清华大学研究生公共课教材—— 数学系列

《最优化基础——模型与方法》系列教材

网络优化(第2版)

谢金星 邢文训 王振波 编著

清华大学出版社
北京

序　言

最优化是人们在工程技术、科学研究和经济管理等诸多领域中经常遇到的问题.结构设计要在满足强度要求等条件下使所用材料的总重量最轻;资源分配要使各用户利用有限资源产生的总效益最大;安排运输方案要在满足物质需求和装载条件下使运输费用最低;编制生产计划要按照产品工艺流程和顾客需求,尽量降低人力、设备、原材料等成本使总利润最高.可以预测,随着科学技术尤其是计算机技术的不断发展,以及数学理论与方法向各学科和各应用领域更广泛、更深入的渗透,在21世纪这个信息时代,最优化理论和技术必将在社会的诸多方面起着越来越大的作用.

解决实际生活中优化问题的手段大致有以下几种:一是靠经验的积累,凭主观作出判断;二是做试验选方案,比优劣定决策;三是建立数学模型,求解最优策略.虽然由于建模时要作适当的简化,可能使结果不一定非常完善,但是它基于客观数据,求解问题简便、经济,而且规模可以很大(将来会越来越大):人们还可以吸收从经验得到的规则,用实验不断校正建立的模型.随着数学方法和计算机技术的进步,用建模和数值模拟解决优化问题这一手段,将会越来越显示出它的效能和威力.显然,在决策定量化、科学化的呼声日益高涨的今天,数学建模方法的推广应用是符合时代潮流和形势发展需要的.

最优化理论、模型与方法所包含的内容很多,国内已出版了不少教材和专著介绍其各个分支.但是,一方面,近年来发展起来的、有着广泛应用背景的规划模型(如随机规划、模糊规划等),以及一些已经为许多人采用、受到广泛关注的优化算法(如模拟退火算法、遗传算法等)还缺乏详细和系统的介绍;另一方面,一些偏重优化理论和方法的教材,其要求难以与工科学生的数学知识衔接,也缺少对于应用来说十分重要的建模过程和软件介绍,而一些比较通俗的运筹学教材,则在加强理论基础,适应学生将来从事科研工作需要上考虑较少.这套教材试图弥补以上两方面的缺陷,力求体现下述特点:

1. 内容既包含传统的线性规划与非线性规划等部分,又纳入有广泛应用前景的随机规划和模糊规划;在传统内容中,既注重典型的数学思想和方法的系统叙述,又引入丰富的建模实例.

2. 数学基础既与工科学生所学知识衔接,又考虑到研究生阅读文献、从事科研工作的需要,适当提高理论基础的起点.

3. 对一般教材介绍的诸多算法进行精选,配合介绍一些应用软件,并引入近年来迅速发展的若干新算法.

本系列教材将陆续出版，首批四册为《线性与非线性规划》、《网络优化》、《现代优化计算方法》、《随机规划与模糊规划》.

由于水平有限，书中难免有缺陷和错误，诚恳希望读者予以批评指正.

《最优化基础——模型与方法》系列教材编委会

2005 年 3 月

前　言

我们生活在一个网络社会中. 从某种意义上说,现代社会是一个由计算机信息网络、电话通信网络、运输服务网络、能源和物质分派网络等各种网络所组成的复杂的网络系统. 网络优化就是研究如何有效地计划、管理和控制这个网络系统,使之发挥最大的社会和经济效益.

网络优化是运筹学(Operations Research)中的一个经典和重要的分支,所研究的问题涉及物资管理、经济管理、工业工程、交通运输、计算机科学与信息技术、通讯与网络技术、控制论及军事运筹学等诸多领域. 因此学习一些有关网络优化的基本知识,对许多专业的学生和许多领域的科技人员、管理人员都是相当必要的.

本书系统介绍了网络优化的基本模型和基本算法. 在基本模型方面,主要介绍树的问题(包括最小树、最小树形图、最大分枝)、最短路问题、最大流问题、最小费用流问题和匹配问题等. 在基本算法方面,由于处理上述这些问题的算法非常多,而且新算法还在不断涌现,从大量算法中选择哪些典型算法进行介绍本身是见仁见智的事,我们只能有选择地介绍其中部分算法. 我们既介绍一些比较经典的算法,也介绍一些比较新颖、实用的算法.

本书由 7 章组成. 第 1 章为概论,主要介绍图和网络的一些基本概念、图和网络在计算机上的表示方法,并初步介绍计算复杂性理论. 第 2 章介绍关于算法的一些基本知识,包括计算复杂性理论、近似算法、整数规划、动态规划,以及一些常用的网络搜索算法等. 第 3 章到第 7 章分别介绍树的问题、最短路问题、最大流问题、最小费用流问题和匹配问题,这是网络优化的基本内容.

本次成书过程中参考了大量国内外有关文献和教材,并对内容进行了进一步的筛选和扩充. 我们希望本书能适合于各个层次的读者阅读. 对于数学和计算机等理论性要求较高的专业的教学,最好能讲授本书全部或大部分内容,并要求读者在学习本课程之前已经掌握了线性规划的基础知识. 我们希望不仅能让这部分读者了解各种网络优化算法的基本思想和基本理论,而且能了解一些实现技巧并学会算法的复杂性分析方法. 在各章的练习题中,我们还配备了一定数量难度较高的题目作为对书中讲授内容的扩充. 对于其他理论性要求不太高的专业的教学或不太关心算法复杂性的读者来说,可以在阅读中略去部分理论内容. 对于只想对网络优化有所了解或只希望从本书中查询一些具体算法的读者来说,完全可以略去全部的理论部分,而只了解相应的模型和算法就可以了.

自 2000 年以来,笔者以本书为教材在清华大学讲授研究生课程"网络优化". 在此期间,得到多方面的反馈和宝贵意见. 在此基础上,我们对本书的结构和内容进行了一些调整和修

改.在第2版中增加了算法基础一章,介绍一些准备知识,包括复杂性理论、近似算法及一些常用的算法.第1版中的整数规划和动态规划两章不再作为单独的章节出现,一些相关的内容安排在算法基础一章.

最后,对我们家人的理解、耐心和支持表示由衷的谢意!由于我们的水平有限,恳请读者对本书的不足之处提出批评指正.

谢金星 邢文训 王振波

2009年春于清华园

目　录

第1章 概 论

我们生活在一个网络社会中. 从某种意义上说，现代社会是一个由计算机信息网络、电话通信网络、运输服务网络、能源和物质分派网络等各种网络所组成的复杂的网络系统. 网络优化就是研究如何有效地计划、管理和控制这个网络系统，使之发挥最大的社会和经济效益.

网络优化是运筹学(Operations Research)中的一个经典和重要的分支，所研究的问题涉及经济管理、工业工程、交通运输、计算机科学与信息技术、通讯与网络技术等诸多领域. 本书中将要讨论的最短路问题、最大流问题、最小费用流问题和匹配问题等都是网络优化的基本问题.

本章主要介绍网络优化问题的一些实际例子以及图与网络的基本概念，初步介绍计算复杂性理论，为后续章节的学习奠定基础.

1.1 网络优化问题的例子

我们首先通过一些例子来了解网络优化问题.

例 1.1 公路连接问题

某地区有若干个主要城市，现准备修建高速公路把这些城市连接起来，使得从其中任何一个城市都可以经高速公路直接或间接到达另一个城市. 假定已经知道了任意两个城市之间修建高速公路的成本，那么应如何决定在哪些城市间修建高速公路，使得总成本最小？ □

例 1.2 最短路问题(shortest path problem，SPP)

一名货柜车司机奉命在最短的时间内将一车货物从甲地运往乙地. 从甲地到乙地的公路网纵横交错，因此有多种行车路线，这名司机应选择哪条线路呢？ 假设货柜车的运行速度是恒定的，那么这一问题相当于需要找到一条从甲地到乙地的最短路. □

例 1.3 运输问题(transportation problem)

某种原材料有 M 个产地，现在需要将原材料从产地运往 N 个使用这些原材料的工厂. 假定 M 个产地的产量和 N 家工厂的需要量已知，单位产品从任一产地到任一工厂的运费已知，那么如何安排运输方案可以使总运输成本最低？ □

例 1.4 指派问题(assignment problem)

一家公司经理准备安排 N 名员工去完成 N 项任务，每人一项. 由于各员工的特点不同，不同的员工去完成同一项任务时所获得的回报是不同的. 如何分配工作方案可以使总回

报最大? □

例 1.5 中国邮递员问题(chinese postman problem,CPP)

一名邮递员负责投递某个街区的邮件.如何为他(她)设计一条最短的投递路线(从邮局出发,经过投递区内每条街道至少一次,最后返回邮局)?由于这一问题是我国管梅谷教授1960年首先提出的,所以国际上称之为中国邮递员问题. □

例 1.6 旅行商问题(traveling salesman problem,TSP)

一名推销员准备前往若干城市推销产品.如何为他(她)设计一条最短的旅行路线(从驻地出发,经过每个城市恰好一次,最后返回驻地)?这一问题的研究历史十分悠久,通常称之为旅行商问题. □

上述问题有两个共同的特点:一是它们的目的都是从若干可能的安排或方案中寻求某种意义下的最优安排或方案,数学上把这种问题称为最优化或优化(optimization)问题;二是它们都易于用图形的形式直观地描述和表达,数学上把这种与图相关的结构称为网络(network).与图和网络相关的最优化问题就是网络最优化或称网络优化(network optimization)问题.所以上面例子中介绍的问题都是网络优化问题.由于多数网络优化问题是以网络上的流(flow)为研究的对象,因此网络优化这门课程在许多学校里又常常被称为网络流(network flows)或网络流规划等.

下面首先简要介绍图与网络的一些基本概念.

1.2 图与网络

1.2.1 有向图与网络的基本概念

定义 1.1 一个有向图(directed graph 或 digraph) G 是由一个非空有限集合 $V(G)$ 和 $V(G)$ 中某些元素的有序对集合 $A(G)$ 构成的二元组,记为 $G=(V(G),A(G))$. 其中 $V(G)=\{v_1,v_2,\cdots,v_n\}$ 称为图 G 的节点集(node set)或顶点集(vertex set),$V(G)$ 中的每一个元素 v_i $(i=1,2,\cdots,n)$ 称为该图的一个节点(node)或顶点(vertex);$A(G)=\{a_1,a_2,\cdots,a_m\}$ 称为图 G 的弧集(arc set),$A(G)$ 中的每一个元素 a_k(即 $V(G)$ 中某两个元素 v_i,v_j 的有序对)记为 $a_k=(v_i,v_j)$ 或 $a_k=v_iv_j$ $(k=1,2,\cdots,m)$,被称为该图的一条从 v_i 到 v_j 的弧(arc).在不引起混淆的情况下,记号 $V(G)$ 和 $A(G)$ 中也可以省略 G,即分别记节点集、弧集为 V 和 A,而记有向图 $G=(V,A)$.

如果对有向图 G 中的每条弧赋予一个或多个实数,得到的有向图称为赋权有向图或有向网络,简称为网络(network).为了讨论方便,本书对图和网络不作严格区分,因为任何图总是可以赋权的.

当弧 $a_k=(v_i,v_j)$ 时,称 v_i,v_j 为弧 a_k 的端点(endpoint),其中 v_i 为尾(tail),v_j 为头(head),并称 v_j 在 G 中与 v_i 相邻(adjacent)或 v_j 是 v_i 的邻居(neighbor);弧 a_k 称为与节点

v_i, v_j 关联 (incident)，并称弧 a_k 为 v_i 的出弧(outgoing arc)，为 v_j 的入弧(incoming arc). 如果某两条弧至少有一个公共端点，则称这两条弧在图 G 中相邻.

图的概念是现实生活中图形概念的数学抽象，因此一个图也可以用图形来直观表示：用小圆圈表示节点，用节点间带箭头的连线表示弧及其方向(箭头从弧的尾指向弧的头). 例如，图 1.1 的图形表示的是有向图 $G=(V,A)$ 其中节点集 $V=\{v_1,v_2,v_3,v_4,v_5\}$，弧集 $A=\{a_1,a_2,a_3,a_4,a_5,a_6\}$，弧 $a_1=(v_1,v_2)$，$a_2=(v_1,v_2)$，$a_3=(v_2,v_3)$，$a_4=(v_3,v_4)$，$a_5=(v_4,v_1)$，$a_6=(v_3,v_3)$. 在该图中，弧 a_1,a_2 都是从节点 v_1 指向节点 v_2，这种弧称为多重弧(multiarc)，相应的图称为多重图(multi-graph). 值得注意的是：这时弧集合 A 也应当理解成多重集(multi-set)，即 A 中可以包括多个相同的元素，这与我们通常所说的普通集合的概念有所不同. 在该图中，弧 a_6 的头和尾都相同(v_3)，这种弧称为环 (loop). 在该图中，没有任何弧与节点 v_5 关联，这样的节点称为孤立点(isolated vertex).

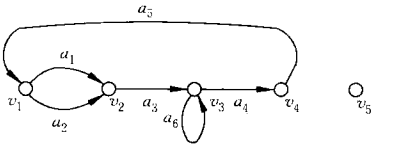


图 1.1 有向图的一般例子

图中节点的个数称为图的阶(order). 通常用 $|V|$ 或 n 表示顶点个数(即图的阶数)，用 $|A|$ 或 m 表示弧的条数. 特别地，我们称只有一个节点的图为平凡图(trivial graph)；不包括任何弧的图为空图(null graph). 没有环、且没有多重弧的图称为简单图(simple graph). 例如，在图 1.1 所示的图 G 中，图的阶数为 $n=|V|=5$，弧的条数为 $m=|A|=6$，图 G 不是平凡图或空图，也不是简单图. 如不特别说明，本书中以后假设讨论的都是简单图.

图中与一个节点关联的出弧的数目称为该节点的出度(outdegree)，入弧的数目称为该节点的入度(indegree)，而出度与入度之和称为该节点的度(degree). 度数为偶数的节点称为偶点(even point)，否则称为奇点(odd point). 例如，在图 1.1 所示的图 G 中，节点 v_2 的度为 3，一般记为 $d_G(v_2)=3$；节点 v_2 的出度为 1，一般记为 $d_G^+(v_2)=1$；节点 v_2 的入度为 2，一般记为 $d_G^-(v_2)=2$. 在不引起混淆的情况下，也可以省略 G，即分别记为 $d(v_2)=3$；$d^+(v_2)=1$；$d^-(v_2)=2$. 图 1.1 所示的图 G 中节点 v_1,v_2 为奇点，节点 v_3,v_4,v_5 为偶点.

假设 $G'=(V',A')$ 和 $G=(V,A)$ 是两个有向图，如果 $V'\subseteq V, A'\subseteq A$，则图 $G'=(V',A')$ 称为图 $G=(V,A)$ 的子图(subgraph)，可简记为 $G'\subseteq G$. 任给 $V'\subseteq V$，以 V' 为节点集的 G 的最大子图定义为：节点集为 V'，弧集为两个端点都在 V' 中的 G 中所有弧的集合. 以 V' 为节点集的 G 的最大子图称为 G 的由 V' 导出的子图(induced subgraph)，即如果 $V'\subseteq V, A'=\{(v_i,v_j)\in A \mid v_i,v_j\in V'\}$，则图 $G'=(V',A')$ 称为图 $G=(V,A)$ 的由 V' 导出的子图(简称节点导出子图)，记为 $G[V']$. 类似地，可以定义弧子集 $A'\subseteq A$ 导出的子图(简称弧导出子图) $G[A']=(V',A')$，其中 $V'=\{v\in V \mid \exists\, v'\in V$，使得 $(v,v')\in A'$ 或 $(v',v)\in A'\}$. 由此可以看出，给定 $V'\subseteq V$(或 $A'\subseteq A$)，图 $G(V,A)$ 的导出子图是唯一的. 例如，在图 1.1 所示的图 G 中，$G[\{v_1,v_2\}]=(\{v_1,v_2\},\{a_1,a_2\})$，$G[\{a_1,a_3\}]=(\{v_1,v_2,v_3\},\{a_1,a_3\})$，$G[\{a_1,a_3,a_5\}]=(\{v_1,v_2,v_3,v_4\},\{a_1,a_3,a_5\})$.

图 $G=(V,A)$的支撑子图 (spanning subgraph,又称生成子图)是包含 G 的所有节点的子图,即当 $V'=V,A'\subseteq A$ 时,$G'=(V',A')$称为 $G=(V,A)$的支撑子图. 由此可以看出,图 $G=(V,A)$的支撑子图一般是不唯一的. 例如,在图 1.1 所示的图 G 中,$(\{v_1,v_2,v_3,v_4,v_5\},\{a_1,a_3,a_5\})$和$(\{v_1,v_2,v_3,v_4,v_5\},\{a_1,a_2,a_5,a_6\})$等都是 G 的支撑子图.

有向图中的途径(walk)是该图的一些节点 $i_1,i_2,\cdots,i_r$ 和弧 $a_1,a_2,\cdots,a_{r-1}$ 所组成的子图,这些节点与弧可以交错排列成点弧序列 $i_1,a_1,i_2,a_2,\cdots,a_{r-1},i_r$,其中 $a_k=(i_k,i_{k+1})$或 $a_k=(i_{k+1},i_k)(\forall k=1,2,\cdots,r-1)$. 如果该序列中的所有弧都指向同一方向,即 $a_k=(i_k,i_{k+1})(\forall k=1,2,\cdots,r-1)$,则该途径称为有向途径(directed walk). 在不引起混淆的情况下(如在简单图的情形下),可以仅仅用节点序列 $i_1,i_2,\cdots,i_r$ 或弧序列 $a_1,a_2,\cdots,a_{r-1}$ 来表示途径. 我们称 $r-1$ 为途径的长,分别称 i_1,i_r 为途径的起点和终点.

有向图中的路 (path)是该图的不包含重复节点的途径. 路中的弧可以分成两类:其中一类弧 $a_k=(i_k,i_{k+1})$的方向与路的起点到终点的方向一致,称为路的前向弧(forward arc);另一类弧 $a_k=(i_{k+1},i_k)$的方向与路的起点到终点的方向不一致,称为路的后向弧或反向弧(backward arc). 路 P 的所有前向弧的集合一般记为 P^+,反向弧的集合记为 P^-. 有向图中的有向路(directed path)是该图的不包含重复节点的有向途径,即不包含反向弧的路. 例如,在图 1.1 所示的图 G 中,$v_1\ a_5\ v_4\ a_4\ v_3$ 就是从节点 v_1 到 v_3 的一条路,但不是从节点 v_1 到 v_3 的有向路,因为路上的弧都是反向弧;而 $v_3\ a_4\ v_4\ a_5\ v_1$ 就是从节点 v_3 到 v_1 的一条有向路. 又如,$v_2\ a_3\ v_3\ a_1\ v_4$ 是从节点 v_2 到 v_4 的一条有向路.

有向图 $G=(V,A)$中,如果$(i_1,i_r)\in A$ 或$(i_r,i_1)\in A$,则路 $i_1,i_2,\cdots,i_r$ 加上弧(i_1,i_r)或(i_r,i_1)组成的途径称为该有向图的圈(cycle),即圈是起点和终点重合且不含其他重复节点的途径. 同路的前向弧和反向弧的定义类似,可以定义圈的前向弧和反向弧. 有向图中的有向圈(directed cycle)是该图的不包含反向弧的圈,即由路 $i_1,i_2,\cdots,i_r$ 加上弧(i_r,i_1)组成. 如在图 1.1 所示的图 G 中,$v_1\ a_1\ v_2\ a_2\ v_1$ 就是一个圈,但不是有向圈, 而 $v_1\ a_1\ v_2\ a_3\ v_3\ a_4\ v_4\ a_5\ v_1$ 就是一个有向圈. 不包含有向圈的图称为无圈图(acyclic graph). 因此,图 1.1 所示的图 G 不是无圈图.

对于有向图 $G=(V,A)$中的两个节点,如果在图中至少存在一条路把它们连接起来,则称这两个节点是连通的(connected). 如果图中任意两个节点都是连通的,则称该图为连通的;否则称为不连通的(disconnected). 若 $G=(V,A)$不连通,$G'=(V',A')$是 $G=(V,A)$的连通子图,且不存在 $G=(V,A)$的连通子图 $G_1=(V_1,A_1)$满足 $G'\subseteq G_1,G'\neq G_1$,则称 $G'=(V',A')$为 G 的连通分支(component). 也就是说,连通分支是图中的极大连通子图,不连通图可以分解成一些连通分支的并,而连通图只有一个连通分支. 例如,图 1.1 所示的图 G 是不连通的,G 包括两个连通分支,其中一个是只包括节点 v_5 的空图,另一个是除去节点 v_5 后所有节点和弧所构成的子图.

如果有向图中从任意一个节点出发,都存在至少一条有向路到达任意另一个节点,则称该图为强连通的(strongly connected). 同样可以与连通分支类似地定义强连通分支. 例如,

图 1.1 所示的图 G 除去节点 v_5 后所有节点和弧所构成的子图是强连通的(因此,这样的子图不仅是 G 的连通分支,也同时是一个强连通分支).

设 S,T 是节点集合 V 的一个划分(即 $S,T\subseteq V;S,T\neq\varnothing;S\cup T=V;S\cap T=\varnothing$),则称两个端点分别位于 S,T 的弧为一个割(cut),记为$[S,T]=\{(i,j)\in A\mid i\in S,j\in T\}\cup\{(j,i)\in A\mid i\in S,j\in T\}$.例如,在图 1.1 所示的图 G 中,若 $S=\{v_1,v_4,v_5\},T=\{v_2,v_3\}$,则$[S,T]=\{a_1,a_2,a_4\}$.

1.2.2 无向图与无向网络的基本概念

有时我们关心的只是两个节点之间的联系,而不关心这种联系的方向性,这样就自然得到了无向图和无向网络的概念.

定义 1.2 一个无向图(undirected graph)G 是由一个非空有限集合 V 和 V 中某些元素的无序对集合 E 构成的,即 $G=(V,E)$.其中 $V=V(G)=\{v_1,v_2,\cdots,v_n\}$仍称为图 G 的节点集(node set)或顶点集(vertex set),V 中的每一个元素 $v_i(i=1,2,\cdots,n)$ 称为该图的一个节点(node)或顶点(vertex);$E=E(G)=\{e_1,e_2,\cdots,e_m\}$通常称为图 G 的边集合(edge set),E 中的每一个元素 e_k(即 V 中某两个元素 v_i,v_j 的无序对) 记为$e_k=(v_i,v_j)$或 $e_k=v_iv_j$,或 $e_k=v_jv_i(k=1,2,\cdots,m)$,称为该图的一条边(edge).边上赋权的无向图称为赋权无向图或无向网络(undirected network).同有向图情况一样,本书对无向图和无向网络也不作严格区分.

有向图的其他相关概念都可以自然地推广到无向图上来.自然,有时也会有一些微小的差异:如无向图的一条边的端点不再有头和尾之分;无向图的一个节点的度也不再有入度和出度之分;无向图中的一条路(或一个圈)上的一条边也不再有前向和反向之分;无向图的连通性不再有所谓的强连通概念;等等.

一个无向图也可以用图形来直观表示:用小圆圈表示节点,用节点间的连线表示边.如图 1.2 的图形表示的是无向图 $G=(V,E)$,其中 $V=\{v_1,v_2,v_3,v_4,v_5\}$, $E=\{e_1,e_2,e_3,e_4,e_5,e_6\},e_1=(v_1,v_2),e_2=(v_1,v_2),e_3=(v_2,v_3),e_4=(v_3,v_4),e_5=(v_1,v_4),e_6=(v_3,v_3)$. 这个图显然是图 1.1 所示的有向图中忽略弧的方向而得到的,有人把这样得到的无向图称为原有向图的基础图(underlying graph).

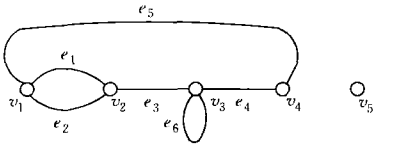


图 1.2 无向图的一般例子

下面再介绍两种特殊的图.若无向图 G 的任意两个节点间有且只有一条边相连,则称其为完全图(complete graph).n 阶完全图记为 K_n.若图 $G=(V,E)$的节点集 V 可以表示成两个互不相交的非空子集 S,T 的并集,且使得 G 中每一条边的一个端点在 S 中,另一端点在 T 中,则称 G 为二部图(bibartite graph).当$|S|=p,|T|=q$,且 S 与 T 中任意两对节点都相邻时,称 G 为完全二部图,记为 $K_{p,q}$.类似地,也可以定义有向的完全图和有向的完全二部图.

其他与图相关的概念还有很多，其中有些概念我们会在后续章节中用到时再作介绍. 对图论有兴趣的读者可以参考图论方面的专门书籍，这里就不详细介绍了. 值得提请读者注意的是，有关图的术语目前并不完全统一，不同的作者可能用不同的术语表达同一个概念，也可能使用的术语相同，表达的概念却并不完全相同. 因此在阅读有关图和网络的书籍和文献时，应注意防止由此而可能引起的误解. 此外，本书中在不引起混淆的情况下，有时将有向图和无向图都简称为图(graph)，具体所指应可从上下文中看出. 并且，我们也不再对弧和边的概念作严格区分，而认为它们表示相同的概念.

1.3 图与网络的数据结构

网络优化研究的是网络上的各种优化模型与算法. 为了在计算机上实现网络优化的算法，首先我们必须有一种方法(即数据结构)在计算机上描述图与网络. 一般来说，算法的好坏(即我们后面将要介绍的复杂度概念)与网络的具体表示方法，以及中间结果的操作方案是有关系的. 这里我们介绍计算机上用来描述图与网络的5种常用表示方法：邻接矩阵表示法、关联矩阵表示法、弧表表示法、邻接表表示法和星形表示法. 在下面的讨论中，我们首先假设 $G=(V,A)$ 是一个简单有向图，$|V|=n$，$|A|=m$，并假设 V 中的节点用自然数 $1,2,\cdots,n$ 表示或编号，A 中的弧用自然数 $1,2,\cdots,m$ 表示或编号. 对于有多重边或无向网络的情况，我们只是在讨论完简单有向图的表示方法之后，给出一些说明.

1.3.1 邻接矩阵表示法

邻接矩阵表示法是将图以邻接矩阵(adjacency matrix)的形式存储在计算机中. 图 $G=(V,A)$ 的邻接矩阵 $\boldsymbol{C}$ 是如下定义的：$\boldsymbol{C}$ 是一个 $n\times n$ 的0-1矩阵，即

$$\boldsymbol{C}=(c_{ij})_{n\times n}\in\{0,1\}^{n\times n},$$

$$c_{ij}=\begin{cases}0, & (i,j)\notin A,\\ 1, & (i,j)\in A.\end{cases}$$

也就是说，如果两节点之间有一条弧，则邻接矩阵中对应的元素为1；否则为0. 每行元素之和正好是对应节点的出度，每列元素之和正好是对应节点的入度. 可以看出，这种表示法非常简单、直接. 但是，在邻接矩阵的所有 n^2 个元素中，只有 m 个为非零元. 如果网络比较稀疏，这种表示法浪费大量的存储空间，从而增加了在网络中查找弧的时间.

例如，图1.3所示的网络图，可以用邻接矩阵表示为

$$\begin{bmatrix}0&1&1&0&0\\0&0&0&1&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&1&1&0\end{bmatrix}.$$

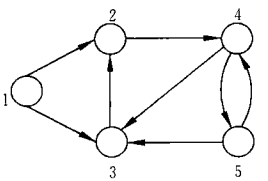


图1.3 有向简单图的例子

同样，对于网络中的权，也可以用类似邻接矩阵的 $n\times n$ 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.

1.3.2 关联矩阵表示法

关联矩阵表示法是将图以关联矩阵(incidence matrix)的形式存储在计算机中. 图 $G=(V,A)$ 的关联矩阵 $\boldsymbol{B}$ 是如下定义的：$\boldsymbol{B}$ 是一个 $n\times m$ 的矩阵，即

$$\boldsymbol{B}=(b_{ik})_{n\times m}\in\{-1,0,1\}^{n\times m},$$

$$b_{ik}=\begin{cases}1, & \exists j\in V,k=(i,j)\in A,\\ -1, & \exists j\in V,k=(j,i)\in A,\\ 0, & \text{其他}.\end{cases}$$

也就是说，在关联矩阵中，每行对应于图的一个节点，每列对应于图的一条弧. 如果一个节点是一条弧的起点，则关联矩阵中对应的元素为 1；如果一个节点是一条弧的终点，则关联矩阵中对应的元素为 -1；如果一个节点与一条弧不关联，则关联矩阵中对应的元素为 0. 对于简单图，关联矩阵每列只含有两个非零元(一个 $+1$，一个 -1). 在关联矩阵中，每行元素 1 的个数正好是对应节点的出度，每行元素 -1 的个数正好是对应节点的入度. 可以看出，这种表示法也非常简单、直接. 但是，在关联矩阵的所有 nm 个元素中，只有 $2m$ 个为非零元. 如果网络比较稀疏，这种表示法也会浪费大量的存储空间. 但由于关联矩阵有许多特别重要的理论性质(我们以后将会介绍)，因此它在网络优化中是非常重要的概念.

例如，图 1.3 所示的网络图，如果关联矩阵中每列对应弧的顺序为(1,2)，(1,3)，(2,4)，(3,2)，(4,3)，(4,5)，(5,3)和(5,4)，则关联矩阵表示为

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0\\ 0 & -1 & 0 & 1 & -1 & 0 & -1 & 0\\ 0 & 0 & -1 & 0 & 1 & 1 & 0 & -1\\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}.$$

同样，对于网络中的权，也可以通过对关联矩阵的扩展来表示. 例如，如果网络中每条弧赋有一个权，我们可以把关联矩阵增加一行，把每一条弧所对应的权存储在增加的行中. 如果网络中每条弧赋有多个权，我们可以把关联矩阵增加相应的行数，把每一条弧所对应的权存储在增加的行中.

1.3.3 弧表表示法

弧表表示法将图以弧表(arc list)的形式存储在计算机中. 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，共需 $2m$ 个存储单元，因此当网络比较稀疏时比较方便. 此外，对于网络图中每条弧上的权，也要对应地用额外的

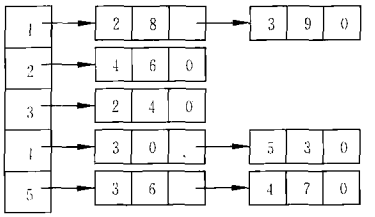
存储单元表示. 例如,图 1.3 所示的网络图,假设弧(1,2),(1,3),(2,4),(3,2),(4,3),(4,5),(5,3)和(5,4)上的权分别为 8,9,6,4,0,3,6 和 7,则弧表表示如下:

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

为了便于检索,一般按照起点、终点的字典序顺序存储弧表,如上面的弧表就是按照这样的顺序存储的.

1.3.4 邻接表表示法

邻接表表示法将图以邻接表(adjacency lists)的形式存储在计算机中. 所谓图的邻接表,也就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. 邻接表表示法就是对图的每个节点,用一个单向链表列出从该节点出发的所有弧,链表中每个单元对应于一条出弧. 为了记录弧上的权,链表中每个单元除列出弧的另一个端点外,还可以包含弧上的权等作为数据域. 图的整个邻接表可以用一个指针数组表示. 例如,图 1.3 所示的网络图,邻接表表示为



这是一个 5 维指针数组,每一维(上面表示法中的每一行)对应于一个节点的邻接表,如第 1 行对应于第 1 个节点的邻接表(即第 1 个节点的所有出弧). 每个指针单元的第 1 个数据域表示弧的另一个端点(弧的头),后面的数据域表示对应弧上的权. 如第 1 行中的"2"表示弧的另一个端点为 2(即弧为(1,2)),"8"表示对应弧(1,2)上的权为 8;"3"表示弧的另一个端点为 3(即弧为(1,3)),"9"表示对应弧(1,3)上的权为 9. 又如,第 5 行说明节点 5 出发的弧有(5,3)、(5,4),他们对应的权分别为 6 和 7.

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). 例如上面例子,$A(1)=\{2,3\}$,$A(5)=\{3,4\}$等. 这种记法我们在本书后面将会经常用到.

1.3.5 星形表示法

星形(star)表示法的思想与邻接表表示法的思想有一定的相似之处. 对每个节点,它也