运筹学导论

修贤超 欧芳 编著

前言

运筹学 (Operations Research, OR) 是一门融合数学、系统科学与工程管理的交叉学科。其核心在于运用数学建模、优化算法、系统分析等工具, 实现有限资源的合理配置。运筹学思想源远流长, 我国古代“田忌赛马”“丁渭修皇宫”“沈括运军粮”等经典案例, 早已蕴含朴素而深刻的运筹智慧。在当今科技飞速发展的时代, 运筹学已成为支撑现代工程技术与人工智能发展的重要理论基础。在工程领域, 运筹学广泛应用于生产调度、设备故障诊断、复杂系统可靠性分析等方面。在人工智能领域, 强化学习策略优化、组合优化问题求解、稀疏优化算法设计等, 均以运筹学为基石。

本书共分为八章, 循序渐进地引导读者掌握运筹学的基本思想与方法。第一章为绪论, 通过案例帮助读者建立学科认知。第二章至第六章聚焦运筹学的五大分支: 第二章为线性规划, 从实际问题出发构建数学模型, 以图解法直观揭示最优解的内涵, 并深入介绍单纯形法的原理与求解步骤, 同时引入对偶理论及其基本性质; 第三章为整数规划, 重点讲解分枝定界法、0−1 规划隐枚举法以及指派问题的匈牙利算法; 第四章为非线性规划, 分别针对无约束优化与约束优化问题, 系统介绍相应的理论与方法; 第五章为动态规划, 梳理多阶段决策过程的优化思想与基本概念; 第六章为博弈论, 重点介绍矩阵博弈的纯策略和混合策略。第七章为 Python 编程实战, 以 Gurobi 为优化求解器, 实现理论与算法的有机结合。第八章为展望, 介绍大模型驱动下运筹学的发展与应用前景。

本书适用于高年级本科生, 建议教学时长为 32 学时。既可作为自动化专业的基础课教材, 也可供管理、数学、计算机等相关专业选用。本书内容曾在上海大学自动化系“运筹模型与案例”课程中使用, 谨向参与课程反馈的各位同学致以诚挚谢意。课题组研究生张鹏飞在初稿整理过程中付出了大量辛勤的工作, 特此致谢。本书的出版工作得到了国家自然科学基金项目 (12371306) 的资助, 在此一并致谢。

限于编者水平, 书中难免存在疏漏与不妥之处, 恳请广大读者批评指正。

编者

2026 年 4 月

符号表

$\mathbb{R}$	实数集
$\mathbb{R}^n$	n 维实向量空间
$\mathbb{R}^{m\times n}$	$m\times n$ 维实矩阵空间
$x\in\mathbb{R}$	标量
$\boldsymbol{x}\in\mathbb{R}^n$	n 维 (列) 向量
$\boldsymbol{X}\in\mathbb{R}^{m\times n}$	$m\times n$ 维矩阵
$\boldsymbol{x}^{\mathrm{T}}$	向量 $\boldsymbol{x}$ 的转置
$\boldsymbol{x}^{(k)}$	第 k 次迭代时的向量 $\boldsymbol{x}$
$\boldsymbol{x}^*$	问题的最优解
x_i	向量 $\boldsymbol{x}$ 的第 i 个分量
x_{ij}	矩阵 $\boldsymbol{X}$ 的第 i 行第 j 列分量
$\boldsymbol{x}\geq 0$	向量 $\boldsymbol{x}$ 的所有分量均大于或等于 0
$\nabla f(\boldsymbol{x})$	函数 f 在点 $\boldsymbol{x}$ 处的梯度
$\nabla^2 f(\boldsymbol{x})$	函数 f 在点 $\boldsymbol{x}$ 处的 Hessian 矩阵
$\partial f(\boldsymbol{x})$	函数 f 在点 $\boldsymbol{x}$ 处的次微分
$\|\boldsymbol{x}\|$	向量 $\boldsymbol{x}$ 的欧几里得范数
$\|x\|$	标量 x 的绝对值
$\lceil x\rceil$	标量 x 向上取整
$\lfloor x\rfloor$	标量 x 向下取整
$\ln(x)$	标量 x 的自然对数
λ_k	第 k 次迭代时的步长

目　录

1

绪论

何为运筹学?《中国大百科全书》将其定义为: 用数学方法研究经济、民政和国防等部门在内外环境的约束条件下合理分配人力、物力、财力等资源, 使实际系统有效运行的技术科学, 它可以用来预测发展趋势, 制订行动规划或优选可行方案。

作为系统工程与现代管理科学的核心基础理论, 运筹学既是解决复杂决策问题的重要方法, 也是提升系统运行效率、优化资源配置的关键工具。从本质上看, 运筹学源于朴素的选优思想。人们在完成特定任务时, 总会在现有条件约束下主动寻求最优的解决方案, 这正是运筹学思想的直观体现。下面结合具体应用案例, 阐释运筹学的实践价值。

例 1.1 高速铁路的运营效益与列车运行数量密切相关。图1.1展示了 12306 网站中上海至北京的列车信息, 每新增一对列车开行方案, 年营业收入可提升约数亿元。

车次	出发站 到达站	出发时间 到达时间	历时	商务座 特等座	优选 一等座	一等座	二等座 二等包座	高级 软卧	软卧/动卧 一等卧	硬卧 二等卧	软座	硬座	无座	其他	备注
G532 智 复 静	始 上海虹桥 终 北京南	06:31 12:18	05:47 当日到达	候补	--	13	有	--	--	--	--	--	有	--	预订
G548 智 复 静	始 上海虹桥 终 北京南	06:32 12:39	06:07 当日到达	5	--	有	有	--	--	--	--	--	有	--	预订
G2 复 静	始 上海虹桥 终 北京南	06:43 11:32	04:49 当日到达	9	--	有	有	--	--	--	--	--	有	--	预订
G4 智 复 静	始 上海 终 北京南	07:00 11:37	04:37 当日到达	11	候补	有	有	--	--	--	--	--	有	--	预订
G550 智 复 静	始 上海虹桥 终 北京南	07:22 13:21	05:59 当日到达	候补	--	有	有	--	--	--	--	--	无	--	预订
G566 智 复 静	过 上海虹桥 终 北京南	07:27 13:36	06:09 当日到达	候补	--	候补	有	--	--	--	--	--	无	--	预订
G598 复 静	始 上海虹桥 终 北京南	07:38 13:32	05:54 当日到达	12	--	有	有	--	--	--	--	--	有	--	预订

图 1.1　上海至北京列车信息

例 1.2 一个邮递员从邮局出发，需遍历其管辖范围内的每一条街道至少一次，最终返回邮局，如何规划路线使总行程最短？该问题由我国数学家管梅谷于 1962 年首次提出，又称中国邮递员问题。其理论可追溯至经典的哥尼斯堡七桥问题，如图1.2所示。

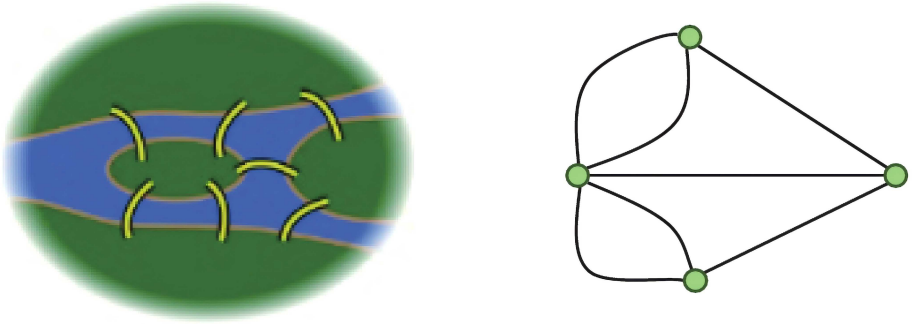

图 1.2 哥尼斯堡七桥问题

例 1.3 某视频网站积累了约 48 万用户对 1.7 万余部电影的上亿条评级数据，用户对电影的评级采用 1 星至 5 星的评分方式，如表1.1所示。基于现有的评级数据，需对用户未评分的电影 (用 “?” 表示) 进行评级预测，从而优化电影推荐系统的效果，这就是著名的 Netflix 百万美金竞赛。

表 1.1 电影评分表

用户	电影 1	电影 2	电影 3	电影 4	电影 5	···	电影 n
A_1	1	4	5	?	2	···	2
A_2	4	2	?	?	4	···	5
A_3	3	?	3	3	4	···	2
⋮	⋮	⋮	⋮	⋮	⋮		⋮
A_m	4	1	3	5	?	···	?

运筹学问题种类繁多，受课时安排的限制，本书仅选取几类典型的数学模型展开介绍，旨在为读者构建运筹学的基础认知框架。

(1) 线性规划 (Linear Programming, LP)：目标函数与约束条件均为线性表达式，且决策变量取连续值，称为线性规划。由于线性规划建模思路简洁直观且具备成熟的求解器，已成为运筹学中应用最广泛、最基础的模型之一。此外，诸多复杂的非线性规划问题，通过适当的变形与近似，也可转化为线性规划问题进行求解。

(2) 整数规划 (Integer Programming, IP)：当决策变量的取值必须满足整数约束时，相应的规划问题称为整数规划。根据目标函数与约束条件是否为线性表达式，整数

规划可划分为线性整数规划与非线性整数规划。进一步, 若线性整数规划中决策变量仅能取 0 或 1 两个值, 则称为 0–1 线性整数规划。

(3) 非线性规划 (Nonlinear Programming, NLP): 当目标函数或至少一个约束条件为非线性表达式时, 称为非线性规划。相较于线性规划, 非线性规划更贴合实际复杂系统的需求, 能够描述决策变量间的复杂关系, 广泛应用于生产制造、资源优化、物流与供应链管理、军事与国防等领域。

(4) 动态规划 (Dynamic Programming, DP): 动态规划是用于求解多阶段决策过程最优化问题的重要方法。这类问题由若干相互联系的阶段组成, 各阶段依次进行决策, 且前一阶段的状态会影响后一阶段的决策。动态规划以系统整体最优为目标, 寻求使目标函数达到最优的决策序列。

(5) 博弈论 (Game Theory, GT): 又称对策论, 主要研究多个理性决策主体在对抗、竞争或合作等交互局势中的数学模型。各决策主体在同一局势中, 因信息掌握程度不同、利益诉求存在冲突或协同 (如零和博弈、非零和博弈), 需自主选择各自策略以实现自身利益的最大化。

运筹学在实际场景应用中, 应根据问题的具体描述和特点, 合理进行模型构建和算法设计。运筹学的一般研究步骤如图1.3所示, 相关细节将在后续各章节中逐步展开。

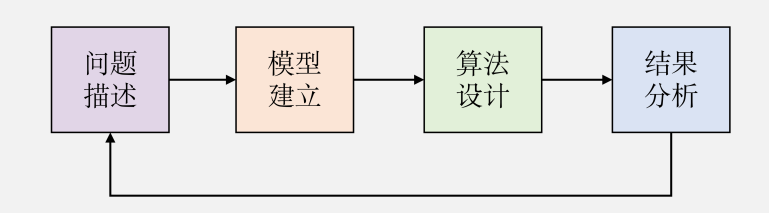

图 1.3　运筹学研究步骤

2

线性规划

2.1 基本概念

2.1.1 线性规划问题

为了帮助读者掌握线性规划问题的建模思路, 下面通过两个简单的实例进行讲解。

例 2.1 某公司计划生产 I、II 两种家电产品, 生产过程中分别占用设备 A、设备 B 的台时及调试工序时间 (单位: 小时)、各设备及工序的每日可用能力 (单位: 小时)、两种产品每件的获利 (单位: 元) 情况如表2.1所示。试确定该公司应制造两种家电各多少件, 使公司获得的日利润最大。

表 2.1 制造两种家电产品所需信息

项目	产品型号		每日可用能力
	I	II	
A	0	5	15
B	6	2	24
调试工序	1	1	5
利润	2	1	

解 设产品 I 的日产量为 x_1 件, 产品 II 的日产量为 x_2 件, 且 x_1, x_2 均为非负实数。公司的日利润为两种产品单件利润与产量的乘积之和, 即 $2x_1 + x_2$。令 $z = 2x_1 + x_2$, 为使公司获得的日利润最大, 因此目标函数为 $\max\ z = 2x_1 + x_2$。根据各设备及工序的每日可用能

力, 建立资源约束, 最终数学模型可表示为

$$\max \ z = 2x_1 + x_2$$

$$\text{s.t.} \begin{cases} 5x_2 \le 15 \\ 6x_1 + 2x_2 \le 24 \\ x_1 + x_2 \le 5 \\ x_1, x_2 \ge 0 \end{cases}$$

例 2.2 某公司拟在下一年度的 1 月至 4 月租用工厂堆放物资, 各月份所需工厂面积 (单位: 平方米) 见表2.2。工厂租借费用 (单位: 元 / 平方米) 随合同期限而定, 合同期越长折扣越大, 具体见表2.3。租借合同可在每月初办理, 每次办理时可签一份合同, 也可签若干份合同。试确定该公司的最优方案, 使所付总租借费用最小。

表 2.2 各月份所需工厂面积

月份	1	2	3	4
所需工厂面积	15	10	20	12

表 2.3 工厂租借费用

合同租借期限	1 个月	2 个月	3 个月	4 个月
合同期内的租费	2 800	4 500	6 000	7 300

解 设 x_{ij} 表示在第 i 个月初签订、租借期为 j 个月的工厂面积, 其中 $i, j = 1, 2, 3, 4$。因 5 月份起该公司不需要租借工厂, 故 x_{24}, x_{33}, x_{34}, x_{42}, x_{43}, x_{44} 均为 0。总租借费用为各类型合同的租费与对应租用面积的乘积之和, 目标为最小化总费用, 同时还需要满足各月份的工厂面积需求。于是, 数学模型可表示为

$$\min \ z = 2\,800(x_{11} + x_{21} + x_{31} + x_{41}) + 4\,500(x_{12} + x_{22} + x_{32}) + 6\,000(x_{13} + x_{23}) + 7\,300x_{14}$$

$$\text{s.t.} \begin{cases} x_{11} + x_{12} + x_{13} + x_{14} \ge 15 \\ x_{12} + x_{13} + x_{14} + x_{21} + x_{22} + x_{23} \ge 10 \\ x_{13} + x_{14} + x_{22} + x_{23} + x_{31} + x_{32} \ge 20 \\ x_{14} + x_{23} + x_{32} + x_{41} \ge 12 \\ x_{ij} \ge 0 \end{cases}$$

2.1.2 数学模型

由上述两个例子可以归纳得出, 线性规划问题的数学模型包含三个基本要素: 决策变量、目标函数与约束条件。其一般形式可表示为

$$\begin{aligned} &\max(\min)\ \ z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ &\text{s.t.}\ \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le (=, \ge) b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le (=, \ge) b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le (=, \ge) b_m \\ x_1, x_2, \cdots, x_n \ge 0 \end{cases} \end{aligned} \tag{2.1}$$

其中 $x_1, x_2, \cdots, x_n$ 为决策变量, 取值是连续的。z 为目标函数, 是决策变量的线性函数, 根据实际问题取最大化 max 或最小化 min。s.t. 是 subject to 的缩写, 表示约束于。约束条件由决策变量的线性等式或不等式构成。此外, $c_1, c_2, \cdots, c_n$ 称为价值系数或费用系数, $b_1, b_2, \cdots, b_m$ 称为资源量或右端项, $a_{11}, a_{12}, \cdots, a_{mn}$ 称为技术系数或约束系数。

线性规划问题(2.1)可整理为

$$\begin{aligned} &\max(\min)\ \ z = \sum_{j=1}^{n} c_j x_j \\ &\text{s.t.}\ \begin{cases} \sum\limits_{j=1}^{n} a_{ij}x_j \le (=, \ge) b_i,\ i = 1, 2, \cdots, m \\ x_j \ge 0,\ j = 1, 2, \cdots, n \end{cases} \end{aligned}$$

本书中所有向量均默认为列向量, 记

$$\boldsymbol{c} = (c_1, c_2, \cdots, c_n)^{\mathrm{T}},\ \boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}},\ \boldsymbol{b} = (b_1, b_2, \cdots, b_m)^{\mathrm{T}}$$

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n)$$

则线性规划模型可表示为如下形式

$$\begin{aligned} &\max(\min)\ \ z = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{s.t.}\ \begin{cases} \sum\limits_{j=1}^{n} \boldsymbol{p}_j x_j \le (=, \ge) \boldsymbol{b} \\ \boldsymbol{x} \ge 0 \end{cases} \end{aligned}$$

或写成更为简洁的矩阵形式

$$\max(\min)\ \ z = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$$
$$\text{s.t.}\ \begin{cases} \boldsymbol{A}\boldsymbol{x} \leq (=, \geq)\boldsymbol{b} \\ \boldsymbol{x} \geq 0 \end{cases}$$

定义 2.1

线性规划问题的标准形式为

$$\max\ \ z = \sum_{j=1}^{n} c_j x_j \tag{2.2a}$$

$$\text{s.t.}\ \begin{cases} \sum_{j=1}^{n} a_{ij} x_j = b_i,\ i = 1, 2, \cdots, m & \text{(2.2b)} \\ x_j \geq 0,\ j = 1, 2, \cdots, n & \text{(2.2c)} \end{cases}$$

标准形式的线性规划模型具有以下四个特征: 目标函数为最大化形式、所有约束条件均为等式约束、所有决策变量满足非负约束、约束条件右端常数项均非负。温馨提醒, 部分教材采用以最小化为目标的标准形式。为便于后续描述线性规划解的相关概念与性质, 给出如下定义。

定义 2.2

满足所有约束条件(2.2b)和(2.2c)的解称为可行解, 所有可行解构成的集合称为可行域。在可行域中使目标函数达到最大的可行解称为最优解, 记为 $\boldsymbol{x}^*$。最优解对应的目标函数值称为最优值, 记为 z^*。

2.1.3 非标准型转化为标准形式

对于非标准形式的线性规划问题, 通常可通过若干等价变换将其转化为标准形式。这个过程主要包括目标函数、约束条件以及决策变量三方面的转换。

(1) 目标函数的转换: 若原问题为最小化问题

$$\min\ \ z = \sum_{j=1}^{n} c_j x_j$$

可通过引入新的目标函数 $z' = -z$, 将其等价地转化为最大化问题

$$\max\ \ z' = -\sum_{j=1}^{n} c_j x_j$$

需要注意的是, 虽然上述两个问题具有相同的最优解, 但其最优值相差一个符号, 即 $\min\ z = -\max\ z'$。

(2) 约束条件的转换: 在标准形式中, 约束条件的右端项要求为非负数。若某约束条件的右端项 $b_i < 0$, 只需将该约束两端同时乘以 -1, 得到等价约束

$$-\sum_{j=1}^{n} a_{ij}x_j = -b_i$$

同时, 标准形式还要求约束条件为等式。若约束条件为“≤”型不等式, 则可引入松弛变量 $s \geq 0$, 使其等于约束右端与左端之差, 从而将不等式转化为

$$\sum_{j=1}^{n} a_{ij}x_j + s = b_i,\ s \geq 0$$

若约束条件为“≥”型不等式, 则可引入剩余变量 $s \geq 0$, 将不等式转化为

$$\sum_{j=1}^{n} a_{ij}x_j - s = b_i,\ s \geq 0$$

(3) 决策变量的转换: 若某变量 x_k 在原问题中无约束 (自由变量), 可将其表示为两个非负变量之差

$$x_k = x_k' - x_k'',\ x_k', x_k'' \geq 0$$

此时, 变量 x_k 的取值符号由 x_k' 与 x_k'' 的相对大小决定。另一方面, 若某变量 $x_k \leq 0$, 则可令

$$x_k' = -x_k$$

从而所有决策变量均满足标准形式中非负的要求。

例 2.3 请将以下线性规划问题转化为标准形式

$$\min\ z = x_1 + 3x_2 + 4x_3$$

$$\text{s.t.}\ \begin{cases} -2x_1 + x_2 + x_3 \leq 7 \\ -3x_1 + x_2 + 2x_3 \geq 3 \\ 4x_1 - 2x_2 - 3x_3 = -6 \\ x_1 \leq 0,\ x_2 \geq 0,\ x_3\text{无约束} \end{cases}$$

解 首先进行目标函数的转换。令 $z' = -z$, 则

$$\max\ z' = -x_1 - 3x_2 - 4x_3$$

$$\text{s.t.}\ \begin{cases} -2x_1 + x_2 + x_3 \le 7 \\ -3x_1 + x_2 + 2x_3 \ge 3 \\ 4x_1 - 2x_2 - 3x_3 = -6 \\ x_1 \le 0,\ x_2 \ge 0,\ x_3\text{无约束} \end{cases}$$

第三个约束条件的右端项存在负数, 两端同乘 -1 得到

$$\max\ z' = -x_1 - 3x_2 - 4x_3$$

$$\text{s.t.}\ \begin{cases} -2x_1 + x_2 + x_3 \le 7 \\ -3x_1 + x_2 + 2x_3 \ge 3 \\ -4x_1 + 2x_2 + 3x_3 = 6 \\ x_1 \le 0,\ x_2 \ge 0,\ x_3\text{无约束} \end{cases}$$

又因第一、二个约束条件存在 “$\le$” 和 “$\ge$” , 引入松弛变量 x_4、剩余变量 x_5, 将不等式转换为等式, 进而得到

$$\max\ z' = -x_1 - 3x_2 - 4x_3 + 0x_4 + 0x_5$$

$$\text{s.t.}\ \begin{cases} -2x_1 + x_2 + x_3 + x_4 = 7 \\ -3x_1 + x_2 + 2x_3 - x_5 = 3 \\ -4x_1 + 2x_2 + 3x_3 = 6 \\ x_1 \le 0,\ x_2, x_4, x_5 \ge 0,\ x_3\text{无约束} \end{cases}$$

注意到 x_3 取值无约束, 令 $x_3 = x_3' - x_3''$, $x_3', x_3'' \ge 0$, 同时 x_1 的约束条件是 $x_1 \le 0$, 令 $x_1' = -x_1$, 于是

$$\max\ z' = x_1' - 3x_2 - 4x_3' + 4x_3'' + 0x_4 + 0x_5$$

$$\text{s.t.}\ \begin{cases} 2x_1' + x_2 + x_3' - x_3'' + x_4 = 7 \\ 3x_1' + x_2 + 2x_3' - 2x_3'' - x_5 = 3 \\ 4x_1' + 2x_2 + 3x_3' - 3x_3'' = 6 \\ x_1', x_2, x_3', x_3'', x_4, x_5 \ge 0 \end{cases}$$

进行简单的变量代换, 最终得到原线性规划问题的标准形式

$$\max \ z = x_1 - 3x_2 - 4x_3 + 4x_4$$
$$\text{s.t.} \begin{cases} 2x_1 + x_2 + x_3 - x_4 + x_5 = 7 \\ 3x_1 + x_2 + 2x_3 - 2x_4 - x_6 = 3 \\ 4x_1 + 2x_2 + 3x_3 - 3x_4 = 6 \\ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{cases}$$

2.2 图解法

当线性规划问题(2.1)仅含两个决策变量时, 其最大化形式的数学模型可简化为

$$\max \ z = c_1 x_1 + c_2 x_2$$
$$\text{s.t.} \begin{cases} a_{i1} x_1 + a_{i2} x_2 \leq (=, \geq) b_i, \ i = 1, 2, \cdots, m \\ x_1, x_2 \geq 0 \end{cases}$$

由于可行域能够在平面直角坐标系中直观呈现, 因此采用图解法进行求解。图解法的基本思路是: 首先建立以 x_1 和 x_2 为坐标轴的平面直角坐标系, 将各约束条件表示为相应的直线或半平面, 从而确定问题的可行域。然后绘制目标函数的等值线, 并沿其改进方向平移。最后在可行域的顶点处比较目标函数值, 确定问题的最优解。

2.2.1 求解过程

例 2.4 用图解法求解下列线性规划问题

$$\max \ z = 2x_1 + 3x_2$$
$$\text{s.t.} \begin{cases} x_1 + 2x_2 \leq 8 \\ 4x_1 \leq 16 \\ 4x_2 \leq 12 \\ x_1, x_2 \geq 0 \end{cases}$$

解 以 x_1, x_2 为坐标轴建立直角坐标系, 非负条件 $x_1, x_2 \geq 0$ 指第一象限。每个不等式约束对应平面上的一个半平面, 可行域为所有约束半平面的交集, 如图2.1中的阴影部分。

阴影区域中的每一个点都是这个线性规划问题的可行解, 因而此区域是线性规划问题的可行域。

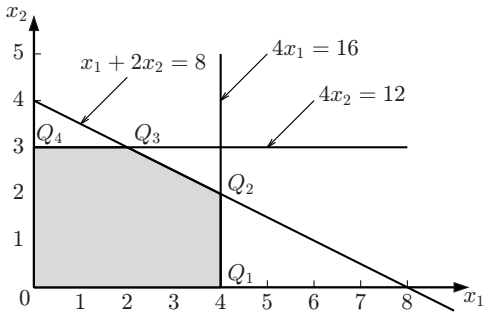

图 2.1 可行域

如图2.2所示, 目标函数可以表示为以 z 为参数、$-\frac{2}{3}$ 为斜率的一族等值线, 即

$$x_2 = -\frac{2}{3}x_1 + \frac{z}{3}$$

当 z 值由小变大时, 该直线沿法线方向逐渐向右上方移动。当移动到 Q_2 点时, z 值在可行域内实现最大化。因此, 该问题的最优解为 $Q_2(4,2)$, 最优值为 $z = 14$。

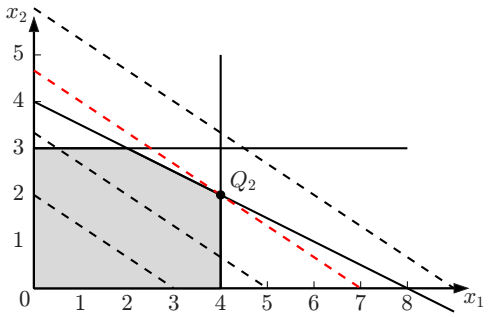

图 2.2 目标函数变化

2.2.2 线性规划问题几种可能解

由例2.4可知, 该问题存在唯一最优解。但对于一般线性规划问题, 除唯一最优解外, 还可能出现无穷多最优解、无界解以及无可行解等情况。

(1) 无穷多最优解: 当目标函数的等值线在最优位置与可行域的某条边界重合或平行时, 沿该边的任意点都能取得相同的最优目标函数值, 从而产生无穷多最优解。以例2.4为例, 若将目标函数改为 $\max\ z = 2x_1 + 4x_2$, 则目标函数的等值线与约束 $x_1 + 2x_2 \le 8$ 对应的边平行。当 z 由小增大时, 等值线最终与线段 Q_2Q_3 重合。因此, 线段 Q_2Q_3 上任意一点均为最优解, 如图2.3所示。

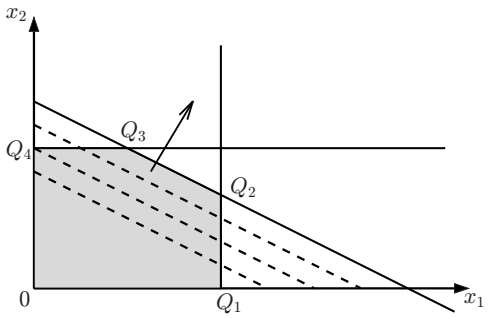

图 2.3 无穷多最优解

(2) 无界解: 若可行域在目标函数的改进方向上无界延伸, 此时目标函数值可以无限增大或无限减小, 则称该问题具有无界解。考察如下线性规划模型

$$\max \ z = x_1 + x_2$$
$$\text{s.t.} \begin{cases} -2x_1 + x_2 \le 4 \\ x_1 - x_2 \le 2 \\ x_1, x_2 \ge 0 \end{cases}$$

由图2.4可以看出, 可行域在某一方向上不受约束限制, 目标函数 $z = x_1 + x_2$ 可无限增大。这表明该问题虽然存在可行解, 但不存在有限的最优解。

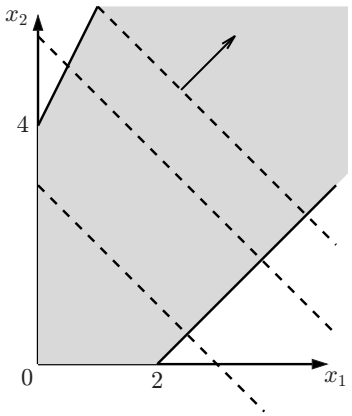

图 2.4 无界解

(3) 无可行解: 当约束条件之间相互矛盾、不存在任何点能同时满足所有约束时, 可行域为空集, 称问题无可行解, 此时自然也不存在最优解。以例2.4为例, 若增加约束 $x_1 + x_2 \ge 10$, 则该约束与原有约束 $x_1 + 2x_2 \le 8$ 相互冲突, 将导致可行域为空, 从而该问题无可行解。

2.3 解与凸集

2.3.1 基解

考虑标准形式的线性规划问题(2.2), 设 $\boldsymbol{A}$ 为约束方程组的 $m \times n$ $(n > m)$ 阶系数矩阵, 其秩为 m。记

$$\boldsymbol{B} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{pmatrix} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_m)$$

定义 2.3

若 $\boldsymbol{B}$ 是矩阵 $\boldsymbol{A}$ 中的一个 $m \times m$ 阶满秩子矩阵, 则称 $\boldsymbol{B}$ 为该线性规划问题的一个基。矩阵 $\boldsymbol{B}$ 中的每一个列向量 $\boldsymbol{p}_j$ 称为基向量。与基向量 $\boldsymbol{p}_j$ 对应的变量 x_j 称为基变量, 记为 $\boldsymbol{x}_B = (x_1, x_2, \cdots, x_m)^{\mathrm{T}}$。除基变量以外的其余变量称为非基变量, 记为 $\boldsymbol{x}_N = (x_{m+1}, x_{m+2}, \cdots, x_n)^{\mathrm{T}}$。

例 2.5 找出线性规划问题的基、基向量和基变量

$$\max \ z = 70x_1 + 120x_2$$

$$\text{s.t.} \begin{cases} 9x_1 + 4x_2 + x_3 = 360 \\ 4x_1 + 5x_2 + x_4 = 200 \\ 3x_1 + 10x_2 + x_5 = 300 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

解 写出约束方程组的系数矩阵

$$\boldsymbol{A} = \begin{pmatrix} 9 & 4 & 1 & 0 & 0 \\ 4 & 5 & 0 & 1 & 0 \\ 3 & 10 & 0 & 0 & 1 \end{pmatrix}$$

寻找阶为 3 的满秩子矩阵, 得到该线性规划问题的一个基

$$\boldsymbol{B} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (\boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5)$$

对应的基变量为 $\boldsymbol{x}_B = (x_3, x_4, x_5)^{\mathrm{T}}$, 非基变量为 $\boldsymbol{x}_N = (x_1, x_2)^{\mathrm{T}}$。另一个基为

$$\boldsymbol{B}' = \begin{pmatrix} 4 & 0 & 0 \\ 5 & 1 & 0 \\ 10 & 0 & 1 \end{pmatrix} = (\boldsymbol{p}_2, \boldsymbol{p}_4, \boldsymbol{p}_5)$$

对应的基变量为 $\boldsymbol{x}_B = (x_2, x_4, x_5)^{\mathrm{T}}$, 非基变量为 $\boldsymbol{x}_N = (x_1, x_3)^{\mathrm{T}}$。

定义 2.4

在约束条件(2.2b)中, 令所有非基变量 $x_{m+1}, x_{m+2}, \cdots, x_n$ 取值为 0, 则称

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_m, 0, \cdots, 0)^{\mathrm{T}}$$

为线性规划问题的基解。进一步, 满足变量非负约束条件(2.2c)的基解称为基可行解, 与之对应的基称为可行基。

约束方程组(2.2b)的基解数目最多为 C_n^m, 而基可行解的数目通常少于基解的数目。上述各类解之间的关系可见图2.5。此外, 若某一基解的非零分量个数小于 m, 则称该基解为退化解。在后续讨论中, 均设线性规划问题不出现退化情形。

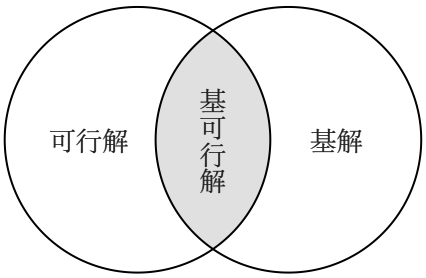

图 2.5 可行解、基解与基可行解

例 2.6 求出全部基解, 指出其中的基可行解, 并确定最优解

$$\max \ z = 2x_1 + 3x_2 + x_3$$

$$\text{s.t.} \begin{cases} x_1 + x_3 = 5 \\ x_1 + 2x_2 + x_4 = 10 \\ x_2 + x_5 = 4 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

解 写出约束方程组的系数矩阵

$$\boldsymbol{A}=\begin{pmatrix}1&0&1&0&0\\1&2&0&1&0\\0&1&0&0&1\end{pmatrix}$$

根据表2.4可知, 最优解为 $\boldsymbol{x}^*=(2,4,3,0,0)^{\mathrm{T}}$, 最优值为 $z^*=19$。

表 2.4 全部基解

序号	x_1	x_2	x_3	x_4	x_5	z	可行解
①	0	0	5	10	4	5	✓
②	0	4	5	2	0	17	✓
③	5	0	0	5	4	10	✓
④	0	5	5	0	−1	20	×
⑤	10	0	−5	0	4	15	×
⑥	5	5/2	0	0	1.5	35/2	✓
⑦	5	4	0	−3	0	22	×
⑧	2	4	3	0	0	19	✓

2.3.2 凸集

定义 2.5

设集合 $\Omega\subseteq\mathbb{R}^n$, 若对任意两点 $\boldsymbol{x}_1,\boldsymbol{x}_2\in\Omega$ 及实数 $\alpha\in(0,1)$, 都满足

$$\alpha\boldsymbol{x}_1+(1-\alpha)\boldsymbol{x}_2\in\Omega$$

则称集合 Ω 为凸集。

由定义知, 凸集中任意两点连成的线段必属于这个集合。若集合不满足凸集, 则称为非凸集。如图2.6所示, (a) 与 (b) 为凸集, (c) 与 (d) 为非凸集。

定义 2.6

对于凸集 Ω 中的点 $\boldsymbol{x}$, 若不存在两个不同的点 $\boldsymbol{x}_1,\boldsymbol{x}_2\in\Omega$ 及实数 $\alpha\in(0,1)$, 使得

$$\boldsymbol{x}=\alpha\boldsymbol{x}_1+(1-\alpha)\boldsymbol{x}_2\in\Omega$$

则称 $\boldsymbol{x}$ 为凸集 Ω 的顶点 (极点)。

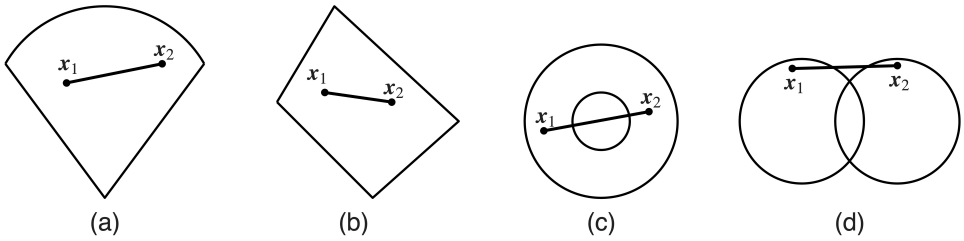

图 2.6 凸集与非凸集

2.3.3 基本定理

定理 2.7

若线性规划问题存在可行解，则其可行域为凸集。

证明 记 Ω 为满足线性规划问题约束条件的集合

$$\sum_{j=1}^{n} \boldsymbol{p}_j x_j = \boldsymbol{b},\ x_j \geq 0,\ j = 1, 2, \cdots, n$$

任取 Ω 中两点

$$\boldsymbol{x}_1 = (x_{11}, x_{12}, \cdots, x_{1n})^{\mathrm{T}},\ \boldsymbol{x}_2 = (x_{21}, x_{22}, \cdots, x_{2n})^{\mathrm{T}}$$

且 $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$，一定满足

$$\sum_{j=1}^{n} \boldsymbol{p}_j x_{1j} = \boldsymbol{b},\ x_{1j} \geq 0,\ j = 1, 2, \cdots, n$$

$$\sum_{j=1}^{n} \boldsymbol{p}_j x_{2j} = \boldsymbol{b},\ x_{2j} \geq 0,\ j = 1, 2, \cdots, n$$

设 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}$ 为 $\boldsymbol{x}_1$ 与 $\boldsymbol{x}_2$ 连线上任意一点及实数 $\alpha \in (0, 1)$，有

$$\boldsymbol{x} = \alpha \boldsymbol{x}_1 + (1 - \alpha)\boldsymbol{x}_2$$

其中 $x_j = \alpha x_{1j} + (1 - \alpha)x_{2j}$。代入约束条件得到

$$\begin{aligned}
\sum_{j=1}^{n} \boldsymbol{p}_j x_j &= \sum_{j=1}^{n} \boldsymbol{p}_j \left(\alpha x_{1j} + (1 - \alpha)x_{2j}\right) \\
&= \alpha \sum_{j=1}^{n} \boldsymbol{p}_j x_{1j} + \sum_{j=1}^{n} \boldsymbol{p}_j x_{2j} - \alpha \sum_{j=1}^{n} \boldsymbol{p}_j x_{2j} \\
&= \alpha \boldsymbol{b} + \boldsymbol{b} - \alpha \boldsymbol{b} \\
&= \boldsymbol{b}
\end{aligned}$$

由于 $x_{1j} \geq 0, x_{2j} \geq 0, \alpha > 0, 1-\alpha > 0$, 可知

$$x_j \geq 0, \ j = 1, 2, \cdots, n$$

因此, 集合 Ω 中任意两点连线上的点仍属于 Ω, 即 Ω 是凸集。

定理 2.8

线性规划问题的可行解 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}$ 为基可行解的充分必要条件是, $\boldsymbol{x}$ 的正分量所对应的系数列向量线性无关。

证明 由基可行解的定义直接可得必要性, 下面证明充分性。若向量 $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k$ 线性无关, 则必有 $k \leq m$。当 $k = m$ 时, 向量 $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k$ 恰好构成一个基, 从而

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_m, 0, \cdots, 0)^{\mathrm{T}}$$

为相应的基可行解。当 $k < m$ 时, 则一定可以从其余的列向量中选出 $m-k$ 个, 与向量 $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k$ 构成线性无关向量组, 其对应的解恰为 $\boldsymbol{x}$。综上, 充分性成立。

定理 2.9

线性规划问题的基可行解 $\boldsymbol{x}$ 对应可行域 (凸集) 的顶点。

定理定理2.9称为线性规划的基本定理, 其重要性在于将可行域的顶点与基可行解建立起严格的对应关系, 从而可以通过代数方法求解基可行解。

定理 2.10

若线性规划问题的可行域有界, 则其目标函数必在可行域 (凸集) 的某个顶点上达到最优。

受定理定理2.10启发, 线性规划问题的基本求解思路可概括为以下几点。先找到可行域的任意一个顶点, 计算该顶点处的目标函数值。再比较相邻顶点的目标函数值, 若不存在函数值更优的相邻顶点, 则该顶点即为最优解或最优解之一, 否则转向函数值更优的相邻顶点。重复上述迭代与比较过程, 直至找到使目标函数达到最优的顶点。

2.4 单纯形法

尽管线性规划问题理论上可通过枚举法求解, 但当决策变量数 n 和约束条件数 m 较大时, 计算量会呈指数级激增, 实际应用中根本无法实现。

2.4.1 计算步骤

为解决上述困境, 单纯形法应运而生。首先构造一个初始基可行解, 对其进行最优性检验。若不是最优解, 则按规则选择入基与出基变量, 通过基变换将当前基可行解转换为相邻的基可行解, 并使目标函数值不断改进。反复迭代, 直至得到最优解。

(1) 建立初始单纯形表: 为规范计算流程, 引入单纯形表。考虑线性规划问题(2.2)的约束方程组

$$\begin{cases} x_1 + a_{1,m+1}x_{m+1} + \cdots + a_{1,n}x_n = b_1 \\ x_2 + a_{2,m+1}x_{m+1} + \cdots + a_{2,n}x_n = b_2 \\ \cdots \\ x_m + a_{m,m+1}x_{m+1} + \cdots + a_{m,n}x_n = b_m \end{cases}$$

其中 $x_1, x_2, \cdots, x_m$ 为基变量, 其余为非基变量。记系数矩阵为

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_{1,m+1} & \cdots & a_{1,n} \\ 0 & 1 & \cdots & 0 & a_{2,m+1} & \cdots & a_{2,n} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & a_{m,m+1} & \cdots & a_{m,n} \end{pmatrix}$$

基于此, 可列出初始单纯形表2.5。表中 $c_j - z_j$ 为检验数, 定义为

$$\sigma_j = c_j - z_j = c_j - \sum_{i=1}^{m} c_i a_{ij}$$

同时, 选取一个 $m \times m$ 单位矩阵对应的列作为初始可行基。

表 2.5 初始单纯形表

$c_j \rightarrow$			c_1	c_2	$\cdots$	c_m	$\cdots$	c_j	$\cdots$	c_n
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	$\boldsymbol{b}$	x_1	x_2	$\cdots$	x_m	$\cdots$	x_j	$\cdots$	x_n
c_1	x_1	b_1	1	0	$\cdots$	0	$\cdots$	a_{1j}	$\cdots$	a_{1n}
c_2	x_2	b_2	0	1	$\cdots$	0	$\cdots$	a_{2j}	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
c_m	x_m	b_m	0	0	$\cdots$	1	$\cdots$	a_{mj}	$\cdots$	a_{mn}
$c_j - z_j$			0	0	$\cdots$	0	$\cdots$	σ_j	$\cdots$	σ_n

(2) 最优性检验: 若所有检验数均满足

$$\sigma_j \leq 0$$

且不含人工变量 (详见下节), 则当前基可行解即为最优解, 迭代终止。若存在

$$\sigma_j > 0$$

且对应列向量 $\boldsymbol{p}_j \leq \mathbf{0}$ (即列中所有元素均非正), 则目标函数可沿该方向无限增大, 问题无界。否则, 转入下一步开展基变换迭代。

(3) 基可行解的转换: 当需要继续改进目标函数值时, 可将当前基可行解转换为相邻的、目标函数值更大的基可行解。首先按如下最大检验数原则

$$\sigma_k = \max_j \{\sigma_j \mid \sigma_j > 0\}$$

从所有检验数为正的非基变量中, 选取检验数最大的变量 x_k 作为换入变量, 标记为 $\underline{x_k}$。随后按如下最小比值原则

$$\theta = \min_i \left\{ \frac{b_i}{a_{ik}} \,\middle|\, a_{ik} > 0 \right\} = \frac{b_l}{a_{lk}}$$

确定主元素 a_{lk}, 标记为 $[a_{lk}]$, 并据此选取 x_l 为换出变量, 标记为 $\underline{x_l}$。随后对单纯形表实施初等行变换, 将主元素化为 1, 同时把主元素所在列的其余元素全部化为 0, 从而得到更新后的单纯形表与新的基可行解。

(4) 重复迭代直至结束: 重复执行上述两个步骤, 直至满足最优性停止条件, 或判定为无界、无可行解。

例 2.7 用单纯形法求解下列线性规划问题

$$\max \ z = 2x_1 + x_2$$
$$\text{s.t.} \begin{cases} 5x_2 \leq 15 \\ 6x_1 + 2x_2 \leq 24 \\ x_1 + x_2 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

解 首先将原问题转化为标准形式, 对“≤”型约束分别引入松弛变量 $x_3, x_4, x_5 \geq 0$ 得到

$$\max \ z = 2x_1 + x_2$$
$$\text{s.t.} \begin{cases} 5x_2 + x_3 = 15 \\ 6x_1 + 2x_2 + x_4 = 24 \\ x_1 + x_2 + x_5 = 5 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

将上式约束条件直接写成矩阵方程 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$, 其中

$$\boldsymbol{A}=\begin{pmatrix}0 & 5 & 1 & 0 & 0\\ 6 & 2 & 0 & 1 & 0\\ 1 & 1 & 0 & 0 & 1\end{pmatrix},\ \boldsymbol{x}=(x_1,x_2,x_3,x_4,x_5)^{\mathrm{T}}$$

据此构造初始单纯形表2.6。

表 2.6 初始单纯形表

$c_j\rightarrow$			2	1	0	0	0
$\boldsymbol{c}_B$	$\boldsymbol{x}_B$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5
0	x_3	15	0	5	1	0	0
0	x_4	24	6	2	0	1	0
0	x_5	5	1	1	0	0	1
c_j-z_j			2	1	0	0	0

当前检验数中存在 $\sigma_j>0$, 说明初始基可行解不是最优解, 需要进行单纯形迭代。根据最大检验数原则, 由于 $\sigma_1>\sigma_2$, 选取 x_1 作为换入变量。然后计算最小比值

$$\theta=\min\left\{\infty,\frac{24}{6},\frac{5}{1}\right\}=4$$

从而确定主元素为 6, 换出变量为 x_4。以主元素为中心进行初等行变换, 得到第一次迭代后的单纯形表2.7。

表 2.7 第一次迭代后单纯形表

$c_j\rightarrow$			2	1	0	0	0
$\boldsymbol{c}_B$	$\boldsymbol{x}_B$	$\boldsymbol{b}$	x_1	$\underline{x_2}$	x_3	x_4	x_5
0	x_3	15	0	5	1	0	0
2	x_1	4	1	2/6	0	1/6	0
0	$\underline{x_5}$	1	0	[4/6]	0	−1/6	1
c_j-z_j			0	1/3	0	−1/3	0

仍有检验数 $\sigma_2>0$, 故选取 x_2 为换入变量。计算最小比值

$$\theta=\min\left\{\frac{15}{5},\frac{4}{2/6},\frac{1}{4/6}\right\}=\frac{6}{4}$$

从而确定主元为 4/6, 换出变量为 x_5。再次进行初等行变换, 得到单纯形表2.8。

此时, 所有检验数均满足 $\sigma_j\le 0$, 迭代终止。得到最优解为 $\boldsymbol{x}^*=(7/2,3/2,15/2,0,0)^{\mathrm{T}}$, 最优值为 $z^*=2x_1+x_2=17/2$。

表 2.8 第二次迭代后单纯形表

$c_j \rightarrow$			2	1	0	0	0
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5
0	x_3	15/2	0	0	1	5/4	−15/2
2	x_1	7/2	1	0	0	1/4	−1/2
1	x_2	3/2	0	1	0	−1/4	3/2
$c_j - z_j$			0	0	0	−1/4	−1/2

2.4.2 大 M 法

若线性规划问题无法直接找到初始基可行解，应当如何处理？常用的方法为大 M 法和两阶段法。其中，大 M 法又称为惩罚法，通过引入人工变量，并取 M 为一个充分大的正数，在原问题的目标函数中添加 $-M$ 与每个人工变量的乘积项。当目标函数为最大化时，只有将所有人工变量从基变量中换出，原问题才有可能达到最优解。

例 2.8 用大 M 法求解下列线性规划问题

$$\max \ z = -3x_1 + x_3$$
$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 \le 4 \\ -2x_1 + x_2 - x_3 \ge 1 \\ 3x_2 + x_3 = 9 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

解 首先将原问题转化为标准形式

$$\max \ z = -3x_1 + x_3 + 0x_4 + 0x_5$$
$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 = 4 \\ -2x_1 + x_2 - x_3 - x_5 = 1 \\ 3x_2 + x_3 = 9 \\ x_1, x_2, x_3, x_4, x_5 \ge 0 \end{cases}$$

观察约束系数矩阵可知，不存在现成的单位矩阵作为初始基矩阵，无法直接构造初始基可行解。因此，采用大 M 法，对相应约束条件引入人工变量 $x_6, x_7 \ge 0$，并在目标函数中给

予其足够大的惩罚系数 M, 构造如下辅助线性规划问题

$$\max \ z = -3x_1 + x_3 + 0x_4 + 0x_5 - Mx_6 - Mx_7$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 = 4 \\ -2x_1 + x_2 - x_3 - x_5 + x_6 = 1 \\ 3x_2 + x_3 + x_7 = 9 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{cases}$$

以 x_4, x_6, x_7 为初始基变量, 构造初始单纯形表, 并按单纯形法迭代求解, 具体过程详见表2.9。

表 2.9 大 M 法迭代过程

$c_j \rightarrow$			-3	0	1	0	0	$-M$	$-M$
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	4	1	1	1	1	0	0	0
$-M$	x_6	1	-2	$[1]$	-1	0	-1	1	0
$-M$	x_7	9	0	3	1	0	0	0	1
$c_j - z_j$			$-3-2M$	$4M$	1	0	$-M$	0	0
0	x_4	3	3	0	2	1	1	-1	0
0	x_2	1	-2	1	-1	0	-1	1	0
$-M$	x_7	6	$[6]$	0	4	0	3	-3	1
$c_j - z_j$			$-3+6M$	0	$1+4M$	0	$3M$	$-4M$	0
0	x_4	0	0	0	0	1	$-1/2$	$-1/2$	$1/2$
0	x_2	3	0	1	$1/3$	0	0	0	$1/3$
-3	x_1	1	1	0	$[2/3]$	0	$1/2$	$-1/2$	$1/6$
$c_j - z_j$			0	0	3	0	$3/2$	$-3/2-M$	$1/2-M$
0	x_4	0	0	0	0	1	$-1/2$	$1/2$	$-1/2$
0	x_2	$5/2$	$-1/2$	1	0	0	$-1/4$	$1/4$	$1/4$
1	x_3	$3/2$	$3/2$	0	1	0	$3/4$	$-3/4$	$1/4$
$c_j - z_j$			$-9/2$	0	0	0	$-3/4$	$3/4-M$	$-1/4-M$

由单纯形表可得, 线性规划问题的最优解为 $\boldsymbol{x}^* = (0, 5/2, 3/2, 0, 0, 0, 0)^{\mathrm{T}}$, 最优值为 $z^* = 3/2$。因所有人工变量 x_6, x_7 均为零, 于是原问题的最优解为 $\boldsymbol{x}^* = (0, 5/2, 3/2)^{\mathrm{T}}$, 最优值为 $z^* = 3/2$。

2.4.3 两阶段法

为克服大 M 法在计算机求解中因 M 取值带来的数值精度问题, 可将引入后的线性规划问题分两个阶段求解。构造辅助线性规划问题

$$\max \ w = -\sum_{i=1}^{m} y_i$$

$$\text{s.t.} \begin{cases} \sum\limits_{j=1}^{n} a_{ij}x_j + y_i = b_i, \ i = 1, 2, \cdots, m \\ x_j \geq 0, \ i = 1, 2, \cdots, m \\ y_i \geq 0, \ j = 1, 2, \cdots, n \end{cases}$$

基于此, 求解含人工变量线性规划的两阶段法步骤如下。

(1) 第一阶段: 采用单纯形法求解辅助问题。若人工变量取值为 0、目标函数值也为 0, 则此时的最优解对应原线性规划问题的一个基可行解。若最优目标函数值不为 0, 即最优解的基变量中仍含有非零人工变量, 表明原线性规划问题无可行解。

(2) 第二阶段: 在第一阶段已求得原问题的一个初始基可行解的基础上, 继续求解原问题的最优解。首先对第一阶段的最优单纯形表进行调整, 将检验数行替换为原问题的价值系数, 删去所有人工变量所在列, 再以单纯形法继续迭代计算。

例 2.9 用两阶段法求解下列线性规划问题

$$\max \ z = -3x_1 + x_3$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 \leq 4 \\ -2x_1 + x_2 - x_3 \geq 1 \\ 3x_2 + x_3 = 9 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

解 将原问题转化为带人工变量的线性规划问题

$$\max \ z = -3x_1 + x_3 + 0x_4 + 0x_5 - Mx_6 - Mx_7$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 = 4 \\ -2x_1 + x_2 - x_3 - x_5 + x_6 = 1 \\ 3x_2 + x_3 + x_7 = 9 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{cases}$$

以最大化 $w=-x_6-x_7$ 为目标, 构造辅助线性规划问题

$$\max \ w=-x_6-x_7$$

$$\text{s.t.} \begin{cases} x_1+x_2+x_3+x_4=4 \\ -2x_1+x_2-x_3-x_5+x_6=1 \\ 3x_2+x_3+x_7=9 \\ x_1,x_2,x_3,x_4,x_5,x_6,x_7\geq 0 \end{cases}$$

在相同约束条件下使用单纯形法迭代求解, 计算过程如表2.10所示。可以看出, 第一阶段最优值为 $w^*=0$, 得到一组基可行解

$$x_1=1,\ x_2=3,\ x_3=x_4=x_5=x_6=x_7=0$$

此时人工变量 $x_6=x_7=0$, 因此 $(x_1,x_2,x_3,x_4,x_5)^{\mathrm{T}}=(1,3,0,0,0)^{\mathrm{T}}$ 为原问题的一个基可行解, 可进入第二阶段计算。

表 2.10 第一阶段单纯形法迭代过程

$c_j\rightarrow$			0	0	0	0	0	−1	−1
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	4	1	1	1	1	0	0	0
−1	x_6	1	−2	[1]	−1	0	−1	1	0
−1	x_7	9	0	3	1	0	0	0	1
	c_j-z_j		−2	4	0	0	−1	0	0
0	x_4	3	3	0	2	1	1	−1	0
0	x_2	1	−2	1	−1	0	−1	1	0
−1	x_7	6	[6]	0	4	0	3	−3	1
	c_j-z_j		6	0	4	0	3	−4	0
0	x_4	0	0	0	0	1	−1/2	1/2	−1/2
0	x_2	3	0	1	1/3	0	0	0	1/3
0	x_1	1	1	0	2/3	0	1/2	−1/2	1/6
	c_j-z_j		0	0	0	0	0	−1	−1

删去第一阶段单纯形表2.10中人工变量 x_6,x_7 的所在列, 并将变量 x_1,x_2,x_3 对应的价

值系数 c_j 替换为原问题目标函数的系数。第二阶段对应的原问题标准形式为

$$\max \ z = -3x_1 + 0x_2 + x_3 + 0x_4 + 0x_5$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 = 4 \\ -2x_1 + x_2 - x_3 - x_5 = 1 \\ 3x_2 + x_3 = 9 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

计算过程如表2.11所示。所有检验数满足 $\sigma_j \leq 0$, 且存在非基变量检验数为 0, 故原线性规划问题存在无穷多最优解。

表 2.11 第二阶段单纯形法迭代过程

$c_j \rightarrow$			−3	0	1	0	0
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5
0	x_4	0	0	0	0	1	−1/2
0	x_2	3	0	1	1/3	0	0
−3	x_1	1	1	0	[2/3]	0	1/2
$c_j - z_j$			0	0	3	0	3/2
0	x_4	0	0	0	0	1	−1/2
0	x_2	5/2	−1/2	1	0	0	−1/4
1	x_3	3/2	3/2	0	1	0	3/4
$c_j - z_j$			−9/2	0	0	0	−3/4

2.4.4 其他问题

运用单纯形法求解线性规划问题, 可能遇到一些需要特别注意的情况, 如最小化问题、退化现象以及无可行解的判别等。

(1) 最小化问题: 部分教材以最小化形式作为线性规划的标准形式, 此时单纯形表中的最优性判别条件与最大化问题存在差异。若表中所有检验数满足 $\sigma_j \geq 0$, 则当前基可行解即为最优解。否则, 应继续迭代以改进目标函数值。

(2) 退化现象: 在单纯形法的迭代过程中, 通常采用最小比值原则确定出基变量。当计算最小比值时, 若出现两个或多个相同的最小值, 则在下一轮单纯形表中可能出现某些基变量取值为 0, 该现象称为退化, 对应的解称为退化解。这种现象一般是由于模型中存在冗余约束, 使得可行域的同一顶点对应多个基可行解。当出

现退化解时, 可能会导致迭代过程发生循环。为此, R. G. Bland 在 1977 年提出了 Bland 规则。一方面, 当存在多个 $\sigma_j > 0$ 时, 选取下标最小的非基变量作为入基变量。另一方面, 当计算 θ 比值出现多个相同最小值时, 选择下标最小的基变量作为出基变量。按上述规则进行变量选择, 可以避免单纯形法计算过程中出现循环现象。

(3) 无可行解的判别: 无论采用大 M 法还是两阶段法, 初始单纯形表中的解通常包含非零人工变量, 并非原问题的可行解。若在迭代计算结束时, 所有检验数满足 $\sigma_j \le 0$, 但基变量中仍然包含非零人工变量, 则说明原线性规划问题无可行解。

2.5 对偶问题

2.5.1 问题提出

针对例2.1的生产规划问题, 略作调整以便更好理解对偶问题。

例 2.10 某公司计划生产 I、II 两种家电产品, 生产过程中分别占用设备 A、设备 B 的台时及调试工序时间 (单位: 小时)、各设备及工序的每日可用能力 (单位: 小时)、两种产品每件的获利 (单位: 元) 情况如表2.12所示。因经营调整, 企业不再自行组织生产, 而是将现有的三种能力资源全部出租用于对外加工。那么, 公司应考虑如何确定各种资源的租价, 才能获得最大租金利润。

表 2.12 制造两种家电产品所需信息

项目	产品型号		每日可用能力
	I	II	
A	0	5	15
B	6	2	24
调试工序	1	1	5
利润	2	1	

解 当公司选择资源出租而非自行生产时, 出租资源获得的总租金收入, 不应低于公司自行生产时能够获得的利润水平, 否则出租行为失去经济意义。设 y_1, y_2, y_3 分别表示单位时间内出租设备 A、设备 B 以及调试工序所收取的租金, 满足约束条件

$$\begin{cases} 6y_2 + y_3 \ge 2 \\ 5y_1 + 2y_2 + y_3 \ge 1 \end{cases}$$

另一方面, 从租赁方的角度出发, 应在满足生产需求的前提下, 尽可能降低自身的租金成本。于是得到数学模型

$$\min\ w = 15y_1 + 24y_2 + 5y_3$$
$$\text{s.t.}\ \begin{cases} 6y_2 + y_3 \geq 2 \\ 5y_1 + 2y_2 + y_3 \geq 1 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

考虑原问题为如下对称形式的线性规划问题

$$\max\ z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$
$$\text{s.t.}\ \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\ x_1, x_2, \cdots, x_n \geq 0 \end{cases} \tag{2.3}$$

其对偶问题可表述为

$$\min\ w = b_1y_1 + b_2y_2 + \cdots + b_my_m$$
$$\text{s.t.}\ \begin{cases} a_{11}y_1 + a_{21}y_2 + \cdots + a_{m1}y_m \geq c_1 \\ a_{12}y_1 + a_{22}y_2 + \cdots + a_{m2}y_m \geq c_2 \\ \cdots \\ a_{1n}y_1 + a_{2n}y_2 + \cdots + a_{mn}y_m \geq c_n \\ y_1, y_2, \cdots, y_m \geq 0 \end{cases} \tag{2.4}$$

其中, 所有决策变量均满足非负约束, 同时当目标函数求最小时, 所有约束条件均取 “≥”。上式可简写为

$$\min\ w = \sum_{i=1}^{m} b_iy_i$$
$$\text{s.t.}\ \begin{cases} \sum\limits_{i=1}^{m} a_{ij}y_i \geq c_j,\ j = 1, 2, \cdots, n \\ y_i \geq 0,\ i = 1, 2, \cdots, m \end{cases}$$

进一步得到矩阵形式

$$\min \ w = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}$$
$$\text{s.t.} \begin{cases} \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c} \\ \boldsymbol{y} \geq \boldsymbol{0} \end{cases}$$

将对称形式线性规划的原问题与对偶问题进行对比, 如表2.13所示。

表 2.13 对称形式原问题与对偶问题比较

项目	原问题(2.3)	对偶问题(2.4)
$\boldsymbol{A}$	约束条件的系数	约束条件的系数转置
$\boldsymbol{b}$	约束条件的右端项	目标函数的系数转置
$\boldsymbol{c}$	目标函数的系数转置	约束条件的右端项
目标函数	$\max \ z = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$	$\min \ w = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}$
约束条件	$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$	$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c}$
决策变量	$\boldsymbol{x} \geq \boldsymbol{0}$	$\boldsymbol{y} \geq \boldsymbol{0}$

例 2.11 写出下列线性规划问题的对偶问题

$$\max \ z = 5x_1 + 6x_2$$
$$\text{s.t.} \begin{cases} 3x_1 - 2x_2 \leq 7 \\ 4x_1 + x_2 \leq 9 \\ x_1, x_2 \geq 0 \end{cases}$$

解 根据表2.13, 引入对偶变量 $y_1, y_2 \geq 0$, 可直接写出对偶问题

$$\min \ w = 7y_1 + 9y_2$$
$$\text{s.t.} \begin{cases} 3y_1 + 4y_2 \geq 5 \\ -2y_1 + y_2 \geq 6 \\ y_1, y_2 \geq 0 \end{cases}$$

请尝试推导对偶问题的对偶规划, 并将其与原问题进行比较。

性质 2.11

原问题与对偶问题互为对偶, 且对偶问题的对偶即为原问题。

2.5.2 非对称形式的对偶问题

下面探讨非对称形式线性规划问题如何写出其对偶问题。

(1) 等式约束的转换: 若约束条件为等式形式, 则该约束可视为同时满足两个方向的不等式约束, 即“≤”与“≥”同时成立。具体而言, 等式约束

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i,\ i = 1, 2, \cdots, m$$

可等价地分解为

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \le b_i,\ a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \ge b_i$$

(2) 不等式方向的转换: 当约束条件本身为不等式时, 其方向应与目标函数类型相协调。若原问题为最大化, 则约束条件为“≤”型。具体而言, 对于

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \ge b_i$$

可通过两端同乘 -1, 将其转化为等价的“≤”型约束

$$-a_{i1}x_1 - a_{i2}x_2 - \cdots - a_{in}x_n \le -b_i$$

相反, 若原问题为最小化, 则约束条件为“≥”型。

(3) 决策变量的转换: 当变量 x_k 取值无约束时, 可将其表示为两个非负变量之差

$$x_k = x_k' - x_k'',\ x_k', x_k'' \ge 0$$

此时, x_k 的取值符号由 x_k' 与 x_k'' 的相对大小决定。当 $x_k \le 0$ 时, 则可令

$$x_k' = -x_k,\ x_k' \ge 0$$

从而满足决策变量的取值非负。

例 2.12 写出下列线性规划问题的对偶问题

$$\max\ z = 5x_1 + 6x_2$$

$$\text{s.t.} \begin{cases} 3x_1 - 2x_2 = 7 \\ 4x_1 + x_2 \le 9 \\ x_1, x_2 \ge 0 \end{cases}$$

解 首先将等式约束拆分为两个方向的不等式约束, 得到

$$\max \ z = 5x_1 + 6x_2$$
$$\text{s.t.} \begin{cases} 3x_1 - 2x_2 \le 7 \\ -3x_1 + 2x_2 \le -7 \\ 4x_1 + x_2 \le 9 \\ x_1, x_2 \ge 0 \end{cases}$$

引入三个对偶变量, 分别记为 $y_1', y_1'', y_2 \ge 0$。根据表2.13, 写出如下对偶问题

$$\min \ w = 7y_1' - 7y_1'' + 9y_2$$
$$\text{s.t.} \begin{cases} 3y_1' - 3y_1'' + 4y_2 \ge 5 \\ -2y_1' + 2y_1'' + y_2 \ge 6 \\ y_1', y_1'', y_2 \ge 0 \end{cases}$$

注意到 y_1' 与 y_1'' 同时满足非负约束, 其差 $y_1 = y_1' - y_1''$ 可取任意实数, 因而 y_1 无约束。将 y_1 代入目标函数与约束条件, 得到最终的对偶问题

$$\min \ w = 7y_1 + 9y_2$$
$$\text{s.t.} \begin{cases} 3y_1 + 4y_2 \ge 5 \\ -2y_1 + y_2 \ge 6 \\ y_1\text{无约束}, \ y_2 \ge 0 \end{cases}$$

例 2.13 写出下列线性规划问题的对偶问题

$$\max \ z = c_1x_1 + c_2x_2 + c_3x_3$$
$$\text{s.t.} \begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \le b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \ge b_3 \\ x_1 \ge 0, \ x_2 \le 0, \ x_3\text{无约束} \end{cases}$$

解 令 $x_2 = -x_2'$, $x_3 = x_3' - x_3''$, 经过变换后可重新表达为

$$\max \ z = c_1x_1 - c_2x_2' + c_3x_3' - c_3x_3''$$

$$\text{s.t.} \begin{cases} a_{11}x_1 - a_{12}x_2' + a_{13}x_3' - a_{13}x_3'' \le b_1 \\ a_{21}x_1 - a_{22}x_2' + a_{23}x_3' - a_{23}x_3'' \le b_2 \\ -a_{21}x_1 + a_{22}x_2' - a_{23}x_3' + a_{23}x_3'' \le -b_2 \\ -a_{31}x_1 - a_{32}x_2' - a_{33}x_3' + a_{33}x_3'' \le -b_3 \\ x_1, x_2', x_3', x_3'' \ge 0 \end{cases}$$

引入对偶变量 $y_1, y_2', y_2'', y_3' \ge 0$, 按对应关系写出对偶问题

$$\min \ w = b_1y_1 + b_2y_2' - b_2y_2'' - b_3y_3'$$

$$\text{s.t.} \begin{cases} a_{11}y_1 + a_{21}y_2' - a_{21}y_2'' - a_{31}y_3' \ge c_1 \\ -a_{12}y_1 - a_{22}y_2' + a_{22}y_2'' - a_{32}y_3' \ge -c_2 \\ a_{13}y_1 + a_{23}y_2' - a_{23}y_2'' - a_{33}y_3' \ge c_3 \\ -a_{13}y_1 - a_{23}y_2' + a_{23}y_2'' + a_{33}y_3' \ge -c_3 \\ y_1, y_2', y_2'', y_3' \ge 0 \end{cases}$$

令 $y_2 = y_2' - y_2''$, $y_3 = -y_3'$, 整理得到最终的对偶问题

$$\min \ w = b_1y_1 + b_2y_2 + b_3y_3$$

$$\text{s.t.} \begin{cases} a_{11}y_1 + a_{21}y_2 + a_{31}y_3 \ge c_1 \\ a_{12}y_1 + a_{22}y_2 + a_{32}y_3 \le c_2 \\ a_{13}y_1 + a_{23}y_2 + a_{33}y_3 = c_3 \\ y_1 \ge 0, \ y_2\text{无约束}, \ y_3 \le 0 \end{cases}$$

通过对比发现, 无论对称形式还是非对称形式的线性规划问题, 在构造对偶问题时, 表2.13中的前四行均成立, 区别仅体现在约束条件的形式及其对应变量的取值。表2.14归纳出对称与非对称形式的对偶规则。

2.5.3 对偶理论

本节探讨原问题(2.3)及其对偶问题(2.4)之间的关系, 包括弱对偶理论、最优性理论、强对偶理论和互补松弛理论。

表 2.14 原问题与对偶问题比较

原问题(2.3)	对偶问题(2.4)
目标函数 max z	目标函数 min w
决策变量 n 个	约束条件 n 个
决策变量 ≥ 0	约束条件 $\geq \boldsymbol{c}$
决策变量 ≤ 0	约束条件 $\leq \boldsymbol{c}$
决策变量无约束	约束条件 $= \boldsymbol{c}$
约束条件 m 个	决策变量 m 个
约束条件 $\geq \boldsymbol{b}$	决策变量 ≤ 0
约束条件 $\leq \boldsymbol{b}$	决策变量 ≥ 0
约束条件 $= \boldsymbol{b}$	决策变量无约束
约束条件的右端项	目标函数的系数转置
目标函数的系数转置	约束条件的右端项

定理 2.12

若 $\hat{x}_j$ 是原问题的可行解，$\hat{y}_i$ 是其对偶问题的可行解，则恒有

$$\sum_{j=1}^{n} c_j \hat{x}_j \leq \sum_{i=1}^{m} b_i \hat{y}_i$$

其中 $i = 1, 2, \cdots, m,\ j = 1, 2, \cdots, n$。

证明 由原问题与对偶问题的可行解定义可知

$$\sum_{j=1}^{n} c_j \hat{x}_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} \hat{y}_i \right) \hat{x}_j = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \hat{x}_j \hat{y}_i$$

$$\sum_{i=1}^{m} b_i \hat{y}_i \geq \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} \hat{x}_j \right) \hat{y}_i = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \hat{y}_i \hat{x}_j$$

联立以上两式，即得定理结论。

性质 2.13

原问题任一可行解的目标函数值，是其对偶问题目标函数值的下界。反之，对偶问题任一可行解的目标函数值，是其原问题目标函数值的上界。

性质 2.14

若原问题有可行解且目标函数值无界，则其对偶问题无可行解。反之，对偶问题有无界解，则原问题无可行解。

性质 2.15

若原问题有可行解，对偶问题无可行解，则原问题目标函数值无界。反之，对偶问题有可行解，而原问题无可行解，则对偶问题的目标函数值无界。

下面给出线性规划的最优性理论与强对偶理论。其中，最优性理论揭示了最优解与可行解之间的内在联系，强对偶理论则建立了原问题与对偶问题之间的对应关系。

定理 2.16

若 $\hat{x}_j$ 是原问题的可行解，$\hat{y}_i$ 是其对偶问题的可行解，且有

$$\sum_{j=1}^{n} c_j \hat{x}_j = \sum_{i=1}^{m} b_i \hat{y}_i$$

则 $\hat{x}_j$ 是原问题的最优解，$\hat{y}_i$ 是其对偶问题的最优解。

证明 设 x_j^* 是原问题的最优解，y_i^* 是其对偶问题的最优解，一定有

$$\sum_{j=1}^{n} c_j \hat{x}_j \le \sum_{j=1}^{n} c_j x_j^*, \quad \sum_{i=1}^{m} b_i y_i^* \le \sum_{i=1}^{m} b_i \hat{y}_i$$

又由题设条件和定理2.12可知

$$\sum_{j=1}^{n} c_j \hat{x}_j = \sum_{i=1}^{m} b_i \hat{y}_i, \quad \sum_{j=1}^{n} c_j x_j^* \le \sum_{i=1}^{m} b_i y_i^*$$

联立即得定理结论。

定理 2.17

若原问题有最优解，则其对偶问题也有最优解，且两者的目标函数值相等。若原问题与对偶问题均有可行解，则两者均有最优解，且它们最优解的目标函数值相等。

证明 一方面，当原问题有最优解时，其对偶问题的对应解为可行解，满足 $z = w$。由最优性知，此时两者的解均为最优解。另一方面，若原问题与对偶问题均有可行解，由第一个推论知，原问题的目标函数有上界，对偶问题的目标函数有下界，因此两者均存在最优解，且最优值相等。

定理 2.18

在线性规划问题的最优解中，若某一约束条件对应的对偶变量为非 0，则该约束条件取严格等式。反之，若约束条件取严格不等式，则其对应的对偶变量必为 0。

证明 由定理2.12知

$$\sum_{j=1}^{n} c_j \hat{x}_j \le \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}\hat{x}_j\hat{y}_i \le \sum_{i=1}^{m}\sum_{j=1}^{n} b_i\hat{y}_i$$

又根据最优性定理2.16，上式中不等式全为等式。由右端等式整理可得

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}\hat{x}_j - b_i\right)\hat{y}_i = 0$$

由于对偶变量 $\hat{y}_i \ge 0$ 且 $\sum_{j=1}^{n} a_{ij}\hat{x}_j - b_i \le 0$，因此对所有 $i = 1, 2, \cdots, m$ 有

$$\left(\sum_{j=1}^{n} a_{ij}\hat{x}_j - b_i\right)\hat{y}_i = 0$$

于是，若 $\hat{y}_i > 0$，则有 $\sum_{j=1}^{n} a_{ij}\hat{x}_j = b_i$，即 $\hat{x}_{si} = 0$。若 $\sum_{j=1}^{n} a_{ij}\hat{x}_j < b_i$，即 $\hat{x}_{si} = 0$，则有 $\hat{y}_i = 0$。

由此可推得 $\hat{x}_{si} \cdot \hat{y}_i = 0$，此即为互补松弛定理，也是后续理解非线性规划中 Karush-Kuhn-Tucker (KKT) 条件的重要基础。

例 2.14 试用对偶理论证明下列线性规划问题无最优解。

$$\max \ z = x_1 + x_2$$
$$\text{s.t.} \begin{cases} -x_1 + x_2 + x_3 \le 2 \\ -2x_1 + x_2 - x_3 \le 1 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

解 写出原问题的对偶问题

$$\min \ w = 2y_1 + y_2$$
$$\text{s.t.} \begin{cases} -y_1 - 2y_2 \ge 1 \\ y_1 + y_2 \ge 1 \\ y_1 - y_2 \ge 0 \\ y_1, y_2 \ge 0 \end{cases}$$

由对偶问题的第一个约束条件知, 在 $y_1, y_2 \geq 0$ 的条件下, 该约束不可能满足, 因此对偶问题无可行解。由性质2.15知, 原问题的目标函数无界, 从而无最优解。

例 2.15 已知线性规划问题

$$\min \ w = 2x_1 + 3x_2 + 5x_3 + 2x_4 + 3x_5$$
$$\text{s.t.} \begin{cases} x_1 + x_2 + 2x_3 + x_4 + 3x_5 \geq 4 \\ 2x_1 - x_2 + 3x_3 + x_4 + x_5 \geq 3 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

其对偶问题的最优解为 $y_1^* = 4/5,\ y_2^* = 3/5,\ z = 5$, 试用对偶理论找出原问题的最优解。

解 写出原问题的对偶问题

$$\max \ z = 4y_1 + 3y_2$$
$$\text{s.t.} \begin{cases} y_1 + 2y_2 \leq 2 & \text{(a)} \\ y_1 - y_2 \leq 3 & \text{(b)} \\ 2y_1 + 3y_2 \leq 5 & \text{(c)} \\ y_1 + y_2 \leq 2 & \text{(d)} \\ 3y_1 + y_2 \leq 3 & \text{(e)} \\ y_1, y_2 \geq 0 \end{cases}$$

将最优解 $y_1^* = 4/5, y_2^* = 3/5$ 代入对偶问题约束条件, 得

$$\text{(b)} = 1/5 < 3,\ \text{(c)} = 17/5 < 5,\ \text{(d)} = 7/5 < 2$$

上述约束均取严格不等式, 由互补松弛理论可知

$$x_2^* = x_3^* = x_4^* = 0$$

又因 $y_1^*, y_2^* > 0$, 于是原问题的两个约束条件必取等式, 即

$$x_1^* + 3x_5^* = 4,\ 2x_1^* + x_5^* = 3$$

解得 $x_1^* = 1,\ x_5^* = 1$。因此, 原问题的最优解为 $\boldsymbol{x}^* = (1, 0, 0, 0, 1)^{\mathrm{T}}$, 最优值为 $w^* = 5$。

2.6 应用案例

2.6.1 问题描述

下料问题是生产制造领域中一类典型的资源优化问题, 旨在给定原材料规格与成品尺寸需求的前提下, 通过设计合理的切割方案, 实现原材料利用率的最大化。根据切割维度可分为一维下料与二维下料, 其中一维下料仅考虑长度的切割, 二维下料需同时考虑长度与宽度的布局。

例 2.16 某单位需要加工 100 套工架, 每套工架包含长度分别为 2.9 米、2.1 米、1.5 米三种规格的圆钢各 1 根。现有原材料为长度 7.4 米的同规格圆钢, 试确定最优下料方案, 使原材料总消耗量与废料最少。

2.6.2 模型建立

若采用单一下料方式, 即每根原材料仅截取 2.9 米、2.1 米、1.5 米的圆钢各 1 根, 则单根废料为 0.9 米。完成 100 套需消耗 100 根原材料, 总废料达 90 米。若采用套截方案, 可减少原材料消耗与废料。表2.15列出了五种套截方案及其消耗量与废料 (单位: 米)。

表 2.15 套截方案

项目	下料组合				
	A	B	C	D	E
I	1	2	0	1	0
II	0	0	2	2	1
III	3	1	2	0	3
消耗量	7.4	7.3	7.2	7.1	6.6
废料	0.0	0.1	0.2	0.3	0.8

设各方案下料的原材料根数分别为 x_1, x_2, x_3, x_4, x_5, 以总废料最小为优化目标, 同时满足三种规格圆钢各 100 根的需求约束, 建立线性规划模型

$$\min \ z = 0x_1 + 0.1x_2 + 0.2x_3 + 0.3x_4 + 0.8x_5$$

$$\text{s.t.} \begin{cases} x_1 + 2x_2 + x_4 = 100 \\ 2x_3 + 2x_4 + x_5 = 100 \\ 3x_1 + x_2 + 2x_3 + 3x_5 = 100 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

2.6.3 算法设计

该模型不存在明显的初始可行基, 因此采用两阶段法求解。引入人工变量 x_6, x_7, x_8, 构造辅助线性规划问题

$$\max \ w = -x_6 - x_7 - x_8$$

$$\text{s.t.} \begin{cases} x_1 + 2x_2 + x_4 + x_6 = 100 \\ 2x_3 + 2x_4 + x_5 + x_7 = 100 \\ 3x_1 + x_2 + 2x_3 + 3x_5 + x_8 = 100 \\ x_i \geq 0, \ i = 1, 2, \cdots, 8 \end{cases}$$

第一阶段的单纯形迭代过程如表2.16所示, 求得原线性规划问题的一个基可行解为 $x_1 = 30, \ x_2 = 10, \ x_4 = 50, \ x_3 = x_5 = 0$。

表 2.16 第一阶段单纯形法迭代过程

$c_j \rightarrow$			0	0	0	0	0	−1	−1	−1
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
−1	x_6	100	1	2	0	1	0	1	0	0
−1	x_7	100	0	0	2	2	1	0	1	0
−1	x_8	100	[3]	1	2	0	3	0	0	1
	$c_j - z_j$		4	3	4	3	4	0	0	0
−1	x_6	200/3	0	5/3	−2/3	1	−1	1	0	−1/3
−1	x_7	100	0	0	2	[2]	1	0	1	0
0	x_1	100/3	1	1/3	2/3	0	1	0	0	1/3
	$c_j - z_j$		0	5/3	4/3	3	0	0	0	−4/3
−1	x_6	50/3	0	[5/3]	−5/3	0	−3/2	1	−1/2	−1/3
0	x_4	50	0	0	1	1	1/2	0	1/2	0
0	x_1	100/3	1	1/3	2/3	0	1	0	0	1/3
	$c_j - z_j$		0	5/3	−5/3	0	−3/2	0	−3/2	−4/3
0	x_2	10	0	1	−1	0	−9/10	3/5	−3/10	−1/5
0	x_4	50	0	0	1	1	1/2	0	1/2	0
0	x_1	30	1	0	1	0	13/10	−1/5	1/10	2/5
	$c_j - z_j$		0	0	0	0	0	−1	−1	−1

剔除人工变量的所在列, 进入第二阶段对原问题继续迭代求解, 过程如表2.17所示。最终得到原问题的最优解为 $\boldsymbol{x}^* = (0, 40, 30, 20, 0)^{\mathrm{T}}$, 对应的最优值为 $z^* = 16$。

表 2.17 第二阶段单纯形法迭代过程

$c_j \rightarrow$			0	0.1	0.2	0.3	0.8
c_B	x_B	b	x_1	x_2	x_3	x_4	x_5
0.1	x_2	10	0	1	−1	0	−9/10
0.3	x_4	50	0	0	1	1	1/2
0	x_1	30	1	0	[1]	0	13/10
$c_j - z_j$			0	0	0	0	37/50
0.1	x_2	40	1	1	0	0	2/5
0.3	x_4	20	−1	0	0	1	−4/5
0.2	x_3	30	1	0	1	0	13/10
$c_j - z_j$			0	0	0	0	37/50

2.6.4 结果分析

计算结果表明, 采用方案 B 下料 40 根、方案 C 下料 30 根、方案 D 下料 20 根可取得最优下料。此时原材料总消耗量为 90 根, 总废料长度为 16 米, 对应的材料利用率为

$$\frac{100 \times 2.9 + 100 \times 2.1 + 100 \times 1.5}{90 \times 7.4} \approx 97.60\%$$

与单一下料方案相比, 上述最优套截方案显著提升了材料的利用率, 可在实际生产中大幅降低生产成本。读者可尝试用大 M 法进行求解, 并验证结果的一致性。

习题

2.1 将下述线性规划问题化成标准形式。

(1) $\min\ z = -3x_1 + 4x_2 + 2x_3 + 5x_4$

$$\text{s.t.}\ \begin{cases} 4x_1 - x_2 + 3x_3 + 2x_4 \le -2 \\ -x_1 + x_2 + x_3 - x_4 \ge 2 \\ x_1, x_2, x_3 \ge 0,\ x_4\text{无约束} \end{cases}$$

(2) $\min\ z = -x_1 + 2x_2 - 3x_3$

$$\text{s.t.}\ \begin{cases} x_1 + x_2 + x_3 \le 7 \\ x_1 - x_2 + x_3 \ge 2 \\ x_1, x_2 \ge 0,\ x_3\text{无约束} \end{cases}$$

2.2 用图解法和单纯形法求解下述线性规划问题。

(1) $\max\ z = 2x_1 + 3x_2$

$$\text{s.t.}\begin{cases} 2x_1 + 2x_2 \le 14 \\ 4x_1 \le 12 \\ 3x_2 \le 15 \\ x_1, x_2 \ge 0 \end{cases}$$

(2) $\max\ z = 50x_1 + 100x_2$

$$\text{s.t.}\begin{cases} x_1 + x_2 \le 300 \\ 2x_1 + x_2 \le 400 \\ x_2 \le 250 \\ x_1, x_2 \ge 0 \end{cases}$$

2.3 分别用大 M 法和两阶段法求解下列线性规划问题, 并指出问题的解属于哪一类。

(1) $\max\ z = 4x_1 + 5x_2 + x_3$

$$\text{s.t.}\begin{cases} 3x_1 + 2x_2 + x_3 \ge 18 \\ x_1 + x_2 \le 4 \\ x_1 + x_2 \le 5 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

(2) $\max\ z = 2x_1 + x_2 + x_3$

$$\text{s.t.}\begin{cases} 4x_1 + 2x_2 + 2x_3 \ge 4 \\ 2x_1 + 4x_2 + 2x_3 \le 20 \\ 4x_1 + 8x_2 + 2x_3 \le 16 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

2.4 已知某线性规划问题的初始单纯形表, 以及经单纯形法迭代后得到的表2.18, 试求表中 $(a)-(l)$ 的值。

表 2.18 初始单纯形表与迭代后的单纯形表

项目		x_1	x_2	x_3	x_4	x_5
x_4	6	(b)	(c)	(d)	1	0
x_5	1	-1	3	(e)	0	1
$c_j - z_j$		(a)	-1	2	0	0
x_1	(f)	(g)	2	-1	1/2	0
x_5	4	(h)	(i)	1	1/2	1
$c_j - z_j$		0	-7	(j)	(k)	(l)

2.5 写出下列线性规划问题的对偶问题。

(1) $\max\ z = 2x_1 + 5x_3$

$$\text{s.t.}\begin{cases} x_1 + x_2 \le 4 \\ 2x_1 - x_2 + 3x_3 \le 5 \\ x_1 - 2x_3 \le 3 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

(2) $\min\ z = 4x_1 + 2x_2 - 3x_3$

$$\text{s.t.}\begin{cases} -x_1 + 2x_2 \le 6 \\ 2x_1 + 3x_3 \ge 9 \\ x_1 + 5x_2 - 2x_3 = 4 \\ x_1\text{无约束},\ x_2, x_3 \ge 0 \end{cases}$$

2.6 已知线性规划问题

$$\min \ w = 2x_1 + 4x_2 + x_3 + x_4$$

$$\text{s.t.} \begin{cases} x_1 + 3x_2 + x_4 \le 8 \\ 2x_1 + x_2 \le 6 \\ x_2 + x_3 + x_4 \le 6 \\ x_1 + x_2 + x_3 \le 9 \\ x_1, x_2, x_3, x_4 \ge 0 \end{cases}$$

(1) 写出其对偶问题。(2) 已知其对偶问题的最优解为 $\boldsymbol{x}^* = (2, 2, 4, 0)^{\mathrm{T}}$, 试根据对偶理论, 直接求出对偶问题的最优解。

2.7 某工厂使用 A、B、C 三种原料生产三种产品 I、II、III, 对应的用料 (单位: 公斤) 和利润 (单位: 千元) 情况如表2.19所示。试制定生产计划, 使总利润最大。

表 2.19 三种产品用料和利润情况

项目	产品型号			原料可用量
	I	II	III	
A	2	3	0	1 500
B	0	2	4	800
C	3	2	5	2 000
利润	3	5	4	

3

整数规划

3.1 基本概念

3.1.1 整数规划问题

在前面讨论的线性规划问题中, 目标函数与约束条件均为线性, 且决策变量可取连续非负实数。然而, 在实际工程与管理问题中, 大量决策变量要求取值为整数。例如, 人数、车辆数、生产批次、机器台数、项目个数等, 都必须使用整数描述。又如, 电路的通断、逻辑判断的对错、方案的选用与否等, 也需要使用 0–1 二值变量加以刻画。这类要求部分或全部决策变量取整数的优化问题, 因其建模与求解方法具有特殊性, 形成了运筹学中的整数规划。

例 3.1 某企业利用集装箱托运 A、B 两种货物, 每个集装箱的体积 (单位: 立方米) 、重量 (单位: 吨) 、利润 (单位: 千元) 及托运限制如表3.1所示。试确定两种货物各托运多少箱, 使总利润最大。

表 3.1 集装箱托运货物信息

项目	货物参数		利润
	体积	重量	
A	5	2	20
B	4	5	10
托运限制	24	13	

解 设 x_1, x_2 分别为托运 A、B 两种货物的箱数。因托运箱数必须为整数，据此可建立如下数学模型

$$\max \ z = 20x_1 + 10x_2$$

$$\text{s.t.} \begin{cases} 5x_1 + 4x_2 \le 24 \\ 2x_1 + 5x_2 \le 13 \\ x_1, x_2 \ge 0 \text{且取整数} \end{cases}$$

例 3.2 某服务部门各时段（每 $2h$ 为一时段）需要的服务员人数如表3.2所示。按规定，服务员连续工作 $8h$（即四个时段）为一班。试确定服务员的工作时间，使所需服务员总数最少。

表 3.2 各时段需要服务员人数

时段	1	2	3	4	5	6	7	8
服务员最少数目	10	8	9	11	13	8	5	3

解 设第 j 时段开始时上班的服务员人数为 x_j。由于第 j 时段上班的服务员将在第 $(j+3)$ 时段结束后下班，只需考虑决策变量 x_1, x_2, x_3, x_4, x_5。根据各时段需要的服务员人数，可建立数学模型

$$\min \ z = x_1 + x_2 + x_3 + x_4 + x_5$$

$$\text{s.t.} \begin{cases} x_1 \ge 10 \\ x_1 + x_2 \ge 8 \\ x_1 + x_2 + x_3 \ge 9 \\ x_1 + x_2 + x_3 + x_4 \ge 11 \\ x_2 + x_3 + x_4 + x_5 \ge 13 \\ x_3 + x_4 + x_5 \ge 8 \\ x_4 + x_5 \ge 5 \\ x_5 \ge 3 \\ x_1, x_2, x_3, x_4, x_5 \ge 0 \text{且取整数} \end{cases}$$

显然，服务员人数 x_1, x_2, x_3, x_4, x_5 必须取整数，因此该问题属于典型的整数规划问题。

例 3.3 现有资金总额为 B, 可供选择的投资项目有 n 个, 项目 j 所需投资额和预期收益分别为 a_j 和 c_j, 其中 $j = 1, 2, \cdots, n$。此外, 由于条件限制, 附加以下约束

(1) 若选择项目 1, 则必须同时选择项目 2, 反之不一定。

(2) 项目 3、4 中至少选择一个。

(3) 项目 5、6、7 中恰好选择两个。

应如何选择投资项目, 使总预期收益最大。

解 每个项目都有投资与不投资两种状态, 为此引入 0–1 变量

$$x_j = \begin{cases} 1, & \text{对项目}j\text{投资} \\ 0, & \text{对项目}j\text{不投资} \end{cases}$$

根据现有资金总额与条件限制, 可建立如下数学模型

$$\max \ z = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \begin{cases} \sum\limits_{j=1}^{n} a_j x_j \le \mathrm{B} \\ x_2 \ge x_1 \\ x_3 + x_4 \ge 1 \\ x_5 + x_6 + x_7 = 2 \\ x_j = 0\text{或}1, \ j = 1, 2, \cdots, n \end{cases}$$

例 3.4 工厂 A_1、A_2 生产某种物资, 因供不应求, 拟再新建一家工厂, 可选方案为 A_3、A_4。物资需求地有 B_1、B_2、B_3、B_4 共四个。各工厂年生产能力 (单位: 吨 / 年) 、各地年需求量 (单位: 吨 / 年) 、各厂至各需求地的单位物资运费 (单位: 万元 / 吨) 如表3.3所示。工厂 A_3、A_4 的生产费用分别为 1 200 万元和 1 500 万元。试确定决策方案, 使总费用最小。

解 设 x_{ij} 表示由工厂 A_i 送往物资需求地 B_j 的物资数量, 对应的单位物资运费为 c_{ij}, 其中 $i, j = 1, 2, 3, 4$。建厂方案包括 A_3、A_4 两种, 于是引入 0–1 变量

$$y = \begin{cases} 1, & \text{若建工厂 } \mathrm{A}_3 \\ 0, & \text{若建工厂 } \mathrm{A}_4 \end{cases}$$

表 3.3 各工厂信息

工厂	物资需求地				生产能力
	B_1	B_2	B_3	B_4	
A_1	2	9	3	4	400
A_2	8	3	5	7	600
A_3	7	6	1	2	200
A_4	4	5	2	5	200
需求量	350	400	300	150	

总费用包含总运费与新建工厂的生产费用, 得到目标函数为

$$\min \ z = \sum_{i=1}^{4}\sum_{j=1}^{4} c_{ij}x_{ij} + 1\,200y + 1\,500(1-y)$$

同时考虑工厂 A_1、A_2、A_3、A_4 的年生产能力以及需求地 B_1、B_2、B_3、B_4 的年需求量, 最终可建立如下数学模型

$$\min \ z = \sum_{i=1}^{4}\sum_{j=1}^{4} c_{ij}x_{ij} + 1\,200y + 1\,500(1-y)$$

$$\text{s.t.} \begin{cases} x_{11}+x_{21}+x_{31}+x_{41} = 350 \\ x_{12}+x_{22}+x_{32}+x_{42} = 400 \\ x_{13}+x_{23}+x_{33}+x_{43} = 300 \\ x_{14}+x_{24}+x_{34}+x_{44} = 150 \\ x_{11}+x_{12}+x_{13}+x_{14} \le 400 \\ x_{21}+x_{22}+x_{23}+x_{24} \le 600 \\ x_{31}+x_{32}+x_{33}+x_{34} \le 200y \\ x_{41}+x_{42}+x_{43}+x_{44} \le 200(1-y) \\ x_{ij} \ge 0,\ i,j = 1,2,3,4 \\ y\text{取0或1} \end{cases}$$

请读者思考, 若允许同时新建工厂 A_3、A_4, 应如何进行数学建模。

3.1.2 数学模型

定义 3.1

要求部分或全部决策变量取整数的规划问题称为整数规划，其一般形式为

$$\begin{aligned} \max\ & z = \sum_{j=1}^{n} c_j x_j \\ \text{s.t.}\ & \begin{cases} \sum\limits_{j=1}^{n} a_{ij} x_j \le (\ge, =)\ b_i,\ i = 1, 2, \cdots, m \\ x_j \ge 0,\ j = 1, 2, \cdots, n \\ x_j \text{中部分或全部取整数} \end{cases} \end{aligned} \tag{3.1}$$

定义 3.2

若不考虑整数条件，由余下的目标函数和约束条件构成的规划问题

$$\begin{aligned} \max\ & z = \sum_{j=1}^{n} c_j x_j \\ \text{s.t.}\ & \begin{cases} \sum\limits_{j=1}^{n} a_{ij} x_j \le (\ge, =)\ b_i,\ i = 1, 2, \cdots, m \\ x_j \ge 0,\ j = 1, 2, \cdots, n \end{cases} \end{aligned}$$

称为原整数规划(3.1)的松弛问题。进一步，若松弛问题是线性规划，则称原整数规划为整数线性规划。

根据决策变量的取值情况，整数线性规划可分为三类。第一，纯整数线性规划，即全部决策变量均要求取整数。第二，0–1 整数线性规划，这里决策变量仅允许取 0 或 1。第三，混合整数线性规划，其决策变量中一部分需取整数，其余变量不作整数要求。

3.1.3 解的特点

以例3.1为例，其松弛问题为

$$\begin{aligned} \max\ & z = 20x_1 + 10x_2 \\ \text{s.t.}\ & \begin{cases} 5x_1 + 4x_2 \le 24 \\ 2x_1 + 5x_2 \le 13 \\ x_1, x_2 \ge 0 \end{cases} \end{aligned}$$

通过图解法或单纯形法求得

$$\boldsymbol{x}^* = (4.80, 0)^{\mathrm{T}},\ z^* = 96$$

然而, x_1, x_2 表示托运货物的箱数, 必须为整数, 因此不是可行解。

一个直观的想法, 能否通过对上述松弛问题最优解的简单取整, 得到整数规划问题的最优解? 如图3.1所示, 若将 $(4.80, 0)^{\mathrm{T}}$ 向上取整为 $(5, 0)^{\mathrm{T}}$, 会违反第一个约束条件, 因而不是可行解。若将其向下取整为 $(4, 0)^{\mathrm{T}}$, 虽满足所有约束, 但并非最优解。实际上, 该整数规划问题的最优解和最优值分别为

$$\boldsymbol{x}^* = (4, 1)^{\mathrm{T}},\ z^* = 90$$

由上一章可知, 松弛问题的可行域为凸集, 任意两个可行解的凸组合仍为可行解。整数规划问题的可行域是其松弛问题可行域的一个子集, 任意两个可行解的凸组合不一定满足整数约束, 因而不一定为可行解。由于整数规划问题的可行解必是其松弛问题的可行解, 反之则不一定, 因此整数规划问题最优解的目标函数值不优于松弛问题最优解的目标函数值。一般情况下, 松弛问题的最优解不会刚好满足变量的整数约束条件, 自然就不是整数规划问题的最优解。此时, 若对松弛问题最优解中不符合整数要求的分量简单取整, 所得到的解不一定是整数规划问题的最优解, 甚至不一定是可行解。

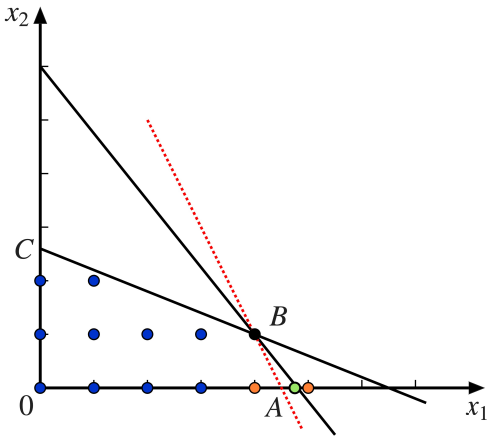

图 3.1 向上取整与向下取整

3.2 分枝定界法

3.2.1 方法介绍

分枝定界法, 也称分支定界法, 是求解纯整数规划或混合整数规划问题的经典方法。相较于枚举法, 它通过分枝不断缩小最优解的搜索空间, 借助定界有效剪枝并提升搜索的效率, 从而实现对整数规划问题的高效求解。

例 3.5 求解下述整数线性规划问题

$$\max \ z = 40x_1 + 90x_2$$
$$\text{s.t.} \begin{cases} 9x_1 + 7x_2 \le 56 \\ 7x_1 + 20x_2 \le 70 \\ x_1, x_2 \ge 0 \text{且取整数} \end{cases}$$

解 记原问题为 A, 首先忽略整数约束, 得到松弛问题 B

$$\max \ z = 40x_1 + 90x_2$$
$$\text{s.t.} \begin{cases} 9x_1 + 7x_2 \le 56 \\ 7x_1 + 20x_2 \le 70 \\ x_1, x_2 \ge 0 \end{cases}$$

(1) 求解松弛问题: 通过图解法或单纯形法求解, 得到最优解为 $x_1 = 4.81$, $x_2 = 1.82$, 最优值为 $z = 356$。由于 x_1, x_2 均非整数, 该解不满足原问题 A 的整数要求, 但松弛问题的最优值给出了原整数规划最优值的上界。

(2) 确定初始下界: 取整数可行解 $x_1 = x_2 = 0$, 对应目标函数值为 $z = 0$, 作为初始下界。于是, 问题 A 最优解的上下界为 $0 \le z^* \le 356$。

(3) 分枝定界迭代求解: (第一次分枝) 由于 $x_1 = 4.81$ 非整数, 将问题 B 分解为如下两枝。舍去 $4 < x_1 < 5$ 之间的可行域, 分枝后的可行域如图3.2所示。

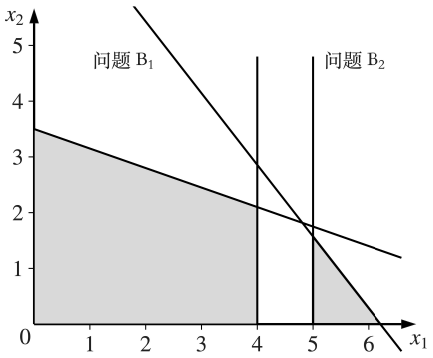

图 3.2 第一次分枝

- 问题 B_1: 在问题 B 的基础上增加条件 $x_1 \le 4$。
- 问题 B_2: 在问题 B 的基础上增加条件 $x_1 \ge 5$。

分别求解两个子问题, 得到问题 B_1 的最优解为 $x_1 = 4.00$, $x_2 = 2.10$, 最优值为 $z = 349$。问题 B_2 的最优解为 $x_1 = 5.00$, $x_2 = 1.57$, 最优值为 $z = 341$。显然, 两

者仍为非整数解。比较可知 349 > 341, 说明问题 B_1 更有可能包含更优的整数解, 故优先对问题 B_1 继续分枝。此时, 问题 A 的上界由 $z = 356$ 修正为 $z = 349$, 而下界仍为 0, 即 $0 \le z^* \le 349$。

(第二次分枝) 问题 B_1 的最优解中 $x_2 = 2.10$ 非整数, 于是将问题 B_1 分解为如下两枝, 其可行域如图3.3所示。

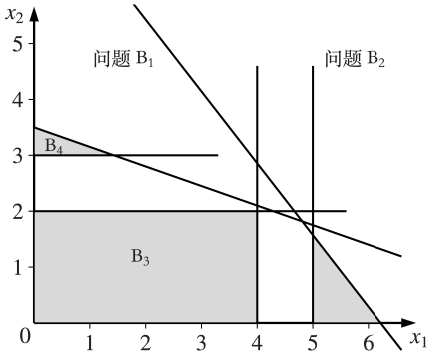

图 3.3 第二次分枝

– 问题 B_3: 在问题 B_1 的基础上增加条件 $x_2 \le 2$。
– 问题 B_4: 在问题 B_1 的基础上增加条件 $x_2 \ge 3$。

求解可得问题 B_3 的最优解为 $x_1 = 4.00$, $x_2 = 2.00$, 此时两变量均为整数, 得到问题 A 的一个可行解, 其最优值为 $z = 340$。而问题 B_4 的最优解为 $x_1 = 1.42$, $x_2 = 3.00$, 最优值为 $z = 327$。由于 $327 < 340$, 说明此分枝不可能产生更优的整数解, 予以剪枝。此时, 下界更新为 $z = 340$, 从而有 $340 \le z* \le 349$。

(第三次分枝) 回到问题 B_2, 由于 $x_2 = 1.57$ 非整数, 于是将问题 B_2 分解为如下两枝, 其可行域如图3.4所示。

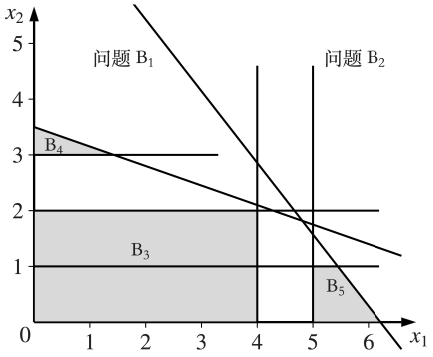

图 3.4 第三次分枝

– 问题 B_5: 在问题 B_2 的基础上增加条件 $x_2 \le 1$。
– 问题 B_6: 在问题 B_2 的基础上增加条件 $x_2 \ge 2$。

求解可得问题 B_5 的最优解为 $x_1 = 5.44$, $x_2 = 1.00$, 最优值为 $z = 308$。由于 $308 < 340$, 不可能产生更优的整数解, 予以剪枝。而问题 B_6 无可行解, 直接舍弃。至此, 所有分枝均已处理或剪枝, 当前下界对应的整数可行解即为最优解。

综上, 原整数规划问题 A 的最优解为 $\boldsymbol{x}^* = (4.00, 2.00)^{\mathrm{T}}$, 最优值为 $z^* = 340$。分枝定界法的完整搜索过程如图3.5所示。

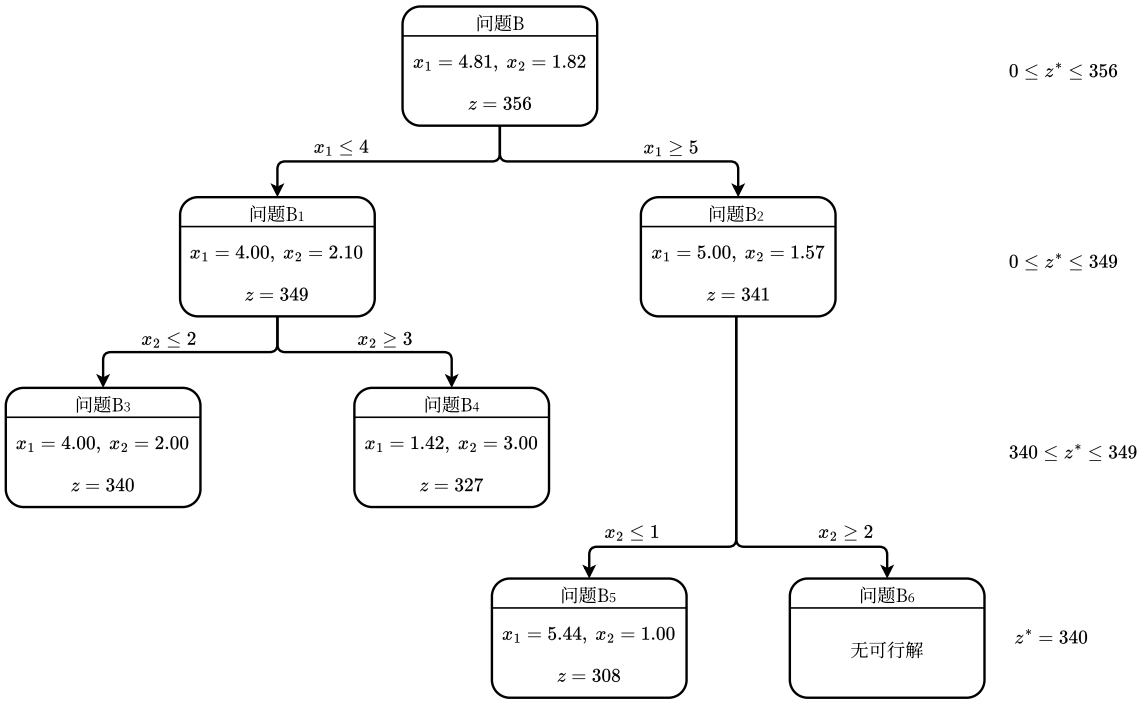

图 3.5 分枝定界法迭代求解

3.2.2 解题步骤

以最大化问题为例, 将原整数规划记为问题 A, 其松弛规划记为问题 B, 分枝定界法可归纳如下步骤。

(1) 求解松弛问题: 若问题 B 无可行解, 则问题 A 也无可行解, 计算终止。若问题 B 的最优解已是整数, 则该解即为问题 A 的最优解, 计算终止。若问题 B 有最优解但非整数, 记其最优目标函数值为 $\bar{z}$, 作为问题 A 最优值 z^* 的初始上界。

(2) 确定初始下界: 通过观察法或简单试探, 得到一个满足整数约束的可行解, 通常取某些变量为 0, 再作适当调整。计算其目标函数值, 记为 $\underline{z}$。于是, 问题 A 的最优值满足上下界约束

$$\underline{z} \le z^* \le \bar{z}$$

算法迭代中, 一旦找到更优的整数可行解, 更新下界 $\underline{z}$, 每个子问题的松弛最优值则用于更新上界 $\bar{z}$。

(3) 分枝定界迭代求解: 对未剪枝的子问题, 求解其松弛问题的最优解。若该松弛问题无可行解, 则该子问题可直接舍弃。若松弛最优解已是整数, 则得到一个可行解, 可据此更新下界 $\underline{z}$, 并且该子问题无需再分枝。若松弛问题的最优解非整数, 则选择其中一个取非整数值的变量 $x_j = b_j$ 进行分枝, 分别加入约束

$$x_j \le \lfloor b_j \rfloor,\ x_j \ge \lceil b_j \rceil$$

从而生成两个新的子问题, 继续求解。对最大化问题, 若某子问题对应松弛问题的最优值不超过当前下界 $\underline{z}$, 说明该分枝不可能产生更优的整数解, 可直接剪枝。若等于 $\underline{z}$, 仍可能产生同等优的整数解, 需保留。重复上述分枝定界迭代, 直至所有分枝均处理完毕。此时, 当前下界所对应的整数可行解即为问题 A 的最优解。

3.3 0–1 整数规划

3.3.1 模型提出

若决策变量仅能取值 0 或 1, 则称其为 0–1 变量。事实上, 0–1 变量是一类典型的逻辑变量, 用于描述系统是否处于某一特定状态或是否采纳某个决策方案。例如

$$x = \begin{cases} 1, & \text{若选择决策方案 A} \\ 0, & \text{若不选择决策方案 A} \end{cases}$$

若问题包含 $\mathrm{E}_1, \mathrm{E}_2, \cdots, \mathrm{E}_n$ 共 n 个要素, 且每个要素 E_j 均对应选择方案 A_j 或不选择方案 A_j 两种互斥状态, 则可定义 0–1 变量

$$x_j = \begin{cases} 1, & \text{若 } \mathrm{E}_j \text{选择 } \mathrm{A}_j \\ 0, & \text{若 } \mathrm{E}_j \text{不选择 } \mathrm{A}_j \end{cases}$$

例 3.6 设工序 B 采用原有加工方式时, 每周工时约束为

$$0.3x_1 + 0.5x_2 \le 150 \tag{3.2}$$

采用新加工方式时, 每周工时约束为

$$0.2x_1 + 0.4x_2 \le 120 \tag{3.3}$$

两种加工方式只能选择其一, 即式(3.2)和(3.3)为相互排斥约束。试将这组相互排斥约束统一为一个约束。

解 针对原有加工方式和新加工方式, 分别引入 0–1 变量

$$y_1 = \begin{cases} 0, & \text{若工序 B 采用原加工方式} \\ 1, & \text{若工序 B 不采用原加工方式} \end{cases}$$

$$y_2 = \begin{cases} 0, & \text{若工序 B 采用新加工方式} \\ 1, & \text{若工序 B 不采用新加工方式} \end{cases}$$

借鉴上一章大 M 法的处理思路, 本节通过引入充分大的常数 M, 巧妙地将相互排斥的约束条件(3.2)和(3.3)统一为

$$\begin{cases} 0.3x_1 + 0.5x_2 \le 150 + My_1 \\ 0.2x_1 + 0.4x_2 \le 120 + My_2 \\ y_1 + y_2 = 1 \end{cases}$$

若要从 p 个约束条件中恰好选择 q $(q < p)$ 个约束条件, 可引入 p 个 0–1 变量

$$y_i = \begin{cases} 0, & \text{若选择第 } i \text{ 个约束条件} \\ 1, & \text{若不选择第 } i \text{ 个约束条件} \end{cases}$$

并将所有约束统一为

$$\begin{cases} \sum\limits_{j=1}^{n} a_{ij}x_j \le b_i + My_i \\ \sum\limits_{i=1}^{p} y_i = p - q \end{cases}$$

例 3.7 现有 A、B、C 三种资源生产 I、II、III 三种产品, 资源量、单件可变费用、固定费用及单价见表3.4。试制定生产计划, 使总收益最大。

解 设 x_j 为生产第 j 种产品的产量, 引入 0–1 变量

$$y_j = \begin{cases} 1, & \text{若生产第 } j \text{ 种产品} \\ 0, & \text{若不生产第 } j \text{ 种产品} \end{cases}$$

表 3.4 固定费用问题

资源	产品类型			资源量
	I	II	III	
A	2	4	8	500
B	2	3	4	300
C	1	2	3	100
单件可变费用	4	5	6	
固定费用	100	150	200	
单价	8	10	12	

其中 $j = 1, 2, 3$。总收益为销售收入减去可变成本与固定成本, 建立 0–1 整数规划模型

$$\max \ z = (8x_1 + 10x_2 + 12x_3) - (4x_1 + 5x_2 + 6x_3) - (100y_1 + 150y_2 + 200y_3)$$

$$\text{s.t.} \begin{cases} 2x_1 + 4x_2 + 8x_3 \le 500 \\ 2x_1 + 3x_2 + 4x_3 \le 300 \\ x_1 + 2x_2 + 3x_3 \le 100 \\ x_1 \le M_1 y_1 \\ x_2 \le M_2 y_2 \\ x_3 \le M_3 y_3 \\ x_1, x_2, x_3 \ge 0\text{且取整数} \\ y_1, y_2, y_3\text{取0或1} \end{cases}$$

值得说明的是, 模型中除 y_1, y_2, y_3 为 0–1 变量外, $x_1, x_2, x_3 \ge 0$ 均为整数变量。

定义 3.3

若整数规划问题(3.1)中的所有决策变量仅允许取 0 或 1, 则称该问题为 0–1 整数规划, 其标准形式为

$$\max \ z = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \begin{cases} \sum\limits_{j=1}^{n} a_{ij} x_j \le (\ge, =)\ b_i, \ i = 1, 2, \cdots, m \\ x_j\text{取0或1}, \ j = 1, 2, \cdots, n \end{cases}$$

3.3.2 隐枚举法

隐枚举法是求解 0–1 整数规划问题最常用的方法之一。对于含 n 个 0–1 变量的整数规划问题, 若采用完全枚举法, 需检查 2^n 种可能的变量组合, 计算量较大。而隐枚举法只需检查全部组合中的一部分, 即可确定问题的最优解。首先尽快找到一个可行解, 例如所有变量取 0, 然后在后续搜索过程中构造适当的过滤条件, 从而不断缩小搜索范围、减少不必要的枚举计算。

例 3.8 试用隐枚举法求解下述问题

$$
\begin{aligned}
&\max \ z = 3x_1 - 2x_2 + 5x_3 \\
&\text{s.t.} \begin{cases}
x_1 + 2x_2 - x_3 \le 2 & \text{(a)} \\
x_1 + 4x_2 + x_3 \le 4 & \text{(b)} \\
x_1 + x_2 \le 3 & \text{(c)} \\
4x_2 + x_3 \le 6 & \text{(d)} \\
x_1, x_2, x_3 \text{取0或1}
\end{cases}
\end{aligned}
$$

解 采用隐枚举法, 首先选择可行解 $(x_1, x_2, x_3) = (0, 0, 0)$, 确定过滤条件 $z \ge 0$。求解过程如表3.5所示, 可知最优解为 $\boldsymbol{x}^* = (1, 0, 1)^{\mathrm{T}}$, 最优值为 $z^* = 8$。

表 3.5 隐枚举法计算过程

(x_1, x_2, x_3)	z 值	(a)	(b)	(c)	(d)	过滤条件
(0, 0, 0)	0	✓	✓	✓	✓	$z \ge 0$
(0, 0, 1)	5	✓	✓	✓	✓	$z \ge 5$
(0, 1, 0)	−2					
(0, 1, 1)	3					
(1, 0, 0)	3					
(1, 0, 1)	8	✓	✓	✓	✓	$z \ge 8$
(1, 1, 0)	1					
(1, 1, 1)	6					

对于最大化 0–1 整数规划问题, 通常将变量按目标函数系数从大到小排序, 可更早找到最优解。将上例数学模型重新排列为

$$\max\ z = 5x_3 + 3x_1 - 2x_2$$

$$\text{s.t.}\begin{cases} -x_3 + x_1 + 2x_2 \le 2 & \text{(a)} \\ x_3 + x_1 + 4x_2 \le 4 & \text{(b)} \\ x_1 + x_2 \le 3 & \text{(c)} \\ x_3 + 4x_2 \le 6 & \text{(d)} \\ x_1, x_2, x_3 \text{取0或1} \end{cases}$$

按 (x_3, x_1, x_2) 顺序枚举, 计算过程如表3.6所示。与表3.5相比, 计算量进一步降低。

表 3.6 按目标系数排序后的隐枚举法

(x_3, x_1, x_2)	z 值	(a)	(b)	(c)	(d)	过滤条件
$(0,0,0)$	0	✓	✓	✓	✓	$z \ge 0$
$(1,0,0)$	5	✓	✓	✓	✓	$z \ge 5$
$(1,1,0)$	8	✓	✓	✓	✓	$z \ge 8$

3.4 指派问题

给定 n 个人和 n 件事, 已知第 i 个人做第 j 件事的费用为 c_{ij}, 要求确定人和事之间一一对应的指派方案, 使完成 n 件事的总费用最小, 这就是 0–1 整数规划中的指派问题。引入 n^2 个 0–1 变量

$$x_{ij} = \begin{cases} 1, & \text{指派第 } i \text{ 个人做第 } j \text{ 件事} \\ 0, & \text{不指派第 } i \text{ 个人做第 } j \text{ 件事} \end{cases}$$

定义 3.4

标准指派问题的数学模型为

$$\min\ z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t.}\begin{cases} \sum\limits_{i=1}^{n} x_{ij} = 1,\ j = 1,2,\cdots,n \\ \sum\limits_{j=1}^{n} x_{ij} = 1,\ i = 1,2,\cdots,n \\ x_{ij} \text{取0或1},\ i,j = 1,2,\cdots,n \end{cases}$$

这里, 第一个约束表示每件事必有且只有一个人去做, 第二个约束表示每个人必做且只做一件事。对于指派问题的任一可行解, 均可用解矩阵 $\boldsymbol{X}=(x_{ij})_{n\times n}$ 表示。由约束条件可知, 可行解矩阵中每一行、每一列都恰好有一个元素为 1, 其余为 0。因此, 规模为 n 的指派问题共有 $n!$ 个可行解。

例 3.9 某商业公司计划新建 5 家商店, 拟由 5 家建筑公司分别承建。已知建筑公司承建商店的报价 (单位: 万元) 如表3.7所示。问若仅考虑节省费用, 商业公司应如何分配建造任务, 才能使总的建造费用最低。

表 3.7 各建筑公司承建商店的报价

建筑公司	商店				
	B_1	B_2	B_3	B_4	B_5
A_1	4	8	7	15	12
A_2	7	9	17	14	10
A_3	6	9	12	8	7
A_4	6	7	14	6	10
A_5	6	9	12	10	6

解 这是一个标准的指派问题, 引入 0–1 变量

$$x_{ij}=\begin{cases}1, & \text{指派第 } A_i \text{ 家建筑公司承建商店 } B_j\\ 0, & \text{不指派第 } A_i \text{ 家建筑公司承建商店 } B_j\end{cases}$$

可建立如下数学模型

$$\min\ z=4x_{11}+8x_{12}+\cdots+10x_{54}+6x_{55}$$

$$\text{s.t.}\begin{cases}\sum\limits_{i=1}^{5}x_{ij}=1,\ j=1,2,\cdots,5\\ \sum\limits_{j=1}^{5}x_{ij}=1,\ i=1,2,\cdots,5\\ x_{ij}\text{取0或1},\ i,j=1,2,\cdots,5\end{cases}$$

其系数矩阵为

$$\boldsymbol{C}=(c_{ij})_{5\times5}=\begin{pmatrix}4&8&7&15&12\\7&9&17&14&10\\6&9&12&8&7\\6&7&14&6&10\\6&9&12&10&6\end{pmatrix}$$

一个显然的可行解矩阵为

$$\boldsymbol{X} = (x_{ij})_{5\times 5} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

3.4.1 匈牙利算法

定义 3.5

称位于不同行、不同列的 0 元素为独立 0 元素。

为使系数矩阵 $\boldsymbol{C}$ 中出现 0 元素, 可对矩阵进行等价变换, 在不改变指派问题最优解的前提下, 使矩阵中产生若干 0 元素, 其依据由如下定理给出。

定理 3.6

若从指派问题的系数矩阵 $\boldsymbol{C}$ 的某行或某列各元素中分别减一个常数 k, 则所得新矩阵 $\boldsymbol{C}'$ 与原矩阵具有相同的最优解。

若在新矩阵 $\boldsymbol{C}'$ 中能找到 n 个相互独立的 0 元素, 令这些 0 元素对应位置取 1, 其余取 0, 由此得到指派方案即为新矩阵对应指派问题的最优解, 也是原指派问题的最优解。

定理 3.7

系数矩阵中独立 0 元素的最大数目, 等于能覆盖所有 0 元素的最少直线数。

匈牙利算法由美国数学家哈罗德 · 库恩 (Harold W. Kuhn) 于 1955 年提出。该方法被命名为 “匈牙利法”, 以纪念两位匈牙利数学家的开创性工作。以下给出匈牙利算法的具体计算步骤。

(1) 产生 0 元素: 对系数矩阵的每一行分别减去该行的最小元素, 再对每一列分别减去该列的最小元素, 得到新矩阵 $\boldsymbol{C}'$。注意, 此时矩阵不出现负元素, 且每行每列至少有一个 0 元素。

$$\boldsymbol{C}' = \begin{pmatrix} 0 & 3 & 0 & 11 & 8 \\ 0 & 1 & 7 & 7 & 3 \\ 0 & 2 & 3 & 2 & 1 \\ 0 & 0 & 5 & 0 & 4 \\ 0 & 2 & 3 & 4 & 0 \end{pmatrix}$$

(2) 确定独立 0 元素: 在矩阵 C' 中寻找尽可能多的独立 0 元素。从只有一个 0 元素的行 (或列) 开始, 给这个 0 元素加圈, 记作 ◎。然后划去 ◎ 所在列 (或行) 的其它 0 元素, 记作 ϕ, 直到所有 0 元素都被圈出和划掉为止。若仍出现同行 (列) 至少有两个 0 元素的, 用试探法, 从含有 0 元素最少的行 (列) 开始, 比较该行各 0 元素所在列中 0 元素的数目, 选择 0 元素少的那列的 0 元素加圈, 然后划掉同行同列的其它 0 元素, 可反复进行, 直到所有 0 元素都被圈出和划掉为止。画 ◎ 元素的数目即为独立 0 元素数。若为 n 个, 得到最优解, 若少于 n 个, 则转入下一步。

$$C' \Rightarrow \begin{pmatrix} \phi & 3 & ◎ & 11 & 8 \\ ◎ & 1 & 7 & 7 & 3 \\ \phi & 2 & 3 & 2 & 1 \\ \phi & ◎ & 5 & \phi & 4 \\ \phi & 2 & 3 & 4 & ◎ \end{pmatrix}$$

(3) 最少覆盖原则: 当独立 0 不足 n 个时, 构造覆盖全部 0 元素的最少直线集合。对没有 ◎ 的行打 ✓, 在已打 ✓ 的行中对 ϕ 所在列打 ✓, 随后对打有 ✓ 的列中含 ◎ 元素的行打 ✓。重复上述两步直到找不出新的打 ✓ 的行、列为止。最后对没有打 ✓ 的行画一横线, 有打 ✓ 的列画一纵线, 得到覆盖所有 0 元素的最少直线数, 如左下矩阵所示。进一步, 在未被直线覆盖的元素中找出一个最小元素, 将所有打 ✓ 的行元素减去最小元素, 并将所有打 ✓ 的列元素加上最小元素, 从而在不改变最优性的前提下产生新的 0 元素。执行该变换后得到右下矩阵。

$$\begin{pmatrix} \cdots & \phi & \cdots & 3 & \cdots & ◎ & \cdots & 11 & \cdots & 8 & \cdots & \\ & \vdots & & & & & & & & & & \\ & ◎ & & 1 & & 7 & & 7 & & 3 & & ✓ \\ & \vdots & & & & & & & & & & \\ & \phi & & 2 & & 3 & & 2 & & 1 & & ✓ \\ & \vdots & & & & & & & & & & \\ \cdots & \phi & \cdots & ◎ & \cdots & 5 & \cdots & \phi & \cdots & 4 & \cdots & \\ & \vdots & & & & & & & & & & \\ \cdots & \phi & \cdots & 2 & \cdots & 3 & \cdots & 4 & \cdots & ◎ & \cdots & \\ & \vdots & & & & & & & & & & \\ & ✓ & & & & & & & & & & \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 3 & 0 & 11 & 8 \\ 0 & 0 & 6 & 6 & 2 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 0 & 5 & 0 & 4 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix}$$

(4) 重复 (2)(3) 两步: 重新寻找独立 0 元素。不难发现, 上述矩阵有 5 个独立 0 元素,

得到最优指派矩阵为

$$\boldsymbol{X}=\begin{pmatrix}0&0&1&0&0\\0&1&0&0&0\\1&0&0&0&0\\0&0&0&1&0\\0&0&0&0&1\end{pmatrix}$$

于是, 最优指派方案为 $A_1\to B_3$, $A_2\to B_2$, $A_3\to B_1$, $A_4\to B_4$, $A_5\to B_5$, 总费用为 $7+9+6+6+6=34$ 万元。

例 3.10 有一份中文说明书, 需翻译成英、日、德、俄四种文字, 分别记为 E、J、G、R。现有甲、乙、丙、丁四人, 每人翻译不同语言所需时间 (单位: 小时) 如表3.8所示。问应如何指派翻译任务, 使总翻译时间最少。

表 3.8 每人翻译不同语言所需时间

人员	翻译语言			
	E	J	G	R
甲	2	15	13	4
乙	10	4	14	15
丙	9	14	16	13
丁	7	8	11	9

解 设系数矩阵为 $\boldsymbol{C}=(c_{ij})$, 其中 c_{ij} 表示第 i 人完成第 j 种语言所需时间。

$$\boldsymbol{C}=\begin{pmatrix}2&15&13&4\\10&4&14&15\\9&14&16&13\\7&8&11&9\end{pmatrix}$$

按匈牙利算法, 寻找独立 0 元素, 得到

$$\boldsymbol{C}'=\begin{pmatrix}\phi&13&7&\circledcirc\\6&\circledcirc&6&9\\\circledcirc&5&3&2\\\phi&1&\circledcirc&\phi\end{pmatrix}$$

由于已圈出 4 个独立 0 元素, 直接为最优解。于是, 最优指派方案为甲 $\to$ R, 乙 $\to$ J, 丙 $\to$ E, 丁 $\to$ G, 总时间为 $4+4+9+11=28$。

3.4.2 非标准形式的指派问题

实际生活中的指派问题通常不满足上述标准形式, 此时应先等价转化为标准指派问题, 再用匈牙利算法求解。

(1) 最大化指派问题: 设最大化指派问题系数矩阵 $\boldsymbol{C}=(c_{ij})_{n\times n}$, 其中最大元素为 m。令矩阵

$$\boldsymbol{B}=(b_{ij})_{n\times n}=(m-c_{ij})_{n\times n}$$

则以 $\boldsymbol{B}$ 为系数矩阵的最小化指派问题和以 $\boldsymbol{C}$ 为系数矩阵的原最大化指派问题有相同的最优解。

(2) 人数和事数不等的指派问题: 若人少事多, 则增加一些虚拟的“人”。这些虚拟的“人”做各事的系数可取 0, 理解为这些费用实际上不会发生。若人多事少, 则增加一些虚拟的“事”。这些虚拟的“事”的系数同样也取 0。

(3) 一个人可做几件事的指派问题: 若某个人可做几件事, 则可将该人化作相同的几个“人”来接受指派。这几个“人”做同一件事的系数都一样。

例 3.11 在例3.9中, 为了保证工程质量, 舍弃建筑公司 A_4 和 A_5, 而让技术较强的建筑公司 A_1、A_2、A_3 来承建, 并允许每家公司承建至多 2 家商店。试确定指派方案, 使总费用最少。

解 由于舍弃建筑公司 A_4、A_5, 系数矩阵可写为

$$\boldsymbol{C}=\begin{pmatrix}4&8&7&15&12\\7&9&17&14&10\\6&9&12&8&7\end{pmatrix}$$

又允许建筑公司 A_1、A_2、A_3 承建最多承建 2 家商店, 故将每家公司拆成 2 个虚拟公司, 得到如下系数矩阵

$$\boldsymbol{C}'=\begin{pmatrix}4&8&7&15&12\\4&8&7&15&12\\7&9&17&14&10\\7&9&17&14&10\\6&9&12&8&7\\6&9&12&8&7\end{pmatrix}$$

为了使“人”和“事”的数目相同, 引入虚拟的“事”使之成为标准的指派问题, 系数矩阵转化为

$$C'' = \begin{pmatrix} 4 & 8 & 7 & 15 & 12 & 0 \\ 4 & 8 & 7 & 15 & 12 & 0 \\ 7 & 9 & 17 & 14 & 10 & 0 \\ 7 & 9 & 17 & 14 & 10 & 0 \\ 6 & 9 & 12 & 8 & 7 & 0 \\ 6 & 9 & 12 & 8 & 7 & 0 \end{pmatrix}$$

通过匈牙利算法, 得最优指派方案为由 A_1 承建 B_1 和 B_3, A_2 承建 B_2, A_3 承建 B_4 和 B_5, 总费用为 $4 + 7 + 9 + 8 + 7 = 35$。

3.5 应用案例

3.5.1 问题描述

在人工智能技术快速发展的背景下, 众多中小型 AI 企业依赖云计算平台完成深度学习模型的训练与推理。某云计算服务商计划推出多款固定配置的套餐, 各套餐包含标准化的硬件资源组合, 用户可直接选用, 无需自行配置底层硬件。

例 3.12 某数据中心拥有 GPU 计算卡 100 块、内存 5 000 GB 以及固态硬盘存储 100 TB。运营部门设计了 5 种套餐, 每种套餐对应的硬件资源配置与预估利润 (单位: 元) 如表3.9所示。试确定应推出哪些套餐以及数量, 使总利润最大。

表 3.9 套餐资源配置与利润表

套餐	资源配置			单位利润
	计算卡	内存	存储	
A	1	32	1	25
B	2	64	4	60
C	4	128	8	135
D	8	256	10	250
E	16	512	20	480

3.5.2 模型建立

该问题需要同时决策是否推出某套餐以及推出后部署的数量, 属于典型的混合整数规划问题。引入 0–1 决策变量

$$y_j = \begin{cases} 1, & \text{推出}j\text{套餐} \\ 0, & \text{不推出}j\text{套餐} \end{cases}$$

其中 $j = \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}$。同时设 x_j 表示 j 套餐的数量, 应满足 $x_j \geq 0$且取整数。由于各类套餐在运行时将占用计算卡、内存与存储等资源, 且其总消耗量不得超过数据中心的可用资源上限, 故有如下约束

$$\begin{cases} x_A + 2x_B + 4x_C + 8x_D + 16x_E \leq 100 \\ 32x_A + 64x_B + 128x_C + 256x_D + 512x_E \leq 5\,000 \\ x_A + 4x_B + 8x_C + 10x_D + 20x_E \leq 100 \end{cases}$$

引入上界 M_j 刻画 x_j 与 y_j 之间的逻辑关系, 以套餐 A 为例, 有

$$M_A = \min\left(\frac{100}{1}, \frac{5\,000}{32}, \frac{100}{1}\right) = 100$$

同理可得

$$M_B = \min\left(\frac{100}{2}, \frac{5\,000}{64}, \frac{100}{4}\right) = 25,\ M_C = \min\left(\frac{100}{4}, \frac{5\,000}{128}, \frac{100}{8}\right) = 12$$

$$M_D = \min\left(\frac{100}{8}, \frac{5\,000}{256}, \frac{100}{10}\right) = 10,\ M_E = \min\left(\frac{100}{16}, \frac{5\,000}{512}, \frac{100}{20}\right) = 5$$

由此建立逻辑约束 $x_j \leq M_j y_j$。该约束保证当 $y_j = 0$ 时, $x_j = 0$。当 $y_j = 1$ 时, x_j 不超过其可能的最大规模。以总利润最大化为目标, 建立如下混合整数规划模型

$$\max\ z = 25x_A + 60x_B + 135x_C + 250x_D + 480x_E$$

$$\text{s.t.} \begin{cases} x_A + 2x_B + 4x_C + 8x_D + 16x_E \leq 100 \\ 32x_A + 64x_B + 128x_C + 256x_D + 512x_E \leq 5\,000 \\ x_A + 4x_B + 8x_C + 10x_D + 20x_E \leq 100 \\ x_A \leq 100y_A \\ x_B \leq 25y_B \\ x_C \leq 12y_C \\ x_D \leq 10y_D \\ x_E \leq 5y_E \\ x_j \geq 0\text{且为整数},\ j = \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E} \\ y_j\text{取0或1},\ j = \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E} \end{cases}$$

3.5.3 算法设计

上述模型可利用优化求解器 Gurobi 直接求解 (详见第 7 章), 得到最优解为

$$x_A^* = 0,\ x_B^* = 25,\ x_C^* = 12,\ x_D^* = 0,\ x_E = 0$$

$$y_A^* = 0,\ y_B^* = 1,\ y_C^* = 1,\ y_D^* = 0,\ y_E = 0$$

对应的最优值为 $z^* = 60 \times 25 + 135 \times 12 = 3\,120$。

3.5.4 结果分析

按上述解, 应推出套餐 B 与套餐 C, 分别部署 25 个与 12 个, 其余套餐不提供。对资源占用情况进行检验, 计算卡资源为 2×25+4×12 = 98, 内存资源为 64×25+128×12 = 3 136, 均满足数据中心的资源约束。然而, 存储资源的计算为 $4 \times 25 + 8 \times 12 = 196$, 超过了可用的 100 TB。因此该解不满足存储资源约束, 并非可行解。

需要指出的是, 在构造逻辑关联约束时所引入的上界参数 M_j, 是基于单一套餐在资源完全独占条件下所能达到的最大规模。当多种套餐同时部署时, 各类资源为共享占用, 必须同时满足全部资源约束。约束 $x_j \le M_j y_j$ 仅用于刻画变量之间的逻辑关系, 保证当 $y_j = 0$ 时 $x_j = 0$, 并限制 x_j 的取值范围, 但并不能替代原有的资源容量约束。那么, 应如何修正上述混合整数规划模型? 请读者思考。

习题

3.1 试用四舍五入法求解下述整数线性规划问题。

$$\max\ z = x_1 + 4x_2$$

$$\text{s.t.} \begin{cases} -2x_1 + 3x_2 \le 3 \\ x_1 + 2x_2 \le 8 \\ x_1, x_2 \ge 0 \text{且取整数} \end{cases}$$

3.2 用分枝界定法求解下列整数规划问题。

(1) $\max\ z = x_1 + x_2$

$$\text{s.t.} \begin{cases} x_1 + \dfrac{9}{14}x_2 \le \dfrac{51}{14} \\ -2x_1 + x_2 \le \dfrac{1}{3} \\ x_1, x_2 \ge 0 \text{且取整数} \end{cases}$$

(2) $\max\ z = 2x_1 + 3x_2$

$$\text{s.t.} \begin{cases} 5x_1 + 7x_2 \le 35 \\ 4x_1 + 9x_2 \le 36 \\ x_1, x_2 \ge 0 \text{且取整数} \end{cases}$$

3.3 用隐枚举法求解下列 0–1 整数规划问题。

$$\min \ z = 5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5$$

$$\text{s.t.} \begin{cases} x_1 - 3x_2 + 5x_3 + x_4 - 4x_5 \geq 2 \\ -2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0 \\ -2x_2 + 2x_3 - x_4 - x_5 \geq 1 \\ x_1, x_2, x_3, x_4, x_5 \text{取0或1} \end{cases}$$

3.4 已知指派问题的系数矩阵, 用匈牙利算法确定最优指派方案。

$$(1) \begin{pmatrix} 7 & 9 & 10 & 12 \\ 13 & 12 & 16 & 17 \\ 15 & 16 & 14 & 15 \\ 11 & 12 & 15 & 16 \end{pmatrix} \qquad (2) \begin{pmatrix} 3 & 6 & 2 & 6 \\ 7 & 1 & 4 & 4 \\ 3 & 8 & 5 & 6 \\ 6 & 4 & 3 & 7 \\ 5 & 2 & 4 & 3 \\ 5 & 7 & 6 & 2 \end{pmatrix}$$

3.5 已知 A、B、C、D、E 共 5 名运动员在不同泳姿下的 50 米游泳成绩 (单位: 秒) 如表3.10所示。试从中选拔队员组成 200 米混合泳接力队, 使预计比赛成绩最优。

表 3.10 运动员游泳姿势和成绩

姿势	运动员				
	A	B	C	D	E
仰泳	37.7	32.9	33.8	37.0	35.4
蛙泳	43.4	33.1	42.2	34.7	41.8
蝶泳	33.3	28.5	38.9	30.4	33.6
自由泳	29.2	26.4	29.6	28.5	31.1

4

非线性规划

4.1 基本概念

在现实世界的许多优化问题中，目标函数或约束条件往往呈现出非线性特征，这类问题无法用线性规划模型描述。以最小化形式的非线性规划问题为例，其数学模型为

$$\begin{aligned} &\min \ f(\boldsymbol{x}) \\ &\text{s.t.} \begin{cases} g_i(\boldsymbol{x}) \geq 0, \ i = 1, 2, \cdots, m \\ h_j(\boldsymbol{x}) = 0, \ j = m+1, m+2, \cdots, m+l \end{cases} \end{aligned} \tag{4.1}$$

其中 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ 为决策向量，$f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ 为目标函数，$g_i(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ 为不等式约束函数，$h_j(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ 为等式约束函数。对于迭代算法，本章以 $\boldsymbol{x}^{(0)}$ 表示初始点，由算法生成的迭代点序列为 $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(k)}$，简记为 $\{\boldsymbol{x}^{(k)}\}$。

4.1.1 最优解

与线性规划不同，非线性规划问题可能存在多个最优解。根据最优解的性质，可将其划分为局部最优解、严格局部最优解、全局最优解和严格全局最优解。

定义 4.1

给定 $\boldsymbol{x}^* \in \mathbb{R}^n$，若存在 $\varepsilon > 0$，使得对任意满足 $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \varepsilon$ 的 $\boldsymbol{x}$ 都有 $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的局部最优解，$f(\boldsymbol{x}^*)$ 为对应的局部最优值。进一步，若对任意满足 $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \varepsilon$ 且 $\boldsymbol{x} \neq \boldsymbol{x}^*$ 的 $\boldsymbol{x}$ 都有 $f(\boldsymbol{x}) > f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的严格局部最优解，$f(\boldsymbol{x}^*)$ 为对应的严格局部最优值。

针对图4.1中所示的一维函数, x, y, z 均为局部最优解, 其中 x, y 为严格局部最优解。

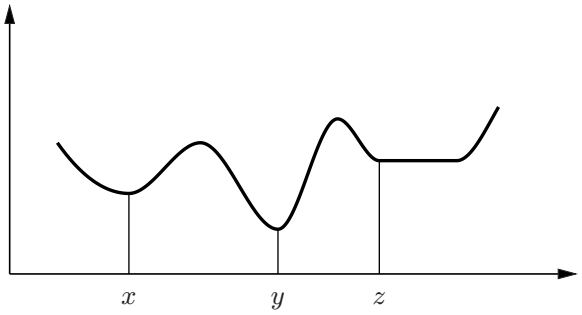

图 4.1 非线性规划最优解

定义 4.2

给定 $\boldsymbol{x}^* \in \mathbb{R}^n$, 若对任意 $\boldsymbol{x}$ 都有 $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$, 则称 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的全局最优解, $f(\boldsymbol{x}^*)$ 为对应的全局最优值。进一步, 若对任意 $\boldsymbol{x} \neq \boldsymbol{x}^*$ 都有 $f(\boldsymbol{x}) > f(\boldsymbol{x}^*)$, 则称 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的严格全局最优解, $f(\boldsymbol{x}^*)$ 为对应的严格全局最优值。

结合图4.1可知, y 不仅是全局最优解, 还是严格全局最优解。对于最大化形式的非线性规划问题, 仅需将定义中的不等号反向, 即可得到其对应最优解的概念。

考虑二次型函数

$$\begin{aligned} f(\boldsymbol{x}) &= a_{11}x_1^2 + 2a_{12}x_1x_2 + \cdots + 2a_{1n}x_1x_n + a_{22}x_2^2 + 2a_{23}x_2x_3 + \cdots + a_{nn}x_n^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j \\ &= \boldsymbol{x}^{\mathrm{T}}A\boldsymbol{x} \end{aligned}$$

其中 $\boldsymbol{A}$ 为 $n \times n$ 对称矩阵, 即 $a_{ij} = a_{ji}$。显然, 一个二次型唯一确定一个对称矩阵 $\boldsymbol{A}$。反之, 一个对称矩阵 $\boldsymbol{A}$ 也唯一确定一个二次型。

定义 4.3

设 $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$ 为实二次型, 定义

(1) 正定二次型: 若对任意 $\boldsymbol{x} \neq \boldsymbol{0}$, 恒有 $f(\boldsymbol{x}) > 0$。

(2) 负定二次型: 若对任意 $\boldsymbol{x} \neq \boldsymbol{0}$, 恒有 $f(\boldsymbol{x}) < 0$。

(3) 半正定二次型: 若对任意 $\boldsymbol{x} \neq \boldsymbol{0}$, 恒有 $f(\boldsymbol{x}) \geq 0$, 且对某些 $\boldsymbol{x} \neq \boldsymbol{0}$ 有 $f(\boldsymbol{x}) = 0$。

(4) 半负定二次型: 若对任意 $\boldsymbol{x} \neq \boldsymbol{0}$, 恒有 $f(\boldsymbol{x}) \leq 0$, 且对某些 $\boldsymbol{x} \neq \boldsymbol{0}$ 有 $f(\boldsymbol{x}) = 0$。

若二次型 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$ 分别为正定、负定、半正定、半负定, 则其对应的对称矩阵 $\boldsymbol{A}$ 称

为正定矩阵、负定矩阵、半正定矩阵、半负定矩阵。此外，根据线性代数理论，实二次型 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$ 正定的充分必要条件为其矩阵 $\boldsymbol{A}$ 的各阶顺序主子式均大于零。

定义 4.4

设 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}^{(0)}$ 处二阶连续可微，则二阶 Taylor 展开式为

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^{(0)}) + \nabla f(\boldsymbol{x}^{(0)})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{(0)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^{\mathrm{T}}\nabla^2 f(\hat{\boldsymbol{x}})(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

其中 $\hat{\boldsymbol{x}} = \boldsymbol{x}^{(0)} + \alpha(\boldsymbol{x} - \boldsymbol{x}^{(0)}), \alpha \in (0, 1)$。

将 $\hat{\boldsymbol{x}}$ 代入，上式可转化为

$$\begin{aligned} f(\boldsymbol{x}) = {} & f(\boldsymbol{x}^{(0)}) + \nabla f(\boldsymbol{x}^{(0)})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{(0)}) \\ & + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^{(0)})(\boldsymbol{x} - \boldsymbol{x}^{(0)}) + o(\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|^2) \end{aligned}$$

当 $\boldsymbol{x} \to \boldsymbol{x}^{(0)}$ 时，可知 $o(\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|^2)$ 是 $\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|^2$ 的高阶无穷小，即

$$\lim_{\boldsymbol{x} \to \boldsymbol{x}^{(0)}} \frac{o(\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|^2)}{\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|^2} = 0$$

4.1.2 最优性理论

定理 4.5

设 $f(\boldsymbol{x})$ 可微，若 $\boldsymbol{x}^*$ 为局部最优解，则必有

$$\nabla f(\boldsymbol{x}^*) = \left(\frac{\partial f(\boldsymbol{x}^*)}{\partial x_1}, \frac{\partial f(\boldsymbol{x}^*)}{\partial x_2}, \cdots, \frac{\partial f(\boldsymbol{x}^*)}{\partial x_n}\right)^{\mathrm{T}} = 0$$

此时称 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的稳定点或驻点。

证明 因 $f(\boldsymbol{x})$ 可微且 $\boldsymbol{x}^*$ 为其局部最优解，对任意方向 $\boldsymbol{d} \in \mathbb{R}^n$ 及充分小的 $\lambda > 0$，由一阶 Taylor 展开式可得

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{x}^* + \lambda\boldsymbol{d}) \\ &= f(\boldsymbol{x}^*) + \lambda\nabla f(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d} + o(\lambda\|\boldsymbol{d}\|) \\ &\geq f(\boldsymbol{x}^*) \end{aligned}$$

不等式两边同时减去 $f(\boldsymbol{x}^*)$，整理得

$$\nabla f(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d} + \frac{o(\lambda\|\boldsymbol{d}\|)}{\lambda} \geq 0$$

当 $\lambda \to 0$ 时, 有 $\nabla f(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d} \geq 0$。又因该式对任意 $\boldsymbol{d}$ 成立, 可推得 $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$。

定理4.5称为无约束优化问题的一阶必要条件。容易验证, 函数 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}$ 处的梯度 $\nabla f(\boldsymbol{x})$ 必与函数过该点的等值面正交。梯度方向为函数值上升最快的方向, 负梯度方向为函数值下降最快的方向。

定理 4.6

设 $f(\boldsymbol{x})$ 二阶连续可微, 若 $\nabla f(\boldsymbol{x}^*) = 0$ 且 Hessian 矩阵

$$\nabla^2 f(\boldsymbol{x}^*) = \begin{pmatrix} \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1^2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1 \partial x_3} & \cdots & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2^2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2 \partial x_3} & \cdots & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n \partial x_2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n \partial x_3} & \cdots & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n^2} \end{pmatrix}$$

正定, 则 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的严格局部最优解。

证明 先证 $\boldsymbol{x}^*$ 为局部最优解, 采用反证法。设 $\boldsymbol{x}^*$ 不是局部最优解, 则存在点列 $\{\boldsymbol{x}^{(k)}\} \to \boldsymbol{x}^*$, 使得

$$f(\boldsymbol{x}^{(k)}) < f(\boldsymbol{x}^*)$$

考虑二阶 Taylor 展开, 有

$$\begin{aligned} f(\boldsymbol{x}^{(k)}) &= f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*)^{\mathrm{T}}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) \\ &\quad + \frac{1}{2}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^*)(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) + o(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2) \\ &< f(\boldsymbol{x}^*) \end{aligned}$$

记 $\boldsymbol{d}^{(k)} = \dfrac{\boldsymbol{x}^{(k)} - \boldsymbol{x}^*}{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|}$, 上式整理得

$$(\boldsymbol{d}^{(k)})^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d}^{(k)} + \frac{2o(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2)}{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2} < 0$$

因 $\|\boldsymbol{d}^{(k)}\| = 1$, 故存在收敛子列 $\{\boldsymbol{d}^{(k_j)}\} \to \boldsymbol{d}$。对上述不等式沿 $\{k_j\}$ 取极限, 得

$$\boldsymbol{d}^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d} \leq 0$$

与 $\nabla^2 f(\boldsymbol{x}^*)$ 正定相矛盾, 因此 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的局部最优解。

接下来证明 $\boldsymbol{x}^*$ 为严格局部最优解, 仍采用反证法。设 $\boldsymbol{x}^*$ 不是严格局部极小点, 则存在点列 $\{\boldsymbol{x}^{(k)}\} \to \boldsymbol{x}^*$, 使得

$$f(\boldsymbol{x}^{(k)}) = f(\boldsymbol{x}^*)$$

由定理4.5可知, $\nabla f(\boldsymbol{x}^{(k)}) = \boldsymbol{0}$。考虑梯度函数的一阶 Taylor 展开, 有

$$\nabla f(\boldsymbol{x}^{(k)}) = \nabla f(\boldsymbol{x}^*) + \nabla^2 f(\boldsymbol{x}^*)(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) + o(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|)$$

注意, $\nabla f(\boldsymbol{x}^*)$ 与 $\nabla f(\boldsymbol{x}^{(k)})$ 均为 $\boldsymbol{0}$。令 $\boldsymbol{d}^{(k)} = \dfrac{\boldsymbol{x}^{(k)} - \boldsymbol{x}^*}{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|}$, 上式整理得

$$\nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d}^{(k)} + \frac{o(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|)}{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|} = \boldsymbol{0}$$

同理, 存在收敛子列 $\{\boldsymbol{d}^{(k_j)}\} \to \boldsymbol{d}$, 取极限得

$$\nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d} = \boldsymbol{0}$$

与 $\nabla^2 f(\boldsymbol{x}^*)$ 正定矛盾, 故 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的严格局部最优解。

定理 4.7

设 $f(\boldsymbol{x})$ 二阶连续可微, 若 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的局部最优解, 则 $\nabla f(\boldsymbol{x}^*) = 0$, 且 Hessian 矩阵

$$\nabla^2 f(\boldsymbol{x}^*) = \begin{pmatrix} \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1^2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1 \partial x_3} & \cdots & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2^2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2 \partial x_3} & \cdots & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n \partial x_2} & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n \partial x_3} & \cdots & \dfrac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_n^2} \end{pmatrix}$$

半正定。

证明 由定理4.5可知, $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$。考虑二阶 Taylor 展开, 有

$$\begin{aligned} f(\boldsymbol{x}^* + \lambda \boldsymbol{d}) &= f(\boldsymbol{x}^*) + \frac{1}{2}\lambda^2 \boldsymbol{d}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d} + o(\lambda^2 \|\boldsymbol{d}\|^2) \\ &\geq f(\boldsymbol{x}^*) \end{aligned}$$

上式整理得

$$\boldsymbol{d}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d} + \frac{2o(\lambda^2 \|\boldsymbol{d}\|^2)}{\lambda^2} \geq 0$$

令 $\lambda \to 0$, 则

$$\boldsymbol{d}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}^*) \boldsymbol{d} \geq 0$$

即 Hessian 矩阵 $\nabla^2 f(\boldsymbol{x}^*)$ 半正定。

例 4.1 判断函数 $f(\boldsymbol{x}) = x_1^2 - x_2^2$ 是否存在最优解。

解 根据无约束优化问题的一阶必要性定理4.5, 函数的梯度为

$$\frac{\partial f(\boldsymbol{x})}{\partial x_1} = 2x_1, \quad \frac{\partial f(\boldsymbol{x})}{\partial x_2} = -2x_2$$

令 $f(\boldsymbol{x}) = 0$, 即 $2x_1 = 0$, $-2x_2 = 0$, 解得稳定点 $\boldsymbol{x} = (x_1, x_2)^{\mathrm{T}} = (0, 0)^{\mathrm{T}}$。

利用二阶充分性定理4.6判断该稳定点是否为极值点。Hessian 矩阵为

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

该 Hessian 矩阵为不定矩阵, 因此 $\boldsymbol{x} = (0, 0)^{\mathrm{T}}$ 不是最优解。

4.1.3 下降迭代算法

对于大多数非线性规划问题, 其解析解往往难以直接求得。通常采用下降迭代算法, 从初始点 $\boldsymbol{x}^{(0)}$ 出发, 依照既定规则, 先找一个比 $\boldsymbol{x}^{(0)}$ 更好的点 $\boldsymbol{x}^{(1)}$, 再找比 $\boldsymbol{x}^{(1)}$ 更好的点 $\boldsymbol{x}^{(2)}$, 如此继续, 生成迭代序列 $\{\boldsymbol{x}^{(k)}\}$。

定义 4.8

若迭代序列 $\{\boldsymbol{x}^{(k)}\}$ 存在极限点 $\boldsymbol{x}^*$, 即

$$\lim_{k \to \infty} \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\| = 0$$

则称序列 $\{\boldsymbol{x}^{(k)}\}$ 收敛于 $\boldsymbol{x}^*$。

定义 4.9

对于最小化问题, 若迭代序列 $\{\boldsymbol{x}^{(k)}\}$ 对应的目标函数值 $\{f(\boldsymbol{x}^{(k)})\}$ 严格单调递减, 即

$$f(\boldsymbol{x}^{(0)}) > f(\boldsymbol{x}^{(1)}) > \cdots > f(\boldsymbol{x}^{(k)}) > \cdots$$

则将具备该性质的迭代算法称为下降迭代算法。

下降迭代算法的一般迭代步骤如下。

(1) 选取初始点: 选定初始迭代点 $\boldsymbol{x}^{(0)}$, 令 $k=0$。

(2) 确定搜索方向: 若当前迭代 $\boldsymbol{x}^{(k)}$ 并非问题的最优解, 则从 $\boldsymbol{x}^{(k)}$ 出发确定搜索方向 $\boldsymbol{p}^{(k)}$, 要求沿该方向能够找到使目标函数值下降的点。对于约束优化问题, 根据所选用算法的不同, 还需保证新迭代点为可行点。

(3) 确定步长: 沿搜索方向 $\boldsymbol{p}^{(k)}$ 迭代, 得到 $\boldsymbol{x}^{(k+1)}$。即在射线 $\boldsymbol{x}=\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)}$ $(\lambda\geq 0)$ 上, 选取步长 λ_k, 确定下一个迭代点

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\lambda_k\boldsymbol{p}^{(k)}$$

且满足函数值下降条件

$$f(\boldsymbol{x}^{(k+1)})=f(\boldsymbol{x}^{(k)}+\lambda_k\boldsymbol{p}^{(k)})<f(\boldsymbol{x}^{(k)})$$

(4) 判断终止准则: 若迭代点 $\boldsymbol{x}^{(k+1)}$ 满足终止准则, 则迭代停止。否则, 返回步骤 (2) 继续迭代。

在下降迭代算法中, 搜索方向 $\boldsymbol{p}^{(k)}$ 和步长 λ_k 的选择至关重要, 不同的选择策略对应不同的优化算法, 如梯度法、牛顿法、共轭梯度法等。为提升算法的收敛速度, 通常选取步长 λ_k 使得目标函数沿搜索方向的下降最大, 即求解如下一维优化问题

$$\lambda_k=\arg\min_{\lambda} f(\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)})$$

该过程称为最优一维搜索或线搜索, 由此确定的步长称为最优步长。

定理 4.10

设 $f(\boldsymbol{x})$ 连续可微, 迭代点 $\boldsymbol{x}^{(k+1)}$ 由以下规则生成

$$\begin{cases}\lambda_k=\arg\min\limits_{\lambda} f(\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)})\\ \boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\lambda_k\boldsymbol{p}^{(k)}\end{cases}$$

则有

$$\nabla f(\boldsymbol{x}^{(k+1)})^{\mathrm{T}}\boldsymbol{p}^{(k)}=0$$

即采用最优步长迭代时, 搜索方向 $\boldsymbol{p}^{(k)}$ 与目标函数的梯度方向正交。

由于迭代不可能无限进行, 需要设定合适的终止准则来判断当前迭代点是否已足够接近最优解, 常见的有以下三种。

(1) 绝对误差准则: 当相邻两次迭代点或目标函数值的距离变化小于预设精度时, 迭代停止, 即

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| \leq \varepsilon_1, \ |f(\boldsymbol{x}^{(k+1)}) - f(\boldsymbol{x}^{(k)})| \leq \varepsilon_2$$

(2) 相对误差准则: 当相邻两次迭代点或目标函数值的相对变化小于预设精度时, 迭代停止, 即

$$\frac{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^{(k)}\|} \leq \varepsilon_3, \ \frac{|f(\boldsymbol{x}^{(k+1)}) - f(\boldsymbol{x}^{(k)})|}{|f(\boldsymbol{x}^{(k)})|} \leq \varepsilon_4$$

(3) 梯度准则: 当目标函数梯度的范数小于预设精度时, 迭代停止, 即

$$\|\nabla f(\boldsymbol{x}^{(k)})\| \leq \varepsilon_5$$

其中 $\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_5 > 0$ 为允许误差, 代表求解精度。

下降迭代算法的收敛性与收敛速度, 由目标函数的性质以及搜索方向和步长选取的策略共同决定。后续章节将深入探讨几种经典的下降算法及其性能。

4.2 凸优化介绍

4.2.1 凸函数

定义 4.11

设 $f(\boldsymbol{x})$ 为凸集 Ω 上的函数, 若对任意两点 $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$ 以及任意 $\alpha \in (0,1)$, 有

$$f(\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2) \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

则称 $f(\boldsymbol{x})$ 为定义在 Ω 上的凸函数。

定义 4.12

若对任意两点 $\boldsymbol{x}_1 \neq \boldsymbol{x}_2 \in \Omega$ 以及任意 $\alpha \in (0,1)$, 恒有

$$f(\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2) < \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

则称 $f(\boldsymbol{x})$ 为定义在 Ω 的严格凸函数。

若 $-f(\boldsymbol{x})$ 为凸函数或严格凸函数, 则称 $f(\boldsymbol{x})$ 为凹函数或严格凹函数。根据定义可判断, 图4.2中所示的一维函数, (a) 为凸函数, (b) 为凹函数。从几何视角来看, 凸函数的图像位于其定义域内任意两点所连线的下方。

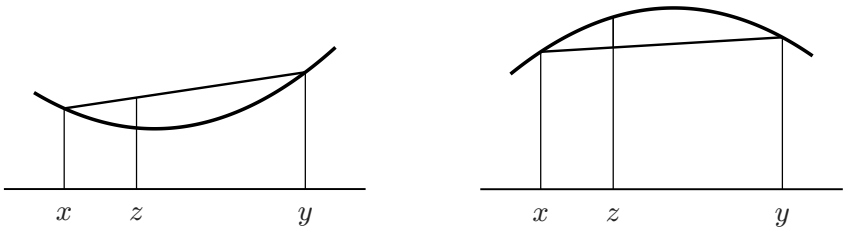

图 4.2 凸函数几何意义

性质 4.13

设 $f(\boldsymbol{x})$ 为凸集 Ω 上的凸函数, 则对任意 $\beta \geq 0$, 函数 $\beta f(\boldsymbol{x})$ 仍为 Ω 上的凸函数。

性质 4.14

设 $f_1(\boldsymbol{x})$ 和 $f_2(\boldsymbol{x})$ 均为凸集 Ω 上的凸函数, 则其和函数 $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$ 仍为 Ω 上的凸函数。

4.2.2 凸函数判定定理

定理 4.15

设 $f(\boldsymbol{x})$ 为凸集 Ω 上的一阶可微函数, 则 $f(\boldsymbol{x})$ 为 Ω 上的凸函数的充分必要条件是, 对任意 $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$, 恒有

$$f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1)$$

证明 根据凸函数的定义, 对任意 $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$ 以及任意 $\alpha \in (0, 1)$, 有

$$f(\alpha \boldsymbol{x}_2 + (1-\alpha)\boldsymbol{x}_1) \leq \alpha f(\boldsymbol{x}_2) + (1-\alpha) f(\boldsymbol{x}_1) \tag{4.2}$$

将左端函数进行一阶 Taylor 展开, 得到

$$\begin{aligned} f(\alpha \boldsymbol{x}_2 + (1-\alpha)\boldsymbol{x}_1) &= f(\boldsymbol{x}_1 + \alpha(\boldsymbol{x}_2 - \boldsymbol{x}_1)) \\ &= f(\boldsymbol{x}_1) + \alpha \nabla f(\boldsymbol{x}_1)^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1) + o(\alpha \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|) \end{aligned} \tag{4.3}$$

将式(4.3)代入式(4.2), 移项化简后两边同除以 α, 可知

$$f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \frac{o(\alpha \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|)}{\alpha}$$

令 $\alpha \to 0$, 即得

$$f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1)$$

另一方面, 对任意 $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$ 以及 $\alpha \in (0,1)$, 记

$$\boldsymbol{x} = \alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2$$

由条件得到

$$\begin{aligned} f(\boldsymbol{x}_1) &\geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x}_1 - \boldsymbol{x}) \\ &= f(\boldsymbol{x}) + (1-\alpha)\nabla f(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) \end{aligned} \tag{4.4}$$

以及

$$f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}) + \alpha \nabla f(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1) \tag{4.5}$$

将式(4.4)乘以 α、式(4.5)乘以 $(1-\alpha)$ 后相加, 得到

$$f(\boldsymbol{x}) \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

由定义可知, $f(\boldsymbol{x})$ 为凸函数。

该定理表明, 凸函数在其定义域内任意一点处的切平面位于函数图像的下方, 即该切平面为函数图像的支撑超平面。

定理 4.16

设 $f(\boldsymbol{x})$ 为凸集 Ω 上的二阶可微函数, 则 $f(\boldsymbol{x})$ 为 Ω 上的凸函数的充分必要条件是, 对任意 $\boldsymbol{x} \in \Omega$, 恒有 $\nabla^2 f(\boldsymbol{x})$ 半正定。

证明 根据定理4.15, 对任意 $\boldsymbol{x} \in \Omega$、方向 $\boldsymbol{d} \in \mathbb{R}^n$ 以及任意 $\alpha \in (0,1)$, 有

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) \geq f(\boldsymbol{x}) + \alpha \nabla f(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{d} \tag{4.6}$$

将左端函数作二阶 Taylor 展开, 得到

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) = f(\boldsymbol{x}) + \alpha \nabla f(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{d} + \frac{1}{2}\alpha^2 \boldsymbol{d}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}) \boldsymbol{d} + o(\alpha^2 \|\boldsymbol{d}\|^2) \tag{4.7}$$

比较式(4.6)和(4.7)并整理得

$$\boldsymbol{d}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}) \boldsymbol{d} \geq \frac{2o(\alpha^2 \|\boldsymbol{d}\|^2)}{\alpha^2}$$

令 $\alpha \to 0$, 右端趋于 0, 因此 $\boldsymbol{d}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}) \boldsymbol{d} \geq 0$, 即 Hessian 矩阵 $\nabla^2 f(\boldsymbol{x})$ 半正定。

另一方面, 任取 $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$, 由二阶中值定理, 存在 $\alpha \in (0,1)$, 记

$$\boldsymbol{x} = \boldsymbol{x}_1 + \alpha(\boldsymbol{x}_2 - \boldsymbol{x}_1) = \alpha \boldsymbol{x}_2 + (1-\alpha)\boldsymbol{x}_1$$

使得

$$f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \frac{1}{2}(\boldsymbol{x}_2 - \boldsymbol{x}_1)^{\mathrm{T}}\nabla^2 f(\boldsymbol{x})(\boldsymbol{x}_2 - \boldsymbol{x}_1)$$

由于 $\nabla^2 f(\boldsymbol{x})$ 半正定, 有

$$\frac{1}{2}(\boldsymbol{x}_2 - \boldsymbol{x}_1)^{\mathrm{T}}\nabla^2 f(\boldsymbol{x})(\boldsymbol{x}_2 - \boldsymbol{x}_1) \geq 0$$

从而

$$f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^{\mathrm{T}}(\boldsymbol{x}_2 - \boldsymbol{x}_1)$$

上式表明定理4.15成立, 因此 $f(x)$ 为凸函数。

进一步, 若对任意 $\boldsymbol{x} \in \Omega$ 恒有 $\nabla^2 f(\boldsymbol{x}) > 0$, 即 Hessian 矩阵正定, 则 $f(\boldsymbol{x})$ 为凸集 Ω 上的严格凸函数。此处证明略, 读者可参考上述证明自行推导。

例 4.2 试证明函数 $f(\boldsymbol{x}) = x_1^2 + x_2^2$ 为严格凸函数。

解 任取 $\boldsymbol{x}_1 = (a_1, b_1)^{\mathrm{T}}$, $\boldsymbol{x}_2 = (a_2, b_2)^{\mathrm{T}}$, 有

$$f(\boldsymbol{x}_1) = a_1^2 + b_1^2, \ f(\boldsymbol{x}_2) = a_2^2 + b_2^2$$

计算得到梯度为

$$\nabla f(\boldsymbol{x}_1) = (2a_1, 2b_1)^{\mathrm{T}}$$

将上述表达式代入严格凸函数的一阶条件, 需验证下式成立

$$a_2^2 + b_2^2 > a_1^2 + b_1^2 + (2a_1, 2b_1)\begin{pmatrix} a_2 - a_1 \\ b_2 - b_1 \end{pmatrix}$$

整理化简后, 等价于

$$(a_2 - a_1)^2 + (b_2 - b_1)^2 > 0$$

由于 $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, 上式恒成立, 因此 $f(\boldsymbol{x})$ 为严格凸函数。

接下来尝试用利用二阶条件证明。梯度为

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x_1}} = 2x_1, \ \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x_2}} = 2x_2$$

进一步, 计算有

$$\frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x_1^2}} = 2, \ \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x_2^2}} = 2, \ \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x_1}\partial \boldsymbol{x_2}} = \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x_2}\partial \boldsymbol{x_1}} = 0$$

由此, 可得函数的 Hessian 矩阵为

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

因 Hessian 矩阵 $\nabla^2 f(\boldsymbol{x})$ 正定, 根据二阶条件判定, $f(\boldsymbol{x})$ 为严格凸函数。

性质 4.17

设 $f(\boldsymbol{x})$ 为凸集 Ω 上的可微凸函数, 若存在 $\boldsymbol{x}^* \in \Omega$, 使得对任意 $\boldsymbol{x} \in \Omega$, 都有

$$\nabla f(\boldsymbol{x}^*)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0$$

则 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的全局最优解。

证明 采用反证法。设 $f(\boldsymbol{x})$ 为凸函数, 且 $\boldsymbol{x}^*$ 不是全局最优解, 则存在 $\boldsymbol{x} \in \Omega$, 使 $f(\boldsymbol{x}) < f(\boldsymbol{x}^*)$。对于任意 $\alpha \in (0, 1)$, 根据凸函数的定义可得

$$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{x}^*) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{x}^*) < f(\boldsymbol{x}^*)$$

当 $\alpha \to 0$ 时, 有 $\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{x}^* \to \boldsymbol{x}^*$。该结论与 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 的局部最优解相矛盾, 因此 $\boldsymbol{x}^*$ 必为 $f(\boldsymbol{x})$ 的全局最优解。

通过图4.3不难看出, $\nabla f(\boldsymbol{x}^*)$ 与 $\boldsymbol{x} - \boldsymbol{x}^*$ 的夹角为锐角。

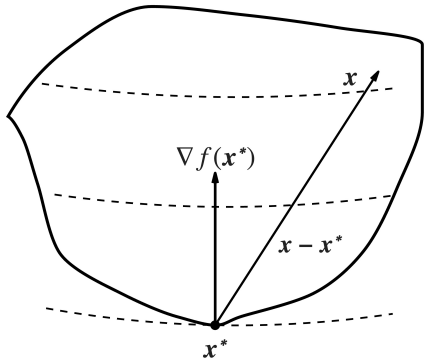

图 4.3 凸函数性质

4.2.3 凸规划

对于非线性规划问题(4.1), 若目标函数 $f(\boldsymbol{x})$ 为凸函数, 不等式约束函数 $g_i(\boldsymbol{x}) \geq 0$ 为凹函数, 等式约束函数 $h_j(\boldsymbol{x}) = 0$ 为线性函数, 则称该规划问题为凸规划。由上一节可知, 若存在可行解, 则可行域为凸集。若存在最优解, 则最优解集为凸集。

性质 4.18

凸规划问题的任意局部最优解必为全局最优解。进一步，若目标函数为严格凸函数且最优解存在，则最优解必唯一。

例 4.3 试判断下述非线性规划问题为凸规划

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = x_1^2 + x_2^2 - 4x_1 + 4 \\ \text{s.t.} \quad & \begin{cases} g_1(\boldsymbol{x}) = x_1 - x_2 + 2 \geq 0 \\ g_2(\boldsymbol{x}) = -x_1^2 + x_2 - 1 \geq 0 \\ g_3(\boldsymbol{x}) = x_1 \geq 0 \\ g_4(\boldsymbol{x}) = x_2 \geq 0 \end{cases} \end{aligned}$$

解 首先分析目标函数，其 Hessian 矩阵为

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

该矩阵正定，因此 $f(\boldsymbol{x})$ 为严格凸函数。其次分析约束函数，$g_1(\boldsymbol{x}) = x_1 - x_2 + 2$、$g_3(\boldsymbol{x}) = x_1$ 与 $g_4(\boldsymbol{x}) = x_2$ 均为线性函数，既是凸函数也是凹函数。对于 $g_2(\boldsymbol{x}) = -x_1^2 + x_2 - 1$，其 Hessian 矩阵为

$$\nabla^2 g_2(\boldsymbol{x}) = \begin{pmatrix} -2 & 0 \\ 0 & 0 \end{pmatrix}$$

该矩阵半负定，因此 $g_2(\boldsymbol{x})$ 为凹函数。因此，该非线性规划问题的目标函数为凸函数，不等式约束函数为凹函数，等式约束函数为线性函数，满足凸规划定义。

感兴趣的读者可参阅 S. Boyd 与 L. Vandenberghe 所著的《Convex Optimization》。

4.3 无约束优化问题

考虑无约束优化问题

$$\min \quad f(\boldsymbol{x}) \tag{4.8}$$

其中决策变量 $\boldsymbol{x} \in \mathbb{R}^n$。依据目标函数的一阶与二阶微分信息，可分别构造梯度法与牛顿法两类经典的下降迭代算法。

4.3.1 梯度法

梯度法, 又称最速下降法, 是求解无约束优化问题(4.8)的基本算法。在下降迭代算法框架中, 梯度法选取迭代点的负梯度方向作为搜索方向, 即 $\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$。每一步迭代均沿该方向执行一维搜索, 寻找使目标函数下降最多的步长, 逐渐逼近问题的最优解。当 $\nabla f(\boldsymbol{x}^{(k)}) \neq 0$ 时, $\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\nabla f(\boldsymbol{x}^{(k)}) < 0$, 因此负梯度方向为下降方向。梯度法的具体迭代步骤如下。

(1) 选定初始点 $\boldsymbol{x}^{(0)}$, 令 $k = 0$。

(2) 计算目标函数值 $f(\boldsymbol{x}^{(k)})$ 和梯度 $\nabla f(\boldsymbol{x}^{(k)})$, 若 $\|\nabla f(\boldsymbol{x}^{(k)})\|^2 \leq \varepsilon$, 则停止迭代, 得到最优解 $\boldsymbol{x}^{(k)}$ 和最优值 $f(\boldsymbol{x}^{(k)})$。否则, 进入下一步。

(3) 执行一维搜索, 通过下式求解最优步长

$$\lambda_k = \arg\min_{\lambda} \ f(\boldsymbol{x}^{(k)} - \lambda\nabla f(\boldsymbol{x}^{(k)}))$$

更新迭代点 $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \lambda_k\nabla f(\boldsymbol{x}^{(k)})$。返回步骤 (2) 继续迭代。

当目标函数二阶连续可微时, 可推导出最优步长的解析公式。具体来说, 将 $f(\boldsymbol{x})$ 在迭代点 $\boldsymbol{x}^{(k)}$ 处沿负梯度方向作二阶 Taylor 展开, 得到

$$\begin{aligned} f(\boldsymbol{x}^{(k)}-\lambda\nabla f(\boldsymbol{x}^{(k)})) \approx\ & f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\lambda\nabla f(\boldsymbol{x}^{(k)}) \\ & + \frac{1}{2}\lambda\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^{(k)})\lambda\nabla f(\boldsymbol{x}^{(k)}) \end{aligned}$$

关于步长 λ 求一阶微分并令其等于零, 即可得到如下最优步长的计算公式

$$\lambda_k = \frac{\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\nabla f(\boldsymbol{x}^{(k)})}{\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^{(k)})\nabla f(\boldsymbol{x}^{(k)})}$$

梯度法具有全局收敛性, 且为线性收敛。实际数值计算中, 梯度法在开始迭代时效果较好, 能够快速逼近最优解。但当迭代点接近最优解时, 算法会出现明显的迭代扭摆现象, 收敛速度显著放缓, 如图4.4所示。

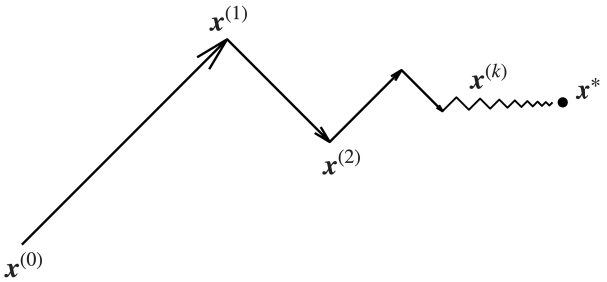

图 4.4 梯度法迭代路径

下面对此现象给出简要分析。设 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}$ 处沿方向 $\boldsymbol{d}$ 的一阶 Taylor 展开为

$$f(\boldsymbol{x}+\lambda\boldsymbol{d}) = f(\boldsymbol{x}) + \lambda\nabla f(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{d} + o(\lambda\|\boldsymbol{d}\|)$$

当迭代点 $\boldsymbol{x}$ 距离最优解 $\boldsymbol{x}^*$ 较远时, 线性项远大于高阶无穷小项, 算法具有较快的下降速度。当迭代点趋近于 $\boldsymbol{x}^*$ 时, 此时线性项与高阶无穷小项同阶, 线性近似的有效性显著降低, 进而导致迭代出现扭摆现象。

例 4.4 用梯度法求解无约束优化问题的最优解

$$\min \quad f(\boldsymbol{x}) = x_1^2 + 5x_2^2$$

取允许误差 $\varepsilon = 0.7$。

解 目标函数的梯度与 Hessian 矩阵分别为

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} 2x_1 \\ 10x_2 \end{pmatrix}, \ \nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 2 & 0 \\ 0 & 10 \end{pmatrix}$$

选取初始点 $\boldsymbol{x}^{(0)} = (2,1)^{\mathrm{T}}$, 计算得

$$\nabla f(\boldsymbol{x}^{(0)}) = \begin{pmatrix} 4 \\ 10 \end{pmatrix}, \ \|\nabla f(\boldsymbol{x}^{(0)})\|^2 = 116 > \varepsilon$$

不满足收敛条件, 需继续迭代。由于 $f(\boldsymbol{x})$ 为二次函数, 当 $k=0$ 时, 代入计算最优步长

$$\lambda_0 = \frac{\begin{pmatrix} 4 & 10 \end{pmatrix}\begin{pmatrix} 4 \\ 10 \end{pmatrix}}{\begin{pmatrix} 4 & 10 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 10 \end{pmatrix}\begin{pmatrix} 4 \\ 10 \end{pmatrix}} = \frac{116}{1\,032} = 0.11$$

更新迭代点

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} - \lambda_0\nabla f(\boldsymbol{x}^{(0)}) = \begin{pmatrix} 1.55 \\ -0.12 \end{pmatrix}$$

此时, 计算得

$$\nabla f(\boldsymbol{x}^{(1)}) = \begin{pmatrix} 3.10 \\ -1.24 \end{pmatrix}, \ \|\nabla f(\boldsymbol{x}^{(1)})\|^2 = 11.15 > \varepsilon$$

仍未满足终止条件, 故继续按上述过程迭代, 直至满足 $\|\nabla f(\boldsymbol{x}^{(k)})\|^2 \le \varepsilon$ 时停止迭代。理论上, 该函数为严格凸二次函数, 其全局最优解为 $\boldsymbol{x}^* = (0,0)^{\mathrm{T}}$。

4.3.2 牛顿法

牛顿法通过有效利用目标函数的二阶微分信息, 收敛速度显著优于梯度法。考虑二次函数

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}^{\mathrm{T}}\boldsymbol{x} + c$$

其中 $\boldsymbol{A}$ 为 $n \times n$ 对称正定矩阵, $c \in \mathbb{R}$。设该函数的最优解为 $\boldsymbol{x}^*$, 由无约束优化问题的一阶必要条件可得

$$\nabla f(\boldsymbol{x}^*) = \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{B} = 0 \tag{4.9}$$

整理得 $\boldsymbol{A}\boldsymbol{x}^* = -\boldsymbol{B}$。另一方面, 任取初始点 $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, 函数在该点的一阶微分为 $\nabla f(\boldsymbol{x}^{(0)}) = \boldsymbol{A}\boldsymbol{x}^{(0)} + \boldsymbol{B}$。将式(4.9)代入上式消去 $\boldsymbol{B}$ 得

$$\nabla f(\boldsymbol{x}^{(0)}) = \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{A}\boldsymbol{x}^*$$

进一步整理, 求出最优解为 $\boldsymbol{x}^* = \boldsymbol{x}^{(0)} - \boldsymbol{A}^{-1}\nabla f(\boldsymbol{x}^{(0)})$。这表明, 关于对称正定二次函数, 从任意初始点 $\boldsymbol{x}^{(0)}$ 出发, 沿牛顿方向 $-\boldsymbol{A}^{-1}\nabla f(\boldsymbol{x}^{(0)})$、以 1 为步长迭代一步, 即可到达函数的最优解, 这就是牛顿法的魅力所在。

例 4.5 用牛顿法求解无约束优化问题的最优解

$$f(\boldsymbol{x}) = x_1^2 + 5x_2^2$$

取允许误差 $\varepsilon = 0.7$。

解 选取初始点 $\boldsymbol{x}^{(0)} = (2, 1)^{\mathrm{T}}$, 梯度为

$$\nabla f(\boldsymbol{x}^{(0)}) = (4, 10)^{\mathrm{T}}$$

二次函数 $f(\boldsymbol{x})$ 对应的 Hessian 矩阵及其逆矩阵分别为

$$\boldsymbol{A} = \begin{pmatrix} 2 & 0 \\ 0 & 10 \end{pmatrix}, \quad \boldsymbol{A}^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/10 \end{pmatrix}$$

根据牛顿迭代公式计算

$$\begin{aligned} \boldsymbol{x}^* &= \boldsymbol{x}^{(0)} - \boldsymbol{A}^{-1}\nabla f(\boldsymbol{x}^{(0)}) \\ &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 1/2 & 0 \\ 0 & 1/10 \end{pmatrix}\begin{pmatrix} 4 \\ 10 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{aligned}$$

经验证, $\boldsymbol{x}^*$ 为函数 $f(\boldsymbol{x})$ 的最优解。

考虑一般 n 元实函数 $f(\boldsymbol{x})$, 若该函数二阶连续可微, 则在 $\boldsymbol{x}^{(k)}$ 处二阶 Taylor 展开对其进行逼近, 即

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(k)}) + \nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\Delta\boldsymbol{x} + \frac{1}{2}\Delta\boldsymbol{x}^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^{(k)})\Delta\boldsymbol{x}$$

其中 $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^{(k)}$。逼近函数的最优解应满足无约束优化问题的一阶必要条件, 因此有

$$\nabla f(\boldsymbol{x}^{(k)}) + \nabla^2 f(\boldsymbol{x}^{(k)})\Delta\boldsymbol{x} = 0$$

若 Hessian 矩阵 $\nabla^2 f(\boldsymbol{x}^{(k)})$ 可逆, 对上式进行整理, 解得

$$\boldsymbol{x} = \boldsymbol{x}^{(k)} - [\nabla^2 f(\boldsymbol{x}^{(k)})]^{-1}\nabla f(\boldsymbol{x}^{(k)})$$

为求 $f(\boldsymbol{x})$ 的最优解, 选取方向 $-[\nabla^2 f(\boldsymbol{x}^{(k)})]^{-1}\nabla f(\boldsymbol{x}^{(k)})$ 作为搜索方向 (牛顿方向), 并按照以下迭代公式进行计算

$$\begin{cases} \boldsymbol{p}^{(k)} = -[\nabla^2 f(\boldsymbol{x}^{(k)})]^{-1}\nabla f(\boldsymbol{x}^{(k)}) \\ \lambda_k = \arg\min\limits_{\lambda} \ f(\boldsymbol{x}^{(k)} + \lambda\boldsymbol{p}^{(k)}) \\ \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \lambda_k\boldsymbol{p}^{(k)} \end{cases}$$

这就是阻尼牛顿法, 也称为广义牛顿法, 可用于求解非正定二次函数的最优化问题。

4.4 约束优化问题

考虑约束优化问题

$$\begin{aligned} &\min \ f(\boldsymbol{x}) \\ &\text{s.t.} \ g_i(\boldsymbol{x}) \geq 0, \ i = 1, 2, \cdots, m \end{aligned} \tag{4.10}$$

上述问题直接求解往往存在较大的难度, 一种自然的思路是将其转化为无约束问题来求解。其中, 最具代表性的是罚函数法与障碍函数法。

4.4.1 罚函数法

罚函数法, 又称外点罚, 通过对违反约束的点施加足够大的惩罚, 使其对应的目标函数值显著增大, 从而引导迭代点逐步向可行域收敛, 最终逼近原约束问题的最优解。为度量约束 $g_i(\boldsymbol{x}) \geq 0$ 的违反程度, 考虑惩罚函数

$$\phi(t) = \begin{cases} 0, & \text{当 } t \geq 0 \\ t^2, & \text{当 } t < 0 \end{cases}$$

将问题(4.10)中的不等式约束转化为 $\phi(g_i(\boldsymbol{x}))$, 其等价于 $[\min(0, g_i(\boldsymbol{x}))]^2$。取充分大的 $M > 0$, 约束优化问题(4.10)可转化为无约束形式

$$P(\boldsymbol{x}, M) = f(\boldsymbol{x}) + M \sum_{i=1}^{m} [\min(0, g_i(\boldsymbol{x}))]^2$$

不难验证, 当迭代点不满足约束条件时, 由于 M 取值极大, 惩罚项会显著增大。这类方法称为罚函数法, M 称为罚因子。具体迭代步骤如下。

(1) 选取初始罚因子 $M_1 > 0$, 允许误差 $\varepsilon > 0$, 令 $k = 1$。

(2) 求解无约束优化问题

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \ P(\boldsymbol{x}, M)$$

(3) 若存在某一不等式约束, 有

$$-g_i(\boldsymbol{x}^{(k)}) > \varepsilon$$

则更新罚因子 $M_{k+1} > M_k$, 返回步骤 (2)。否则, 停止迭代。

例 4.6 用罚函数法求解约束优化问题的最优解

$$\begin{aligned} \min \ & f(x) = \left(x - \frac{1}{2}\right)^2 \\ \text{s.t.} \ & x \le 0 \end{aligned}$$

解 首先将约束 $x \le 0$ 改写为

$$g(x) = -x \ge 0$$

取充分大的 $M > 0$, 构造罚函数

$$\begin{aligned} P(x, M) &= f(x) + M[\min(0, g(x))]^2 \\ &= \left(x - \frac{1}{2}\right)^2 + M[\min(0, -x)]^2 \end{aligned}$$

当 $x > 0$ 时, $\min(0, -x) = -x$, 于是

$$P(x, M) = \left(x - \frac{1}{2}\right)^2 + Mx^2$$

对固定罚因子 M, 计算一阶微分并令其为零, 有

$$\frac{\partial P(x, M)}{\partial x} = 2\left(x - \frac{1}{2}\right) + 2Mx = 0$$

求解可得罚函数的最优解为

$$x(M) = \frac{1}{2(1+M)}$$

如图4.5所示, 当 $M = 0$ 时, $x(M) = 1/2$; 当 $M = 1$ 时, $x(M) = 1/4$; 当 $M = 10$ 时, $x(M) = 1/22$; 当 $M \to \infty$ 时, $x(M) \to 0$。由此可见, 随着罚因子 M 的增大, 罚函数问题的最小点逐步逼近原优化问题的最优解, 这正是罚函数法的基本思想。

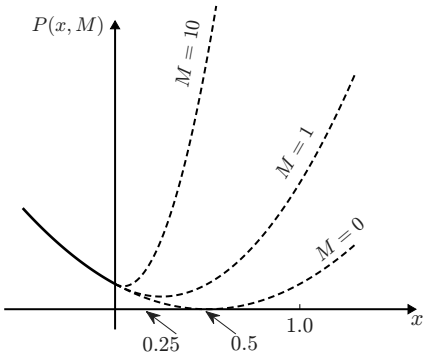

图 4.5 罚函数法求解

4.4.2 障碍函数法

与罚函数法允许迭代点位于可行域外不同, 障碍函数法要求整个迭代过程始终保持在可行域内部, 因此又称为内点罚。考虑约束优化问题(4.10), 构造倒数型障碍函数 $\frac{1}{g_i(\boldsymbol{x})}$ 与对数型障碍函数 $\ln g_i(\boldsymbol{x})$, 得到无约束优化问题

$$P(\boldsymbol{x}, r_k) = f(\boldsymbol{x}) + r_k \sum_{i=1}^{m} \frac{1}{g_i(\boldsymbol{x})}$$

与

$$P(\boldsymbol{x}, r_k) = f(\boldsymbol{x}) - r_k \sum_{i=1}^{m} \ln g_i(\boldsymbol{x})$$

其中 $r_k > 0$ 称为障碍因子, $P(\boldsymbol{x}, r_k)$ 称为障碍函数。具体迭代步骤如下。

(1) 选取初始障碍因子 $r_1 > 0$, 允许误差 $\varepsilon > 0$, 令 $k = 1$。

(2) 构建障碍函数, 障碍项可根据求解需求选择倒数型函数或对数型函数。

(3) 求解无约束优化问题

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \ P(\boldsymbol{x}, r_k)$$

(4) 若采用倒数型障碍项, 检查是否满足

$$r_k \sum_{i=1}^{m} \frac{1}{g_i(\boldsymbol{x}^{(k)})} \leq \varepsilon$$

若采用对数型障碍项, 检查是否满足

$$\left| r_k \sum_{i=1}^{m} \ln(g_i(\boldsymbol{x}^{(k)})) \right| \leq \varepsilon$$

若满足, 得到原约束优化问题的最优解 $\boldsymbol{x}^{(k)}$, 停止迭代。否则, 取 $r_{k+1} < r_k$, 返回步骤 (3)。

例 4.7 用障碍函数法求解约束优化问题的最优解

$$\begin{aligned} \min \ & f(x) = x - 2 \\ \text{s.t.} \ & x \geq 0 \end{aligned}$$

解 采用倒数型障碍项, 构造障碍函数

$$P(x, r_k) = x - 2 + \frac{r_k}{x}$$

对于固定的障碍因子 r_k, 求一阶微分并令其为零, 有

$$\frac{\partial P(x, r_k)}{\partial x} = 1 - \frac{r_k}{x^2} = 0$$

求解可得 $x = \pm\sqrt{r_k}$。结合约束条件 $x \geq 0$, 舍去负根, 得障碍问题的最优解为

$$x = \sqrt{r_k}$$

图4.6展示了 $r_k = 1, 0.1, 0.01$ 时障碍函数的图像。由此可见, 随着障碍因子 $r_k \to 0$, 障碍函数的极小点逐渐逼近原问题的最优解。

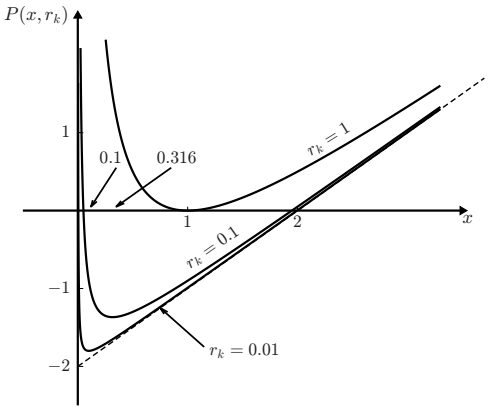

图 4.6 障碍函数求解

4.5 应用案例

4.5.1 问题描述

电力经济调度是现代电力系统运行管理中的重要问题, 其基本任务是在满足系统负荷需求及机组运行约束的条件下, 合理分配各发电机组的输出功率, 使总发电成本达到最小。由于发电机组的燃料成本函数通常可表示为输出功率的二次函数, 因此电力经济调度问题属于典型的非线性规划范畴。

例 4.8 某区域电力系统包含三台并网发电机组, 需共同承担系统总负荷 300 兆瓦。各机组发电成本函数 (单位: 元) 及出力上下限 (单位: 兆瓦) 如表4.1所示。试确定各机组最优出力 x_1, x_2, x_3, 使发电总成本最小。

表 4.1 发电机组技术参数

机组编号	成本函数	出力下限	出力上限
1	$0.01x_1^2 + 2x_1 + 100$	50	200
2	$0.015x_2^2 + 1.5x_2 + 120$	30	150
3	$0.02x_3^2 + x_3 + 80$	20	100

4.5.2 模型建立

设三台发电机组的出力为决策变量 $\boldsymbol{x} = [x_1, x_2, x_3]^{\mathrm{T}}$, 发电总成本为各机组成本之和, 则目标函数为

$$\min \quad f(\boldsymbol{x}) = 0.01x_1^2 + 0.015x_2^2 + 0.02x_3^2 + 2x_1 + 1.5x_2 + x_3 + 300$$

系统运行需满足功率平衡约束, 即 $x_1 + x_2 + x_3 = 300$。同时, 各机组出力需满足技术上下限约束

$$50 \le x_1 \le 200,\ 30 \le x_2 \le 150,\ 20 \le x_3 \le 100$$

综上, 该电力经济调度问题可表示为如下非线性规划模型

$$\min \quad f(\boldsymbol{x}) = 0.01x_1^2 + 0.015x_2^2 + 0.02x_3^2 + 2x_1 + 1.5x_2 + x_3 + 300$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 = 300 \\ 50 \le x_1 \le 200 \\ 30 \le x_2 \le 150 \\ 20 \le x_3 \le 100 \end{cases}$$

4.5.3 算法设计

采用罚函数法进行求解, 首先将不等式约束表示为

$$g_1 = x_1 - 50,\ g_2 = 200 - x_1$$
$$g_3 = x_2 - 30,\ g_4 = 150 - x_2$$
$$g_5 = x_3 - 20,\ g_6 = 100 - x_3$$

构造二次罚函数

$$P(\boldsymbol{x}, M) = f(\boldsymbol{x}) + M\left[(x_1 + x_2 + x_3 - 300)^2 + \sum_{i=1}^{6}(\max(0, -g_i(\boldsymbol{x})))^2\right]$$

计算梯度, 可得

$$\nabla P(\boldsymbol{x}, M) = \nabla f(\boldsymbol{x}) + 2M\left[(x_1 + x_2 + x_3 - 300) + \sum_{i=1}^{6}\max(0, -g_i(\boldsymbol{x}))\nabla(-g_i(\boldsymbol{x}))\right]$$

其中

$$\nabla f(\boldsymbol{x}) = (0.02x_1 + 2, 0.03x_2 + 1.5, 0.04x_3 + 1)^{\mathrm{T}}$$

选取初始点 $\boldsymbol{x}^{(0)} = (100, 100, 100)^{\mathrm{T}}$, 给定初始罚因子 $M_0 = 1$、放大系数 $\beta = 5$、迭代步长 $\alpha = 0.01$ 以及收敛精度 $\epsilon = 10^{-4}$。迭代过程如表4.2所示。

表 4.2 罚函数法迭代过程

迭代	罚因子	x_1	x_2	x_3	总成本
0	1	100.00	100.00	100.00	1200.00
1	5	119.03	96.02	84.51	1189.43
2	25	119.19	96.13	84.60	1190.96
3	125	119.22	96.15	84.61	1191.27
4	625	119.23	96.15	84.61	1191.33
5	3 125	119.23	96.15	84.62	1191.34

随着罚因子不断增大, 迭代点逐步趋近可行域, 同时目标函数单调下降并趋于稳定。算法最终收敛至最优解 $\boldsymbol{x}^* = [119.23, 96.15, 84.62]^{\mathrm{T}}$, 对应的最优值为 $f(\boldsymbol{x}^*) = 1191.34$。

4.5.4 结果分析

为验证最优解的可行性与合理性, 现对约束条件逐一进行检验。首先, 考虑功率平衡约束, 有 $119.23 + 96.15 + 84.62 = 300$。可见其满足系统总负荷需求。进一步考察各

机组的出力约束, 可以得到 $50 \le 119.23 \le 200,\ 30 \le 96.15 \le 150,\ 20 \le 84.62 \le 100$。因此, 该解满足所有等式约束与不等式约束, 为原问题的最优解。需要说明的是, 罚函数法的收敛速度与罚因子的更新策略密切相关, 读者可分别取 $\beta = 1, 10$ 进行数值分析。此外, 表4.3给出了几种经典非线性规划算法的主要特点。

表 4.3 非线性规划算法对比

方法	优点	缺点
梯度法	简单易实现	收敛速度慢
牛顿法	收敛速度快	需计算 Hessian 矩阵
罚函数法	无需可行初始点	需调整罚因子
障碍函数法	数值稳定性好	需可行初始点

习题

4.1 判定下述非线性规划是否为凸规划。

$$(1)\ \max\ f(\boldsymbol{x}) = x_1 + 2x_2 \quad \text{s.t.} \begin{cases} x_1^2 + x_2^2 \le 9 \\ x_2 \ge 0 \end{cases}$$

$$(2)\ \min\ f(\boldsymbol{x}) = 2x_1^2 + x_2^2 + x_3^3 \quad \text{s.t.} \begin{cases} x_1^2 + x_2^2 \le 4 \\ 5x_1 + x_3 = 10 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

4.2 试分析非线形规划

$$\min\ f(\boldsymbol{x}) = (x_1 - 2)^2 + (x_2 - 3)^2$$
$$\text{s.t.} \begin{cases} g_1(\boldsymbol{x}) = x_1^2 + (x_2 - 3)^2 \ge 4 \\ g_2(\boldsymbol{x}) = x_2 \le 2 \end{cases}$$

在点 $\boldsymbol{x}_1 = (3, 2)^{\mathrm{T}},\ \boldsymbol{x}_2 = (0, 0)^{\mathrm{T}}$ 的可行下降方向, 并绘图表示其可行下降方向的范围。

4.3 试用梯度法求解无约束优化问题的最优解。

$$\min\ f(\boldsymbol{x}) = x_1^2 + x_2^2 + 2x_3^2$$

选取初始点 $\boldsymbol{x}^0 = (2, -2, 1)^{\mathrm{T}}$, 要求进行三次迭代, 并验证相邻两步的搜索方向正交。

4.4 试用牛顿法求解无约束优化问题的最优解。

$$\min \quad f(\boldsymbol{x}) = -\frac{1}{x_1^2 + x_2^2 + 2}$$

选取初始点 $\boldsymbol{x}^0 = (4, 0)^{\mathrm{T}}$, 并将采用最佳步长和固定步长 $\lambda = 1$ 的情形作比较。

4.5 试用罚函数法求解约束优化问题

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_2 = 2 \end{aligned}$$

并写出当罚因子 $M = 1$ 和 $M = 10$ 时的近似解。

4.6 试用障碍函数法求解约束优化问题

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = \frac{1}{3}(x_1 + 1)^3 + x_2 \\ \text{s.t.} \quad & \begin{cases} g_1(\boldsymbol{x}) = x_1 - 1 \geq 0 \\ g_2(\boldsymbol{x}) = x_2 \geq 0 \end{cases} \end{aligned}$$

5

动态规划

5.1 基本概念

5.1.1 多阶段决策过程

所谓多阶段决策过程，是指一类可依据时间或逻辑顺序，分解为若干相互关联阶段的动态过程，其中每个阶段称为一个“时段”。在每一阶段均需作出相应决策，整个过程的决策集合构成一个决策序列，因此多阶段决策问题也被称为序贯决策问题。各阶段的决策并非相互独立，单个阶段的决策不仅决定本阶段的效果，还会直接影响下一阶段的初始状态。当所有阶段的决策确定后，由此形成的决策序列称为策略。多阶段决策过程的目标是寻求一个最优策略，使得各阶段效益的总和达到全局最优。

例 5.1 给定某线路网络图，节点间连线上的数值表示对应两点间的距离，如图5.1所示。试确定一条从 A 到 F 的输油管道铺设路线，使总距离最短。

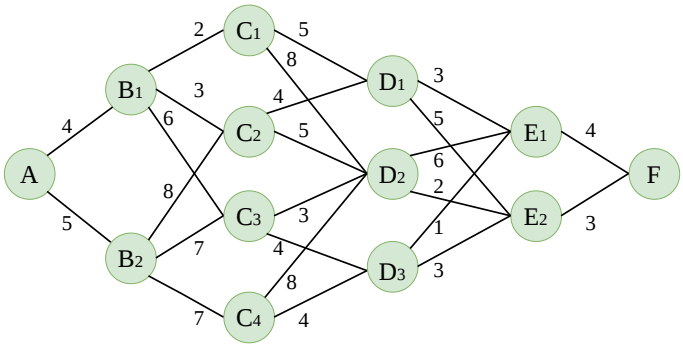

图 5.1 A 到 F 各点间距离

例 5.2 某公司可用于投资的总资金为 10 万元。设对项目 $1,2,3$ 的投资额为 x_1, x_2, x_3 万元, 对应的收益函数分别为 $g_1(x_1)=4x_1$, $g_2(x_2)=9x_2$, $g_3(x_3)=2x_3^2$。试确定资金在各项目的分配方案, 使总投资收益最大。

解 该问题可建立如下非线性规划模型

$$\max z = 4x_1 + 9x_2 + 2x_3^2$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 = 10 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

采用动态规划求解时, 可按项目顺序划分为 3 个阶段, 将静态优化问题转化为多阶段决策过程, 避免直接求解非线性规划的复杂迭代。

例 5.3 某旅行者携带背包登山, 背包最大承重为 a 公斤。现有 n 种物品可供选择装入背包, 第 i 种物品单件重量为 a_i 公斤, 其价值为携带数量 x_i 的函数 $c_i(x_i)$。试确定各类物品的携带数量, 使背包中物品总价值最大。

5.1.2 关键要素

以最短路线问题为例, 给出建立动态规划模型所需的关键要素, 主要包括阶段、状态、决策与策略、状态转移方程、指标函数。

(1) 阶段: 指按时间顺序或空间结构将问题分解为若干相互关联的部分, 通常用 k 表示。在例5.1中, 从 A 到 F 的过程可划分为 5 个阶段

$$\mathrm{A} \to \mathrm{B},\ \mathrm{B} \to \mathrm{C},\ \mathrm{C} \to \mathrm{D},\ \mathrm{D} \to \mathrm{E},\ \mathrm{E} \to \mathrm{F}$$

(2) 状态: 指各阶段决策开始时所面临的客观条件, 通常用 s_k 表示。状态变量的所有可能取值构成状态集合, 记为 S_k。动态规划中的状态必须满足无后效性, 也就是说, 若某阶段状态一经确定, 则该阶段之后的决策过程仅依赖于当前状态, 而与此前各阶段如何到达该状态的历史路径无关。在例5.1中, 第 1 阶段的状态为 A, 即 $S_1=\{\mathrm{A}\}$, 其余阶段状态集合为

$$S_2=\{\mathrm{B}_1,\mathrm{B}_2\},\ S_3=\{\mathrm{C}_1,\mathrm{C}_2,\mathrm{C}_3,\mathrm{C}_4\},\ S_4=\{\mathrm{D}_1,\mathrm{D}_2,\mathrm{D}_3\},\ S_5=\{\mathrm{E}_1,\mathrm{E}_2\}$$

(3) 决策与策略: 决策指在给定第 k 阶段状态 s_k 的前提下, 对下一阶段状态作出的选择, 通常用 $u_k(s_k)$ 表示。决策变量的取值范围称为允许决策集合, 记为 $D_k(s_k)$, 满足 $u_k(s_k) \in D_k(s_k)$。由各阶段决策构成的序列称为策略, 记为

$p_{1,n} = \{u_1(s_1), u_2(s_2), \cdots, u_n(s_n)\}$。所有策略构成的集合记为 $P_{1,n}$, 其中使目标函数达到最优的策略称为最优策略。在例5.1中, 从第 2 阶段状态 B_1 出发, 可选择下一节点为 $\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3$, 即 $D_2(\mathrm{B}_1) = \{\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3\}$。若选定 C_3, 则相应决策为

$$u_2(\mathrm{B}_1) = \mathrm{C}_3$$

(4) 状态转移方程: 描述当前阶段的状态由上一阶段的状态及其决策共同决定的规律, 通常用 $s_{k+1} = T_k(s_k, u_k)$ 表示。在例5.1中, 下一阶段状态即为当前决策, 因此状态转移方程可简化为

$$s_{k+1} = u_k(s_k)$$

(5) 指标函数: 用以衡量策略优劣的数量标准, 通常分为阶段指标函数和过程指标函数。阶段指标函数 $d_k(s_k, u_k)$ 表示第 k 阶段在状态 s_k 下采取决策 u_k 所获得的单阶段效益。对于 n 阶段过程, 从第 1 阶段到第 n 阶段称为全过程, 记其过程指标函数为 $V_{1,n}(s_1, p_{1,n})$, 从第 k 阶段到第 n 阶段的部分过程称为后部子过程, 记其过程指标函数为 $V_{k,n}(s_k, p_{k,n})$。最优指标函数 $f_k(s_k)$ 定义为从第 k 阶段状态 s_k 出发, 采用最优策略直至过程结束所能获得的最优效益, 即

$$f_k(s_k) = V_{k,n}(s_k, p^*_{k,n}) = \operatorname*{opt}_{p_{k,n} \in P_{k,n}} V_{k,n}(s_k, p_{k,n})$$

其中, opt 根据具体问题可取 max 或 min。当 $k = 1$ 时, $f_1(s_1)$ 即为从初始状态 s_1 出发的全过程最优值。在例5.1中, 指标函数为路径长度。$d(\mathrm{B}_1, \mathrm{C}_2)$ 表示 B_1 到 C_2 的路段长度, $V_{2,5}(\mathrm{B}_1)$ 表示从 B_1 到 F 的路径长度, $f_2(\mathrm{B}_1)$ 表示从 B_1 到 F 的最短路径长度, $f_1(\mathrm{A})$ 表示问题的总体目标。

5.2 动态规划模型

5.2.1 基本思想

对于例5.1所示的最短路线问题, 若直接采用枚举法求解, 需列出从 A 到 F 的所有可行路径并逐一比较。随着问题阶段数与各阶段状态数的增加, 枚举法的计算量将呈指数级增长。为高效求解此类多阶段决策优化问题, 本节引入动态规划方法。其核心思路是从过程的最后阶段出发, 采用逆序递推的方式, 逐阶段计算各状态至 F 的最短距离, 最终确定从 A 到 F 的最优路线。

从最后阶段 $k = 5$ 开始, 状态变量 s_5 可取两种状态 $\mathrm{E}_1, \mathrm{E}_2$, 其到 F 的距离分别为 4 和 3, 即

$$f_5(\mathrm{E}_1) = 4, \ f_5(\mathrm{E}_2) = 3$$

当 $k=4$ 时, 状态变量 s_4 可取三种状态 $\mathrm{D}_1,\mathrm{D}_2,\mathrm{D}_3$, 对应从该状态经一步决策到达终点。以状态 D_1 为例, 需比较经 $\mathrm{E}_1,\mathrm{E}_2$ 的两条路径, 选取总距离最小者, 即

$$f_4(\mathrm{D}_1)=\min\begin{Bmatrix}d(\mathrm{D}_1,\mathrm{E}_1)+f_5(\mathrm{E}_1)\\ d(\mathrm{D}_1,\mathrm{E}_2)+f_5(\mathrm{E}_2)\end{Bmatrix}=\min\begin{Bmatrix}3+4\\ 5+3\end{Bmatrix}=7$$

上式表明, 由 D_1 到 F 的最短距离为 7, 其路径为 $\mathrm{D}_1\to\mathrm{E}_1\to\mathrm{F}$, 决策为 $u_4^*(\mathrm{D}_1)=\mathrm{E}_1$。同理可得

$$f_4(\mathrm{D}_2)=\min\begin{Bmatrix}d(\mathrm{D}_2,\mathrm{E}_1)+f_5(\mathrm{E}_1)\\ d(\mathrm{D}_2,\mathrm{E}_2)+f_5(\mathrm{E}_2)\end{Bmatrix}=\min\begin{Bmatrix}6+4\\ 2+3\end{Bmatrix}=5$$

即 D_2 到 F 的最短距离为 5, 其路径为 $\mathrm{D}_2\to\mathrm{E}_2\to\mathrm{F}$, 决策为 $u_4^*(\mathrm{D}_2)=\mathrm{E}_2$。

$$f_4(\mathrm{D}_3)=\min\begin{Bmatrix}d(\mathrm{D}_3,\mathrm{E}_1)+f_5(\mathrm{E}_1)\\ d(\mathrm{D}_3,\mathrm{E}_2)+f_5(\mathrm{E}_2)\end{Bmatrix}=\min\begin{Bmatrix}1+4\\ 3+3\end{Bmatrix}=5$$

即 D_3 到 F 的最短距离为 5, 其路径为 $\mathrm{D}_3\to\mathrm{E}_1\to\mathrm{F}$, 决策为 $u_4^*(\mathrm{D}_3)=\mathrm{E}_1$。

当 $k=3$ 时, 经计算可得

$$\begin{aligned}&f_3(\mathrm{C}_1)=12,\ u_3^*(\mathrm{C}_1)=\mathrm{D}_1\\ &f_3(\mathrm{C}_2)=10,\ u_3^*(\mathrm{C}_2)=\mathrm{D}_2\\ &f_3(\mathrm{C}_3)=8,\ u_3^*(\mathrm{C}_3)=\mathrm{D}_2\\ &f_3(\mathrm{C}_4)=9,\ u_3^*(\mathrm{C}_4)=\mathrm{D}_3\end{aligned}$$

当 $k=2$ 时, 继续计算可得

$$\begin{aligned}&f_2(\mathrm{B}_1)=13,\ u_2^*(\mathrm{B}_1)=\mathrm{C}_2\\ &f_2(\mathrm{B}_2)=15,\ u_2^*(\mathrm{B}_2)=\mathrm{C}_3\end{aligned}$$

当 $k=1$ 时, 仅有初始状态 A, 于是

$$f_1(\mathrm{A})=\min\begin{Bmatrix}d(\mathrm{A},\mathrm{B}_1)+f_2(\mathrm{B}_1)\\ d(\mathrm{A},\mathrm{B}_2)+f_2(\mathrm{B}_2)\end{Bmatrix}=\min\begin{Bmatrix}4+13\\ 5+15\end{Bmatrix}=17$$

即从 A 到 F 的最短距离为 17, 最优决策为 $u_1^*(\mathrm{A})=\mathrm{B}_1$。按计算顺序反向追溯, 可得最优决策序列为

$$u_1^*(\mathrm{A})=\mathrm{B}_1,\ u_2^*(\mathrm{B}_1)=\mathrm{C}_2,\ u_3^*(\mathrm{C}_2)=\mathrm{D}_2,\ u_4^*(\mathrm{D}_2)=\mathrm{E}_2,\ u_5^*(\mathrm{E}_2)=\mathrm{F}$$

对应的最优路线为

$$A \to B_1 \to C_2 \to D_2 \to E_2 \to F$$

由上述计算过程可知, 各阶段最优指标函数满足如下递推关系

$$\begin{cases} f_k(s_k) = \min\{d_k(s_k, u_k) + f_{k+1}(s_{k+1})\}, \ k = 5, 4, 3, 2, 1 \\ f_6(s_6) = 0 \end{cases}$$

该式称为动态规划基本方程, 其中 $f_6(s_6) = 0$ 为边界条件。一般形式的动态规划基本方程可写为

$$\begin{cases} f_k(s_k) = \underset{u_k \in D_k(s_k)}{\text{opt}} \{v_k(s_k, u_k) + f_{k+1}(s_{k+1})\}, \ k = n, n-1, \cdots, 1 \\ f_{n+1}(s_{n+1}) = 0 \end{cases}$$

其中 $v_k(s_k, u_k)$ 为状态 s_k 在决策 u_k 下对应的第 k 阶段指标函数值。

性质 5.1

对于最优策略所经过的任一状态, 无论该状态之前的状态与决策如何, 此后的决策序列必构成从该状态出发的最优子策略。

以最短路线问题为例, 最优路线上任意一点到终点的子路径, 也必为该点到终点的最短路径。根据上述性质, 多阶段决策问题可转化为分阶段递推优化过程, 通常采用逆序递推方式, 由终点向起点逐步推算。

5.2.2 模型建立

动态规划建模的关键在于合理选取状态变量, 使各阶段状态之间满足状态转移方程。下面以例5.2的投资决策问题为例说明模型建立与求解方法。

若采用动态规划方法求解, 可人为引入 “时段” 概念, 按项目决策顺序将问题划分为 3 个阶段, 依次对各项目进行投资分配, 阶段变量记为 $k = 1, 2, 3$。状态变量通常选取在递推过程中具有累积性或规律性变化的量, 此处选取第 k 阶段初始可用于投资的资金总额为状态变量 s_k, 初始状态为 $s_1 = 10$。取第 k 阶段对项目 k 的投资额为决策变量, 并记

$$u_k = x_k, \ k = 1, 2, 3$$

当 $k = 1$ 时, 有 $s_1 = 10$, $u_1 = x_1$。当 $k = 2$ 时, 状态 s_2 表示完成项目 1 投资后剩余的可投资资金, 即 $s_2 = s_1 - u_1$, $u_2 = x_2$。依此类推, 可得状态转移方程为

$$s_{k+1} = s_k - x_k$$

定义阶段指标函数为

$$V_{k,3} = \sum_{i=k}^{3} g_i(x_i)$$

最优指标函数 $f_k(s_k)$ 表示当可投资资金为 s_k 时, 从第 k 项到第 3 项所能获得的最大收益。对应的动态规划基本方程为

$$\begin{cases} f_k(s_k) = \max\limits_{0 \le x_k \le s_k} \{g_k(x_k) + f_{k+1}(s_{k+1})\}, \ k = 3, 2, 1 \\ f_4(s_4) = 0 \end{cases}$$

通过逐阶段逆序递推计算, 得到各项目的最优投资额, 其中 $f_1(10)$ 即为全局最大收益。

一般地, 建立动态规划模型的基本步骤可归纳为以下四点。

(1) 划分阶段: 分析问题结构, 识别多阶段决策特征, 按时间或空间顺序划分为若干满足递推关系的阶段, 对非时序的静态问题赋予“时段”概念。

(2) 选取状态变量: 状态变量 s_k 应能准确刻画过程的演变特征, 并满足无后效性。同时状态变量应具有可知性, 即可通过直接或间接的方法测知。此外, 状态变量可取离散值或连续值, 通常从约束条件或与决策直接关联的累积量中选取。

(3) 确定状态转移方程: 明确决策变量 u_k 及其允许决策集合 $D_k(s_k)$, 建立由当前状态 s_k 与决策 u_k 确定下一状态 s_{k+1} 的状态转移方程 $s_{k+1} = T(s_k, u_k)$。

(4) 构造基本方程: 定义阶段指标 $d(s_k, u_k)$、过程指标函数 $V_{k,n}$ 以及最优指标函数 $f_k(s_k)$, 并结合递推关系与边界条件, 构建动态规划基本方程。

5.3 动态规划求解

动态规划有两种基本求解方法: 逆序解法与顺序解法。若寻优方向与多阶段决策过程的实际行进方向相反, 从最后阶段开始逐段向前递推, 最终得到全过程的最优策略, 称为逆序解法。反之, 寻优方向与过程行进方向一致, 则称为顺序解法。

5.3.1 逆序解法

设已知初始状态为 s_1, 定义最优值函数 $f_k(s_k)$ 表示第 k 阶段以状态 s_k 出发, 从第 k 阶段到第 n 阶段所能获得的最大效益。

从第 n 阶段开始递推, 有

$$f_n(s_n) = \max_{u_n \in D_n(s_n)} \{d_n(s_n, u_n)\}$$

其中 $D_n(s_n)$ 为状态 s_n 对应的第 n 阶段允许决策集合。求解上述一维极值问题, 即可得到最优决策 $u_n = u_n(s_n)$ 与最优值 $f_n(s_n)$。

在第 $n-1$ 阶段, 有

$$f_{n-1}(s_{n-1}) = \max_{u_{n-1} \in D_{n-1}(s_{n-1})} \{d_{n-1}(s_{n-1}, u_{n-1}) * f_n(s_n)\}$$

其中状态转移方程为 $s_n = T_{n-1}(s_{n-1}, u_{n-1})$。同理可求得最优决策 $u_{n-1} = u_{n-1}(s_{n-1})$ 与最优值 $f_{n-1}(s_{n-1})$。符号 $*$ 统一表示加法或乘法, 具体运算类型需根据实际目标函数确定。

推广至第 k 阶段, 递推方程为

$$f_k(s_k) = \max_{u_k \in D_k(s_k)} \{d_k(s_k, u_k) * f_{k+1}(s_{k+1})\}$$

其中 $s_{k+1} = T_k(s_k, u_k)$。求解该极值问题, 得到最优解 $u_k = u_k(s_k)$ 与最优值 $f_k(s_k)$。

依此类推, 直至第 1 阶段, 有

$$f_1(s_1) = \max_{u_1 \in D_1(s_1)} \{d_1(s_1, u_1) * f_2(s_2)\}$$

其中 $s_2 = T_1(s_1, u_1)$。由此可得到最优决策 $u_1 = u_1(s_1)$ 与最优值 $f_1(s_1)$。又初始状态 s_1 已知, $u_1 = u_1(s_1)$ 与 $f_1(s_1)$ 可确定, 进而由状态转移方程计算 $s_2 = T_1(s_1, u_1)$。依此前推, 即可逐段确定各阶段最优决策与相应效益。

例 5.4 用逆序解法求解如下问题

$$\max z = x_1 \cdot x_2^2 \cdot x_3$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 = c \\ x_i \geq 0,\ i = 1, 2, 3 \end{cases}$$

解 按变量数目将问题划分为 3 个决策阶段。引入状态变量 s_1, s_2, s_3, 并令 $s_1 = c$。以 x_1, x_2, x_3 为各阶段决策变量, 阶段指标函数按乘积方式复合。定义最优指标函数 $f_k(s_k)$ 为第 k 阶段从状态 s_k 出发, 至第 3 阶段结束所能获得的最大效益。阶段指标分别为 x_1, x_2^2, x_3, 全过程指标函数为 $V_{1,3} = x_1 \cdot x_2^2 \cdot x_3$。

状态满足 $s_3 = x_3$, $s_3 + x_2 = s_2$, $s_2 + x_1 = s_1 = c$, 即 $s_3 = x_3$, $0 \leq x_2 \leq s_2$, $0 \leq x_1 \leq s_1 = c$。采用逆序解法, 从后向前递推, 有

$$f_3(s_3) = \max_{x_3 \in D_3(s_3)} \{d_3(s_3, u_3)\} = \max\{x_3\} = s_3$$

最优解为 $x_3^* = s_3$。接下来计算

$$\begin{aligned} f_2(s_2) &= \max_{x_2 \in D_2(s_2)} \{d_2(s_2, u_2) \cdot f_3(s_3)\} \\ &= \max_{0 \le x_2 \le s_2} \{x_2^2 \cdot f_3(s_3)\} \\ &= \max_{0 \le x_2 \le s_2} \{x_2^2 \cdot (s_2 - x_2)\} \\ &= \max_{0 \le x_2 \le s_2} \{h_2(s_2, x_2)\} \end{aligned}$$

对 x_2 求导并令一阶导数为零有 $\dfrac{\partial h_2}{\partial x_2} = 2x_2 s_2 - 3x_2^2 = 0$, 解得驻点 $x_2 = \dfrac{2}{3}s_2$, $x_2 = 0$ (舍去)。再由二阶条件 $\left.\dfrac{\partial^2 h_2}{\partial^2 x_2}\right|_{x_2 = \frac{2}{3}s_2} = -2s_2 < 0$, 知该点为极大值点, 因此

$$f_2(s_2) = \frac{4}{27}s_2^3, \ x_2^* = \frac{2}{3}s_2$$

继续递推至第 1 阶段, 有

$$\begin{aligned} f_1(s_1) &= \max_{x_1 \in D_1(s_1)} \{d_1(s_1, u_1) \cdot f_2(s_2)\} \\ &= \max_{0 \le x_1 \le s_1} \{x_1 \cdot f_2(s_2)\} \\ &= \max_{0 \le x_1 \le s_1} \{x_1 \cdot \frac{4}{27}(s_1 - x_1)^3\} \\ &= \max_{0 \le x_1 \le s_1} \{h_1(s_1, x_1)\} \end{aligned}$$

同理, 利用微分法可求得最优解与最优值分别为

$$x_1^* = \frac{1}{4}s_1, \ f_1(s_1) = \frac{1}{64}s_1^4$$

已知 $s_1 = c$, 按递推顺序回代可得

$$\begin{aligned} x_1^* &= \frac{1}{4}s_1, \ f_1(c) = \frac{1}{64}c^4 \\ x_2^* &= \frac{2}{3}s_2 = \frac{1}{2}c, \ f_2(s_2) = \frac{1}{16}c^3 \\ x_3^* &= \frac{1}{4}c, \ f_3(s_3) = \frac{1}{4}c \end{aligned}$$

因此, 目标函数的最大值为 $f_1(c) = \dfrac{1}{64}c^4$。

5.3.2 顺序解法

顺序解法可理解为将原 n 阶段决策过程的递推方向反转, 以 s_{k+1} 为输入状态, s_k 为输出状态。相应的状态转移方程为逆序解法的逆变换, 记为 $s_k = T_k(s_{k+1}, u_k)$。设终止状态 s_{n+1} 已知, 定义最优指标函数 $f_k(s)$ 表示第 k 阶段末状态为 s 时, 从第 1 阶段至第 k 阶段所能获得的最大效益。

从第 1 阶段开始递推, 有

$$f_1(s_2) = \max_{u_1 \in D_1(s_1)} \{d_1(s_1, u_1)\}$$

其中 $s_1 = T_1(s_2, u_1)$。求解得到最优决策 $u_1 = u_1(s_2)$ 与最优值 $f_1(s_2)$。

第 2 阶段递推方程为

$$f_2(s_3) = \max_{u_2 \in D_2(s_2)} \{d_2(s_2, u_2) * f_1(s_2)\}$$

依此类推, 直到第 n 阶段有

$$f_n(s_{n+1}) = \max_{u_n \in D_n(s_n)} \{d_n(s_n, u_n) * f_{n-1}(s_n)\}$$

由于终止状态 s_{n+1} 已知, $u_n = u_n(s_{n+1})$ 与 $f_n(s_{n+1})$ 可确定。再沿与过程相反的方向回代推算, 即可逐段确定各阶段最优决策与效益。

仍以例5.4说明顺序解法的求解过程。定义状态满足 $s_3 + x_3 = s_4 = c$, $s_2 + x_2 = s_3$, $s_2 = x_1$, 状态转移方程为 $s_k = T_k(s_{k+1}, u_k)$, 约束为

$$s_2 = x_1,\ 0 \le x_2 \le s_3,\ 0 \le x_3 \le s_4 = c$$

采用顺序解法, 从前向后依次有

$$\begin{aligned}
f_1(s_2) &= \max_{x_1 = s_2} \{x_1\} = s_2,\ x_1^* = s_2 \\
f_2(s_3) &= \max_{0 \le x_2 \le s_3} \{x_2^2 \cdot f_1(s_2)\} = \frac{4}{27} s_3^3,\ x_2^* = \frac{2}{3} s_3 \\
f_3(s_4) &= \max_{0 \le x_3 \le s_4} \{x_3 \cdot f_2(s_3)\} = \frac{1}{64} s_4^4,\ x_3^* = \frac{1}{4} s_4
\end{aligned}$$

由 $s_4 = c$, 可得最优解为

$$x_1^* = \frac{1}{4}c,\ x_2^* = \frac{1}{2}c,\ x_3^* = \frac{1}{4}c$$

目标函数的最大值为 $z^* = \dfrac{1}{64}c^4$。

5.3.3 方法比较

顺序解法与逆序解法在理论上并无本质差别, 只是递推方向不同。一般而言, 初始状态给定时宜采用逆序解法, 终止状态给定时宜采用顺序解法。若同时给定唯一的初始状态和终止状态, 则两种方法均可使用。但当初始状态已知而终止状态有多个时, 需要比较各终止状态对应的路径及指标函数值, 以选取总体效益最优者, 此时顺序解法通常更为方便。因此, 应根据问题的特点灵活选择求解方向, 以简化计算过程。此外, 两种方法在建模时的主要区别如下。

(1) 状态转移方式不同: 逆序解法中, 状态由前向后转移, 即 $s_{k+1}=T_k(s_k,u_k)$。顺序解法中, 状态由后向前转移, 即 $s_k=T_k(s_{k+1},u_k)$。

(2) 指标函数定义不同: 逆序解法中, $f_k(s_k)$ 表示从第 k 阶段状态 s_k 到过程终点的后部子过程最优效益, 整体最优值为 $f_1(s_1)$。顺序解法中, $f_k(s_{k+1})$ 表示从过程起点到第 k 阶段末状态 s_{k+1} 的前部子过程最优效益, 整体最优值为 $f_n(s_{n+1})$。

(3) 基本方程形式不同: 对于可加型指标函数, 在逆序解法中, 有 $v_{k,n}=\sum\limits_{j=k}^{n}v_j(s_j,u_j)$, 基本方程为

$$\begin{cases} f_k(s_k)=\underset{u_k\in D_k}{\text{opt}}\{v_k(s_k,u_k)+f_{k+1}(s_{k+1})\},\ k=n,n-1,\cdots,1 \\ f_{n+1}(s_{n+1})=0 \end{cases}$$

在顺序解法中, 有 $v_{1,k}=\sum\limits_{j=1}^{k}v_j(s_{j+1},u_j)$, 基本方程为

$$\begin{cases} f_k(s_{k+1})=\underset{u_k\in D_k}{\text{opt}}\{v_k(s_{k+1},u_k)+f_{k-1}(s_k)\},\ k=1,2,\cdots,n \\ f_0(s_1)=0 \end{cases}$$

对于可乘型指标函数, 在逆序解法中, 有 $v_{k,n}=\prod\limits_{j=k}^{n}v_j(s_j,u_j)$, 基本方程为

$$\begin{cases} f_k(s_k)=\underset{u_k\in D_k}{\text{opt}}\{v_k(s_k,u_k)\cdot f_{k+1}(s_{k+1})\},\ k=n,n-1,\cdots,1 \\ f_{n+1}(s_{n+1})=1 \end{cases}$$

在顺序解法中, 有 $v_{1,k}=\prod\limits_{j=1}^{k}v_j(s_{j+1},u_j)$, 基本方程为

$$\begin{cases} f_k(s_{k+1})=\underset{u_k\in D_k}{\text{opt}}\{v_k(s_{k+1},u_k)\cdot f_{k-1}(s_k)\},\ k=1,2,\cdots,n \\ f_0(s_1)=1 \end{cases}$$

5.4 背包问题

除最优路径问题与资源分配问题外, 动态规划方法在经济管理、物流调度、资源配置等领域同样具有广泛的应用。本节选取具有代表性的背包问题进行介绍, 即例5.3。

设 x_i 为第 i 种物品装入的件数, 则背包问题可归纳为如下形式的整数规划模型

$$\max z = \sum_{i=1}^{n} c_i(x_i)$$
$$\text{s.t.} \begin{cases} \sum\limits_{i=1}^{n} a_i x_i \le a \\ x_i \ge 0 \text{且取整数} \end{cases}$$

将待装入的 n 种物品按顺序划分阶段, 每个阶段仅考虑一种物品, 共分为 n 个阶段, 即 $k = 1, 2, \cdots, n$。在第 k 阶段开始时, 背包剩余可装入前 k 种物品的总重量限额记为状态变量 s_{k+1}。决策变量为第 k 阶段装入第 k 种物品的件数 x_k, 则状态转移方程为

$$s_k = s_{k+1} - a_k x_k$$

允许决策集合为 $D_k(s_{k+1}) = \{x_k \mid 0 \le x_k \le s_{k+1}/a_k,\ x_k \text{为整数}\}$。定义最优指标函数 $f_k(s_{k+1})$, 表示在背包中允许装入物品的总重量不超过 s_{k+1}, 采用最优策略只装前 k 种物品时的最大使用价值。由此, 可得到动态规划的顺序递推方程为

$$\begin{cases} f_k(s_{k+1}) = \max\limits_{x_k = 0, 1, \cdots, [s_{k+1}/a_k]} \{c_k(x_k) + f_{k-1}(s_{k+1} - a_k x_k)\} \\ f_0(s_1) = 0 \end{cases}$$

依次计算 $f_1(s_2), f_2(s_3), \cdots, f_n(s_{n+1})$ 及对应最优决策 $x_1(s_2), x_2(s_3), \cdots, x_n(s_{n+1})$, 最终得到的 $f_n(a)$ 即为背包总重量不超过 a 时的最大价值, 相应最优方案可通过反向回溯确定。

例 5.5 某卡车的最大货运量为 10 吨, 可用以装载 A、B、C 三种货物, 各货物的单位重量 (单位: 吨) 与单位价值 (单位: 万元) 如表5.1所示。试问应如何确定装载方案, 使卡车装载的总价值最大。

解 设 x_1, x_2, x_3 分别为货物 A、B、C 的装载件数, 则该问题可表示为整数规划模型

$$\max z = 4x_1 + 5x_2 + 6x_3$$
$$\text{s.t.} \begin{cases} 3x_1 + 4x_2 + 5x_3 \le 10 \\ x_1, x_2, x_3 \ge 0 \text{且取整数} \end{cases}$$

表 5.1 各货物的单位重量及价值

货物编号	重量	价值
A	3	4
B	4	5
C	5	6

当 $k=1$ 时, 计算结果如表5.2所示, 即

$$\begin{aligned} f_1(s_2) &= \max_{0\le 3x_1\le s_2,\ x_1\text{取整数}} \{4x_1\} \\ &= \max_{0\le x_1\le s_2/3,\ x_1\text{取整数}} \{4x_1\} \end{aligned}$$

表 5.2 第 1 阶段计算结果

s_2	0	1	2	3	4	5	6	7	8	9	10
$f_1(s_2)$	0	0	0	4	4	4	8	8	8	12	12
x_1^*	0	0	0	1	1	1	2	2	2	3	3

当 $k=2$ 时, 计算结果如表5.3所示, 即

$$\begin{aligned} f_2(s_3) &= \max_{0\le 4x_2\le s_3,\ x_2\text{取整数}} \{5x_2 + f_1(s_3-4x_2)\} \\ &= \max_{0\le x_2\le s_3/4,\ x_2\text{取整数}} \{5x_2 + f_1(s_3-4x_2)\} \end{aligned}$$

表 5.3 第 2 阶段计算结果

s_3	0	1	2	3	4	5	6	7	8	9	10
x_2	0	0	0	0	0 1	0 1	0 1	0 1	0 1 2	0 1 2	0 1 2
$c_2(x_2)+f_2$	0	0	0	4	4 5	4 5	8 5	8 9	8 9 10	12 9 10	12 13 10
$f_2(s_3)$	0	0	0	4	5	5	8	9	10	12	13
x_2^*	0	0	0	0	1	1	0	1	2	0	1

当 $k=3$ 时, 有

$$\begin{aligned} f_3(s_4) &= \max_{0\le x_3\le s_4/5} \{6x_3 + f_2(s_4-5x_3)\} \\ &= \max_{x_3=0,1,2} \{6x_3 + f_2(10-5x_3)\} \\ &= \max\{f_2(10), 6+f_2(5), 12+f_2(0)\} \\ &= \max\{13, 6+5, 12+0\} \\ &= 13 \end{aligned}$$

由此可得最大价值为 13 万元, 最优决策为 $x_3^* = 0$。反向回溯可得 $x_1^* = 2$, $x_2^* = 1$, 即货物 A 装 2 件, 货物 B 装 1 件, 不装载货物 C。

5.5 应用案例

5.5.1 问题描述

随着电子商务和工业制造的发展, 智能工厂系统已成为现代物流体系中的重要环节。某企业使用移动机器人完成工厂内货物的搬运、分拣与入库, 其作业区域包含可通行区域、固定障碍物以及关键功能节点。机器人从充电起点出发, 需按指定顺序依次访问多个工作站完成任务, 最终返回指定终点。该路径规划问题具有典型的多阶段决策特征, 可采用动态规划方法求解。

例 5.6 设工厂作业区域为 6×6 网格, 用坐标 (i, j) 表示任意网格位置, 其中 $i, j = 0, 1, 2, 3, 4, 5$。机器人起点 (充电区) 位于 $\mathrm{S}(0, 0)$, 终点 (货物出口) 位于 $\mathrm{T}(5, 5)$, 固定障碍物位于 $(1, 1)$、$(2, 3)$、$(3, 2)$、$(4, 4)$, 如图5.2所示。机器人仅允许沿上、下、左、右四个方向单步移动, 且需依次访问 $\mathrm{W}_1(2, 1)$、$\mathrm{W}_2(4, 3)$、$\mathrm{W}_3(3, 5)$ 三个工作站。试求机器人完成全部任务的最短路径。

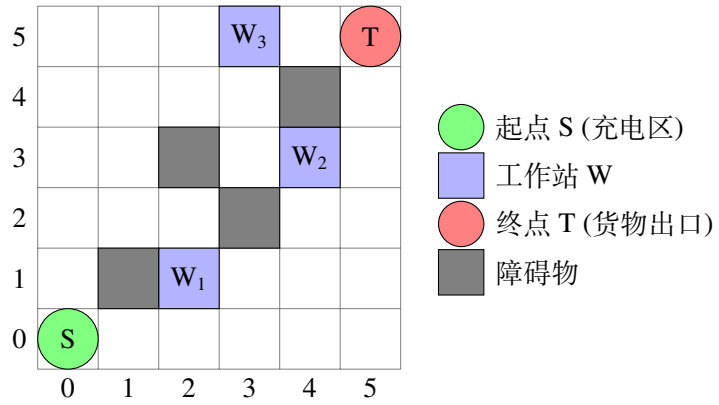

图 5.2 智能工厂机器人路径规划问题

5.5.2 模型建立

关键功能节点坐标为

$$\mathrm{S}(0, 0),\ \mathrm{W}_1(2, 1),\ \mathrm{W}_2(4, 3),\ \mathrm{W}_3(3, 5),\ \mathrm{T}(5, 5)$$

由于任务要求必须依次访问各工作站, 可将整个路径划分为 4 个阶段, 即

$$\mathrm{S} \to \mathrm{W}_1,\ \mathrm{W}_1 \to \mathrm{W}_2,\ \mathrm{W}_2 \to \mathrm{W}_3,\ \mathrm{W}_3 \to \mathrm{T}$$

5.5.3 算法设计

针对该多阶段最短路径问题, 下面分别采用动态规划的顺序解法与逆序解法进行建模与求解。

(1) 顺序解法: 设 $f_k(i,j)$ 表示从起点 S 出发, 经过前 k 个阶段后到达功能节点 (i,j) 的最短路径长度。状态转移方程为

$$f_k(i,j) = \min_{(x,y)\in D_k} \{d((x,y),(i,j)) + f_{k-1}(x,y)\},\ k = 1,2,3,4$$

其中 D_k 表示第 k 阶段的可达关键节点集合, $d((x,y),(i,j))$ 表示在给定网格结构与障碍约束下, 从 (x,y) 到 (i,j) 的最短路径长度。边界条件为 $f_0(0,0) = 0$, 其余状态取为无穷大。使用曼哈顿距离计算出图中各关键点之间的最短路径, 得到

$$d(\mathrm{S},\mathrm{W}_1) = 3,\ d(\mathrm{W}_1,\mathrm{W}_2) = 4,\ d(\mathrm{W}_2,\mathrm{W}_3) = 3,\ d(\mathrm{W}_3,\mathrm{T}) = 2$$

按阶段顺序递推

$$\begin{aligned} f_1(\mathrm{W}_1) &= f_0(\mathrm{S}) + d(\mathrm{S},\mathrm{W}_1) = 3 \\ f_2(\mathrm{W}_2) &= f_1(\mathrm{W}_1) + d(\mathrm{W}_1,\mathrm{W}_2) = 3 + 4 = 7 \\ f_3(\mathrm{W}_3) &= f_2(\mathrm{W}_2) + d(\mathrm{W}_2,\mathrm{W}_3) = 7 + 3 = 10 \\ f_4(\mathrm{T}) &= f_3(\mathrm{W}_3) + d(\mathrm{W}_3,\mathrm{T}) = 10 + 2 = 12 \end{aligned}$$

因此, 最优路径长度为 12。

(2) 逆序解法: 设 $f_k(i,j)$ 表示从位置 (i,j) 出发, 经过后续 k 个阶段到达终点 T 的最短路径长度。状态转移方程为

$$f_k(i,j) = \min_{(x,y)\in D_k} \{d((x,y),(i,j)) + f_{k+1}(x,y)\},\ k = 4,3,2,1$$

边界条件为 $f_5(5,5) = 0$, 其余状态取为无穷大。根据已计算的关键节点之间的最短距离, 可进行逆序递推

$$\begin{aligned} f_4(\mathrm{W}_3) &= d(\mathrm{W}_3,\mathrm{T}) = 2 \\ f_3(\mathrm{W}_2) &= d(\mathrm{W}_2,\mathrm{W}_3) + f_4(\mathrm{W}_3) = 3 + 2 = 5 \\ f_2(\mathrm{W}_1) &= d(\mathrm{W}_1,\mathrm{W}_2) + f_3(\mathrm{W}_2) = 4 + 5 = 9 \\ f_1(\mathrm{S}) &= d(\mathrm{S},\mathrm{W}_1) + f_2(\mathrm{W}_1) = 3 + 9 = 12 \end{aligned}$$

得到最短路径长度仍为 12。

5.5.4 结果分析

顺序解法与逆序解法从不同递推方向对多阶段决策过程进行求解, 二者最终得到的最短路径长度完全一致。若任务要求机器人完成所有工作站访问后返回起点 $S(0,0)$, 试重新计算最优路径。进一步, 若不限制各工作站的访问顺序, 求解结果会发生何种变化。请读者自行思考。

习题

5.1 现有天然气站 A, 需铺设管道至用气单位 E, 可选择的设计路线如图5.3所示, 其中 $B_1, B_2, \cdots, D_2$ 等为中间加压站, 各路段旁的数字为该段线路的铺设费用 (单位: 万元)。试用动态规划逆序解法, 设计总费用最低的管道路线。

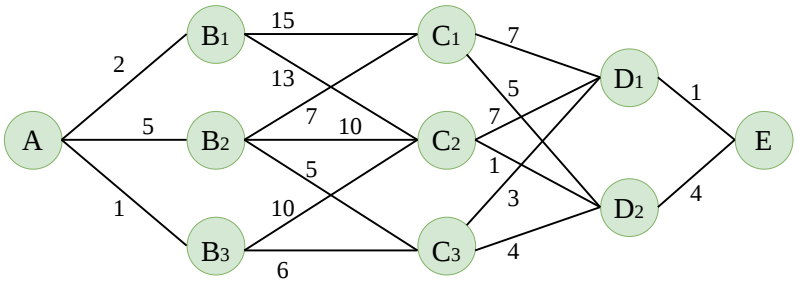

图 5.3 A 到 E 各点间费用

5.2 某公司现有可支配资金 4 万元, 拟对 A、B、C 三个项目进行投资。各项目在不同投资额下对应的效益值 (单位: 万元) 如表5.4所示。试建立动态规划模型, 并分别用逆序解法与顺序解法确定资金分配方案, 使总效益达到最大。

表 5.4 各项目在不同投资额下效益值

项目	投资额				
	0	1	2	3	4
A	0	41	48	60	66
B	0	42	50	60	66
C	0	64	68	78	76

5.3 首先用动态规划的逆序解法与顺序解法求解, 随后增加整数约束继续求解。

$$\max z = 4x_1 + 9x_2 + 2x_3^2$$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 = 10 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

5.4 分别用动态规划的逆序解法与顺序解法求解下列非线性规划问题, 并进行比较。

(1) $\max w = x_1 \cdot x_2 \cdot x_3$

$$\text{s.t.} \begin{cases} x_1 + 5x_2 + 2x_3 \leq 20 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

(2) $\max w = x_1 \cdot x_2 \cdot x_3 \cdot x_4$

$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 = 48 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

5.5 现有一辆卡车, 可装载 A、B、C、D 四种货物, 各货物的单件重量 (单位: 吨) 、占用容积 (单位: 立方米) 及单件价值 (单位: 万元) 如表5.5所示。已知卡车最大载重为 15 吨, 最大允许装载容积为 10 立方米, 每种货物可装载件数不限。试确定合理的货物装载方案, 使单车装载的总价值最大。

表 5.5 各货物单件重量、占用容积及单件价值

货物编号	货物属性		
	重量	容积	价值
A	2	2	3
B	3	2	4
C	4	2	5
D	5	3	6

6

博弈论

6.1 基本概念

6.1.1 典型案例

博弈论, 又称对策论, 广泛应用于军事对抗、政治谈判、企业竞争等众多领域。博弈问题中, 各参与主体以自身利益最大化为目标, 在预判其他主体行动选择的基础上, 制定并选取最优决策方案。除经典的“田忌赛马”外, 以下再给出若干典型示例。

例 6.1 某乡镇下一年的饮食品购买力将达到 4 000 万元, 乡镇企业与中心城市企业展开市场竞争。乡镇企业可生产特色饮食品和低档饮食品, 中心城市企业可生产高档饮食品和低档饮食品, 双方市场的收益 (单位: 万元) 如表6.1所示。试分析乡镇企业和中心城市应选择何种产品策略, 以实现自身收益最优。

表 6.1 企业市场竞争收益

乡镇企业策略	中心城市企业的策略	
	出售高档饮食品	出售低档饮食品
出售特色饮食品	2 000	3 000
出售低档饮食品	1 000	3 000

例 6.2 假定企业 I 与企业 II 均可向市场投放某一产品, 且两者均可在时间区间 $[0, 1]$ 内任意时刻完成销售。设企业 I 的销售时刻为 x, 企业 II 的销售时刻为 y, 企业 I 的赢得

函数为

$$H(x,y)=\begin{cases} c(y-x), & \text{若} x<y \\ \dfrac{1}{2}c(1-x), & \text{若} x=y \\ c(1-x), & \text{若} x>y \end{cases}$$

其中 $c\in\mathbb{R}$ 为正常数，表示单位时间的收益系数。试分析两家企业应如何选择销售时刻，才能最大化自身收益。

例 6.3 英式拍卖是最常见的拍卖形式，由拍卖方先行公示拍卖品的基本信息，竞买者依次递增报价，最终报价最高者获得拍卖品，并以其最终报价完成支付。设有 n 个竞买者参与拍卖，报价分别为 $p_1,p_2,\cdots,p_n$，且不妨设 $p_n>p_{n-1}>\cdots>p_1$。第 n 个竞买者只要报价略高于 p_{n-1}，即可以接近次高报价的成本获得拍卖品。竞买者之间可能相互知晓彼此估价，也可能处于信息不完全状态。试分析每位竞买者应如何报价，才能以较低成本获得拍卖品。

例 6.4 现有两名嫌疑犯因涉嫌重大案件被警方拘留，二人被分别审讯且无法互通信息。依据法律规定，若两人均认罪，每人判处有期徒刑 7 年。若两人均不认罪，因证据不足，每人判处有期徒刑 1 年。若仅一人认罪，认罪者予以释放，拒不认罪者判处有期徒刑 9 年。两名囚犯需在认罪与不认罪中做出抉择，该案例即经典的“囚徒困境”。

6.1.2 三要素

针对不同的实际问题可构建不同形式的博弈模型，而无论模型如何变化，均包含以下三个基本要素。

(1) 局中人：指在博弈中拥有独立决策权限、以自身利益最大化为目标的参与主体。通常用 $I=\{1,2,\cdots,n\}$ 表示所有局中人的集合，其中 n 为局中人的数量。局中人既可以是自然人，也可以是企业、组织等集体决策主体。

(2) 策略集：指供局中人可选择的、完整且可执行的行动方案集合。通常用 s_i 表示第 i 个局中人的某一策略，其全部策略构成的集合记为 S_i。由所有局中人的策略共同组成局势 $s=(s_1,s_2,\cdots,s_n)$，全体局势构成的集合记为 S。

(3) 赢得函数：指在任意局势下每个局中人的收益或损失值。通常用 $H_i(s)$ 表示第 i 个局中人在局势 s 下的赢得值，函数值为正代表局中人获得收益，函数值为负代表局中人产生损失。

以“田忌赛马”为例，局中人为齐王和田忌，记为 $I=\{1,2\}$。用 (上, 中, 下) 表示按照上等马、中等马、下等马依次参赛，即为一个完整的策略。齐王和田忌各自拥有 6 个策

略, 分别为 (上, 中, 下)、(上, 下, 中)、(中, 上, 下)、(中, 下, 上)、(下, 中, 上)、(下, 上, 中)。设齐王的策略集为 $S_1 = \{\alpha_1, \alpha_2, \cdots, \alpha_6\}$, 田忌的策略集为 $S_2 = \{\beta_1, \beta_2, \cdots, \beta_6\}$, 任意策略组合 (α_i, β_j) 构成一个局势 s_{ij}, 对应齐王与田忌的赢得值分别为 $H_1(s_{ij})$ 和 $H_2(s_{ij})$。若 $\alpha_1 =$ (上, 中, 下), $\beta_1 =$ (上, 中, 下), 则在局势 s_{11} 下, 齐王的赢得值为 $H_1(s_{11}) = 3$, 田忌的赢得值为 $H_2(s_{11}) = -3$。

6.1.3 分类

为便于对不同类型的博弈问题开展研究, 通常会依据一定标准对其进行分类。

(1) 按局中人的数量, 博弈问题可分为二人博弈和多人博弈。例如“田忌赛马”“饮食品市场竞争”均属于二人博弈。

(2) 按各局中人的赢得函数的代数和是否为零, 博弈问题可分为零和博弈与非零和博弈。例如“田忌赛马”属于零和博弈, “囚徒困境”属于非零和博弈。

(3) 按各局中人是否允许合作, 博弈问题可分为合作博弈和非合作博弈。前述典型案例均属于非合作博弈。

(4) 按局中人策略集中的策略数量, 博弈问题可分为有限博弈和无限博弈。例如“田忌赛马”属于有限博弈, “企业销售时机”属于无限博弈。

除此之外, 博弈问题还可从其他维度划分。按策略选择是否与时间相关, 可分为静态博弈与动态博弈。按数学结构与分析方法, 可分为矩阵博弈、连续博弈、微分博弈、凸博弈以及随机博弈等。

6.2 矩阵博弈模型

6.2.1 纯策略

矩阵博弈, 即二人有限零和博弈。设局中人 I 有 m 个纯策略 $\alpha_1, \alpha_2, \cdots, \alpha_m$, 局中人 II 有 n 个纯策略 $\beta_1, \beta_2, \cdots, \beta_n$, 则双方的策略集分别为 $S_1 = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ 与 $S_2 = \{\beta_1, \beta_2, \cdots, \beta_n\}$。当局中人 I 选取纯策略 α_i、局中人 II 选取纯策略 β_j 时, 构成纯局势 (α_i, β_j), 全部纯局势的总数为 $m \times n$ 个。对任意纯局势 (α_i, β_j), 记局中人 I 的赢得值为 a_{ij}, 则称矩阵

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

为局中人 I 的赢得矩阵。根据零和博弈的基本性质, 局中人 II 的赢得矩阵为 $-\boldsymbol{A}$。当局中人 I 和局中人 II 的策略集 S_1, S_2 及赢得矩阵 A 均确定时, 一个矩阵博弈即被完全确定, 记为 $G = \{S_1, S_2; \boldsymbol{A}\}$。仍以“田忌赛马”为例, 齐王的赢得矩阵为

$$\boldsymbol{A} = \begin{pmatrix} 3 & 1 & 1 & 1 & 1 & -1 \\ 1 & 3 & 1 & 1 & -1 & 1 \\ 1 & -1 & 3 & 1 & 1 & 1 \\ -1 & 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & -1 & 1 & 3 & 1 \\ 1 & 1 & 1 & -1 & 1 & 3 \end{pmatrix}$$

当 G 确定后, 各局中人将考虑如何选取对自身最优的策略, 实现自身赢得的最大化。本章设所有局中人均为完全理性, 即始终以自身收益的最大化为唯一决策目标, 不存在利用局中人决策的失误来扩大自身利益的行为。

例 6.5 设有矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$\boldsymbol{A} = \begin{pmatrix} -6 & 1 & -8 \\ 3 & 2 & 4 \\ 9 & -1 & -10 \\ -3 & 0 & 6 \end{pmatrix}$$

解 由赢得矩阵 $\boldsymbol{A}$ 可知, 局中人 I 各纯策略对应的最小赢得依次为 $-8, 2, -10, -3$, 其中最大值为 2, 因此局中人 I 应选取策略 α_2。同理, 局中人 II 各纯策略对应的最大损失依次为 $9, 2, 6$, 其中最小值为 2, 因此局中人 II 应选取策略 β_2。由此得到纯局势 (α_2, β_2), 对应的赢得值为 2。

定义 6.1

设有矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中 $S_1 = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$, $S_2 = \{\beta_1, \beta_2, \cdots, \beta_n\}$, $\boldsymbol{A} = (a_{ij})_{m\times n}$。若

$$\max_i \min_j a_{ij} = \min_j \max_i a_{ij}$$

成立, 记其值为 V_G, 则称 V_G 为博弈值, 称使上式成立的纯局势 (α_i^*, β_j^*) 为博弈 G 在纯策略意义下的解, 称 α_i^* 和 β_j^* 分别为局中人 I 和局中人 II 的最优纯策略。

定理 6.2

矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$ 在纯策略意义下有解的充分必要条件是，存在纯局势 (α_i^*, β_j^*)，使得对任意 i, j，有

$$a_{ij^*} \le a_{i^*j^*} \le a_{i^*j}$$

证明 设存在 i^*, j^* 满足

$$\min_j a_{i^*j} = \max_i \min_j a_{ij}, \quad \max_i a_{ij^*} = \min_j \max_i a_{ij}$$

由 $\max\limits_i \min\limits_j a_{ij} = \min\limits_j \max\limits_i a_{ij}$，可得

$$\max_i a_{ij^*} = \min_j a_{i^*j} \le a_{i^*j^*} \le \max_i a_{ij^*} = \min_j a_{i^*j}$$

因此对任意 i, j，有

$$a_{ij^*} \le a_{i^*j^*} \le a_{i^*j}$$

另一方面，若 $a_{ij^*} \le a_{i^*j^*} \le a_{i^*j}$ 对任意 i, j 成立，则

$$\max_i a_{ij^*} \le a_{i^*j^*} \le \min_j a_{i^*j}$$

又由不等式基本性质知

$$\min_j \max_i a_{ij} \le \max_i a_{ij^*}, \quad \min_j a_{i^*j} \le \max_i \min_j a_{ij}$$

因此

$$\min_j \max_i a_{ij} \le a_{i^*j^*} \le \max_i \min_j a_{ij}$$

对任意 i, j，有

$$\max_i \min_j a_{ij} \le \min_j \max_i a_{ij}$$

于是

$$\max_i \min_j a_{ij} = \min_j \max_i a_{ij} = a_{i^*j^*}$$

即博弈在纯策略意义下有解。

称满足定理6.2的元素 $a_{i^*j^*}$ 为赢得矩阵 $\boldsymbol{A}$ 的鞍点，表示博弈双方在局势 (α_i^*, β_j^*) 下达到平衡状态。也就是说，当局中人 I 选定最优纯策略 α_i^* 后，局中人 II 为最小化自身损失，仅能选择纯策略 β_j^*，否则损失将扩大。同理，当局中人 II 选定纯策略 β_j^* 后，局中人 I 为最大化自身赢得，仅能选择纯策略 α_i^*，否则收益将减少。

例 6.6 在纯策略意义下求解矩阵博弈 $G=\{S_1,S_2;\boldsymbol{A}\}$, 其中

$$\boldsymbol{A}=\begin{pmatrix}9 & 8 & 11 & 8\\ 2 & 4 & 6 & 3\\ 5 & 6 & 7 & 4\\ 10 & 7 & 9 & 6\end{pmatrix}$$

解 通过赢得矩阵 $\boldsymbol{A}$ 计算可得

$$\max_i \min_j a_{ij}=\min_j \max_i a_{ij}=8$$

即满足鞍点条件的元素为 $a_{i^*j^*}=8$, 其中 $i^*=1,\ j^*=2,4$。因此

$$(\alpha_1,\beta_2),\ (\alpha_1,\beta_4)$$

均为博弈 G 的解, 且博弈值为 $V_G=8$。

性质 6.3

纯策略意义下的解未必唯一, 但博弈值必定唯一。

性质 6.4

若 $(\alpha_{i_1},\beta_{j_1})$ 与 $(\alpha_{i_2},\beta_{j_2})$ 为博弈 G 在纯策略意义下的解, 则 $a_{i_1j_1}=a_{i_2j_2}$。

性质 6.5

若 $(\alpha_{i_1},\beta_{j_1})$ 与 $(\alpha_{i_2},\beta_{j_2})$ 为博弈 G 在纯策略意义下的解, 则 $(\alpha_{i_1},\beta_{j_2})$ 与 $(\alpha_{i_2},\beta_{j_1})$ 也为博弈 G 在纯策略意义下的解。

6.2.2 混合策略

考虑矩阵博弈 $G=\{S_1,S_2;\boldsymbol{A}\}$, 局中人 I 能保证的最小赢得为

$$v_1=\max_i \min_j\ a_{ij}$$

局中人 II 能保证的最大所失为

$$v_2=\min_j \max_i\ a_{ij}$$

同时, 局中人 I 的赢得不超过局中人 II 的损失, 即恒有

$$v_1\le v_2$$

当 $v_1 = v_2$ 时, 矩阵博弈存在纯策略意义下的解, 且 $V_G = v_1 = v_2$。在实际问题中, 更为普遍的情形是 $v_1 < v_2$, 此时矩阵博弈不存在纯策略意义下的解。例如, 赢得矩阵为

$$\boldsymbol{A} = \begin{pmatrix} 3 & 6 \\ 5 & 4 \end{pmatrix}$$

经计算可得

$$v_1 = \max_i \min_j \ a_{ij} = 4, \ i^* = 2$$

$$v_2 = \min_j \max_i \ a_{ij} = 5, \ j^* = 1$$

$$v_2 = 5 > 4 = v_1$$

因此, 该博弈在纯策略意义下无解。在此情形下, 博弈双方能够以一定概率分布选取不同纯策略的方式参与博弈。例如局中人 I 以概率 x 和 $(1-x)$ 分别选取纯策略 α_1 和 α_2, 该策略称为混合策略。同理, 局中人 II 也可定义相应的混合策略。下面给出矩阵博弈混合策略及混合策略意义下解的定义。

定义 6.6

设有矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中 $S_1 = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$, $S_2 = \{\beta_1, \beta_2, \cdots, \beta_n\}$, $\boldsymbol{A} = (a_{ij})_{m\times n}$。记

$$S_1^* = \{\boldsymbol{x} \in \mathbb{R}^m \mid \sum_{i=1}^m x_i = 1, \ x_i \geq 0, \ i = 1, 2, \cdots, m\}$$

$$S_2^* = \{\boldsymbol{y} \in \mathbb{R}^n \mid \sum_{j=1}^n y_j = 1, \ y_j \geq 0, \ j = 1, 2, \cdots, n\}$$

称 S_1^* 和 S_2^* 分别为局中人 I 和局中人 II 的混合策略集, 称 $\boldsymbol{x} \in S_1^*$, $\boldsymbol{y} \in S_2^*$ 为混合策略, 称 $(\boldsymbol{x}, \boldsymbol{y})$ 为混合局势。

在混合策略意义下, 局中人 I 的赢得函数为

$$E(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^{\mathrm{T}} A \boldsymbol{y} = \sum_i^m \sum_j^n a_{ij} x_i y_j$$

称 $G^* = \{S_1^*, S_2^*; E\}$ 为矩阵博弈 G 的混合扩充。显然, 纯策略是混合策略的特殊情形。混合策略 $\boldsymbol{x} = (x_1, x_2, \cdots, x_m)^{\mathrm{T}}$ 可理解为, 若重复进行多局博弈, 则 x_i 表示局中人 I 选取纯策略 α_i 的频率。若仅进行单局博弈, 则反映局中人 I 对各纯策略的偏好程度。

设博弈双方仍遵循完全理性原则, 局中人 I 选择混合策略 $\boldsymbol{x}$ 时, 其最不利情形下的

赢得为 $\min\limits_{\boldsymbol{y}\in S_2^*} E(\boldsymbol{x},\boldsymbol{y})$。因此, 局中人 I 的目标为选取 $\boldsymbol{x}\in S_1^*$, 使得

$$v_1 = \max_{\boldsymbol{x}\in S_1^*}\min_{\boldsymbol{y}\in S_2^*} E(\boldsymbol{x},\boldsymbol{y})$$

同理, 局中人 II 可保证的最大损失为

$$v_2 = \min_{\boldsymbol{y}\in S_2^*}\max_{\boldsymbol{x}\in S_1^*} E(\boldsymbol{x},\boldsymbol{y})$$

且仍满足 $v_1 \le v_2$, 即局中人 I 的赢得不超过局中人 II 的损失。

定义 6.7

设矩阵博弈 $G^* = \{S_1^*, S_2^*; E\}$ 为矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$ 的混合扩充, 若

$$\max_{\boldsymbol{x}\in S_1^*}\min_{\boldsymbol{y}\in S_2^*} E(\boldsymbol{x},\boldsymbol{y}) = \min_{\boldsymbol{y}\in S_2^*}\max_{\boldsymbol{x}\in S_1^*} E(\boldsymbol{x},\boldsymbol{y})$$

成立, 记其值为 V_G, 称使上式成立的混合局势 $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ 为 G 在混合策略意义下的解, 称 $\boldsymbol{x}^*,\boldsymbol{y}^*$ 分别为局中人 I 和局中人 II 的最优混合策略。

定理 6.8

矩阵博弈 G 在混合策略意义下有解的充分必要条件是, 存在 $\boldsymbol{x}^*\in S_1^*$, $\boldsymbol{y}^*\in S_2^*$ 使得对任意 $\boldsymbol{x}\in S_1^*$, $\boldsymbol{y}\in S_2^*$, 有

$$E(\boldsymbol{x},\boldsymbol{y}^*) \le E(\boldsymbol{x}^*,\boldsymbol{y}^*) \le E(\boldsymbol{x}^*,\boldsymbol{y})$$

例 6.7 在混合策略意义下求解矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$\boldsymbol{A} = \begin{pmatrix} 3 & 6 \\ 5 & 4 \end{pmatrix}$$

解 设 $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{y} = (y_1, y_2)$ 分别为局中人 I 和局中人 II 的混合策略, 则混合策略集可表示为

$$S_1^* = \{(x_1,x_2) \mid x_1, x_2 \ge 0,\ x_1 + x_2 = 1\}$$

$$S_2^* = \{(y_1,y_2) \mid y_1, y_2 \ge 0,\ y_1 + y_2 = 1\}$$

局中人 I 的赢得为

$$\begin{aligned} E(\boldsymbol{x},\boldsymbol{y}) &= 3x_1y_1 + 6x_1y_2 + 5x_2y_1 + 4x_2y_2 \\ &= 3x_1y_1 + 6x_1(1-y_1) + 5(1-x_1)y_1 + 4(1-x_1)(1-y_1) \\ &= -4\left(x_1 - \frac{1}{4}\right)\left(y_1 - \frac{1}{2}\right) + \frac{9}{2} \end{aligned}$$

取 $\boldsymbol{x}^* = \left(\frac{1}{4}, \frac{3}{4}\right)$, $\boldsymbol{y}^* = \left(\frac{1}{2}, \frac{1}{2}\right)$, 则

$$E(\boldsymbol{x}^*,\boldsymbol{y}^*) = \frac{9}{2},\ E(\boldsymbol{x}^*,\boldsymbol{y}) = E(\boldsymbol{x},\boldsymbol{y}^*) = \frac{9}{2}$$

满足不等式 $E(\boldsymbol{x},\boldsymbol{y}^*) \le E(\boldsymbol{x}^*,\boldsymbol{y}^*) \le E(\boldsymbol{x}^*,\boldsymbol{y})$。因此, $\boldsymbol{x}^* = \left(\frac{1}{4}, \frac{3}{4}\right)$ 为局中人 I 的最优混合策略, $\boldsymbol{y}^* = \left(\frac{1}{2}, \frac{1}{2}\right)$ 为局中人 II 的最优混合策略, 博弈值为 $V_G = \frac{9}{2}$。

6.2.3 基本理论

为便于后续表述, 引入如下记号

$$E(i,\boldsymbol{y}) = \sum_j^n a_{ij}y_j,\ E(\boldsymbol{x},j) = \sum_i^m a_{ij}x_i$$

则 $E(i,\boldsymbol{y})$ 为局中人 I 取纯策略 α_i 时的赢得值, $E(\boldsymbol{x},j)$ 为局中人 II 取纯策略 β_j 时的赢得值。矩阵博弈的混合策略赢得可表示为

$$E(\boldsymbol{x},\boldsymbol{y}) = \sum_i^m\sum_j^n a_{ij}x_iy_j = \sum_i^m\left(\sum_j^n a_{ij}y_j\right)x_i = \sum_i^m E(i,\boldsymbol{y})x_i \tag{6.1}$$

同时, 该赢得也满足

$$E(\boldsymbol{x},\boldsymbol{y}) = \sum_i^m\sum_j^n a_{ij}x_iy_j = \sum_j^n\left(\sum_i^m a_{ij}x_i\right)y_j = \sum_j^n E(\boldsymbol{x},j)y_j \tag{6.2}$$

定理 6.9

设 $\boldsymbol{x}^* \in S_1^*$, $\boldsymbol{y}^* \in S_2^*$, 则 $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ 为矩阵博弈 G 的解的充分必要条件是, 对任意 $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$, 有

$$E(i,\boldsymbol{y}^*) \le E(\boldsymbol{x}^*,\boldsymbol{y}^*) \le E(\boldsymbol{x}^*,j)$$

证明 设 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 为矩阵博弈 G 的解, 则根据解的基本定义, 可得

$$E(\boldsymbol{x}, \boldsymbol{y}^*) \leq E(\boldsymbol{x}^*, \boldsymbol{y}^*) \leq E(\boldsymbol{x}^*, \boldsymbol{y})$$

由于纯策略是混合策略的特殊情形, 将纯策略代入上式, 即得必要性。另一方面, 若定理中不等式成立, 结合式(6.1)和(6.2), 可得

$$E(\boldsymbol{x}, \boldsymbol{y}^*) = \sum_i^m E(i, \boldsymbol{y}^*) x_i \leq E(\boldsymbol{x}^*, \boldsymbol{y}^*) \sum_i^m x_i = E(\boldsymbol{x}^*, \boldsymbol{y}^*)$$

$$E(\boldsymbol{x}^*, \boldsymbol{y}) = \sum_j^n E(\boldsymbol{x}^*, j) y_j \geq E(\boldsymbol{x}^*, \boldsymbol{y}^*) \sum_j^n y_j = E(\boldsymbol{x}^*, \boldsymbol{y}^*)$$

因此 $E(\boldsymbol{x}, \boldsymbol{y}^*) \leq E(\boldsymbol{x}^*, \boldsymbol{y}^*) \leq E(\boldsymbol{x}^*, \boldsymbol{y})$ 成立, 即 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 为矩阵博弈 G 的解。

定理 6.10

设 $\boldsymbol{x}^* \in S_1^*$, $\boldsymbol{y}^* \in S_2^*$, 则 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 为矩阵博弈 G 的解的充分必要条件是, 存在 v, 使得 $\boldsymbol{x}^*, \boldsymbol{y}^*$ 分别为不等式组

$$\begin{cases} \sum_i^m a_{ij} x_i \geq v, \ j = 1, 2, \cdots, n \\ \sum_i^m x_i = 1 \\ x_i \geq 0, \ i = 1, 2, \cdots, m \end{cases} \tag{6.3}$$

与

$$\begin{cases} \sum_j^n a_{ij} y_j \leq v, \ i = 1, 2, \cdots, m \\ \sum_j^n y_j = 1 \\ y_j \geq 0, \ j = 1, 2, \cdots, n \end{cases} \tag{6.4}$$

的解, 且 $v = V_G$。

性质 6.11

对任意矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 一定存在混合策略意义下的解。

证明 只需证明存在 $\boldsymbol{x}^* \in S_1^*$, $\boldsymbol{y}^* \in S_2^*$ 使得上述定理中的不等式组成立。为此, 构造线性

规划问题

$$\max\ w$$
$$\text{s.t.}\ \begin{cases} \sum\limits_{i}^{m} a_{ij}x_i \geq w,\ j=1,2,\cdots,n \\ \sum\limits_{i}^{m} x_i = 1 \\ x_i \geq 0,\ i=1,2,\cdots,m \end{cases} \tag{6.5}$$

以及

$$\min\ v$$
$$\text{s.t.}\ \begin{cases} \sum\limits_{j}^{n} a_{ij}y_j \leq v,\ i=1,2,\cdots,m \\ \sum\limits_{j}^{n} y_j = 1 \\ y_j \geq 0,\ j=1,2,\cdots,n \end{cases} \tag{6.6}$$

容易验证, 问题(6.5)与(6.6)互为对偶问题。取

$$\boldsymbol{x} = (1,0,\cdots,0)^{\mathrm{T}} \in \mathbb{R}^m,\ w = \min_j\ a_{1j}$$
$$\boldsymbol{y} = (1,0,\cdots,0)^{\mathrm{T}} \in \mathbb{R}^n,\ v = \max_i\ a_{i1}$$

则 $(\boldsymbol{x}, w)$, $(\boldsymbol{y}, v)$ 分别为问题(6.5)与(6.6)的可行解, 即两个问题均存在可行解。根据对偶理论, 问题(6.5)与(6.6)分别存在最优解 $(\boldsymbol{x}^*, w^*)$ 与 $(\boldsymbol{y}^*, v^*)$, 且最优值满足 $w^* = v^*$。因此, 存在 $\boldsymbol{x}^* \in S_1^*$, $\boldsymbol{y}^* \in S_2^*$ 及 v^*, 使得对任意 $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$, 有

$$\sum_{j}^{n} a_{ij}y_j^* \leq v^* \leq \sum_{i}^{m} a_{ij}x_i^*$$

上式等价于

$$E(i, \boldsymbol{y}^*) \leq v^* \leq E(\boldsymbol{x}^*, j)$$

进一步结合式(6.1)和(6.2), 可得

$$E(\boldsymbol{x}^*, \boldsymbol{y}^*) = \sum_{i}^{m} E(i, \boldsymbol{y}^*)x_i^* \leq v^* \sum_{i}^{m} x_i^* = v^*$$
$$E(\boldsymbol{x}^*, \boldsymbol{y}^*) = \sum_{j}^{n} E(\boldsymbol{x}^*, j)y_j^* \geq v^* \sum_{j}^{n} y_j^* = v^*$$

进而 $v^* = E(\boldsymbol{x}^*, \boldsymbol{y}^*)$, 即

$$E(i, \boldsymbol{y}^*) \leq E(\boldsymbol{x}^*, \boldsymbol{y}^*) \leq E(\boldsymbol{x}^*, j)$$

成立, 故该矩阵博弈在混合策略意义下有解。

定理 6.12

设 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 为矩阵博弈 G 的解, 且 $v = V_G$, 则

(1) 若 $x_i^* > 0$, 则 $\sum\limits_{j}^{n} a_{ij} y_j^* = v$。反之, 若 $\sum\limits_{j}^{n} a_{ij} y_j^* < v$, 则 $x_i^* = 0$。

(2) 若 $y_j^* > 0$, 则 $\sum\limits_{i}^{m} a_{ij} x_i^* = v$。反之, 若 $\sum\limits_{i}^{m} a_{ij} x_i^* > v$, 则 $y_j^* = 0$。

证明 由定义 $v = \max\limits_{\boldsymbol{x} \in S_1^*} E(\boldsymbol{x}, \boldsymbol{y}^*)$, 可得

$$v - \sum_{j}^{n} a_{ij} y_j^* = \max_{\boldsymbol{x} \in S_1^*} E(\boldsymbol{x}, \boldsymbol{y}^*) - E(\boldsymbol{x}, \boldsymbol{y}^*) \geq 0$$

又因为

$$\sum_{i}^{m} x_i^* (v - \sum_{j}^{n} a_{ij} y_j^*) = v - \sum_{i}^{m} \sum_{j}^{n} a_{ij} x_i^* y_j^* = 0$$

因此, 当 $x_i^* > 0$ 时, 必有 $\sum\limits_{j}^{n} a_{ij} y_j^* = v$。当 $\sum\limits_{j}^{n} a_{ij} y_j^* < v$ 时, 必有 $x_i^* = 0$, 结论 (1) 得证。采用类似的推导方法, 可证结论 (2) 成立。

6.3 矩阵博弈求解

6.3.1 图解法

图解法适用于赢得矩阵 $\boldsymbol{A}$ 为 $2 \times n$ 或 $m \times 2$ 阶的矩阵博弈问题, 该方法求解效率较高, 且几何意义直观清晰。具体求解步骤如下。

(1) 将局中人的混合策略表示为 $(x, 1-x)^{\mathrm{T}}$ 或 $(y, 1-y)^{\mathrm{T}}$, 其中参数取值范围为 $[0, 1]$。

(2) 在一维坐标轴上选取 0 与 1 两点, 分别作垂直于坐标轴的直线, 用以刻画对应纯策略下的赢得取值。

(3) 结合赢得矩阵, 作出相应的线性函数图形, 使每条直线表示对方采取某一纯策略时的赢得函数。

(4) 比较线性函数在区间内的上下界, 依据极小极大原则确定参数取值, 从而得到最优混合策略与博弈值。

例 6.8 用图解法求解矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$\boldsymbol{A} = \begin{pmatrix} 2 & 3 & 11 \\ 7 & 5 & 2 \end{pmatrix}$$

解 设局中人 I 的混合策略为 $(x, 1-x)^{\mathrm{T}}$, 其中 $x \in [0,1]$。在坐标轴上 0 和 1 处分别作垂线 I-I 和 II-II, 如图6.1所示。垂线上的纵坐标分别对应局中人 I 采取纯策略 α_1, α_2 时, 局中人 II 采取各纯策略时的赢得值。

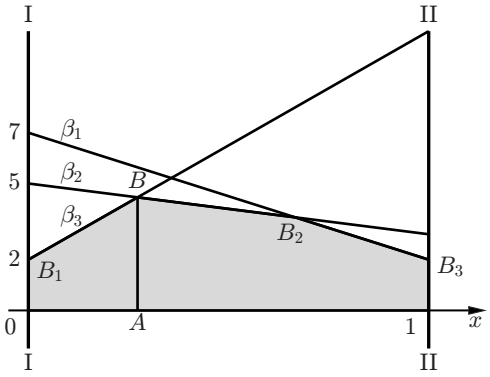

图 6.1 矩阵博弈图解法

当局中人 I 选择某一混合策略 $(x, 1-x)^{\mathrm{T}}$ 后, 其最小赢得由 $\beta_1, \beta_2, \beta_3$ 所确定的三条直线在 x 处的纵坐标中最小者决定。依据极小极大准则, 局中人 I 的最优决策为选取合适的 x, 使得该最小值达到最大, 几何上对应选取 $x = A$, 此时 B 点的纵坐标即为博弈值 V_G。联立过 B 的直线方程有

$$\begin{cases} 3x + 5(1-x) = V_G \\ 11x + 2(1-x) = V_G \end{cases}$$

解得 $x = \dfrac{3}{11}$, $V_G = \dfrac{49}{11}$。因此, 局中人 I 的最优混合策略为

$$\boldsymbol{x}^* = (x_1^*, x_2^*)^{\mathrm{T}} = \left(\frac{3}{11}, \frac{8}{11}\right)^{\mathrm{T}}$$

设局中人 II 的最优混合策略为 $\boldsymbol{y}^* = (y_1^*, y_2^*, y_3^*)^{\mathrm{T}}$, 计算可得

$$E(\boldsymbol{x}^*, 1) = 2 \times \frac{3}{11} + 7 \times \frac{8}{11} = \frac{62}{11} > \frac{49}{11} = V_G$$

根据互补松弛定理6.12, 有 $y_1^* = 0$。又 $x_1^* = \dfrac{3}{11} > 0$, $x_2^* = \dfrac{8}{11} > 0$, 建立方程组

$$\begin{cases} 3y_2 + 11y_3 = \dfrac{49}{11} \\ 5y_2 + 2y_3 = \dfrac{49}{11} \\ y_2 + y_3 = 1 \end{cases}$$

解得 $y_2^* = \dfrac{9}{11}$, $y_3^* = \dfrac{2}{11}$。综上, 局中人 II 的最优混合策略为

$$\boldsymbol{y}^* = \left(0, \frac{9}{11}, \frac{2}{11}\right)^{\mathrm{T}}$$

6.3.2 方程组法

矩阵博弈的求解等价于求解相应的不等式组(6.3)和不等式组(6.4)。若最优策略中的 x_i^* 和 y_j^* 均不为零, 则不等式组可退化为

$$\begin{cases} \sum\limits_{i}^{m} a_{ij} x_i = v, \ j = 1, 2, \cdots, n \\ \sum\limits_{i}^{m} x_i = 1 \end{cases} \tag{6.7}$$

与

$$\begin{cases} \sum\limits_{j}^{n} a_{ij} y_j = v, \ i = 1, 2, \cdots, m \\ \sum\limits_{j}^{n} y_j = 1 \end{cases} \tag{6.8}$$

若方程组(6.7)和(6.8)存在非负解 x_i^*, y_j^*, 则可得到该矩阵博弈的一组解。若上述方程组不存在非负解, 则可根据实际情况, 将方程组(6.7)和(6.8)中的部分等式替换为不等式, 继续试求解, 直至得到博弈的解。该方法预先假定 x_i^*, y_j^* 均不为零, 因此当最优策略的某些分量为零时, 方程组(6.7)和(6.8)可能无解, 这使得该方法在实际应用中存在一定的局限性。但对于 2×2 阶矩阵博弈, 当局中人 I 的赢得矩阵

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

不存在鞍点时, 可知各局中人的最优混合策略中的 x_i^*, y_j^* 均大于零。于是, 方程组

$$\begin{cases} a_{11} x_1 + a_{21} x_2 = v \\ a_{12} x_1 + a_{22} x_2 = v \\ x_1 + x_2 = 1 \end{cases} \quad \text{与} \quad \begin{cases} a_{11} y_1 + a_{12} y_2 = v \\ a_{21} y_1 + a_{22} y_2 = v \\ y_1 + y_2 = 1 \end{cases}$$

一定有严格的非负解, 解即为双方局中人的最优混合策略, 其解析表达式为

$$x_1^* = \frac{a_{22}-a_{21}}{(a_{11}+a_{22})-(a_{12}+a_{21})},\ x_2^* = \frac{a_{11}-a_{12}}{(a_{11}+a_{22})-(a_{12}+a_{21})}$$

$$y_1^* = \frac{a_{22}-a_{12}}{(a_{11}+a_{22})-(a_{12}+a_{21})},\ y_2^* = \frac{a_{11}-a_{21}}{(a_{11}+a_{22})-(a_{12}+a_{21})}$$

代入得博弈值为

$$v^* = \frac{a_{11}a_{22}-a_{12}a_{21}}{(a_{11}+a_{22})-(a_{12}+a_{21})} = V_G$$

定义 6.13

设有矩阵博弈 $G=\{S_1,S_2;\boldsymbol{A}\}$, 其中 $i=1,2,\cdots,m,\ j=1,2,\cdots,n$。若 $a_{kj}\geq a_{lj}$, 则称局中人 I 的策略 k 优超于策略 l。反之, 若 $a_{ik}\leq a_{il}$, 则称局中人 II 的策略 k 优超于策略 l。

局中人 I 的策略 k 优超于策略 l 说明对局中人 I 而言当其采用策略 k, 无论局中人 II 采用任何策略, 其获得都比采用策略 l 来的好。于是, 策略 l 出现的概率为 0, 可以在赢得矩阵中删除该策略对应的行。

例 6.9 用方程组法求解矩阵博弈 $G=\{S_1,S_2;\boldsymbol{A}\}$, 其中

$$\boldsymbol{A}=\begin{pmatrix}3&4&0&3&0\\5&0&2&5&9\\7&3&9&5&9\\4&6&8&7&6\\6&0&8&8&3\end{pmatrix}$$

解 首先利用矩阵博弈的优超原则对赢得矩阵 $\boldsymbol{A}$ 进行化简, 依次得到

$$\boldsymbol{A}_1=\begin{pmatrix}7&3&9&5&9\\4&6&8&7&6\\6&0&8&8&3\end{pmatrix},\ \boldsymbol{A}_2=\begin{pmatrix}7&3\\4&6\\6&0\end{pmatrix},\ \boldsymbol{A}_3=\begin{pmatrix}7&3\\4&6\end{pmatrix}$$

易知 $\boldsymbol{A}_3$ 没有鞍点, 构建并求解方程组

$$\begin{cases}7x_3+4x_4=v\\3x_3+6x_4=v\\x_3+x_4=1\end{cases}\quad 与\quad \begin{cases}7y_1+3y_2=v\\4y_1+6y_2=v\\y_1+y_2=1\end{cases}$$

解得

$$x_3^* = \frac{1}{3},\ x_4^* = \frac{2}{3},\ y_1^* = \frac{1}{2},\ y_2^* = \frac{1}{2},\ v = 5$$

对应赢得矩阵 $\boldsymbol{A}$ 的解为

$$\boldsymbol{x}^* = \left(0, 0, \frac{1}{3}, \frac{2}{3}, 0\right)^{\mathrm{T}},\ \boldsymbol{y}^* = \left(\frac{1}{2}, \frac{1}{2}, 0, 0, 0\right)^{\mathrm{T}},\ V_G = 5$$

6.3.3 线性规划法

任意矩阵博弈的求解均可等价地转化为一对互为对偶的线性规划问题(6.5)与 (6.6)。在问题(6.5)中, 不妨设 $w > 0$, 作变量替换 $x_i' = \dfrac{x_i}{w},\ i = 1, 2, \cdots, m$, 则问题(6.5)等价于线性规划问题

$$\min \sum_i^m x_i'$$

$$\text{s.t.} \begin{cases} \sum_i^m a_{ij} x_i' \geq 1,\ j = 1, 2, \cdots, n \\ x_i' \geq 0,\ i = 1, 2, \cdots, m \end{cases}$$

同理, 令 $y_j' = \dfrac{y_j}{v},\ j = 1, 2, \cdots, n$, 可知问题 (6.6) 等价于线性规划问题

$$\max \sum_j^n y_j'$$

$$\text{s.t.} \begin{cases} \sum_j^n a_{ij} y_j' \leq 1,\ i = 1, 2, \cdots, m \\ y_j' \geq 0,\ j = 1, 2, \cdots, n \end{cases}$$

上述两个问题互为对偶线性规划, 可采用单纯形法求解。再通过变量逆变换, 即可得到矩阵博弈的最优混合策略与博弈值。

例 6.10 用线性规划法求解矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$\boldsymbol{A} = \begin{pmatrix} 7 & 2 & 9 \\ 2 & 9 & 0 \\ 9 & 0 & 11 \end{pmatrix}$$

解 该博弈等价于如下互为对偶的线性规划问题

$$\min\ x_1+x_2+x_3 \quad \text{s.t.}\ \begin{cases} 7x_1+2x_2+9x_3\geq 1 \\ 2x_1+9x_2\geq 1 \\ 9x_1+11x_3\geq 1 \\ x_1,x_2,x_3\geq 0 \end{cases} \quad \text{与} \quad \max\ y_1+y_2+y_3 \quad \text{s.t.}\ \begin{cases} 7y_1+2y_2+9y_3\leq 1 \\ 2y_1+9y_2\leq 1 \\ 9y_1+11y_3\leq 1 \\ y_1,y_2,y_3\geq 0 \end{cases}$$

求解可得

$$\boldsymbol{x}=\left(\frac{1}{20},\frac{1}{10},\frac{1}{20}\right)^{\mathrm{T}},\ w=\frac{1}{5}$$

$$\boldsymbol{y}=\left(\frac{1}{20},\frac{1}{10},\frac{1}{20}\right)^{\mathrm{T}},\ v=\frac{1}{5}$$

因此, 矩阵博弈的最优策略与博弈值为

$$V_G=\frac{1}{w}=\frac{1}{v}=5$$

$$\boldsymbol{x}^*=V_G\boldsymbol{x}=5\left(\frac{1}{20},\frac{1}{10},\frac{1}{20}\right)^{\mathrm{T}}=\left(\frac{1}{4},\frac{1}{2},\frac{1}{4}\right)^{\mathrm{T}}$$

$$\boldsymbol{y}^*=V_G\boldsymbol{y}=5\left(\frac{1}{20},\frac{1}{10},\frac{1}{20}\right)^{\mathrm{T}}=\left(\frac{1}{4},\frac{1}{2},\frac{1}{4}\right)^{\mathrm{T}}$$

习题

6.1 在纯策略意义下求解矩阵博弈 $G=\{S_1,S_2;\boldsymbol{A}\}$, 其中

$$\boldsymbol{A}=\begin{pmatrix} -2 & 12 & -4 \\ 1 & 4 & 8 \\ -5 & 2 & 3 \end{pmatrix}$$

6.2 在混合策略意义下求解矩阵博弈 $G=\{S_1,S_2;\boldsymbol{A}\}$, 其中

$$\boldsymbol{A}=\begin{pmatrix} 2 & 2 & 1 \\ 2 & 3 & 4 \\ 3 & 5 & 4 \\ 2 & 3 & 1 \end{pmatrix}$$

6.3 用图解法求解矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$(1)\ \boldsymbol{A} = \begin{pmatrix} 2 & 4 \\ 2 & 3 \\ 3 & 2 \\ -2 & 6 \end{pmatrix} \qquad (2)\ \boldsymbol{A} = \begin{pmatrix} 1 & 3 & 11 \\ 8 & 5 & 2 \end{pmatrix}$$

6.4 用方程组法求解矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$(1)\ \boldsymbol{A} = \begin{pmatrix} 4 & 5 & 6 \\ 6 & 4 & 5 \\ 5 & 6 & 4 \end{pmatrix} \qquad (2)\ \boldsymbol{A} = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$$

6.5 用线性规划方法求解矩阵博弈 $G = \{S_1, S_2; \boldsymbol{A}\}$, 其中

$$(1)\ \boldsymbol{A} = \begin{pmatrix} 8 & 2 & 4 \\ 2 & 6 & 6 \\ 6 & 4 & 4 \end{pmatrix} \qquad (2)\ \boldsymbol{A} = \begin{pmatrix} 2 & 0 & 2 \\ 0 & 3 & 1 \\ 1 & 2 & 1 \end{pmatrix}$$

7

Python 编程

7.1 软件简介

在现代运筹学研究中，数值计算与优化求解器是不可或缺的工具。依托专业编程软件，能够高效完成优化模型的求解与分析，大幅提升了工作的效率。Python 凭借其简洁规范的语法、丰富的开源第三方库以及高度灵活的扩展性，已发展成为运筹学领域应用最广泛的语言之一。基于 Python 编程平台，各类优化软件快速发展，其中既包括 Gurobi、CPLEX、Xpress 等国际商用求解器，也涵盖华为 OptVerse、阿里 MindOpt、杉数 COPT 等国产求解器。值得一提的是，国产求解器在问题规模、求解稳定性与综合性能上逐渐达到国际先进水平。

当前，Gurobi 与 CPLEX 仍是运筹优化领域主流的求解器，均支持线性规划、整数规划、混合整数规划、二次规划等基础模型。其中，Gurobi 以高效求解速度为核心优势，在大规模混合整数规划问题中具备更高的迭代效率与收敛速度，其官方 Python 接口 gurobipy 设计友好、易用性强，是科研工作者与算法开发者的首选工具。CPLEX 则以高稳定性与强鲁棒性为显著特征，兼容 OPL 专业优化建模语言，与企业级业务系统集成适配性更强，但专业建模语言的学习成本相对更高。

本章选用 Gurobi 作为优化求解器，通过其 Python 接口实现典型运筹学模型的求解。通过实际代码的演示，帮助读者快速掌握将运筹学模型转化为可执行计算程序的流程，实现理论与实践的协同贯通，为后续复杂优化问题的建模与求解奠定基础。Gurobi 的安装方法可参考官方文档，本章不再赘述。

7.2 代码实现

7.2.1 线性规划

例 7.1 求解下列线性规划问题

$$\max \ z = 3x_1 + 5x_2 + 4x_3$$

$$\text{s.t.} \begin{cases} 2x_1 + 3x_2 \le 15 \\ 2x_2 + 4x_3 \le 8 \\ 3x_1 + 2x_2 + 5x_3 \ge 2 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

解 下列代码演示了在 Python 环境中, 基于 Gurobi 优化求解器建立线性规划模型、定义决策变量与约束条件、执行并输出结果的完整过程。

```
from gurobipy import *

try:
    # Create a new model
    m = Model("LinearProblem")

    # Create variables
    x1 = m.addVar(lb = 0, vtype=GRB.CONTINUOUS, name="x1")
    x2 = m.addVar(lb = 0, vtype=GRB.CONTINUOUS, name="x2")
    x3 = m.addVar(lb = 0, vtype=GRB.CONTINUOUS, name="x3")

    # Set objective
    m.setObjective(3 * x1 + 5 * x2 + 4 * x3, GRB.MAXIMIZE)

    # Add constraints
    m.addConstr(2 * x1 + 3 * x2 <= 15, "c0")
    m.addConstr(2 * x2 + 4 * x3 <= 8, "c1")
    m.addConstr(3 * x1 + 2 * x2 + 5 * x3 >= 2, "c2")

    # Write model to file
    m.write("LinearProblem.lp")

    # Solve the model
    m.optimize()

```

```
    # Print optimal solution
    print('Optimal solution', end=": ")
    for v in m.getVars():
        print(f'{v.varName} = {v.x}', end="  ")

except GurobiError as e:
    print('Error code ' + str(e.errno) + ": " + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

运行代码, 得到如下结果。

```
Optimal objective  3.050000000e+01
Optimal solution   x1 = 7.5  x2 = 0.0  x3 = 2.0
```

7.2.2 整数规划

例 7.2 求解下列整数规划问题

$$\max \ z = 3x_1 + 5x_2 + 4x_3$$

$$\text{s.t.} \begin{cases} 2x_1 + 3x_2 \le 15 \\ 2x_2 + 4x_3 \le 8 \\ 3x_1 + 2x_2 + 5x_3 \ge 2 \\ x_1, x_2, x_3 \ge 0 \text{且取整数} \end{cases}$$

解 基于 Gurobi 优化求解器构建整数规划模型并求解, 相应代码如下。

```
from gurobipy import *

try:
    # Create a new model
    m = Model("IntegerProblem")

    # Create integer variables corresponding to x1, x2, x3
    x1 = m.addVar(lb = 0, vtype=GRB.INTEGER, name="x1")
    x2 = m.addVar(lb = 0, vtype=GRB.INTEGER, name="x2")
    x3 = m.addVar(lb = 0, vtype=GRB.INTEGER, name="x3")

    # Set objective
    m.setObjective(3 * x1 + 5 * x2 + 4 * x3, GRB.MAXIMIZE)
```

```
14
15      # Add constraints
16      m.addConstr(2 * x1 + 3 * x2 <= 15, "c0")
17      m.addConstr(2 * x2 + 4 * x3 <= 8, "c1")
18      m.addConstr(3 * x1 + 2 * x2 + 5 * x3 >= 2, "c2")
19
20      # Write model to file
21      m.write("IntegerProblem.lp")
22
23      # Solve the model
24      m.optimize()
25
26      # Print optimal solution
27      print('Optimal solution', end=": ")
28      for v in m.getVars():
29          print(f'{v.varName} = {v.x}', end="  ")
30
31  except GurobiError as e:
32      print('Error code ' + str(e.errno) + ": " + str(e))
33
34  except AttributeError:
35      print('Encountered an attribute error')
```

运行代码, 结果如下。

```
1  Best objective 2.900000000000e+01, best bound 2.900000000000e+01, gap
       0.0000
2  Optimal solution: x1 = 7.0  x2 = -0.0  x3 = 2.0
```

与例7.1对比可知, 在增加整数约束后, 最优值由 30.5 变为 29。若变量 x_1 仅取值 0 或 1, 请编程求解并进行比较。

7.2.3 非线性规划

例 7.3 求解下列非线性规划问题

$$\min\ z = -2x_1 - 6x_2 + ({x_1}^2 - 2x_1x_2 + 2{x_2}^2)/2$$

$$\text{s.t.}\ \begin{cases} -x_1 + 2x_2 \le 2 \\ x_1 + x_2 \le 2 \\ 2x_1 + x_2 \le 3 \\ x_1, x_2 \ge 0 \end{cases}$$

解 该问题属于二次规划, 具体实现代码如下。

```
from gurobipy import *

try:
    # Create a new model
    m = Model("QuadraticProblem")

    # Create variables corresponding to x1 and x2
    x1 = m.addVar(lb=0, vtype=GRB.CONTINUOUS, name="x1")
    x2 = m.addVar(lb=0, vtype=GRB.CONTINUOUS, name="x2")

    # Set objective function
    # 0.5 * x1 * x1 + x2 * x2 - x1 * x2 - 2 * x1 - 6 * x2
    m.setObjective(0.5 * x1 * x1 + x2 * x2 - x1 * x2 - 2 * x1 - 6 * x2
       , GRB.MINIMIZE)

    # Add constraints
    m.addConstr(-x1 + 2 * x2 <= 2, "c0")
    m.addConstr(x1 + x2 <= 2, "c1")
    m.addConstr(2 * x1 + x2 <= 3, "c2")

    # Write model to file
    m.write("QuadraticProblem.lp")

    # Solve the model
    m.optimize()

    # Print optimal solution
    print('Optimal solution', end=" ")
    for v in m.getVars():
        print('%s = %g' % (v.varName, v.x), end=" ")

except GurobiError as e:
    print('Error code ' + str(e.errno) + ": " + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

运行代码, 得到结果。

```
Optimal objective -8.22222222e+00
Optimal solution x1 = 0.666667 x2 = 1.33333
```

其精确分数形式可表示为 $\boldsymbol{x}^* = \left(\frac{2}{3}, \frac{4}{3}\right)^{\mathrm{T}}$。请读者判定该问题是否为凸规划。

7.2.4 综合案例

带时间窗的车辆路径问题 (Vehicle Routing Problem with Time Windows, VRPTW) 是车辆路径问题的重要扩展形式, 在物流配送、城市运输和供应链管理中具有广泛的应用。该问题通过引入客户服务时间窗约束, 更贴近现实的物流作业场景。

例 7.4 给定一个配送中心与若干客户节点, 每个客户节点具有确定的货物需求量, 并且要求配送车辆在指定的时间区间内完成服务。配送中心拥有若干辆具有载重容量限制的车辆, 所有车辆均从配送中心出发, 完成客户服务后最终返回配送中心。VRPTW 的目标是在满足所有约束条件的前提下, 合理规划各车辆的行驶路径与服务顺序, 使总运输成本最小。

解 本例选用经典的 Solomon VRPTW 基准数据集, 以 R101 实例为研究对象, 并截取前 50 个客户节点开展实验, 数据详见 https://www.sintef.no/projectweb/top/vrptw/100-customers/。由于时间窗约束和路径决策关联, VRPTW 属于典型的 NP-hard 问题, 其求解复杂度远高于一般车辆路径问题。根据文献可构建如下数学模型

$$
\begin{aligned}
\min \quad & \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijk} \\
\text{s.t.} \quad & \begin{cases}
\sum_{k\in K}\sum_{j\in V} x_{ijk} = 1,\ \forall i \in C & \text{(a)} \\
\sum_{j\in V} x_{0jk} = 1,\ \forall k \in K & \text{(b)} \\
\sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0,\ \forall h \in C, \forall k \in K & \text{(c)} \\
\sum_{i\in V} x_{i,n+1,k} = 1,\ \forall k \in K & \text{(d)} \\
\sum_{i\in C} q_i \sum_{j\in V} x_{ijk} \le Q,\ \forall k \in K & \text{(e)} \\
s_{ik} + t_{ij} - M(1 - x_{ijk}) \le s_{jk},\ \forall (i,j) \in A, \forall k \in K & \text{(f)} \\
e_i \le s_{ik} \le l_i,\ \forall i \in V, \forall k \in K & \text{(g)} \\
x_{ijk}\text{取0或1},\ \forall (i,j) \in A, \forall k \in K
\end{cases}
\end{aligned}
$$

其中 (a)、(b)、(c)、(d) 为网络流平衡约束, 保证每辆车必须从配送中心出发, 访问每个客户节点, 最终返回中心。(e) 为车辆载重容量约束, 限制单车服务总需求不超过额定容量。(f) 和 (g) 为时间窗约束, 确保车辆到达各客户节点的时间严格落在指定服务时间区间内。节点 $n+1$ 为配送中心 0 的虚拟节点, 用于简化模型的数学表达。基于上述模型的 Python 编程实现如下。

```
from __future__ import print_function
from gurobipy import *
import re
import math
import matplotlib.pyplot as plt
import numpy
import pandas as pd

class Data:
    customerNum = 0
    nodeNum     = 0
    vehicleNum  = 0
    capacity    = 0
    cor_X       = []
    cor_Y       = []
    demand      = []
    serviceTime = []
    readyTime   = []
    dueTime     = []
    disMatrix   = [[]] # 读取数据

class Solution:
    ObjVal = 0
    X = [[[]]]
    S = [[]]
    routes = [[]]
    routeNum = 0

    def __init__(self):
        self.ObjVal = 0
        # X_ijk
        self.X = [[([0] * data.vehicleNum) for i in range(data.nodeNum
    )]
                  for j in range(data.nodeNum)]
        # S_ik
        self.S = [([0] * data.nodeNum) for j in range(data.vehicleNum)
    ]
        # routes
        self.routes = [[]]

    def getSolution(self, data, model):
        solution = Solution()
        solution.ObjVal = model.ObjVal
        # X_ijk
```

```
43          solution.X = [[([0] * data.vehicleNum) for i in range(data.
        nodeNum)]
44                        for j in range(data.nodeNum)]
45          # S_ik
46          solution.S = [([0] * data.nodeNum) for j in range(data.
        vehicleNum)]
47          # routes
48          solution.routes = [[]]
49
50          # get the solutions of decision variables
51          print("\n\n-----------------这是一个最优解-----------------")
52          for m in model.getVars():
53              str_list = re.split(r"_", m.VarName)
54              if(str_list[0] == "X" and m.x == 1):
55                  print("str[1] = %d" % int(str_list[1]))
56                  print("str[2] = %d" % int(str_list[2]))
57                  print("str[3] = %d" % int(str_list[3]))
58                  print("m.x = ", end = "")
59                  print(m.x)
60                  solution.X[int(str_list[1])][int(str_list[2])][int(
        str_list[3])] = m.x
61                  print(str_list, end = " ")
62                  print(" = %d" % m.x)
63              elif(str_list[0] == "S" and m.x == 1):
64                  solution.S[int(str_list[1])][int(str_list[2])] = m.x
65
66          print("-----solution = ------")
67          for k in range(data.vehicleNum):
68              for i in range(data.nodeNum):
69                  for j in range(data.nodeNum):
70                      if(solution.X[i][j][k] > 0 and (not(i == 0 and j
        == data.nodeNum - 1))):
71                          print("x[{0},{1},{2}] = {3}".format(i, j, k,
        solution.X[i][j][k]))
72
73          print(solution.X)
74          # get the route of vehicles from the value of decision
        variables
75          for k in range(data.vehicleNum):
76              i = 0
77              subRoute = []
78              subRoute.append(i)
79              finish = False
80              while(not finish):
```

```
                for j in range(data.nodeNum):
                    if(solution.X[i][j][k] > 0):
                        subRoute.append(j)
                        i = j
                        if(j == data.nodeNum - 1):
                            finish = True

            if(len(subRoute) >= 3):
                subRoute[len(subRoute) - 1] = 0
                solution.routes.append(subRoute)
                solution.routeNum = solution.routeNum + 1

        print("\n\n ------Route of Vehicles ------ ")
        print(solution.routes)

        print("\n\n ------Drawing the Graph ------ ")
        # draw the route graph
        # draw all the nodes first
        # data1 = Data()
        # readData(data1, path, 100)
        fig = plt.figure(0)
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title('All Customers')

        '''
        # marker='o'
        # marker=','
        # marker='.'
        # marker=(9, 3, 30)
        # marker='+'
        # marker='v'
        # marker='^'
        # marker='<'
        # marker='>'
        # marker='1'
        # marker='2'
        # marker='3'
        # 'magenta'
        # red      blue      green
        '''

        plt.scatter(data.cor_X[0], data.cor_Y[0], c='blue', alpha=1,
    marker=',', linewidths=3, label='depot')
```

```
124         plt.scatter(data.cor_X[1:-1], data.cor_Y[1:-1], c='black',
        alpha=1, marker='o', linewidths=3, label='customer')
125         # c='red' 定义为红色, alpha 是透明度, marker 是绘制的样式
126
127         # draw the route
128         for k in range(solution.routeNum):
129             for i in range(len(solution.routes[k]) - 1):
130                 a = solution.routes[k][i]
131                 b = solution.routes[k][i+1]
132                 x = [data.cor_X[a], data.cor_X[b]]
133                 y = [data.cor_Y[a], data.cor_Y[b]]
134                 plt.plot(x, y, 'k', linewidth = 1) # r--
135
136         # plt.grid(True)
137         plt.grid(False)
138         plt.legend(loc='best')
139         plt.show()
140
141         # print(solution.route_UAV)
142         return solution
143
144 # function to read data from .txt files
145 def readData(data, path, customerNum):
146     data.customerNum = customerNum
147     data.nodeNum = customerNum + 2
148     f = open(path, 'r')
149     lines = f.readlines()
150     count = 0
151     # read the info
152     for line in lines:
153         count = count + 1
154         # 1. 处理车辆信息 (在第 5 行)
155         if(count == 5):
156             line = line.strip() # 去掉行首行尾空格
157             str_list = re.split(r"\s+", line) # 使用 \s+ 匹配一个或多个空
        格
158             data.vehicleNum = int(str_list[0])
159             data.capacity = float(str_list[1])
160
161         # 2. 处理客户/节点信息 (从第 10 行开始)
162         # 这里的范围是 10 到 10 + customerNum (包含 depot)
163         elif(count >= 10 and count <= 10 + customerNum):
164             line = line.strip()
165             str_list = re.split(r"\s+", line)
```

```
166             if not str_list or str_list[0] == '': continue # 跳过空行
167 
168             # Solomon 格式列索引: 0:ID, 1:X, 2:Y, 3:DEMAND, 4:READY, 5:
        DUE, 6:SERVICE
169             data.cor_X.append(float(str_list[1]))
170             data.cor_Y.append(float(str_list[2]))
171             data.demand.append(float(str_list[3]))
172             data.readyTime.append(float(str_list[4]))
173             data.dueTime.append(float(str_list[5]))
174             data.serviceTime.append(float(str_list[6]))
175 
176     # 3. VRPTW 通常将起点(Depot)复制一份作为终点
177     data.cor_X.append(data.cor_X[0])
178     data.cor_Y.append(data.cor_Y[0])
179     data.demand.append(data.demand[0])
180     data.readyTime.append(data.readyTime[0])
181     data.dueTime.append(data.dueTime[0])
182     data.serviceTime.append(data.serviceTime[0])
183 
184     # 4. 计算距离矩阵
185     data.disMatrix = [[0.0 for _ in range(data.nodeNum)] for _ in
        range(data.nodeNum)]
186     for i in range(0, data.nodeNum):
187         for j in range(0, data.nodeNum):
188             temp = (data.cor_X[i] - data.cor_X[j])**2 + (data.cor_Y[i]
        - data.cor_Y[j])**2
189             data.disMatrix[i][j] = math.sqrt(temp)
190 
191     return data
192 
193 def printData(data, customerNum):
194     print("下面打印数据\n")
195     print("vehicle number = %4d" % data.vehicleNum)
196     print("vehicle capacity = %4d" % data.capacity)
197     for i in range(len(data.demand)):
198         print('{0}\t{1}\t{2}\t{3}'.format(data.demand[i], data.
        readyTime[i], data.dueTime[i], data.serviceTime[i]))
199 
200     print("-------距离矩阵-------\n")
201     for i in range(data.nodeNum):
202         for j in range(data.nodeNum):
203             # print("%d %d" % (i, j))
204             print("%6.2f" % (data.disMatrix[i][j]), end = " ")
205         print()
```

```
206
207 # reading data
208 data = Data()
209
210 path = '/Users/pfzhang/Project/p/r101.txt'
211 customerNum = 100
212 readData(data, path, customerNum)
213 printData(data, customerNum)
214
215 # ==========build the model==========
216 big_M = 10000
217 # construct the model object
218 model = Model("VRPYW")
219
220 # Initialize variables
221 # create variables: Multi-dimension vector: from inner to outer
222 # X_ijk
223 X = [[[[] for k in range(data.vehicleNum)] for j in  range(data.
      nodeNum)] for i in range(data.nodeNum)]
224
225 # S_ik
226 S = [[[] for k in range(data.vehicleNum)] for i in range(data.nodeNum
      )]
227
228 for i in range(data.nodeNum):
229    for k in range(data.vehicleNum):
230        name1 = 's_' + str(i) + '_' + str(k)
231        S[i][k] = model.addVar(data.readyTime[i], data.dueTime[i],
      vtype = GRB.CONTINUOUS, name = name1)
232        for j in range(data.nodeNum):
233            # if(i != j):
234            name2 = 'X_' + str(i) + "_" + str(j) + "_" + str(k)
235            X[i][j][k] = model.addVar(0, 1, vtype = GRB.BINARY, name =
       name2)
236
237 # Add constraints
238 # create the objective expression
239 obj = LinExpr(0)
240 for i in range(data.nodeNum):
241    for j in range(data.nodeNum):
242        if(i != j):
243            for k in range(data.vehicleNum):
244                obj.addTerms(data.disMatrix[i][j], X[i][j][k])
245 # print(model.getObjective()) # 这个可以打印出目标函数
```

```
246
247 # add the objective function into the model
248 model.setObjective(obj, GRB.MINIMIZE)
249
250 # constraint (1)
251 for i in range(1, data.nodeNum - 1): # 这里需要注意i的取值范围, 否则可能会
        加
252     # 入空约束
253     expr = LinExpr(0)
254     for j in range(data.nodeNum):
255         if(i != j):
256             for k in range(data.vehicleNum):
257                 if(i != 0 and i != data.nodeNum - 1):
258                     expr.addTerms(1, X[i][j][k])
259
260     model.addConstr(expr == 1, "c1")
261     expr.clear()
262
263 # constraint (2)
264 for k in range(data.vehicleNum):
265     expr = LinExpr(0)
266     for i in range(1, data.nodeNum - 1):
267         for j in range(data.nodeNum):
268             if(i != 0 and i != data.nodeNum - 1 and i != j):
269                 expr.addTerms(data.demand[i], X[i][j][k])
270
271     model.addConstr(expr <= data.capacity, "c2")
272     expr.clear()
273
274 # constraint (3)
275 for k in range(data.vehicleNum):
276     expr = LinExpr(0)
277     for j in range(1, data.nodeNum): # 此处注意, 不能有 i == j的情况出现
278         expr.addTerms(1.0, X[0][j][k])
279     model.addConstr(expr == 1.0, "c3")
280     expr.clear()
281
282 # constraint (4)
283 for k in range(data.vehicleNum):
284     for h in range(1, data.nodeNum - 1):
285         expr1 = LinExpr(0)
286         expr2 = LinExpr(0)
287         for i in range(data.nodeNum):
288             if(h != i):
```

```
289                 expr1.addTerms(1, X[i][h][k])
290
291         for j in range(data.nodeNum):
292             if(h != j):
293                 expr2.addTerms(1, X[h][j][k])
294
295         model.addConstr(expr1 == expr2, "c4")
296         expr1.clear()
297         expr2.clear()
298
299 # constraint (5)
300 for k in range(data.vehicleNum):
301     expr = LinExpr(0)
302     for i in range(data.nodeNum - 1): # 这个地方也要注意，是 data.nodeNum
        - 1,
303         # 不是 data.nodeNum
304         expr.addTerms(1, X[i][data.nodeNum - 1][k])
305     model.addConstr(expr == 1, "c5")
306     expr.clear()
307
308 # constraint (6)
309 for k in range(data.vehicleNum):
310     for i in range(data.nodeNum):
311         for j in range(data.nodeNum):
312             if(i != j):
313                 model.addConstr(S[i][k] + data.disMatrix[i][j] - S[j][
      k] <=
314                                 big_M - big_M * X[i][j][k], "c6")
315
316 # solve the problem
317 # model.setAttr("ub", model.getVarByName('X_0_2_3'), 0)
318 # model.setAttr("modelSense", -1)
319
320 model.write('a.lp')
321 model.optimize()
322 print("\n\n-----optimal value-----")
323 print(model.ObjVal)
324
325 for m in model.getVars():
326     if(m.x == 1):
327         print("%s \t %d" % (m.varName, m.x))
328
329 fig = plt.figure(0)
330 plt.xlabel('x')
```

```
plt.ylabel('y')
plt.title('All Customers')
plt.scatter(data.cor_X[0], data.cor_Y[0], c='red', alpha=1, marker=',
      ',
           linewidths=10, label='depot')
plt.text(data.cor_X[0]+1, data.cor_Y[0]+1, 'Depot', color = 'r',
      fontsize
        =30)

# 只绘制被路由使用的客户点
used_customers = set()
for route in solution.routes:
   for node in route[1:-1]:  # 跳过起点和终点 (depot)
       used_customers.add(node)

used_X = [data.cor_X[i] for i in used_customers]
used_Y = [data.cor_Y[i] for i in used_customers]
plt.scatter(used_X, used_Y, c='black', alpha=1, marker='o',
      linewidths=3, label='customer')
# 动态选择标签位置, 避免索引超出范围
mid_index = len(data.cor_X) // 2
plt.text(data.cor_X[mid_index]-6, data.cor_Y[mid_index]+4, 'Customers
      ', color = 'r',
         fontsize=30)

# plt.grid(True)
plt.grid(False)
plt.legend(loc='best')
plt.show()

#get the solution info
solution = Solution()
solution = solution.getSolution(data, model)
```

运行程序, 得到如下最优配送方案。

```
R101 的最优解
============================================================
Route: 1, 0-42-14-44-16-38-37-17-0
Route: 2, 0-27-50-0
Route: 3, 0-28-12-3-35-1-0
Route: 4, 0-39-23-41-22-4-25-0
Route: 5, 0-33-29-9-34-24-0
Route: 6, 0-45-8-46-48-0
```

```
Route: 7, 0-36-47-11-19-49-10-32-0
Route: 8, 0-2-15-43-13-0
Route: 9, 0-5-7-18-6-0
Route: 10, 0-31-30-20-0
Route: 11, 0-21-40-26-0
Vehicle capacity: 200.00
Routes: 11
Total travel distance: 991.57095325
------------------------------------------------------------
Route: 1, length: 8, distance: 131.92041957, max load: 88.00
Route: 2, length: 3, distance:  35.86300674, max load: 29.00
Route: 3, length: 6, distance:  90.69958612, max load: 66.00
Route: 4, length: 7, distance: 117.65164318, max load: 108.00
Route: 5, length: 6, distance: 120.92985724, max load: 53.00
Route: 6, length: 5, distance:  84.49944230, max load: 62.00
Route: 7, length: 8, distance: 152.31809945, max load: 130.00
Route: 8, length: 5, distance:  72.54724254, max load: 45.00
Route: 9, length: 5, distance:  73.59173603, max load: 46.00
Route: 10, length: 4, distance:  68.19968819, max load: 57.00
Route: 11, length: 4, distance:  43.35023189, max load: 37.00
```

通过案例可以看出, 使用 Python 调用 Gurobi 求解 VRPTW 问题具有显著的优势。以 50 个客户节点为例, 若设配送车辆数为 K, 则决策变量 x_{ijk} 的规模约为 $|K| \times |V|^2$, 其中 $|V| \approx 52$, 决策变量总规模可达数万级。同时, 还需建立服务约束、流平衡约束、容量约束及时间窗约束等多类限制, 其约束数量亦达到成千上万条。对于此类大规模混合整数规划问题, 手动求解是不现实的。此外, 借助 Python 可对最优配送路径进行可视化, 如图7.1所示, 从而为决策者提供更直观的支撑依据。

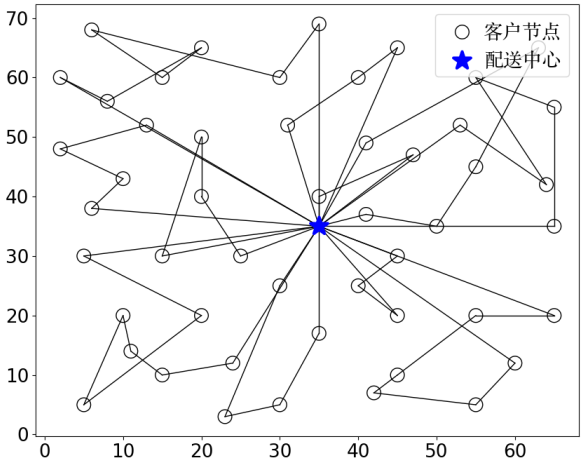

图 7.1 最优解路径图

8

展望

本书系统阐述了线性规划、整数规划、动态规划等经典的运筹学问题与方法。上述方法均属于精确算法范畴，在问题规模适中、模型结构规范的前提下，能够高效获得问题的全局最优解。然而，当面向更具挑战性的复杂优化问题时，如机器学习中的高维非凸优化、集成电路中的芯片布局规划、具身机器人中的轨迹控制等，精确算法普遍面临计算复杂度高的瓶颈，难以满足大规模与实时性的计算需求。在此背景下，以进化学习、强化学习、深度学习为代表的数据驱动优化方法，凭借其优异的近似求解与泛化适应能力，受到学术界与工业界的广泛关注。

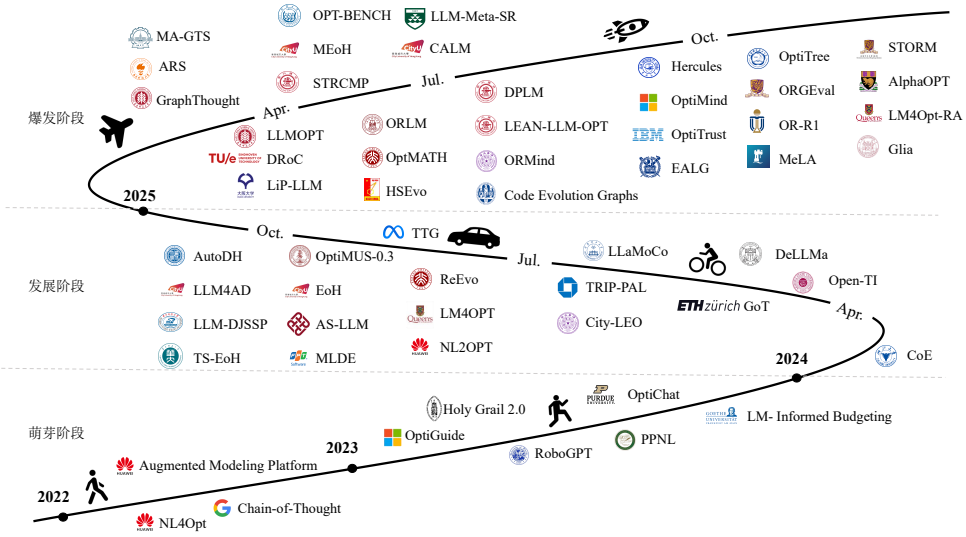

图 8.1　大语言模型驱动的运筹学

近年来, 大语言模型依托超大规模的参数体系与高质量的训练数据, 在多任务场景下表现出强大的泛化能力, 标志着人工智能向通用智能迈出了关键一步。与此同时, 大语言模型在数学推理、代码编程、算法设计等多个复杂领域中展现出广阔的前景, 为人工智能辅助科学 (AI for Science) 研究提供了全新范式。

大语言模型驱动的运筹学算法与应用已得到了初步探索, 并涌现出一批代表性的工作, 如图8.1所示。大语言模型不仅减少了解决问题所需的人力, 而且还提高了解决方案的效率, 在一定程度上重塑了运筹学的研究路径。目前, 相关方法已在交通运输、调度规划、机器任务、生物分子、网络通信、金融投资等领域得到实际应用。值得注意的是, 大语言模型在提升决策效率与智能化水平的同时, 也带来了模型可解释性不足、推理可靠性难以保障等一系列新的科学问题。

总体而言, 运筹学正处于由经典理论框架向智能化范式转型的重要发展阶段。未来研究将更加注重数据与模型的深度融合、优化与学习的协同设计, 以及面向复杂决策的应用拓展。通过与人工智能、数据科学、机器人工程等学科的交叉, 运筹学将发挥更加重要的作用, 为智能决策提供更加高效的技术。

附录 A　运筹学与钱学森

1934 年, 钱学森毕业于交通大学机械工程系。1936 年, 从美国麻省理工学院硕士研究生毕业, 之后转入加州理工学院航空系, 师从冯 · 卡门。于 1939 年获航空与数学双博士学位, 之后留校任教。1949 年, 中华人民共和国成立, 钱学森开始筹划回国, 却遭到美国当局多方阻挠, 被迫参与多场听证会 (见图A.1)。时任美国海军部副部长曾声称: “他知道所有美国导弹工程的秘密, 一个钱学森抵得上五个海军陆战师, 宁可把这个家伙给枪毙了, 也不能放他回中国去。” 面对美方威胁, 钱学森坚定回应 “我效忠中国人民”。这一立场令美方极为不满, 钱学森随即遭到无理拘禁。拘禁期间, 他被 24 小时强光照射、剥夺正常休息, 历经 15 天非人折磨, 体重急剧下降, 一度丧失语言能力。获保释后, 39 岁的钱学森开始长达 5 年的软禁生活。

图 A.1　钱学森参加美国移民归化局听证会

1955 年, 钱学森与家人躲过联邦调查局监视, 在香烟纸上写下求助信并辗转寄回祖国, 信中写道: “无一日、一时、一刻不思归国, 参加伟大的建设高潮” (见图A.2)。经毛泽东主席、周恩来总理积极斡旋, 以朝鲜战争空战中被俘的多名美军飞行员作为交换, 钱学森才得以返回新中国。同期, 许国志获美国堪萨斯大学博士学位后, 亦辞去马里兰大学流体力学与应用数学研究所研究员职务, 与钱学森同船携眷归国, 投身新中国科技与建设事业。钱学森与许国志在归国途中, 共同商议如何服务于国家建设。钱学森从政治、经济、科技等维度提出重要见解, 二人重点讨论了新兴的 Operations Research。他们认为,

该学科虽起源于第二次世界大战的作战分析, 但其强调整体优化的学术本质, 不仅适用于军事领域, 更能在经济建设中发挥关键作用。

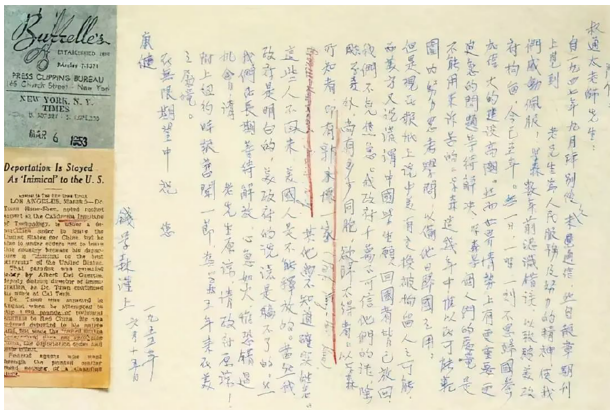

图 A.2　钱学森的求助信

Operations Research 初期中文译名为“运用学”, 钱学森对该译名并不满意。时任清华大学教师的周华章提出, 采用“运筹学”比“运用学”更为贴切, 因为该学科不仅有运用, 而且还有筹划之意。钱学森对此表示赞赏, “运筹”二字源自我国“运筹帷幄”之语, 司马迁在《史记 · 高祖本纪》中写道: “夫运筹策帷帐之中, 决胜于千里之外, 吾不如子房。镇国家, 抚百姓, 给粮饷, 不绝粮道, 吾不如萧何。连百万之军, 战必胜, 攻必取, 吾不如韩信。此三者, 皆人杰也, 吾能用之, 此吾所以取天下也。”自此, Operations Research 的中文译名正式定为“运筹学”, 并沿用至今。

1957 年, 在钱学森和钱伟长的倡导下, 中国科学院力学所成立了国内首个运筹学研究室。钱学森与许国志遴选 8 名应届毕业生组建基础研究队伍, 包括北京大学数学力学系应用数学专业 3 人, 中国人民大学国民经济专业 3 人、上海交通大学电子技术专业 2 人, 实现了数学、经济学与工程学的交叉融合。该研究室面向国家实际需求开展应用研究, 成果覆盖纺织、水利、交通、通信等多个行业, 为我国早期运筹学的学科建设与人才培养奠定了坚实基础。

钱学森以战略科学家的远见, 将运筹学引入中国, 使其成为服务于国家治理、工程建设与经济发展的重要工具。截至 2026 年初, 中国运筹学会已拥有个人会员超过 8000 名, 下属分支机构 35 个, 学科队伍不断壮大, 研究体系日趋完善。伴随人工智能技术的飞速发展, 中国运筹学工作者正迎来前所未有的发展机遇, 也肩负着新的时代使命与挑战, 将继续传承践行钱学森的科学理念, 推动运筹学在强国建设中发挥更大的作用。

附录 B　线性规划之父

乔治 · 伯纳德 · 丹齐格 (G. B. Dantzig) 的父亲是一名俄罗斯数学家, 曾在法国巴黎跟随著名数学家昂利 · 庞加莱学习, 后移民美国, 这份家学渊源也为丹齐格的学术之路埋下了伏笔。1939 年, 丹齐格进入加州大学伯克利分校攻读统计学博士学位, 师从著名统计学大师杰尔齐 · 内曼。某日, 丹齐格因迟到错过了课程的开头部分, 只见黑板上留有两道数学题, 便理所当然地将其当作教授布置的课后作业, 默默抄录下来。尽管这两道题的难度远超平日作业, 丹齐格却并未退缩。沉心钻研数日, 逐一攻克难题后, 才带着写好的解答去见内曼教授, 还略带歉意地说明自己花费了较长时间才完成这份作业。丹齐格将作业轻轻放在内曼堆满文稿的办公桌上, 甚至担心这份作业会被淹没在繁杂的纸张中, 再也无法被教授看到。

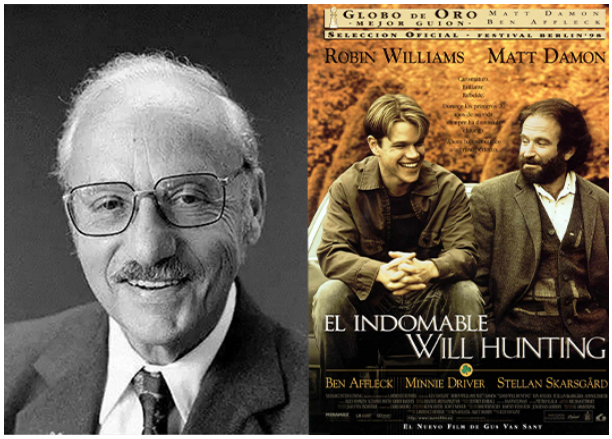

图 B.1　丹齐格与《心灵捕手》

六周后的一个周日清晨, 内曼教授带着难以掩饰的兴奋, 亲自登门拜访丹齐格。他手中紧紧攥着丹齐格那份作业, 激动地对这位年轻的研究生说道: “我已经为其中一篇解答撰写好了引言, 现在就可以寄给数学期刊发表。”丹齐格此刻才恍然大悟, 自己当作普通课后作业完成的两道题, 竟是当时统计学界悬而未决的两大难题, 此前无数学者都曾试图攻克却无果而终。一次偶然的误解, 不仅成就了丹齐格的博士学业, 也为其后续学术道路奠定了坚实基础。这段充满传奇色彩的轶事, 后来被改编为电影《心灵捕手》的开篇场景, 激励了一代又一代投身科研事业的工作者 (见图B.1)。

第二次世界大战爆发后, 丹齐格中断了在伯克利的博士学业, 加入美国空军总部, 担任统计控制战斗分析处主任, 主要负责供应链补给、人员调度与物资管理等一系列实际决策工作。这段实践经历使丹齐格深刻地认识到, 现实世界中存在大量需要优化资源配置、协调多方需求的问题, 而传统的经验决策方式效率低下, 亟需一套系统的数学方法来解决。1947 年, 丹齐格在总结前人工作的基础上, 正式提出了线性规划的标准形式, 并创立了求解线性规划问题的单纯形法。该方法为线性规划问题提供了通用、高效的解决方案, 使得运筹学能够真正应用于实际工程领域, 后来更被美国物理学会与计算机协会联合评为 “20 世纪十大算法之一”。

图 B.2　美国总统福特向丹齐格颁发美国国家科学奖章

1963 年, 丹齐格出版专著《Linear Programming and Extensions》, 至今仍是线性规划方面的标准参考书。1975 年, 丹齐格荣获美国国家科学奖章 (见图B.2), 同年又摘得运筹管理学领域最高的冯 · 诺依曼理论奖。丹齐格还当选美国国家科学院、美国国家工程院和美国艺术与科学院三院院士, 成为运筹学领域无可替代的标杆性人物。为纪念其开创性贡献, 国际数学规划学会设立丹齐格奖, 自 1982 年起每三年颁发一次, 以表彰在数学规划做出重大影响的杰出学者。

附录 C　运筹学期刊

Operations Research
Management Science
Manufacturing and Service Operations Management
INFORMS Journal on Computing
INFORMS Journal on Optimization
Mathematics of Operations Research
Mathematical Programming
SIAM Journal on Optimization
SIAM Journal on Scientific Computing
Transportation Science
Transportation Research Part B: Methodological
European Journal of Operational Research
Computers and Operations Research
International Journal of Production Research
Annals of Operations Research
Journal of the Operational Research Society
IEEE Transactions on Automatic Control
IEEE Transactions on Evolutionary Computation
CSIAM Transactions on Applied Mathematics
Journal of the Operations Research Society of China
运筹与管理
运筹学学报

参考文献

[1] HILLIER F S, LIEBERMAN G J. Introduction to operations research[M]. 11th ed. New York: McGraw-Hill Education, 2021.

[2] LUENBERGER D G, YE Y. Linear and nonlinear programming[M]. 5th ed. New York: Springer, 2021.

[3] TAHA H A. Operations research: An introduction[M]. 10th ed. New York: Pearson Education, 2016.

[4] 胡运权, 郭耀煌. 运筹学教程 [M]. 5 版. 北京: 清华大学出版社, 2018.

[5] 侯福均, 吴祈宗. 运筹学与最优化方法 [M]. 3 版. 北京: 机械工业出版社, 2022.

[6] 刁在筠, 戎晓霞, 王光辉, 等. 运筹学 [M]. 5 版. 北京: 高等教育出版社, 2024.

[7] 文再文, 袁亚湘. 最优化方法与理论 [M]. 北京: 高等教育出版社, 2025.

[8] 刘兴禄, 熊望祺, 臧永森, 等. 运筹优化常用模型、算法及案例实战 [M]. 北京: 清华大学出版社, 2022.