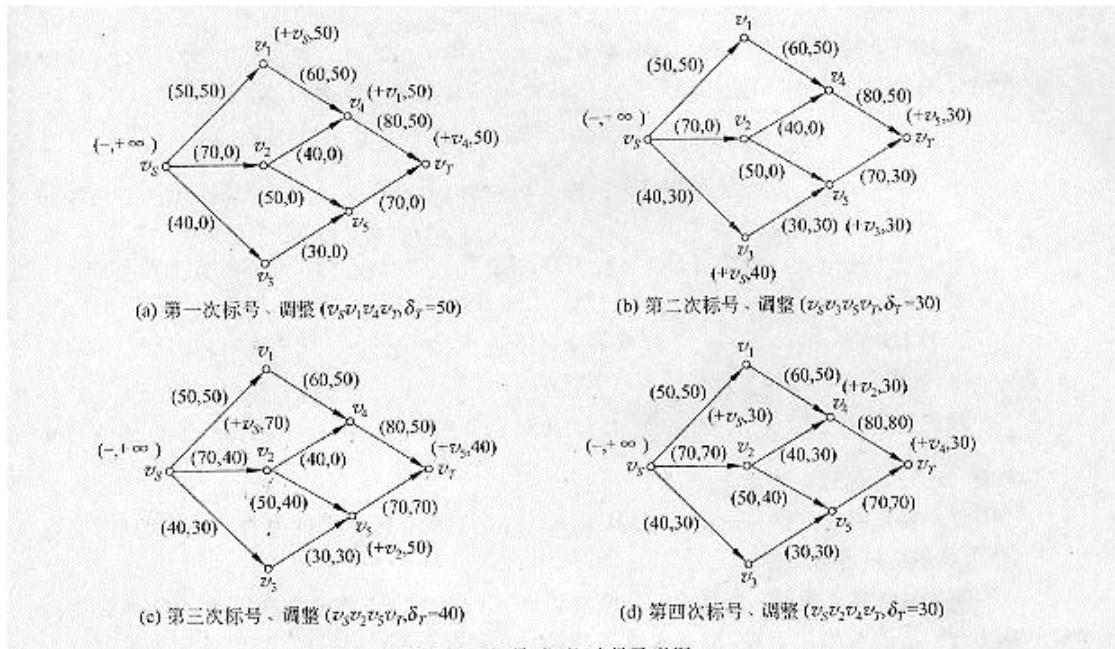


图 21 流量图



2.4 最小费用最大流问题

实际问题中,除了考虑流量的大小外,一般还要考虑和输送成本有关的因素,如运输费用(与距离有关)等等。这样我们在建立运输网络时,除了每条边有容量外,又对应了另外一个参数——距离(或者其它成本参数)。

在上例题中,某企业为了保证其某地区营销中心的配件供应,要求从其生产配件工厂 VS 运送尽可能多的配件到营销中心 Vt,运输网络如上图所示,弧上的数据为流量限制,同时给出了每条弧上的单位流量费用,如下图所示,该企业应如何安排运输,在获得从源到汇的最大运输能力前提下运输成本最小?该问题就是最大流最小费用问题。其描述如下:

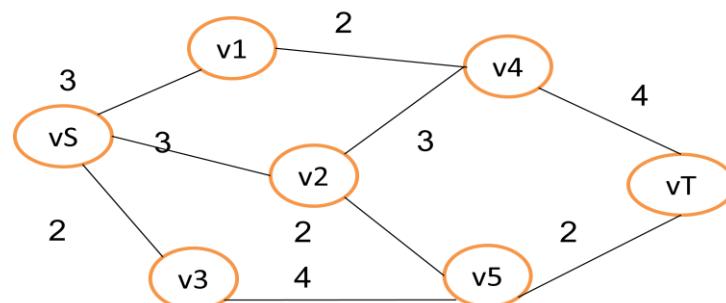


图 22 成本流量图

对于已知容量网络 $D=(V,E,C)$, 每条弧 (v_i, v_j) 上除了给出最大流容量 c_{ij} 外, 还给出了单位流量成本 $d_{ij} (\geq 0)$, 一般将带成本的容量网络称为赋权容量网络。记

为 $D=(V,E,C,d)$ 。求 D 的一个可行流 $f=\{f_{ij}\}$, 使得流量 $W(f)=w^*$, 且总费用 $d(f) = \sum_{(v_i,v_j) \in E} d_{ij} f_{ij}$ 达到最小。这类问题即为最小费用问题, 特别当要求 f 为最大流时, 该为问题既是最小费用最大流问题。这类问题一般采用对偶算法求解, 在介绍该类算法时, 先介绍几个有关可增广链的概念。

已知 $D=(V,E,C,d)$ 是一个赋权容量网络, f 是 D 上的可行流, Q 是从源到汇的一条关于 f 的可增广链。

链的费用: $\sum_{Q^+} d_{ij} f_{ij} - \sum_{Q^-} d_{ij} f_{ij}$ 为 Q 的费用, 记为 $d(Q)$, 其中 Q^+ 为 Q 的前向弧集合, 其中 Q^- 为 Q 的后向弧集合。

最小费用可增广链: 若 Q^* 是从源到汇所有可增广链中费用最小的链, 则称 Q^* 为最小费用可增广链。

对偶算法基本思路是: 先寻找一个流量满足 $W(f^{(0)}) < w^*$ 的最小费用流 $f^{(0)}$, 然后寻找从源到汇的可增广链 Q , 用最大流方法将 $f^{(0)}$ 调整到 $f^{(1)}$, 使得 $f^{(1)}$ 的流量为 $W(f^{(1)}) = W(f^{(0)}) + \theta$, 并保证 $f^{(1)}$ 是在流量 $W(f^{(0)}) + \theta$ 下的最小费用流, 继续以上步骤, 直到 $W(f^{(k)}) = w^*$ 为止。由于对偶算法中要求 $f^{(1)}$ 是在流量 $W(f^{(0)}) + \theta$ 下的最小费用流, 为此给出:

定理: 若 f 是流量为 $W(f)$ 下的最小费用流, Q 是关于 f 的从源到汇的一条最小费用可增广链, 则 f 经过 Q 调整流量 θ 后得到新的可行流 f' 一定是流量为 $W(f) + \theta$ 下的最小费用流。

由于 $d_{ij} \geq 0$, 因此通常在选取初始最小费用流时, 取 $f^{(0)} = \{0\}$, 接下来的难题是如何寻找关于 $f^{(0)}$ 的最小费用可增广链, 为此要引入长度网络概念。

对于赋权网络 $D=(V,E,C,d)$ 中，对于可行流 f ，保持原有网络各顶点，每条弧用两条相反的弧代替，各弧权重 l_{ij} 按如下规则。

$$1) \text{ 当弧 } (v_i, v_j) \in E \text{ 时, 令 } l_{ij} = \begin{cases} d_{ij} & f_{ij} < c_{ij} \\ +\infty & f_{ij} = c_{ij} \end{cases}$$

$$2) \text{ 当弧 } (v_j, v_i) \text{ 为原网络 } D \text{ 中 } (v_i, v_j) \text{ 反向弧时, 令 } l_{ij} = \begin{cases} -d_{ij} & f_{ij} > 0 \\ +\infty & f_{ij} = 0 \end{cases}$$

以上定义的网络称为长度网络，记为 $L(f)$ ，其本质是将费用看出长度，从而可以利用最短路径方法来求。

2.4.2 对偶算法基本步骤

其步骤如下：

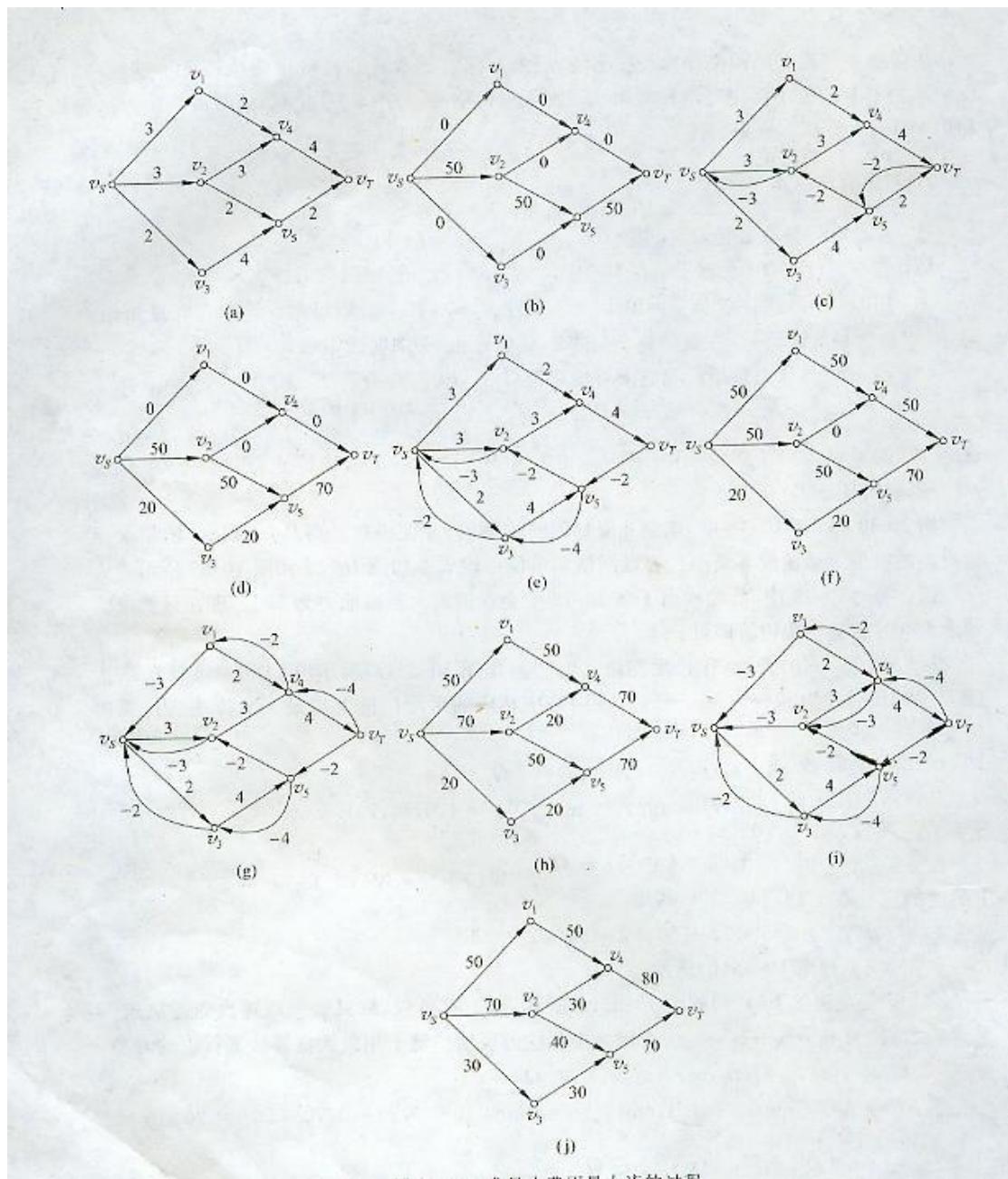
- (1) 初始可行流取零流，即 $f^{(0)}=\{0\}$ 。
- (2) 若有可行流 $f^{(k-1)}$ 流量满足 $W(f^{(k-1)}) < w^*$ ，构造长度网络 $L(f^{(k-1)})$ 。
- (3) 用最短路径算法求长度网络 $L(f^{(k-1)})$ 中从源到汇最短路，若不存在最短路，则 $f^{(k-1)}$ 即为最大流，不存在流量为 w^* 的流，算法停止，否则继续步骤
- (4) 在 D 中与这条最短路径相应的可增广链 Q 上，做 $f^{(k)}=f_Q^{(k-1)} + \theta$ ，其中

$$\theta = \min\{\min(c_{ij} - f_{ij}^{(k-1)}), \min f_{ij}^{(k-1)}\}$$

此时 $f^{(k)}$ 的流量为 $W(f^{(k)}) = W(f^{(k-1)}) + \theta$ ，若 $W(f^{(k)}) = w^*$ ，则停止，否则 $f^{(k)}$ 代替 $f^{(k-1)}$ 返回步骤 (2)

例子：在上例中，该企业如何安排运输，才使得在获得从源到汇的最大运输能力前提下运输成本最小？

解：已经求出了该企业的最大运输能力为 150。现在只要求流量为 150 时最小费用即可。其解题图解如下：（摘自刘满凤书）



3. 本章小节

该部分讲述了图论的一些基本概念，并给出了最短路、最小生成树、最大流，最小费用最大流等网络模型，在讲述这些模型方法时，同时给出了具体应用的环境和应用例子。