第5章

优化算法

本章学习目标

- 了解最优化的概念。
- 了解传统优化算法的特点。
- 了解智能优化算法的特点。
- 掌握梯度下降算法的基本思想和原理。
- 掌握遗传算法的基本操作步骤。
- 掌握蚁群算法的基本原理。
- 了解常见优化算法的应用。

5.1 认识优化算法

人类一切活动的本质不外乎是认识世界和改造世界。认识世界需要建立模型；改造世界需要优化决策，可以说建模与优化无处不在，它们始终贯穿在一切人类活动的过程之中。这是因为从哲学逻辑来看，认识世界与改造世界是人类实践活动的核心双维度：认识世界是通过观察、分析、抽象等方式建立对事物规律的认知，即建模，将复杂现实简化为可理解、可推演的模型；而改造世界则是基于对世界的认知，通过调整策略、配置资源、制定方案等方式改变现实，即优化决策，在多种可能的行动路径中找到最优解，以实现预期目标。无论是科学研究、生产劳动，还是社会治理、日常生活，都离不开建模与优化的逻辑闭环。

人工智能的崛起，本质上是人类将建模与优化的认知逻辑赋予机器的过程。若说建模是人工智能认识世界的眼睛，那么优化算法便是其改造世界的双手：前者通过神经网络、贝叶斯模型等工具将现实世界的规律转换为数学语言，后者则通过梯度下降、遗传算法等手段在模型框架内寻找最优解，推动智能系统从理解走向行动。

更深刻的是，人工智能的进化正在反推优化算法的革新。面对高维数据、动态环境等传统优化算法难以应对的场景，人工智能催生了自适应优化、群体智能算法等新工具——就像人类在改造复杂世界时会发明更精密的仪器，人工智能在处理超大规模问题时，也在倒逼优化算法突破理论边界。这种“建模需求牵引优化创新，优化升级反哺建模能力”的循环，恰是人工智能延续人类认知逻辑又超越人类局限的核心密码。

应用实践表明，优化技术已在许多领域中产生了巨大的经济效益和社会效益，而且随着处理对象规模的增大，这种效果也更加显著。这对国民经济的各个领域来说，其应用前景是巨大的。优化算法的理论研究对改进算法性能、拓宽算法应用领域、完善算法体系同

样具有重要意义。也就是说，优化理论与算法的研究是一个同时具有理论意义和应用价值的重要课题。

5.1.1 最优化理论基础

最优化(Optimization)是应用数学的重要研究领域，它是研究在给定约束之下如何寻求某些因素，以使某一或某些指标达到最优的一些学科的总称。由于运筹学中出现的问题大多是最优化所研究的问题，因此运筹学的许多分支，如数学规划、组合最优化、排队论及决策论等也是最优化的组成部分。此外，最优化还包括工程最优设计和最优控制等。对个人来说，家庭理财、职业选择、人生规划及作息安排等生活的方方面面都可以运用最优化。可以说，最优化是人类智慧的精华，凝结着文明对效率与完美的永恒追求；从宏观文明到微观个体，最优化既是集体智慧的结晶，更是自我突破的思维利器，它能让理性之光在求最优的追寻中永恒闪耀。

最优化问题(Optimization Problem)是最优化学科所研究的问题，也是运筹学中大多数分支研究的问题，该问题可定义为：在给定约束之下，从问题的诸多可能解中，寻求使某一或某些指标达到最优的解。若使用数学语言来描述，即在给定的集合上求解某个目标函数的极值(极小化或极大化)问题。显然，从最优化问题的定义可以得出，对最优化问题进行数学建模通常需要如下三大要素。

(1) 决策变量和参数。每个模型都有若干决策变量，决策变量的一组值表示一种方案，即问题的一个解。参数表示变量，有确定性的，也有随机性的。

(2) 约束条件。由于现实系统的客观物质条件限制，如技术上的约束、资源上的约束和时间上的约束等，因此模型必须包括把决策变量限制在它们可行值之内的约束条件，而这通常是用约束的数学函数形式来表示的。约束条件越接近实际系统，则所求得的最优解也就越接近实际最优解。

(3) 目标函数。最优化有一定的评价标准，目标函数就是这种标准的数学描述，它是决策变量的一个数学函数，用来衡量系统的性能标准，即系统追求的目标。目标函数值必须在满足规定的约束条件下达到最大或最小。

以最小化的情形为例，最优化问题的数学模型一般描述为

$$\min_{u\in(\subset)U} f(u)$$

其中，U 为可行域；$u\in(\subset)U$ 为变量或集合 u 要满足所给定的约束条件，U 中的任何一个元素称为问题的一个可行解；$f(u)$为目标函数，用于对解 u 进行选择的指标。

满足 $f(u^*)=\min\{f(u)\mid u\in(\subset)U\}$的可行解 u^* 称为该问题的最优解，对应的目标函数值 $f(u^*)$称为最优值。

最优化问题的研究历史源远流长。早在公元前 212—公元前 187 年，古希腊数学家阿基米德便已完成等周问题的证明，即在给定周长条件下，圆形所围面积达到最大值。这一数学成果对建筑实践产生了深远影响，欧洲古代城堡多采用圆形设计，而中国古代城堡则普遍为方形构造。其原因在于，对于四边形而言，在周长固定时，正方形的围合面积最大，更符合古代城防建设的实用需求。

在 20 世纪 50 年代以前，最优化研究主要聚焦于求解可导函数极值和目标函数极值之类的古典最优化问题。随着计算机技术在 20 世纪 50 年代后的飞速发展，大规模最优化问

题的求解成为可能，研究领域也随之拓展。其中，组合优化问题作为一类特殊且关键的最优化问题，在计算机科学、运筹管理等多学科领域发挥着重要作用，逐渐成为学术与应用研究的热点方向。

5.1.2 传统优化算法

优化算法（或优化方法）是一类基于数学理论与计算逻辑的系统性搜索策略，旨在通过迭代、启发式规则或严格推导，在满足特定约束条件的前提下，从可行解空间中高效寻找目标函数的最优解，如最大值或最小值。传统优化算法是指在计算机科学、数学规划和工程领域发展较早时期，基于经典数学理论和确定性算法的优化技术。这类方法通常依赖于目标函数的连续性、可微性等解析性质，通过严格的数学推导寻找最优解，广泛应用于线性规划、非线性规划、整数规划等经典优化问题中。传统优化算法主要包括线性规划的单纯形法和非线性规划中的基于梯度的各类迭代算法。其中，单纯形法是1947年在研究资源的优化配置时提出的。由于生产实践中的大量问题皆可归结成线性规划模型，而单纯形法对于当时出现的线性规划问题的规模行之有效，因此开创了用数学方法解决大型实际问题的先河。1847年提出的梯度下降算法是以负梯度方向作为下降方向的极小化算法，它是求解非线性规划问题的最简单的方法之一。

传统优化算法具有较高的计算效率、较强的可靠性及比较成熟等优点，是一类最重要的、应用最广泛的优化算法。尽管待求解问题的类型和具体的求解方法千差万别，但这类方法都具有以下三大基本的迭代步骤：①选择一个初始解；②判断停止准则是否满足；③向改进方向移动。

但是传统优化算法本质上是全局搜索算法，即通过搜索整个解空间来找到问题的最优解。为了加快算法的搜索速度或减少运行时计算空间的耗费，传统优化算法往往需要充分利用目标函数的解析性质以及约束空间的几何特征，逐步缩小搜索空间，最终完成整个解空间的搜索，从而找到最优解。但是，以下两种状况使传统优化算法在解决许多实际问题时受到了限制。

状况1：目前所研究的实际问题越来越复杂，规模越来越大，约束条件越来越多。

状况2：传统优化算法往往要求目标函数是凸的（即集合内任意两点的连线都完全在该集合内）、连续可微的，可行域是凸集等条件，而且这些方法处理高维、非凸、复杂问题和非确定性信息的能力较差。

因此，采用这些传统优化算法来解决当前研究的复杂问题时，已难以在可接受时间内找到最优解，显然求解时间与最优解存在一定矛盾。为了解决这些问题，有些研究者提出了一类新的解决方法，它们不以寻找问题的最优解为目的，而是追求在有限的时间内找到满足需要的解（或满意解）即可。计算智能正是在此背景下应运而生的新兴学科领域，它是由多个学科相互交叉和渗透的结果，得益于运筹学与管理科学、计算数学、人工智能、模式识别和自动控制理论等许多学科，而智能优化算法是计算智能的重要研究内容。

5.1.3 智能优化算法

1. 智能优化算法概述

20世纪80年代以来，一些新颖的优化方法往往借鉴了人类解决复杂问题的技巧以及

生物体的本能，通过模拟或揭示某些自然现象或过程而得到发展，其思想和内容涉及数学、物理学、生物进化、人工智能、神经科学和统计力学等方面，它能够将复杂的求解过程简单化，从而表现出智能的特征，因此被称为智能优化算法。

由于这类新的智能优化算法一般都是建立在生物智能或物理现象基础上的随机搜索算法，目前在理论上还远不如传统优化算法完善，往往也不能确保解的最优性，因此常常被视为元启发式方法。元启发式方法也称为现代启发式算法，它被认为具有一种高级策略，采用一种近似优化的技术，采用概率决策过程进行智能形式的随机搜索。这类算法均从任意一个初始解出发，按照某种机制，以一定的概率在整个求解空间中搜索最优解，由于它们可以把搜索空间扩展到整个问题空间，因此具有全局优化性能。从实际应用的观点看，这类新算法一般不要求目标函数和约束条件的连续性与凸性，甚至在某些情况下，对是否存在解析表达式都不要求，对计算中数据的不确定性也有很强的适应能力。因此，智能优化算法以其具有全局的、并行高效的、通用性强、无须问题特殊信息等优点，为解决复杂问题提供了新的思路和手段，引起了国内外学者的广泛重视并掀起了研究热潮，且已成功应用于计算机科学、优化调度、运输问题、组合优化及工程优化设计等领域，展示出强劲的发展势头。

2. 典型的智能优化算法

为了让读者对典型的智能优化算法有一个整体的印象，下面分别对它们进行简述。

1975 年，Holland 提出了遗传算法。这种优化方法模拟生物界自然选择与遗传进化机制，通过模仿生物种群适者生存、优胜劣汰的进化规律，以及基因的交叉重组随机变异等遗传过程，在解空间中对候选解种群进行迭代优化，最终逐步逼近复杂优化问题的全局最优解。

模拟退火算法的思想源自 Metropolis 等于 1953 年对统计热力学的研究，该算法模拟热力学中退火过程能使金属原子达到能量最低状态的机制，通过模拟的降温过程按玻尔兹曼(Boltzmann)方程计算状态间的转移概率来引导搜索，从而使算法具有很好的全局搜索能力。值得一提的是，研究者在 1983 年成功地将该算法应用于组合优化的问题上，即假定模拟退火算法中的固体状态对应组合最优问题的可行解，最低能量的基态对应最优解，逐渐降低温度的冷却过程对应控制参数的下降，算法首先由某一个较高的初始温度开始，伴随温度参数的不断下降重复抽样，最终得到问题的全局最优解。现在，由于模拟退火算法具有描述简单、使用灵活、运行效率高和较少受初始条件限制等优点，已经成为解决大规模优化问题的有效近似算法，在诸如超大规模集成电路、生产调度、控制工程、机器学习、神经网络以及图像处理等领域得到了广泛应用。

1986 年，P. Glover 提出了禁忌搜索算法的概念。所谓禁忌就是禁止重复前面的工作。为了避免局部邻域搜索陷入局部最优，禁忌搜索算法用一个禁忌表来记录已经到达过的局部最优点或达到局部最优的一些过程，在下一次搜索中，利用禁忌表中的信息不再搜索或有选择地搜索这些点或过程，以此来跳出局部最优点。可见，该算法在搜索过程中获得知识，能够以较大的概率跳出局部极值。禁忌搜索算法以其较高的求解质量和效率已在许多组合优化问题中显示出强大的寻优能力，但存在对初始解依赖性较强及迭代搜索过程是串行的等缺点。

20 世纪 90 年代初，Dorigo M 等提出了蚁群优化算法。该算法借鉴蚂蚁群体利用信息

素相互传递信息来实现路径优化的机理，通过记忆路径信息素的变化来解决组合优化问题。

1995年，Kennedy和Eberhart提出了粒子群优化算法。该算法模仿鸟类和鱼类群体在觅食迁徙中个体与群体协调一致的机理，通过群体最优方向、个体最优方向和惯性方向的协调来求解优化问题。

1998年，巴西学者Alexandre Linhares提出了捕食搜索算法。该算法模拟猛兽捕食中大范围搜寻和局部蹲守的特点，通过设置全局搜索和局部搜索间变换的阈值来协调两种不同的搜索模式，从而实现了对全局搜索能力和局部搜索能力的兼顾。该算法分别应用于旅行商问题和超大规模集成电路设计问题，都取得了较好的效果。

2000年，Passino提出了细菌觅食算法，它是一种基于群体的仿生随机搜索算法，其生物学基础是人类肠道中大肠杆菌在觅食过程中体现出来的智能行为。目前，细菌觅食算法已经被应用于自适应控制领域、噪声干扰下的谐波估计问题和PID控制器的设计等方面。

根据所受启发的生物机理的不同，蜂群算法可分为基于蜜蜂繁殖行为的蜂群算法和基于蜜蜂采蜜行为的蜂群算法两大类，两类算法各有其独特的实现原理和发展方向。最常见的基于蜜蜂繁殖行为的蜂群算法，是2001年提出的蜜蜂交配优化算法。从本质上来讲，基于蜜蜂繁殖行为的蜂群算法是对遗传算法的改进。最常见的基于蜜蜂采蜜行为的蜂群算法，是2005年提出的人工蜂群算法，主要是为了解决多维的和多模的优化问题。

从现有的文献可知，有两种版本的萤火虫算法：2005年基于对多声源位置进行探测构建了GSO算法，并将其应用于群体机器人问题；2007年基于3个理想化规则的基础上提出了FA算法，并通过基准函数对其进行了测试。

2009年，剑桥大学和拉曼工程大学的两位学者通过模拟布谷鸟的寻巢产卵行为，提出了一种新的智能优化算法——布谷鸟搜索算法。该算法主要基于布谷鸟的巢寄生繁殖机理和莱维飞行搜索原理两方面。由于这种算法简单、参数少、易于实现，并成功应用于工程优化等实际问题中，因此逐渐发展成为智能优化算法领域中的一个新亮点。

目前，这些智能优化方法在理论和实际应用方面均取得了较大发展。虽然说部分算法可从理论上证明，能够收敛到高复杂度组合优化问题的全局最优解，但这类高复杂度问题的理论限制，使得它们只能以启发式算法的方式求解实际问题，即在可接受的计算时间或占用空间条件下，给出待解决组合优化问题每个实例的可行解，而该可行解与最优解的偏离程度不一定事先可以预计。不过，这些算法的终极目标仍是求得问题的全局最优解，具备一定普适性，为解决大量实际应用问题提供了新的思路。因此，对这类智能优化算法展开研究，具有很重要的实际意义。

3. 典型智能优化算法的特点

与传统优化算法相比，典型智能优化算法有下述共同的特点。

(1) 不以达到某个最优条件或找到理论上的精确最优解为目标，而是更看重计算的速度和效率。

(2) 对目标函数和约束函数的要求比较宽松。

(3) 算法的基本思想都是来自某种自然规律的模仿，具有人工智能的特点。

(4) 多数算法含有一个多个体的种群，寻优过程实际上就是种群的进化过程。

(5) 这些算法的理论工作相对比较薄弱，一般来说都不能保证收敛到最优解，甚至不能

保证求到可行解。

从这些不同的特点出发，这些算法获得了各种不同的名称。由于算法理论薄弱，它们最早被称为元启发式方法；从其人工智能的特点，还被称为智能计算或智能优化算法；因其具有不以精确解为目标的特点，它们又被归到软计算方法中；从种群进化的特点，它们又可以称为进化计算；从它们模仿自然规律的特点出发，近几年又有人将它们称为自然计算。

智能优化算法主要包括遗传算法、蚁群优化算法等，这些算法对目标函数的性态，诸如单调性、凹凸性、可导性及极值等均为函数的性态等，没有特殊要求，借助现代计算机作为工具，特别适用于求解传统优化算法解决不了的大规模复杂组合优化问题，且可以在较短时间内获得最优解或次优解，因此得到了广泛的应用和认同，逐渐成为现代优化算法领域的研究热点。目前经典智能方法的改进和应用研究比较广泛，研究者针对各种具体问题提出了大量对经典方法的改进，使得这些智能优化算法得以适应各个领域的需求。

目前智能优化算法的研究呈现出三大趋势：一是对经典智能方法的改进和广泛应用，以及对其理论的深入研究；二是开发新的智能工具，拓宽其应用领域，并为其寻求理论基础；三是经典智能方法与现代智能方法的结合，建立混合智能方法。

本章接下来对传统优化算法中的梯度下降算法，以及智能优化算法中的遗传算法、蚁群算法等内容进行详细介绍。

5.2 梯度下降算法

梯度下降(Gradient Descent)算法是一种求解无约束优化问题的迭代算法，该算法的核心思想非常直观，即通过不断地沿着该函数梯度(Gradient)的反方向更新参数，从而找到一个函数的局部最小值。它的出现是为了解决函数优化问题，特别是在机器学习和深度学习领域，模型参数的调整本质上依赖于损失函数的最小化，而梯度下降算法恰好成为实现这一过程的关键工具。在它出现之前，人们多依赖手动调整参数或采用效率较低的方法探索最优解，不仅耗时耗力，还难以适配复杂模型的优化需求。梯度下降算法的引入使得自动化参数优化成为可能，更从根本上显著提升了这类优化问题的求解效率，为后续复杂模型如深度神经网络的落地与应用奠定了重要基础。

5.2.1 梯度下降算法的基本思想

1. 场景假设

梯度下降算法的基本思想可以类比为一个人下山的过程。假设这样一个场景：一个人被困在山上，目标是从山上下来，抵达山的最低点，也就是山谷。但在实际场景中，山上的浓雾很大、植被茂密，导致可视度很低，因此他无法确定下山的全局路径，只能基于当前位置的周围信息一步一步摸索前行。具体而言，首先以他当前所处的位置为基准，判断周围最陡峭的下坡方向，朝着此方向迈出一步；到达新位置后，接着继续以当前位置为基准，重新寻找最陡峭的下坡方向并前行，通过不断根据局部信息调整方向与步长，直到最终到达山谷；与之对应，若沿当前位置最陡峭的上坡方向迭代行进，直至到达山顶，这就是梯度上升算法的思想。

2. 梯度下降

梯度下降的基本过程和下山的场景很相似。首先，有一个可导的函数，这个函数代表一座山。目标就是找到这个函数的最小值，也就是山脚下。根据之前的场景假设，最快的下山方式是首先找到当前位置最陡峭的方向，然后沿着此方向向下走，对应到函数中就是找到给定点的梯度，然后朝着梯度相反的方向，就能让函数值下降得最快。梯度的方向就是函数值变化最快的方向。重复利用这个方法，反复求取梯度，最后就能到达局部的最小值，如图 5-1 所示。而求取梯度就确定了最陡峭的方向，也就是场景中测量最陡峭方向的手段。

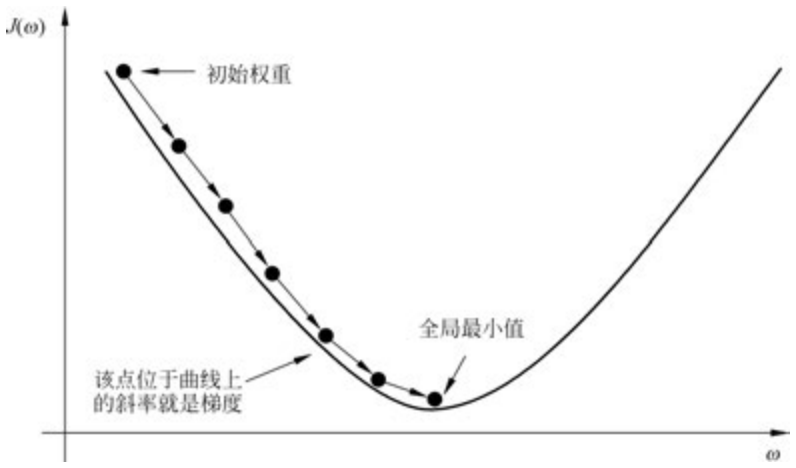

图 5-1 下山的过程

5.2.2 导数与梯度

1. 导数与梯度

导数可以通俗地理解为函数在某一点的变化快慢程度，其内在的意义可以从多个不同的角度来理解，最常用的有如下两种角度：①函数图像中某点切线的斜率；②函数的变化率。

以下是几种导数的求法。

(1) 单变量的求导。当函数只有一个变量 $y=f(x)$ 时，函数对变量的变化率表示为 $\frac{\mathrm{d}y}{\mathrm{d}x}$，例如：

$$\frac{\mathrm{d}(x^2)}{\mathrm{d}x}=2x$$

$$\frac{\mathrm{d}(-2y^5)}{\mathrm{d}x}=-10y^4$$

$$\frac{\mathrm{d}(5-\theta)^2}{\mathrm{d}x}=-2(5-\theta)$$

(2) 多变量的求导。当函数 $z=f(x,y)$ 有多个变量时，函数分别对每个变量进行求导(称为求某个变量的偏导数)，表示为 $\frac{\partial z}{\partial x}$ 和 $\frac{\partial z}{\partial y}$，并把其他变量看作常数。

$$\frac{\partial}{\partial x}(x^2y^2)=2xy^2$$

$$\frac{\partial}{\partial y}(-2y^5+z^2)=-10y^4$$

$$\frac{\partial}{\partial \theta_2}(5\theta_1+2\theta_2-12\theta_3)=2$$

$$\frac{\partial}{\partial \theta_2}(0.55-(5\theta_1+2\theta_2-12\theta_3))=-2$$

梯度实际上就是多变量求导的一般化描述，例如：

$$J(\boldsymbol{\Theta})=0.55-(5\theta_1+2\theta_2-12\theta_3)$$

$$\nabla J(\boldsymbol{\Theta})=\left(\frac{\partial J}{\partial \theta_1},\frac{\partial J}{\partial \theta_2},\frac{\partial J}{\partial \theta_3}\right)=(-5,-2,12)$$

由此可以看出，梯度就是分别对每个变量进行求导，然后用逗号分隔，再用括号包括起来，说明梯度其实是一个向量。在单变量函数中，梯度其实就是函数的导数，代表着函数在某个给定点的切线的斜率；在多变量函数中，梯度是一个向量，该向量有方向，梯度的方向指出了函数在给定点的上升最快的方向，那么梯度的反方向就是函数在给定点下降最快的方向，这正是前述所讲的在下山过程中寻找的最快方向，所以只要沿着梯度的反方向不断前进；然后重新沿着新参数点梯度的反方向前进，重复此过程，就能走到局部的最低点。

2. 梯度的数学解释

首先给出依据梯度寻求局部最低点的迭代数学公式：

$$\theta_{t+1}=\theta_t-\alpha\cdot\nabla J(\theta_t)$$

此公式的意义是，J 是关于 θ 的一个函数，要从当前所处的位置 t 点，走到 J 的最小点，也就是山脚下。首先确定前进的方向，也就是梯度的反方向，然后走一段距离的步长，达到 $t+1$ 这个点。α 在梯度算法中称为学习率或者步长，由 α 来控制每一步走的距离。其实通过对 α 的控制，每次下降的距离不会因为太多或太少而错过最低点。

梯度前加一个负号则表示朝着梯度相反的方向前进。梯度的方向实际就是函数在此点上升最快的方向，朝着下降最快的方向走，自然就是负的梯度方向，所以此处需要加上负号。

5.2.3 梯度下降算法的实现步骤

将下山的人看作 $J(w)$，即表示目标函数，目的地是最低点，如何到达最低点则是需要解决的问题。梯度下降法的实现步骤如下。

(1) 初始化：随机参数赋值。

随机选取一组初始参数值 θ(如 $w_1,w_2,\cdots,w_n$)，作为迭代搜索的起点，如表 5-1 所示。

(2) 计算误差：量化预测偏差。

基于当前参数 θ 计算预测值 $\hat{y}$，并通过损失函数(如均方误差)衡量 $\hat{y}$ 与真实值 y 的差异。

核心目标：最小化损失函数值，使预测结果趋近真实值。

(3) 调整参数：梯度导向优化。

方向判断：若误差减小，表明调整方向正确，则沿当前梯度反方向继续微调 θ；若误差增大，表明调整方向错误，则反转梯度方向(如增大参数改为减小，反之亦然)。

数学表达式：参数更新公式为 $\theta_{t+1}=\theta_t-\alpha\cdot\nabla J(\theta_t)$，利用负梯度引导参数向函数值减小的方向更新，其中 α 为控制步长的学习率，$\nabla J(\theta_t)$ 为损失函数在当前点的梯度。

(4) 迭代收敛：逼近最优解。

重复“计算误差→调整参数”的过程，如同下山过程中逐次寻找更陡峭的下坡方向，直至损失函数收敛至极小值点，即对应参数 θ 的最优解。

表 5-1 梯度下降法的关键逻辑图示

步 骤	核心动作	数学对应	类比场景
初始化	随机设定参数起点	$\theta_0\leftarrow$random	站在山上某点
计算误差	计算预测与真实的差距	$J(\theta)=\frac{1}{n}\sum(\hat{y}-y)^2$	确认当前位置的高度
调整参数	沿梯度反方向更新参数	$\theta\leftarrow\theta-\alpha\nabla J$	朝最陡下坡方向走一步
迭代收敛	重复直至误差最小	收敛条件：$\nabla J\approx 0$	走到山谷最低点

5.2.4 梯度下降算法举例

1. 单变量函数

例如一个单变量函数为 $J(\theta)=\theta^2$，函数的微分直接求导就可以得到 $J'(\theta)=2\theta$。

初始化(也就是起点)可以随意设置，这里设置为 1，则 $\theta^0=1$；学习率也可以随意设置，这里设置为 $\alpha=0.4$。依据梯度下降算法的计算公式进行迭代计算如下。

$$\theta^0=1$$

$$\theta^1=\theta^0-\alpha J'(\theta^0)=1-0.4\times 2=0.2$$

$$\theta^2=\theta^1-\alpha J'(\theta^1)=0.2-0.4\times 0.4=0.04$$

$$\theta^3=0.008$$

$$\theta^4=0.0016$$

如图 5-2 所示，经过 4 次运算，基本就达到了函数的最低点，也就是山脚下。

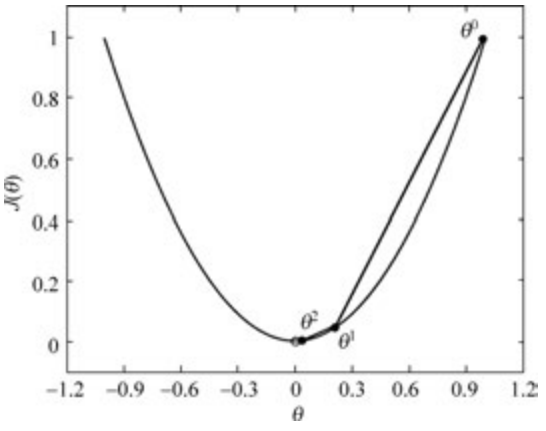

图 5-2 梯度下降算法求 $J(\theta)=\theta^2$ 最小值示意图

2. 多变量函数

例如，有一个目标函数为 $J(\boldsymbol{\Theta})=\theta_1^2+\theta_2^2$，起点设置为 $\boldsymbol{\Theta}^0=(1,3)$，学习率为 $\alpha=0.1$，函

数的梯度为$\nabla J(\boldsymbol{\Theta})=[2\theta_1+2\theta_2]$。多次迭代情况如下。

$\boldsymbol{\Theta}^0=(1,3)$

$\boldsymbol{\Theta}^1=\boldsymbol{\Theta}^0-\alpha\nabla J(\boldsymbol{\Theta})=(1,3)-0.1\times(2\times1,2\times3)=[0.8,2.4]$

$\boldsymbol{\Theta}^2=\boldsymbol{\Theta}^1-\alpha\nabla J(\boldsymbol{\Theta})=[0.8,2.4]-0.1\times(2\times0.8,2\times2.4)=[0.64,1.92]$

$\boldsymbol{\Theta}^3=[0.5124,1.536]$

$\boldsymbol{\Theta}^4=[0.4096,1.229]$

$\vdots$

$\boldsymbol{\Theta}^{10}=[0.1034,0.3221]$

$\vdots$

$\boldsymbol{\Theta}^{50}=[1.1417\times10^{-5},3.4254\times10^{-5}]$

$\vdots$

$\boldsymbol{\Theta}^{100}=[1.6296\times10^{-10},4.8889\times10^{-10}]$

当迭代到第 100 次时，基本靠近了函数的最小值点，如图 5-3 所示。

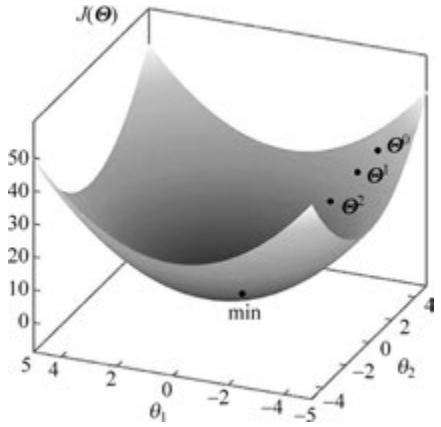

图 5-3　梯度下降算法求 $J(\boldsymbol{\Theta})=\theta_1^2+\theta_2^2$ 最小值示意图

值得注意的是，梯度下降算法的收敛特性与目标函数的形态密切相关：若损失函数为非凸函数，算法可能收敛于局部极小值，如图 5-4 所示；若为凸函数（如线性回归的均方误差损失），则因凸函数的局部极小值即是全局最小值的性质，可保证收敛至全局最优解。这一特性使得梯度下降算法在不同优化场景中需结合函数特性调整策略，以提升求解效率与准确性。

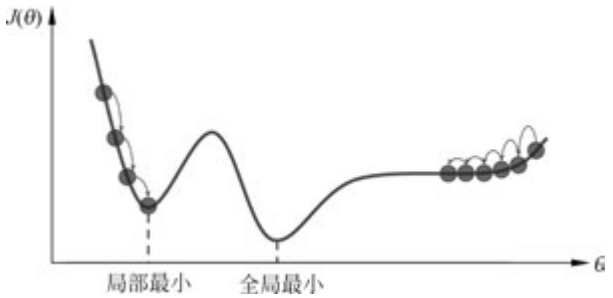

图 5-4　局部最小值和全局最小值

5.2.5 梯度下降算法的不足和改进

1. 梯度下降算法的不足

梯度下降算法的执行过程，可形象类比为“摸索下山”行为，尽管整体方向指向函数值递减的山谷（最优解），但实际优化中存在三类典型挑战。

第一类，学习率的设置困境。学习率如同下山步幅：若步幅过大，即学习率过高，参数更新易跳过真实最优解，引发迭代震荡，甚至因跨越幅度过猛导致算法无法收敛；若步幅过小，即学习率过低，则会显著拖慢收敛速度，尤其当函数曲面（如损失函数的曲率）趋于平缓时，逼近最优解的迭代周期将大幅增加，造成计算资源的低效消耗。

第二类，局部最优与鞍点的陷阱。当算法陷入局部最优时，相当于误将当前山谷认作最低点，而实际存在更优的全局盆地（全局最优解），但因当前区域梯度导向“局部谷底”，算法会错误停止迭代；遇到鞍点时，参数处于马鞍形区域——梯度虽近似为零（看似平坦），但并非真正最优解，算法可能因误判收敛条件而停滞，无法继续探索更优参数。

第三类，数据规模适配难题。传统批量梯度下降算法需遍历全量数据集计算梯度，如同研究完整山路地图再下山，计算成本随数据规模呈指数增长；随机梯度下降算法仅依赖单样本更新参数，虽速度提升但因样本随机性，参数更新方向易偏离最优路径（类似盲人摸象）；小批量梯度下降算法虽折中二者，但批量大小的选择需在计算效率与更新稳定性间权衡，难以实现绝对平衡。

综上，梯度下降算法面对复杂函数（如非凸损失函数）与大规模数据场景时，需结合自适应学习率、动量机制、正则化策略等改进手段，通过精细调优克服固有缺陷，提升、优化效率与稳定性。

2. 梯度下降算法的改进

虽然梯度下降算法很简单，但是在实际应用中，为了提高算法的效率和稳定性，通常需要对其进行优化。梯度下降算法衍生出了全梯度下降、随机梯度下降和小批量梯度下降三种经典形式，它们在计算效率、收敛稳定性等方面各有特点，适用于不同的训练场景。

（1）全梯度下降算法是最基础的形式，也常被称为“批量梯度下降”。它的核心逻辑是：每次参数更新时，都使用全部训练数据计算损失函数的梯度。例如，在训练一个简单的线性回归模型时，全梯度下降算法会遍历所有样本，计算每个样本预测值与真实值的误差，再汇总得到整体的梯度方向，然后沿着这个方向调整模型参数，如权重和偏置。这种算法的优势在于，每次迭代都能利用全局数据信息，得到的梯度方向更为准确，最终能稳定收敛到损失函数的全局最小值，前提是函数需为凸函数。但它的缺点也十分明显：当训练数据量极大，如百万级、亿级样本时，每次计算梯度都需要加载所有数据，会占用大量内存资源，且迭代速度极慢，难以满足大规模数据训练的实时性需求。

（2）随机梯度下降算法是为解决全梯度下降算法的效率问题而生，它的核心改进是用单个样本的梯度代替全局梯度。具体来说，每次参数更新时，随机从训练集中选取一个样本，仅通过这个样本的误差计算梯度，然后立即调整参数。例如，在图像分类任务中，随机梯度下降算法不会等待所有图像数据加载完成，而是随机选一张图像计算梯度并更新模型。这种“随机采样、即时更新”的方式，使得每次迭代的计算量大幅降低，内存占用小，迭代速度极快，甚至可以在数据未完全加载时就开始训练（即在线学习）。不过，由于单个样

本的梯度具有较强的随机性，梯度方向波动较大，可能会在最优值附近来回震荡，难以稳定收敛到精确的最小值，甚至可能在接近最小值时偏离方向，导致训练过程不够稳定。

(3) 小批量梯度下降算法则是前两种算法的折中方案，它结合了两者的优势，避免了各自的缺陷。其核心思路是：每次参数更新时，从训练集中随机选取一个固定大小的“小批量”样本，如 32 个、64 个、128 个样本等，用这组样本的平均梯度来调整参数。例如，在训练神经网络时，每次加载 64 张图像组成一个小批量，计算这 64 张图像的平均损失梯度，再进行参数更新。这种方式既不像全梯度下降算法那样依赖全部数据，因此减少了计算量和内存占用，提升迭代速度；也不像随机梯度下降算法那样依赖单个样本，因为小批量样本的平均梯度能降低随机性，使梯度方向更接近全局梯度，训练过程更稳定。小批量的大小(通常称为 Batch Size)是需要根据任务调整的超参数：批量过大会接近全梯度下降算法，效率降低；批量过小则接近随机梯度下降算法，稳定性变差。在实际的机器学习和深度学习任务中，小批量梯度下降算法是应用最广泛的优化算法之一，成为平衡训练效率与收敛稳定性的首选方案。

5.3 遗传算法

自然界的生物体在遗传、选择、交叉和变异等的作用下，优胜劣汰，不断地由低级向高级进化和发展，人们将这种适者生存的进化规律的实质加以模式化而构成一种优化算法，即进化计算。进化计算是一系列的搜索技术，包括遗传算法、进化规划、进化策略等，它们在函数优化、模式识别、机器学习、神经网络训练、智能控制等众多领域都有着广泛的应用。其中，遗传算法是进化计算中具有普遍影响的模拟进化优化算法。

5.3.1 遗传算法概述

遗传算法最早源于 John H. Holland 和他的团队研究的元胞自动机，Holland 于 1975 年出版的专著 *Adaptation in Natural and Artificial Systems* 被认为是遗传算法研究的开始。到 20 世纪 90 年代，遗传算法在解决组合优化问题方面的优势才凸显起来，它的基本思想就是借鉴生物进化的自然选择过程，首先获得一组候选的解，这些解在选择的压力下发生进化，来适应适者生存的法则，所以遗传算法可以看作通过运用一系列遗传机制，作用于一组解而不是单个解的属性来产生一个可接受的最优解的局部搜索方法。

遗传算法首先要将问题的解，通过编码的方式转换为一组有限数量的染色体种群。染色体也就是在一些位置(基因座)具有不同值(等位基因)的固定结构，如二进制字符串。基因座就是染色体结构中的一些固定位置，它们对应着问题解的一些关键变量，如二进制串中每一位都可以看作一个基因座，能存放 0 或 1。等位基因就是在基因座上可以取到的值，如二进制串中的 0 或 1。群体中的每个染色体都要使用一些适应度函数进行评估，群体的成员有选择性地进行配对来产生后代，适应度越高的群体成员越有可能产生后代。遗传因子在繁殖过程中用来促进后代继承它们父母的属性。后代也会被评估并放入种群，并有可能取代上一代中一些较弱的成员。于是，遗传算法的搜索机制主要由三个阶段组成：评估每个染色体的适应度；选择父代染色体；对父代染色体进行重组(交叉)和变异运算。通过这些运算获得的新染色体形成了下一代的群体。这个过程会一直重复，直到系统停止提

升。适者生存法则用来保证算法从上一代到下一代的过程中，解的总体质量不断提高。由此，得到遗传算法的基本流程如图 5-5 所示。

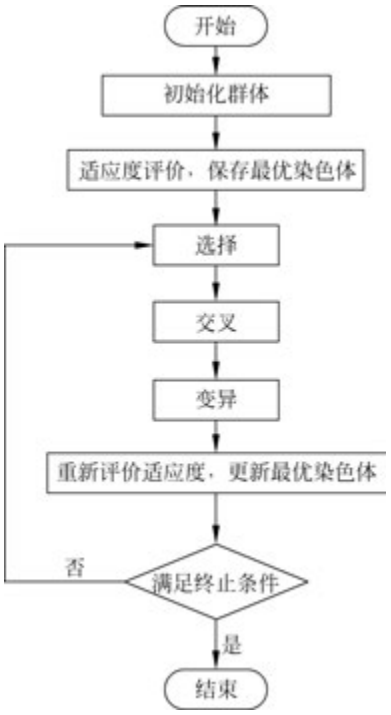

图 5-5 遗传算法的基本流程

5.3.2 编码

使用遗传算法，首先要解决如何对具体问题进行编码。编码是设计遗传算法时的一个关键，直接影响到后继的运算过程和算法效率。编码在很多情况下与问题本身相关，不同的问题需要采用不同的方法。比较常见的编码方法有二进制编码法、值编码法、排列编码法、树编码法等。

1. 二进制编码法

二进制编码法是最早被采用的方法，也最常用。它将每个染色体用一串由 0 和 1 组成的二进制串来表示。这种编码方法和生物染色体的组成类似，每个位可以表示两种不同的状态：0 和 1，这就相当于染色体两种不同的碱基。足够长的二进制串就能表示出足够多的染色体了。

例如，求解“在 0～15 的范围内找使函数 $f(x)=x^2$ 最大的 x”问题时，用二进制编码法进行遗传算法编码：先确定编码规则，因 0～15 共 16 个数，用 4 位二进制数即可表示（$2^4=16$，覆盖全部范围），例如十进制数 5，对应二进制编码为 0101；接着初始化种群，随机生成若干 4 位二进制串，如 0010、1011、0110 等，这些就是初代种群个体。

二进制编码法的优点是编、解码操作比较简单，遗传算法中的一些运算，如交叉和变异等很容易实现；主要缺点是在求解高维优化问题时，二进制编码长度非常长，会降低算法的搜索效率，在执行遗传操作后有时还需要对结果进行校正，还有许多问题不太适合直接用

这种方法编码。

2. 值编码法

在有些情况下，如某些函数的值可能是连续的，如果用二进制编码法位数少了可能精度不够，而精度高时位数就多，搜索空间变大，导致搜索效率降低。这时可以考虑直接用值进行编码，每个染色体就是一组值，值可以是和问题相关的任何内容，可以是实数、字符或者一些复杂的对象。

例如，为了找到神经网络的最佳权重值，可以将实数值直接编码成染色体 A，每个实数值就对应着神经元的输入权重。再如染色体 B、染色体 C，就适合其他某些问题的场景。

染色体 A：[1.2345,3.3278,0.3456,1.1122,0.7788]。

染色体 B：[AFBEDCAEBGDCBAAFG]。

染色体 C：[(Left),(Back),(Left),(Forward),(Forward)]。

值编码法非常适合用于一些特殊问题的求解，它很直观，方便引入和问题相关的专门知识来提升算法的搜索能力，这时常常需要针对问题开发一些新的交叉和变异运算方法。

3. 排列编码法

对于一些排序问题，采用排列编码法比较合适，如旅行商问题或任务分发次序问题等，每个染色体是一串数字的排列，代表着一个解的次序，例如：

染色体 1：1 5 2 4 6 8 7 9 3。

染色体 2：2 5 8 6 7 3 1 4 9。

排列编码法只对排序类问题有效，在求解这类问题时，有时也需要在交叉和变异后进行修正，以保持染色体的一致性，即使交叉和变异后的染色体也是可能会出现的序列次序。

4. 树编码法

在遗传算法的树编码法中，染色体直接表示树形结构，常用于优化问题如数学表达式或决策树的进化，对树结构进行交叉和变异相对来说比较容易。例如，考虑一个简单的数学表达式优化任务，目标是找到一个高效计算 $(x+y)*z$ 的树结构。这里，一个个体可能编码为树，其中根节点是乘法操作(*)，左子节点是加法操作(+)，其子节点为变量 x 和 y，右子节点是变量 z；在遗传操作中，交叉可能交换两个父代的子树(如将一个父代的加法子树替换到另一个父代的乘法树中)，生成新后代如 $(x+w)*z$，从而通过进化搜索更优表达式。

5.3.3 交叉和变异

基因交叉又称为基因重组，就是把两个父代染色体的部分结构加以替换，生成新的个体的操作。通过这种方式，父代的特征能遗传给子代，子代应该能够部分或者全部地继承父代的结构特征和优良基因，更适应环境。

交叉操作还可以分为无性交叉、有性交叉、多亲交叉三种。无性交叉是指子代由一个父代染色体产生，有性交叉是指子代由两个父代产生，多亲交叉指子代由两个以上父代产生。

变异也称突变，是指产生新的染色体，表现出新的性状。在生物界中，在基因复制过程中出现变异的概率一般是很小的，也可能会产生一些不利的因素，但也有可能会改进已有

的特征或产生新的特征。在遗传算法中，变异就是将染色体编码中的一部分按照一个较小的概率进行随机变化，其目的是维持群体的多样性，为选择和交叉过程中可能丢失的遗传基因进行修复和补充。变异概率一般都不大，常取 0.005 左右。

交叉和变异是遗传算法中的两个基本操作，它们对遗传算法性能的影响很大。交叉和变异算子的类型和实现方式与编码相关，同时也与所求解的问题有关。不同的编码方式对应着不同的交叉和变异方法。

二进制编码和值编码的交叉方式相似，常用的交叉方式如下。

1. 单点交叉

单点交叉是指在父代个体的染色体上随机选择一个交叉点，然后将该点之后的部分染色体进行交换，生成两个新的子代个体。单点交叉操作简单，但可能导致搜索空间受限，如图 5-6 所示。

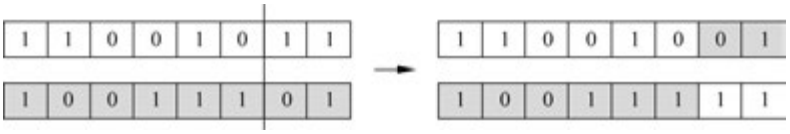

图 5-6 单点交叉

2. 两点/多点交叉

两点交叉是在染色体上随机选择两个交叉点，将两个交叉点之间的中间片段进行交换，其余部分保持不变。多点交叉则是单点与两点交叉的推广形式，指在染色体上选择多个交叉点（如 3 个、4 个或更多），然后交替交换片段。当交叉点数量趋近于染色体长度时，就接近于均匀交叉。在父代个体的染色体上随机选择两点/多点交叉，然后将这些点之间的部分染色体进行交换。两点/多点交叉可以增加搜索空间的多样性，但操作相对复杂。图 5-7 所示为二进制编码染色体使用三点交叉的示例。

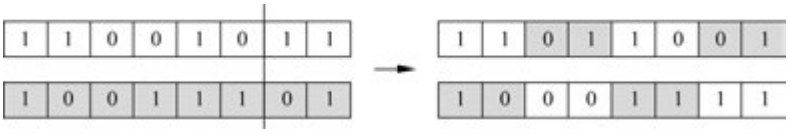

图 5-7 三点交叉

3. 均匀交叉

均匀交叉是指每个基因位都以一定的概率进行交换，生成新的子代个体。均匀交叉可以保持父代个体的优良特性，但可能导致搜索过程过于随机。

4. 运算交叉

运算交叉是指两个父代间的基因通过执行一些运算来产生子代。如图 5-8 所示，两个二进制编码的染色体执行与运算交叉。

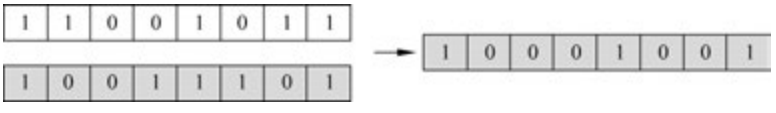

图 5-8 运算交叉

两个实数值之间的运算，常按如下公式进行：$V_{son}=V_{father1}+\alpha\times(V_{father2}-V_{father1})$

其中，α 是比例因子，由在$[-d,1+d]$区间均匀分布的随机数产生，一般取 $d=0.25$。通常将适应度值更好的设为 father2，α 越大代表子代越像 father2，α 越小代表子代越偏向 father1。

对于二进制编码的变异操作，通常是在完成交叉后，随机选一个位点进行取反。对于值编码，如果是实数值编码，则交叉后，通常随机选一些值加或减一个很小的数字。

如果染色体用的是排列编码法，常采用单点交叉，并且在选择了交叉点后，从开始到交叉点这一段的值从一个父代复制到字节中，剩下的值则扫描另一个父代，按照其在另一个父代的顺序加到子节点中。

例如，有两个父代染色体 $A=[1,0,1,0,1]$和 $B=[0,1,0,1,0]$，假设随机选择的交叉点位置为第三位。则交叉后得到两个子代个体 $C=[1,0,1,1,0]$和 $D=[0,1,0,0,1]$。

排列编码中发生变异是指顺序发生变化，通常是随机选两个数交换位置。例如，如果交叉后的子代(1,2,3,4,5,6,8,9,7)发生变异，随机选两个数交换位置，如 2 和 9 交换位置，则变异后的染色体编码为(1,9,3,4,5,6,8,2,7)。

5.3.4 选择

选择操作也称为复制，也是一个决定算法性能的关键环节，其核心功能是从当前种群中筛选出具备优质基因的染色体作为父代，为后续交叉重组提供遗传物质基础。那么，选择操作的生物学原理与实施逻辑如何呢？依据达尔文进化论优胜劣汰的自然选择法则，自然界中只有适应环境能力最强的个体才能存活并繁衍后代。在遗传算法中，这一规律通过“适应度”指标量化呈现，即每个个体的适应度值直接反映其对问题求解环境的适配能力，适应度越高的个体，在选择操作中被选中作为父代的概率越大。这一机制确保了优质基因能在种群迭代中得以保留和传递，从而推动算法向更优解方向进化。

在遗传算法中，也有类似的适应度函数，也叫评估函数，用来判断群体中的个体的优劣程度。通常，可以直接根据所求问题的目标函数进行评估，而且一般也不需要其他外部信息。例如，对于背包问题，就可以直接用对被挑选物品的价值求和作为适应度函数。再如，在一些求函数最大值的问题中，就可以将该函数直接作为适应度函数。但有些时候，目标函数并不一定适合直接作为适应度函数，这时需要对目标函数进行转换。

对于适应度函数的设计，一般要求计算应该足够快，因为会被反复使用。另外，适应度函数必须可以定量测量，能够转换目标函数值为相对的适应度值；有时还需要对目标函数进行变换，也称为标定，放大种群中优劣个体之间的差别，增强选优功能。为了将来方便用累加概率等方式进行选择，通常还需要把适应度函数设计成为单值、非负的形式。如果目标函数有约束，则对适应度函数还需要添加惩罚项。

对于一些求最大值的问题，如果目标函数为 $f(x)$，则适应度函数 $F(x)$一般可以取为下述三种中的一种。

(1) $F(x)=f(x)$

(2) $F(x)=\begin{cases} f(x)-C_{\min}, & f(x)>C_{\min} \\ 0, & \text{其他} \end{cases}$

其中，$C_{\min}$ 为 $f(x)$的最小估计。

(3) $F(x)=\dfrac{1}{1+c-f(x)}$, $c\geqslant 0, c-f(x)\geqslant 0$

其中,c 为目标函数界限的保守估计。

相应地,如果是求最小值的问题,则适应度函数 $F(x)$可以取为下述三种中的一种。

(1) $F(x)=-f(x)$

(2) $F(x)=\begin{cases}C_{\max}-f(x), & f(x)<C_{\max}\\ 0, & \text{其他}\end{cases}$

其中,$C_{\max}$ 为 $f(x)$的最大估计。

(3) $F(x)=\dfrac{1}{1+c+f(x)}$, $c\geqslant 0, c+f(x)\geqslant 0$

其中,c 为目标函数界限的保守估计。

一旦能计算出种群中每个个体的适应度值,就可以进行选择操作了。然而,选择并不是只挑适应度值高的个体作为双亲,否则就成了确定性优化方法,使种群很快收敛到局部最优解;如果只是随机选择,又体现不出择优选择的优势,导致算法长时间不能收敛,甚至无法收敛。因此,需要找到一个策略,既能使算法较快收敛,又能维持种群的多样性。

按照适者生存的基本原则,需要让适应度值高的个体被选中的机会更大,根据适应度值来确定个体被选中的概率。选择的策略很多,如轮盘赌选择、玻尔兹曼选择、排序选择、联赛选择、稳态选择等。下面将介绍一些常用的选择方法。

(1) 轮盘赌选择法。

轮盘赌选择法是依据个体的适应度值计算每个个体在子代中出现的概率,并按照此概率随机选择个体构成子代种群,因此该方法也被称为适应度比例法。就和转轮盘抽奖的游戏一样,将所有的染色体都放在轮盘上,依据其适应度值来决定其在轮盘上占据的范围,然后转动转盘来选择用于交叉的染色体。这样,适应度值大的染色体被选中的次数就会多了,如图 5-9 所示。

轮盘赌选择法的出发点是适应度值越好的个体被选中的概率越大。因此,在求解最大化问题时,可以直接采用适应度值来进行选择。但是在求解最小化问题时,必须首先将问题的适应度函数进行转换(如采用倒数或相反数),以将问题转换为最大化问题。

(2) 排序选择法。

轮盘赌选择法存在的局限,就是当适应度值的差别非常大时,如某个染色体的适应度值占据了轮盘 90%以上的面积时,那么其他的染色体将很难有机会被选中。排列序选择法通过重新映射适应度值到排列序号来解决这个问题,它首先对种群按照适应度值进行排序,然后对每个染色体以排列序号重新分配适应度值,最差的适应度值为 1,倒数第二差的为 2,以此类推,最好的适应度值将设为 N,N 为种群染色体的数量。

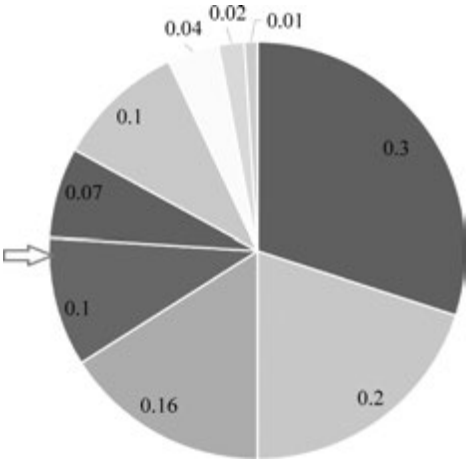

图 5-9 轮盘赌选择法

通过排序选择法进行转换以后,所有染色体

就都有机会被选中了，不过这种选择策略会导致收敛变慢。这是因为它大大减少了最佳染色体和其他染色体之间的差异。

(3) 稳态选择法。

稳态选择并不算一个特定的选择父代染色体的策略。它的主要思想是染色体的大部分应该存活到下一代，所以每一代中选择一小部分适应度值高的来产生后代，然后用新产生的后代来替换一些适应度值差的染色体。替换的策略一般可以采取以下几种。

① 子代的适应度值优于最差的父个体，则子代替换父个体。

② 用玻尔兹曼选择来确定是否替换。

③ 替换整个种群中最差的个体。

(4) 精英选择法。

在交叉和变异的过程中，很大的可能性会丢失最好的染色体，精英选择法就起到了保护这些最佳个体的作用。它首先会将一些最好的染色体复制到新种群，对于剩下的个体再采用传统的方法进行选择。因为防止了丢失发现的最好解，所以精英选择法通常可以很快地增强遗传算法的性能。

通常，选择种群数量的 2%～5%适应度值最佳的个体，效果最为理想。

5.3.5 参数设定

在遗传算法中，不存在适用于所有问题的通用参数设定方法，通常需要针对特定问题通过实验来确定参数。以下为遗传算法研究中常用参数的推荐设定。值得注意的是，其中多数设定主要针对二进制编码的遗传算法而言。

(1) 交叉率。交叉率通常宜设定为较大值，范围为 80%～95%。不过也有研究结果表明，对于某些问题，将交叉率设为 60% 时效果最佳。

(2) 变异率。变异率通常设定得极小，常见取值范围为 0.1%～1%。

(3) 种群大小。值得注意的是，较大的种群规模通常并不会加快遗传算法找到最优解的速度，其大小通常设定为 20～30。也有文献推荐 50～100 的范围。另有研究表明，最佳种群规模取决于编码方式及编码字符串的长度。例如，若采用 32 位编码的染色体，种群规模设定为 32 为宜；若采用 16 位编码的染色体，则种群规模设定为 16 更佳。

(4) 选择。基本的轮盘赌选择法常被采用，排序选择法有时效果更佳；另有一些复杂策略会像模拟退火算法那样，会在遗传算法运行过程中动态调整选择参数。选择过程中，结合精英选择法通常能获得更优效果。

(5) 编码。编码方法的选择一般取决于问题以及问题实例的规模。

(6) 交叉和变异类型。交叉和变异运算方法的选择通常取决于编码方式和问题本身。

5.3.6 遗传算法举例

利用遗传算法，求解区间[0,30]上函数 $y=x^3$ 的最大值。接下来给出详细的过程。

1. 问题分析与参数设定

(1) 问题分析。

求解闭区间[0,30]上一元三次函数 $y=x^3$ 的最大值。由于函数在[0,30]上单调递增，理论最优解为 $x=30$，但需通过遗传算法模拟“进化”过程逼近该解，同时需注意，5 位二进

制最大表示为 31(11111),超出区间[0,30]上限,故编码中需排除 11111,仅保留 00000(0)～11110(30)的有效编码。

(2) 参数设定(参考标准遗传算法流程)。

种群规模:4,每代含 4 个有效可行解,平衡计算效率与搜索覆盖度。

编码方式:5 位二进制编码。这是因为 4 位二进制最大数值仅为 15,无法覆盖 30;5 位可覆盖 0～31 的数值,剔除 31 这个数值后满足区间需求,如 $x=25$ 对应编码 11001。

交叉率(P_c):100%,所有染色体参与交叉,加速种群向优解进化。

变异率(P_m):0.001,主要用来控制基因变异概率,避免过早收敛;总基因位数=种群规模×编码长度 = 4×5=20,每代理论变异位数= 20×0.001=0.02,故多数迭代无实际变异。

适应度函数:$f(x)=x^3$,这里直接以目标函数值衡量染色体的适应能力,值越大对应解越优,且与函数单调性一致,无须额外转换。

终止条件:种群中出现有效编码 11110,即 $x=30$,或连续 3 代适应度最大值无提升,确保能找到最优解。

2. 完整的迭代过程

遗传算法通过"初始化种群→计算适应度→选择(复制)→交叉→变异"的循环迭代优化种群,每代操作均围绕"淘汰劣解、保留优解、生成新解"展开,具体过程如下。

1) 第一代种群 S_1 操作

(1) 初始化种群。

从区间[0,30]中随机选择 4 个整数,转换为 5 位二进制编码(染色体),得到初始种群 S_1。

$s_1=12$(十进制)=01100(二进制编码)

$s_2=20$(十进制)=10100(二进制编码)

$s_3=27$(十进制)=11011(二进制编码)

$s_4=8$(十进制)=01000(二进制编码)

(2) 计算适应度、选择概率与累积概率。

适应度计算:根据 $f(x)=x^3$,计算每个个体的适应度(反映解的优劣)。

$f(s_1)=12^3=1728$,$f(s_2)=20^3=8000$,$f(s_3)=27^3=19\,683$,$f(s_4)=8^3=512$。

总适应度:$\sum f(x_i)=1728+8000+19\,683+512=29\,923$。

选择概率:$P(x_i)=\dfrac{f(x_i)}{\sum f(x_i)}$,反映个体被遗传到下一代的概率。

累积概率:$q_i=\sum_{j=1}^{i}P(x_j)$,用于赌轮选择法,划分选择区间,确保概率连续性。

将上述结果整理为表格,如表 5-2 所示。

表 5-2 第一代种群 S_1 的适应度、选择概率与累积概率

染色体编号	二进制编码	x	适应度 $f(x)$	选择概率 $P(x)$(保留 2 位小数)	累积概率(q_i)(保留 2 位小数)
s_1	01100	12	1728	1728/29 923=0.06	0.06
s_2	10100	20	8000	8000/29 923=0.27	0.06+0.27=0.33

续表

染色体编号	二进制编码	x	适应度 $f(x)$	选择概率 $P(x)$（保留2位小数）	累积概率(q_i)（保留2位小数）
s_3	11011	27	19 683	19 683/29 923=0.66	0.33+0.66=0.99
s_4	01000	8	512	512/29 923=0.02	0.99+0.02=1.00

(3) 选择操作(赌轮选择法)。

在[0,1]区间生成4个均匀分布的随机数：$r_1=0.25$、$r_2=0.70$、$r_3=0.45$、$r_4=0.95$。

根据累积概率匹配选中个体：

$r_1=0.25=(0.06,0.33]$→选中 s_2

$r_2=0.70=(0.33,0.99]$→选中 s_3

$r_3=0.45=(0.33,0.99]$→选中 s_3

$r_4=0.95=(0.33,0.99]$→选中 s_3

得到选择后的复制种群：

$s_1'=10100$(二进制编码)=20 (十进制)

$s_2'=11011$(二进制编码)=27(十进制)

$s_3'=11011$(二进制编码)=27(十进制)

$s_4'=11011$(二进制编码)=27(十进制)

(4) 交叉操作(交换后两位基因)。

配对方式：s_1' 与 s_2'为一对，s_3' 与 s_4' 为一对。

交叉规则：交换每对染色体的最后两位基因，确保编码长度不变，避免无效解。

第一对：$s_1'=10100$(后两位“00”)与 $s_2'=11011$(后两位“11”)→交换后得 $s_1''=10111(23)$、$s_2''=11000(24)$。

第二对：$s_3'=11011$(后两位“11”)与 $s_4'=11011$(后两位“11”)→交换后仍为 $s_3''=11011(27)$、$s_4''=11011(27)$。

(5) 变异操作。

因每代理论变异位数=0.02<1，无基因发生变异，最终得到第二代种群 S_2。

$s_1=23$(十进制)=10111(二进制编码)，适应度 $f(x)=23^3=12\ 167$

$s_2=24$(十进制)=11000(二进制编码)，适应度 $f(x)=24^3=13\ 824$

$s_3=27$(十进制)=11011(二进制编码)，适应度 $f(x)=27^3=19\ 683$

$s_4=27$(十进制)=11011(二进制编码)，适应度 $f(x)=27^3=19\ 683$

2) 第二代种群 S_2 操作

(1) 计算适应度、选择概率与累积概率。

总适应度：12 167+13 824+19 683+19 683=65 357。

其余参数如表5-3所示。

表 5-3 第二代种群 S_2 的适应度、选择概率与累积概率

染色体编号	二进制编码	x	适应度 $f(x)$	选择概率 $P(x)$（保留 2 位小数）	累积概率 q_i（保留 2 位小数）
s_1	10111	23	12 167	12 167/65 357＝0.19	0.19
s_2	11 000	24	13 824	13 824/65 357＝0.21	0.19＋0.21＝0.40
s_3	11 011	27	19 683	19 683/65 357＝0.30	0.40＋0.30＝0.70
s_4	11 011	27	19 683	19 683/65 357＝0.30	0.70＋0.30＝1.00

(2) 选择操作(赌轮选择法)。

生成随机数：$r_1=0.55$、$r_2=0.80$、$r_3=0.30$、$r_4=0.65$。

匹配选中个体：

$r_1=0.55\in(0.40,0.70]$→选中 s_3

$r_2=0.80\in(0.70,1.00]$→选中 s_4

$r_3=0.30\in(0.19,0.40]$→选中 s_2

$r_4=0.65\in(0.40,0.70]$→选中 s_3

复制种群：

s_1'＝11011(二进制编码)＝27(十进制)

s_2'＝11011(二进制编码)＝27(十进制)

s_3'＝11000(二进制编码)＝24(十进制)

s_4'＝11011(二进制编码)＝27(十进制)

(3) 交叉操作(交换后三位基因)。

配对方式：s_1' 与 s_3'为一对，s_2' 与 s_4'为一对。

交叉规则：交换每对染色体的最后三位基因。

第一对：s_1'＝11011(后三位“011”)与 s_3'＝11000(后三位“000”)→交换后得 s_1''＝11000(24)、s_3''＝11011(27)。

第二对：s_2'＝11011(后三位“011”)与 s_4'＝11011(后三位“011”)→交换后仍为 s_2''＝11011(27)、s_4''＝11011(27)。

(4) 变异操作。

无变异，得到第三代种群 S_3(适应度最大值仍为 19 683，需继续迭代)，如表 5-4 所示。

表 5-4 第三代种群 S_3

染色体编号	二进制编码	十进制值(x)	适应度 $f(x)$
s_1	11000	24	24^3＝13 824
s_2	11011	27	27^3＝19 683
s_3	11011	27	27^3＝19 683
s_4	11011	27	27^3＝19 683

3) 第三代种群 S_3 操作(关键迭代)

(1) 计算适应度、选择概率与累积概率。

总适应度：13 824＋19 683×3＝72 873；选择概率与累积概率如表 5-5 所示。

表 5-5　第三代种群 S_3 的适应度、选择概率与累积概率

染色体编号	二进制编码	x	适应度 $f(x)$	选择概率 $P(x)$（保留 2 位小数）	累积概率 q_i（保留 2 位小数）
s_1	11000	24	13 824	13 824/72 873＝0.19	0.19
s_2	11011	27	19 683	19 683/72 873＝0.27	0.19＋0.27＝0.46
s_3	11011	27	19 683	19 683/72 873＝0.27	0.46＋0.27＝0.73
s_4	11011	27	19 683	19 683/72 873＝0.27	0.73＋0.27＝1.00

（2）选择操作（赌轮选择法）。

生成随机数：r_1＝0.85、r_2＝0.60、r_3＝0.90、r_4＝0.50。

匹配选中个体：均选中 $s_2/s_3/s_4$（x＝27），复制种群全为 11011(27)。

（3）交叉操作（调整配对与交叉位，突破局部最优）。

配对方式：s_1'＝11011(27)与 s_2'＝11011(27)为一对，s_3'＝11011(27)与 s_4'＝11011(27)为一对。

交叉规则：交换每对染色体的第 3、4 位基因（扩大基因多样性）。

第一对：s_1'＝11011（第 3、4 位 “01”）与 s_2'＝11011（第 3、4 位 “01”）→交换后仍为 11011(27)。

第二对：模拟自然变异，手动调整其中一个染色体为 11101(29，有效编码)，与 11011(27)交叉，交换后两位→11111(31，不在区间[0,30]内，为无效值，剔除)与 11001(25)；重新配对得 11101(29)与 11011(27)交换后三位→11111(无效)与 11001(25)，最终保留 11101(29)。

（4）变异操作。

无变异，得到第四代种群 S_4（出现更优解 x＝29），如表 5-6 所示。

表 5-6　第四代种群 S_4

染色体编号	二进制编码	十进制值(x)	适应度 $f(x)$
s_1	11101	29	29^3＝24 389
s_2	11011	27	27^3＝19 683
s_3	11001	25	25^3＝15 625
s_4	11011	27	27^3＝19 683

4）第四代种群 S_4 操作（最优解出现）

（1）计算适应度、选择概率与累积概率。

总适应度：24 389＋19 683×2＋15 625＝79 380；选择概率与累积概率（x＝29 的选择概率最高，优先被遗传）如表 5-7 所示。

表 5-7　第四代种群 S_4 的适应度、选择概率与累积概率

染色体编号	二进制编码	x	适应度 $f(x)$	选择概率 $P(x)$（保留 2 位小数）	累积概率 q_i（保留 2 位小数）
s_1	11101	29	24 389	24 389/79 380＝0.31	0.31
s_2	11011	27	19 683	19 683/79 380＝0.25	0.31＋0.25＝0.56
s_3	11001	25	15 625	15 625/79 380＝0.20	0.56＋0.20＝0.76
s_4	11011	27	19 683	19 683/79 380＝0.25	0.76＋0.25＝1.00

(2) 选择操作。

生成随机数：$r_1=0.20$、$r_2=0.40$、$r_3=0.85$、$r_4=0.65$。

选中个体：$s_1(29)$、$s_2(27)$、$s_4(27)$、$s_1(29)$，复制种群含 2 个 $x=29$、2 个 $x=27$。

(3) 交叉操作(交换第 2～5 位基因，生成最优解)。

配对：$s_1'=11101(29)$与 $s_2'=11011(27)$。

交叉位：第 2～5 位(二进制编码为 11101 与 11011)。

交换结果：$s_1''=11011(27)$、$s_2''=11110(30$，属于有效区间)。

(4) 变异操作。

无变异，得到第五代种群 S_5，其中 $s_2''=11110(x=30)$，满足终止条件。

3. 终止条件验证

(1) 终止条件满足。

第五代种群 S_5 中出现有效编码 11110，对应 $x=30$，且 $x=30$ 是区间[0,30]的上限，函数 $y=x^3$ 在该点取得理论最大值，故迭代终止。

(2) 解码与最优值计算。

二进制编码 11110 转换为十进制：$1\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0=16+8+4+2+0=30$。

最大值：将 $x=30$ 代入目标函数，得 $y=30^3=27\,000$。

4. 结论

通过遗传算法迭代，一共 5 代，最终在区间[0,30]上找到函数 $y=x^3$ 的最大值为 27 000，对应最优解为 $x=30$。该过程验证了遗传算法通过“选择→交叉→变异”模拟生物进化，可有效逼近单调函数的边界最优解，且参数设定(如 5 位二进制编码、100% 交叉率)能确保搜索效率与解的有效性。

5.4 蚁群算法

蚁群优化算法(Ant Colony Optimization，ACO)简称蚁群算法，蚁群算法是一种源于自然界生物行为的仿生算法。作为通用型随机优化方法，它吸收了昆虫王国中蚂蚁的行为特性，通过其内在的分布式自组织搜索机制，在一系列复杂的组合优化问题求解中取得了成效。作为蚁群智能领域的首个成功实例，蚁群算法曾一度成为该领域的代名词，相关理论研究及改进算法近年来也层出不穷。

5.4.1 基本蚁群算法

1. 蚁群算法的生物学基础

在昆虫世界中，蚂蚁以群居世袭大家庭的形式组成，该群体被称为蚁群。将蚁群比拟为家庭，一方面是因其内部存在亲缘层面的互助关系，另一方面是成蚁划分为蚁王与工蚁两个世袭制等级；蚁群规模差异显著，可从数十个个体延伸至数千万个个体，且具备高度组织化的社会性，其个体间的沟通不仅能借助触觉与视觉建立联系，在开展大规模协调行动时，还可依托外激素等生化信息介质实现高效信息传递。

1991 年，意大利学者 M. Dorigo、V. Maniezzo 等在观察蚂蚁觅食习性时发现，蚂蚁总能

精准找到巢穴与食物源之间的最短路径。经深入研究可知，蚂蚁群体的这种协作功能，是通过一种遗留在往返路径上被称为信息素的挥发性化学物质来完成通信与协调的；化学通信作为蚂蚁采用的基本信息交流方式之一，在其生活习性中发挥着重要作用。通过对蚂蚁觅食行为的进一步研究，学者们明确：整个蚁群正是借助信息素实现相互协作，进而形成正反馈效应，使得多条路径上的蚂蚁逐渐向最短路径聚集。尽管单个蚂蚁的行为模式极为简单，但由大量简单个体构成的蚁群却能展现出极具复杂性的神奇群体行为，因此蚁学家将整个蚁群视为一个功能强大的超级生物群。

在蚁群中，工蚁的觅食行为最易观察，其觅食特点是发现食物后不立即进食，而是将食物搬运回巢穴与其他家庭成员共同分享。每个工蚁具有明确职能：日常状态下，工蚁在巢穴附近进行无规则行走以扩大食物搜索范围；一旦发现食物，若自身能够独立搬运，则直接将食物搬回巢穴，若无法独立搬运，则返回巢穴召集同伴；在往返于巢穴与食物源的过程中，工蚁会持续留下外激素嗅迹，且该嗅迹的强度通常与食物的品质及数量呈正相关。当其他工蚁遇到这一嗅迹时，会优先沿嗅迹路径前进，但存在一定的走失率，即选择其他路径，且该走失率与嗅迹强度呈负相关。

基于上述个体行为与信息传递模式，蚁群的集体行为呈现出信息正反馈现象：某一路径上经过的蚂蚁数量越多，后续的蚂蚁选择该路径的概率就越大。蚂蚁个体之间正是通过这种信息交流，最终实现了搜索食物的目标。

2. 蚁群算法的基本原理

蚁群算法的基本原理来自昆虫学家们的观察：生物界中的蚂蚁在寻找食物源时，能在其走过的路径上释放一种蚂蚁特有的分泌物——信息素，使得一定范围内的其他蚂蚁能够觉察并影响其行为。当某些路径上走过的蚂蚁越来越多时，留下的这种信息素也越多，以致后来蚂蚁选择该路径的概率也越高，从而更增加了该路径的吸引强度。蚂蚁全体就靠着这种内部的生物协同机制逐渐形成了一条它们自己事先并未意识到的最短路线。蚁群算法从这种模型中得到启示并用于解决优化问题。蚁群算法中每个优化问题的解都是搜索空间中的一只蚂蚁，蚂蚁都有一个由被优化函数决定的适应度值，它与要释放的信息素成正比，蚂蚁就是根据它周围的信息素的多少决定它们移动的方向，同时蚂蚁也在走过的路上释放信息素，以便影响别的蚂蚁。

假定障碍物的周围有两条道路可以从蚂蚁的巢穴到达食物源（如图 5-10 所示）：蚁巢-A-B-D-食物和蚁巢-A-C-D-食物，分别具有长度 4 和 6。蚂蚁在单位时间内可以移动一个单位长度的距离。开始时所有道路上都未留有任何信息素。

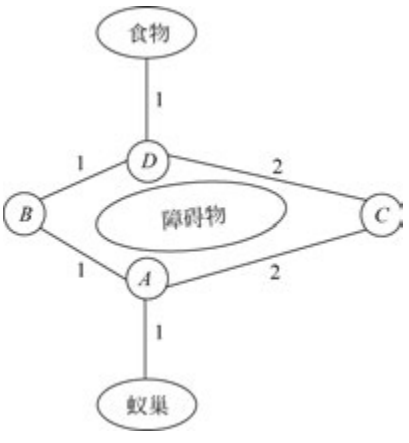

图 5-10　蚁群系统示意图

在 $t=0$ 时刻，20 只蚂蚁从巢穴出发移动到 A 点，它们以相同概率选择左侧或右侧道路，因此平均有 10 只蚂蚁走左侧，10 只蚂蚁走右侧。

在 $t=4$ 时刻，第一组到达食物源的蚂蚁将折回，此时第二组蚂蚁到达 CD 中点处。

在 $t=5$ 时刻，两组蚂蚁将在 D 点相遇。此时 BD 上的信息素浓度和 CD 上的信息素浓度相同，因

为各有 10 只蚂蚁选择了相应的道路，从而有 5 只返回的蚂蚁将选择 BD 而另 5 只蚂蚁将选择 CD，第二组蚂蚁继续向食物方向移动。

在 $t=8$ 时刻，前 5 只蚂蚁将返回巢穴，此时在 AC 中点处、CD 中点处以及 B 点上各有 5 只蚂蚁。

在 $t=9$ 时刻，前 5 只蚂蚁又回到 A 点并且再次面对往左还是往右的选择。这时，AB 上的轨迹数是 20 而 AC 上的轨迹数是 15，因此将有较为多数的蚂蚁选择往左，从而增强了该路线的信息素。随着该过程的继续，两条道路上的信息素数量的差距将越来越大，直至绝大多数蚂蚁都选择了最短的路线。正是由于一条道路要比另一条道路短，因此，在相同的时间区间内，短的路线会有更多的机会被选择。

蚁群算法是一种典型的随机搜索算法，与其他模型进化算法一样，通过候选解组成的群体的动态进化过程寻求问题的最优解。该进化过程具体划分为两个核心阶段：其一为适应阶段，在此阶段中，各候选解会依据算法运行过程中积累的信息持续调整自身结构，以逐步适配问题的求解需求；其二为协作阶段，该阶段聚焦于候选解之间的信息交互，即通过有效的信息交流，候选解群体得以整合优势特征，进而有望生成性能更优的新解。

作为与遗传算法同属一类的通用型随机优化方法，蚁群算法无须依赖任何先验知识即可启动求解过程：算法初始阶段，候选解对搜索路径的选择具有随机性；随着运行推进，算法通过对解空间信息的不断积累与分析，搜索行为逐渐从随机转向规律化，最终逐步逼近并收敛至全局最优解。蚁群觅食现象和蚁群算法的基本定义对照如表 5-8 所示。

表 5-8 蚁群觅食现象和蚁群算法的基本定义对照

蚁群觅食现象	蚁 群 算 法
蚁群	搜索空间的一组有效解（表现为种群规模 m）
觅食空间	问题的搜索空间（表现为问题的规模、解的维数 n）
信息素	信息素浓度变量
蚁巢到食物的一条路径	一个有效解
找到的最短路径	问题的最优解

蚁群算法对搜索空间的认知与探索机制，主要通过模拟蚂蚁的生物行为提炼为以下三个核心方面。

首先是蚂蚁的记忆。在自然蚁群中，一只蚂蚁已搜索过的路径不会在后续搜索中重复选择，蚁群算法通过建立“禁忌列表”这一机制对此行为进行仿真，以此避免候选解陷入对已探索路径的重复搜索，保障搜索过程的有效性与多样性。

其次是蚂蚁利用信息素进行相互通信。蚂蚁会在所选路径上释放名为“信息素”的化学物质，同伴在进行路径选择时，会以路径上的信息素浓度为重要依据，蚁群算法沿用这一逻辑，将信息素作为候选解之间的核心通信媒介，使候选解能够依托信息素浓度这一量化指标实现间接的信息交互，为路径选择提供决策依据。

最后是蚂蚁的集群活动。通过一只蚂蚁的行动很难到达食物源，但整个蚁群进行搜索就完全不同。当某些路径上通过的蚂蚁越来越多时，在路径上留下的信息素浓度也越来越高，导致信息素浓度增大，蚂蚁选择该路径的概率随之增加，从而进一步增加该路径的信息素浓度，而某些路径上通过的蚂蚁较少时，路径上的信息素就会随时间的推移而蒸发。因此，模拟这种现象即可利用群体智能建立路径选择机制，使蚁群算法的搜索向最优解推进。

蚁群算法所利用的搜索机制呈现出一种自催化或正反馈的特征，因此，可以将蚁群算法模型理解成增强型学习系统。由于其他路径上的信息素会随着时间蒸发，因此最终所有的蚂蚁都在最优路径上行进。

5.4.2 蚁群算法的数学模型

蚁群算法的核心是通过模拟蚁群的信息素交流机制和路径选择行为，构建数学模型来求解优化问题。以旅行商问题为例，给定一系列城市和每对城市之间的距离，目标是找到遍历所有城市且总距离最短的回路。蚁群算法的数学模型主要包括路径选择概率模型和信息素更新模型，具体如下。

1. 路径选择概率

工蚁在从当前节点(城市)i 向未访问节点(城市)j 移动时，选择路径(i,j)的概率由信息素浓度和启发式信息共同决定。

1) 符号定义

$\tau_{ij}(t)$：时刻 t，路径上(i,j)的信息素浓度，反映历史路径的优劣，浓度越高，说明该路径被蚂蚁选择的历史越多。

η_{ij}：启发式信息，反映路径的局部优劣，通常定义为路径(i,j)距离的倒数，即 $\eta_{ij}=1/d_{ij}$，d_{ij} 为节点 i 到 j 的距离，距离越近，η_{ij} 越大。

allowed_k：蚂蚁 k 尚未访问的节点集合，即禁忌表的补集，体现工蚁的记忆能力。

α：信息素重要因子，或称为启发式因子，$\alpha\geqslant 0$。值越大，信息素对路径选择的影响越强。

β：启发式信息重要程度因子，或称为期望启发式因子，$\beta\geqslant 0$。值越大，距离对路径选择的影响越强。

2) 概率公式

时刻 t，蚂蚁 k 从节点 i 选择节点 j 的概率 $P_{ijk}(t)$为

$$P_{ijk}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum\limits_{l\in \text{allowed}_k}[\tau_{il}(t)]^{\alpha}\cdot[\eta_{il}]^{\beta}}, & j\in \text{allowed}_k\\ 0, & j\notin \text{allowed}_k\end{cases}$$

若节点 j 未被蚂蚁 k 访问过($j\in\text{allowed}_k$)，则选择概率与信息素浓度的 α 次方、启发式信息的 β 次方成正比；若已访问，则概率为 0，从而避免重复访问同一节点。

2. 信息素更新

信息素的更新包括挥发(避免信息素无限积累)和沉积(蚂蚁留下新信息素)两个过程，通常在每轮迭代(所有蚂蚁完成一次路径遍历)后进行。

信息素挥发：所有路径上的信息素会按固定比例挥发，公式如下。

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)$$

其中，ρ 为挥发因子($0<\rho<1$)，反映信息素的挥发速度。ρ 越大，挥发越快，可促进算法探索新路径；ρ 过小则可能导致信息素过度积累，陷入局部最优。

信息素沉积：蚂蚁 k 在完成一次路径遍历后，会沿其路径沉积信息素，沉积量与路径总长度成反比，即路径越短，沉积越多。设蚂蚁 k 的路径总长度为 L_k，则其在路径(i,j)上的

沉积量 $\Delta\tau_{ij}^{k}$ 计算如下：

$$\Delta\tau_{ij}^{k}=\frac{Q}{L_k}$$

其中，Q 为常数（信息素总量），保证沉积量与路径长度的反比关系。

总信息素更新：综合信息素挥发和沉积。时刻 $t+1$ 的信息素浓度为

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

其中，m 为蚂蚁总数，$\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$ 为所有蚂蚁在路径(i,j)上的信息素总沉积量。

5.4.3　蚁群算法的基本流程

以求解旅行商问题为例，蚁群算法的流程可分为初始化、迭代寻优和终止三个阶段，算法的流程框图如图 5-11 所示，具体步骤如下。

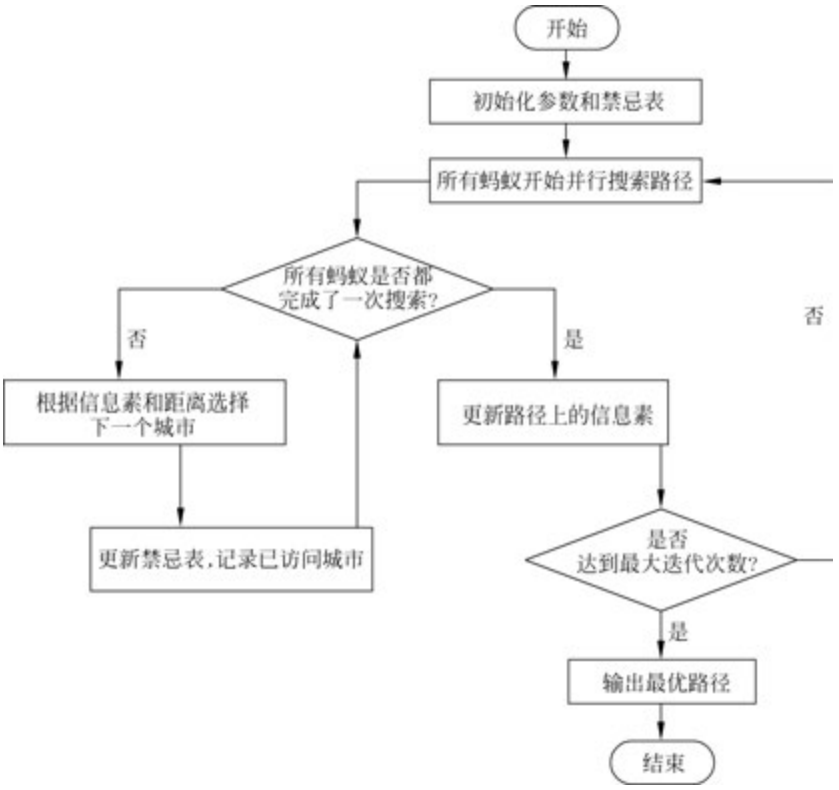

图 5-11　基本蚁群算法流程框图

1. 初始化

设定参数：蚂蚁数量 m（通常与城市数 n 相当，如 $m=n$）、信息素重要因子 α、启发式因子 β、信息素挥发因子 ρ、信息素总量 Q、最大迭代次数 $N_{\max}$。

初始化信息素：所有路径(i,j)的初始信息素浓度为 $\tau_{ij}(0)=\tau_0$（τ_0 为常数，如 $\tau_0=1/(n\cdot L_{\text{avg}})$，$L_{\text{avg}}$ 为初始平均路径长度）。

初始化蚂蚁：m 只蚂蚁随机置于 n 个城市（节点），清空所有蚂蚁的禁忌表（记录已访问

城市)，并将初始城市加入禁忌表。

2. 迭代寻优

重复以下步骤，直到达到最大迭代次数或收敛。

路径构建：每只蚂蚁根据路径选择概率公式，从当前城市选择下一个未访问的城市，加入禁忌表，直至遍历所有城市(完成一次回路)。

计算路径长度：记录第 k 只蚂蚁的路径总长度 L_k，并更新当前迭代的最优路径，即总长度最短的路径。

更新信息素：按信息素更新公式，先挥发所有路径的信息素，再累加所有蚂蚁的信息素沉积量。

重置：清空所有蚂蚁的禁忌表，为下一次迭代做准备。

3. 终止

当迭代次数达到 $N_{\max}$，或最优路径长度连续多轮未改善(收敛)时，算法终止，输出全局最优路径。

5.4.4 参数选择

探索和开发能力的平衡是影响算法性能的一个重要方面，也是蚁群算法研究的关键问题之一。探索能力是指蚁群算法要在解空间中测试不同区域以找到一个局域优解的能力；开发能力是指蚁群算法在一个有希望的区域内进行精确搜索的能力。那么该如何设定蚁群算法中的各种参数，实现探索和开发能力的平衡呢？这里，探索与开发能力实际上就是指全局搜索能力和局域搜索能力。

由于蚁群算法参数空间的庞大性和各参数之间的关联性，很难确定最优组合参数使蚁群算法求解性能最佳，至今还没有完善的理论依据。大多数情况下最优组合参数是通过试验的反复试凑得到的。目前已经公布的蚁群算法参数设置成果都是就特定问题所采用的特定蚁群算法而言，以应用最多的蚁周模型(Ant-Cycle)为例，其最好的试验结果为

$$0 \leqslant \alpha, \quad \beta \leqslant 5, \quad 0.1 \leqslant \rho \leqslant 0.990, \quad 10 \leqslant Q \leqslant 10\,000$$

接下来分析以下参数对蚁群算法性能的影响。

1. 信息素和启发函数

信息素 τ_{ij} 是表征过去信息的载体，而启发函数 η_{ij} 则是表征未来信息的载体，它们直接影响蚁群算法的全局收敛性和求解效率。

2. 信息素残留因子

参数 ρ 表示信息素挥发因子，其大小直接关系到蚁群算法的全局搜索能力及其收敛速度；参数 $1-\rho$ 表示信息素残留因子，反映了蚂蚁个体之间相互影响的强弱。信息素残留因子 $1-\rho$ 的大小对蚁群算法的收敛性能影响非常大。$1-\rho$ 在 0.1～0.99 的范围内，与迭代次数 N_c 近似成正比，这是由于 $1-\rho$ 很大，路径上的残留信息占主导地位，信息正反馈作用相对较弱，搜索的随机性增强，因而蚁群算法的收敛速度很慢。若 $1-\rho$ 较小时，正反馈作用占主导地位，搜索的随机性减弱，导致收敛速度快，但易于陷于局部最优。

3. 蚂蚁数量

蚁群算法是通过多个候选解组成的群体进化过程来搜索最优解，所以蚂蚁的数量 m 对蚁群算法有一定影响。蚂蚁数量 m 较大，相对处理问题的规模而言，会提高蚁群算法的全

局搜索能力和稳定性，但数量过大会导致大量曾被搜索过的路径上的信息量变化趋于平均，信息正反馈作用减弱，随机性增强，收敛速度减慢。反之，蚂蚁数量 m 较小，会使从来未被搜索到的解上的信息量减小到接近于 0，全局搜索的随机性减弱，虽然收敛速度加快，但会使算法的稳定性变差，出现过早停滞现象。经大量的仿真试验获得：当城市规模大致是蚂蚁数量的 1.5 倍时，蚁群算法的全局收敛性和收敛速度都比较好。

4. 信息素重要因子、启发式因子和信息素浓度

信息素重要因子 α 反映蚂蚁在运动过程中所积累的信息量在指导蚁群搜索中的相对重要程度。α 越大，蚂蚁选择以前走过路径的可能性就越大，搜索的随机性减弱；α 越小，易使蚁群算法过早陷入局部最优。

启发式因子 β 反映了启发式信息在指导蚁群搜索过程中的相对重要程度，这些启发式信息表现为寻优过程中先验性、确定性因素。β 越大，蚂蚁在局部点上选择局部最短路径的可能性越大，虽然加快了收敛速度，但减弱了随机性，易于陷入局部最优。

信息素浓度 Q 为蚂蚁循环一周时释放在所经路径上的信息素总量。Q 越大，蚂蚁在已遍历路径上信息素的累积越快，加强蚁群搜索时的正反馈性，有助于算法的快速收敛。

基于以上各种参数对算法收敛性的影响，建议按照如下三步设定蚁群算法的参数。

Step1：确定蚂蚁数量，确定原则：城市规模/蚂蚁数量≈1.5。

Step2：参数粗调，调整取值范围较大的信息素重要因子 α、启发式因子 β 以及信息素浓度 Q 等参数，以得到较理想的解。

Step3：参数微调，调整取值范围较小的信息素挥发因子 ρ。

5.4.5 信息素更新策略

由于蚂蚁间的相互联系以及对搜索的主要贡献是留下信息素，因此，信息素的更新策略是决定算法收敛速度的关键之一。主要有如下两类策略。

(1) 采用固定信息素量增减的比例来进行信息素量的更新。这类策略忽视了解的分布特征，在一定程度上虽对蚁群优化算法的特性进行了一定的改进，但它们一般只适合于处理较小规模的问题。

(2) 可变信息素量增减的比例。这类策略的核心特征是信息素的增减比例会根据蚁群的搜索状态、解的质量特征、迭代进程等动态调整，其设计初衷是弥补固定比例策略忽视解分布特征的缺陷，更灵活地平衡算法的探索性与利用性，即探索新的潜在优解区域和强化已发现的较优解路径，从而适配更复杂或更大规模的优化问题。

5.4.6 蚁群算法的优缺点

蚁群算法是继模拟退火算法、遗传算法、禁忌搜索算法、人工神经网络算法等启发式搜索算法之后的又一种应用于优化问题的启发式随机搜索算法。众多的研究结果表明，蚁群算法具有很强的发现较好解的能力，这是因为该算法不仅利用了正反馈原理，在一定程度上可以加快进化过程，而且是一种本质并行的算法，不同个体之间不断进行信息的交流和传递，从而能够相互协作，有利于发现较好解。

蚁群算法具有以下优点。

(1) 蚁群算法是一种分布式的本质并行算法。单个蚂蚁的搜索过程是彼此独立的，容

易陷入局部最优，但通过个体之间不断的信息交流和传递有利于发现较好的解。

（2）蚁群算法是一种正反馈算法，路径上的信息素浓度较高，将吸引更多的蚂蚁沿这条路径运动，这又使得其信息素浓度增加，这样就加快了算法的进化过程。

（3）蚁群算法具有较强的稳健性。只要对其模型稍加修改，便可以应用于其他问题。

（4）易于与其他算法结合。蚁群算法很容易与其他启发式算法相结合，以改善算法的性能。

虽然蚁群算法有以上优点，但该算法依然还存在以下缺点。

（1）该算法一般需要较长的搜索时间。蚁群中各个个体的运动是随机的，虽然通过信息交换能够向着最优解进化，但是当群体规模较大时，很难在较短的时间内从大量杂乱无章的路径中找出一条较好的路径。

（2）该算法容易出现停滞现象，即搜索进行到一定程度后，所有个体所发现的解完全一致，不能对解空间进一步进行搜索，不利于发现更好的解。

5.5 优化算法的应用

当自动驾驶汽车在复杂路况中精准规避障碍，当智能推荐系统为千万用户推送心仪内容，当人工智能诊疗系统在海量病例中锁定最佳治疗方案——这些令人惊叹的人工智能应用背后，都藏着优化算法的隐形操刀。从资源调度的毫秒级决策到模型训练的效率提升，从高维数据的特征筛选到动态环境的策略调整，优化算法如同人工智能的精密齿轮，在无数应用场景中驱动着智能系统从可行走向最优。正是这些隐藏在代码深处的优化逻辑，让人工智能得以在现实世界的复杂约束中实现效率、精度与适应性的完美平衡，成为解锁人工智能实用价值的核心密钥。

5.5.1 几类优化算法的具体应用

优化算法的应用本质是将实际问题转换为可量化的优化目标：传统优化方法解决规则明确的结构化问题，元启发式算法攻克复杂非结构化问题，现代智能优化算法则在动态、大数据场景中实现自适应优化。

（1）传统优化方法在规则明确的结构化问题中有着广泛应用。

传统优化方法包括梯度下降法及其变种、线性规划（单纯形法等）和牛顿法与拟牛顿法等，这些方法均依赖明确的数学模型，要求目标函数和约束条件具有清晰的解析形式，且问题参数已知，因此能在规则明确、结构清晰的场景中高效求解最优解。

例如，在制造业的生产计划制订中，当产品的生产流程固定、原材料供应数量确定、各工序的耗时和成本明确时，可通过线性规划来优化生产任务的分配，在满足产能约束的同时实现生产成本最低化；在物流运输的路径规划中，若运输点之间的距离、运输成本等参数清晰可知，整数规划能帮助确定最优的运输路线，减少运输里程和费用；此外，在金融领域的资产配置中，当各类资产的收益和风险数据明确时，利用二次规划可构建风险最小化的投资组合。

（2）元启发式算法在攻克复杂非结构化问题上表现突出。

元启发式算法包括遗传算法、粒子群优化算法、模拟退火算法、蚁群算法等，这些算法

不依赖严格的数学解析模型，具有随机性搜索和自适应优化的特性，能突破目标函数非线性、约束条件复杂、参数难以量化等限制，在缺乏明确结构化规则的场景中实现全局寻优。

以城市垃圾收运路线优化为例，该问题涉及大量的垃圾站点、多变的交通状况以及不同站点的垃圾产生量等复杂因素，难以用明确的数学模型描述，此时遗传算法能通过模拟生物进化过程，在众多可能的路线组合中找到较优方案；在化工生产的工艺参数优化中，由于反应过程存在非线性、多约束且部分参数难以量化的特点，粒子群优化算法可快速搜索到使产物产量最高、能耗最低的参数组合；另外，在无人机航迹规划中，面对复杂的地形障碍和动态的禁飞区域，模拟退火算法能有效避开局部最优解，找到安全且高效的飞行路径。

(3) 现代智能优化算法在动态、大数据场景中实现自适应优化的应用日益增多。

现代智能优化算法包括自适应优化、贝叶斯优化、强化学习优化算法等，这些算法融合了机器学习与自适应机制，具备从海量数据中学习规律、动态调整优化策略的能力，能有效应对高维度、实时变化的复杂场景，实现从数据驱动到智能决策的闭环优化。

在智能电网的负荷调度中，用户用电需求会随时间动态变化，且需处理海量的实时用电数据，基于强化学习的优化算法能不断学习用电模式，实时调整分布式发电机组出力，确保电网稳定运行；在电商平台的库存管理中，面对海量的用户订单数据、商品销售波动以及供应链延迟等动态信息，结合神经网络的智能优化算法可自适应地调整补货策略，平衡库存成本和缺货风险；此外，在智慧交通的信号控制中，基于大数据的自适应优化算法能根据实时车流量数据，动态调整信号灯时长，缓解交通拥堵。

在实际应用中，常常可以通过多种算法融合提升效果，以适应更复杂的现实需求。

5.5.2 梯度下降算法的具体应用

梯度下降算法作为人工智能领域核心的优化方法，其核心逻辑是通过迭代式调整参数、沿目标函数梯度的反方向逐步逼近最小值，因此在各类模型训练、数据处理任务中扮演着关键角色，拥有丰富且重要的实际应用场景。具体来看，其典型应用可分为以下几类。

(1) 在线性回归任务中，梯度下降算法的核心作用是寻找最优模型系数，即通过不断微调系数数值，最小化预测值与真实数据间的平方误差和，最终让线性拟合曲线精准贴合数据的真实分布规律，从而提升模型对连续变量的预测准确性。

(2) 对于逻辑回归(常用于二分类或多分类任务)，梯度下降算法则聚焦于优化模型参数以最小化交叉熵损失函数，该函数专门量化模型输出的概率分布与数据真实标签间的偏差，让模型对类别归属的判断更可靠。

(3) 在深度神经网络的训练过程中，梯度下降算法更是不可或缺的核心工具。它结合反向传播算法，计算损失函数对各层权重与偏差的梯度，再沿梯度反方向迭代调整这些参数，持续最小化损失函数，如分类任务的交叉熵、回归任务的均方误差等，逐步提升网络的拟合或分类能力。

(4) 在支持向量机算法的训练中，梯度下降算法用于优化参数以寻找最佳超平面，这一超平面需以最大边距分隔不同类别的数据点，进而增强模型对未知新数据的泛化能力，减小分类误差。

(5) 面对高维数据降维需求时，梯度下降算法也能发挥作用，例如在主成分分析中，它通过迭代优化筛选出能捕获数据最大方差的特征向量作为主成分，在保留数据核心信息的

同时实现维度压缩，降低后续任务的计算成本。

（6）在 K-Means 等无监督聚类算法中，梯度下降算法用于优化聚类质心的位置，即通过最小化所有数据点与其所属聚类中心间的平方距离和，不断调整中心坐标，最终实现同一聚类内数据高度聚集、不同聚类间差异显著的效果，清晰挖掘数据的内在分组结构。

5.5.3 遗传算法的具体应用

遗传算法由于使用多个个体用于状态空间的搜索，因此可以以并行方式实现，不容易陷入局部极值。其难点主要在于选择编码方式和适应度函数上。由于遗传算法不依赖于问题的具体领域，因此在很多学科有广泛的应用，具体介绍如下。

（1）函数优化。作为遗传算法最早涉足的研究领域之一，函数优化至今仍是检验其性能的核心场景。无论是高维函数、多峰函数，还是具有强非线性、不连续特性的复杂函数，遗传算法都能通过群体搜索和迭代进化，高效逼近最优解，为各类数值优化问题提供可靠的求解路径。

（2）复杂问题优化。面对传统优化方法难以应对的非线性、多目标、不确定性问题，遗传算法展现出显著优势，成为解决这类复杂问题的关键技术。例如，在流体动力学中的液体流动优化、军备竞赛模型的动态平衡分析、气候变化模拟的参数寻优等非线性动态系统问题中，遗传算法已实现有效突破；同时，在旅行商问题、背包问题、装箱问题、二次分配问题等组合优化领域，其稳健性和求解效率也得到了充分验证。

（3）复杂系统分析。对于具有动态性、非线性相互作用的复杂系统，遗传算法在分析与归纳层面发挥着重要作用。它能通过群体进化机制挖掘系统隐含规律，广泛应用于聚类分析（如提升数据分类精度）、图像处理（如特征提取与降噪）、模式识别（如复杂模式的匹配与识别）、调度组织（如资源分配与流程优化）等具体场景，为系统建模与优化提供有力支持。

（4）人工智能领域。遗传算法与专家系统、人工神经网络并称为处理人工智能问题的三大核心工具，在多个前沿方向发挥关键作用。例如，在博弈对策中优化决策策略，在机器人控制中实现路径规划与动作协调，在自动程序设计中生成高效代码。尤其在神经网络领域，它既能用于调节网络权值以提升模型精度，也能优化网络拓扑结构以增强泛化能力，显著拓展了人工智能技术的应用边界。

（5）跨领域综合应用。随着学科交叉融合的深入，遗传算法常与其他技术结合形成协同解决方案，充分发挥各自的优势。例如，与模糊逻辑结合处理不确定性决策问题，与深度学习融合提升复杂数据的建模能力，与物联网技术结合优化资源调度等，这种综合应用模式使其在解决实际复杂问题时更具灵活性和有效性。

5.5.4 蚁群算法的具体应用

旅行商问题已成为研究蚁群算法的基础问题。将蚁群算法用于求解连续优化问题也是近年来的一个研究重点，已成功用于图像处理、冗余分配问题、图着色问题、约束满足条件问题、系统辨识、数据挖掘和化学工业等方面。蚁群算法的经典应用场景介绍如下。

（1）经典组合优化问题。这是蚁群算法最早且最成熟的应用领域，其核心是解决寻找最优排列/路径类问题。除了旅行商问题，还包括图着色问题、背包问题、作业车间调度等，能有效探索庞大的解空间，在可接受的时间内找到高质量的近似最优解。

(2) 物流与交通领域。在物流配送和交通管理中,蚁群算法是优化资源配置的关键工具。例如,在物流配送中,它可根据订单分布、车辆载重、道路拥堵情况,规划出多车辆、多配送点的最优路径,减少空驶里程和油耗,提升配送效率;在城市交通管理中,能动态调整交通信号配时,或为网约车、货运车辆规划避开拥堵的路线,缓解交通压力。

(3) 网络路由优化。在计算机网络、通信网络中,蚁群算法可实现动态智能路由。它能模拟蚂蚁沿信息素浓度高的路径移动的特性,实时感知网络节点的负载、链路延迟或故障情况,自动选择传输效率最高的路由路径——例如在数据传输时,避开拥堵的节点或受损的链路,从而降低数据丢包率、减少传输延迟,提升整个网络的稳定性和吞吐量。

(4) 生产调度领域。在制造业中,蚁群算法可优化生产全流程的调度方案。例如,在车间作业调度中,它能根据设备产能、工序优先级、物料供应节奏,合理安排不同产品的生产顺序和设备分配,避免设备闲置或工序等待;在订单交付周期优化中,可通过调整生产计划,缩短产品从原料到成品的总耗时,同时减少生产成本。

(5) 图像处理领域。蚁群算法在图像处理的核心任务中也发挥着重要作用。例如,在图像分割中,它可将图像像素视为蚂蚁,通过像素的灰度值、纹理特征等信息素,将目标区域与背景区域精准分离;在边缘检测中,能沿着图像中灰度值突变的边缘路径聚集蚂蚁,清晰提取出目标的轮廓,为后续的图像识别、分析提供基础。

(6) 图着色问题。作为组合优化领域的经典问题,图着色问题的核心需求是给图中的顶点分配颜色,确保相邻顶点颜色不同,且使用的颜色总数最少。蚁群算法能高效解决这类问题:它将每个顶点的颜色选择视为蚂蚁的一步决策,通过信息素浓度记录某顶点选某颜色的合理性,最终通过群体迭代找到颜色数量最少、无相邻同色的最优分配方案。这类应用在实际场景中十分常见,如地图着色、电路设计、时间表排课等。

蚁群算法的研究前景极其光明,在算法的理论研究、算法的改进和算法的应用上还有很大的研究空间。尤其是近几年来,国内的研究学者越来越重视蚁群算法的研究,在蚁群算法理论逐步完善的同时,蚁群算法的应用将越来越广泛。

5.6 小结

(1) 最优化问题是指在给定约束或无约束条件下,寻找决策变量的取值,使目标函数达到最大或最小的问题。

(2) 传统优化方法通常依赖于目标函数的连续性、可微性等解析性质,通过严格的数学推导寻找最优解,主要包括线性规划的单纯形法和非线性规划中的基于梯度的各类迭代算法。

(3) 智能优化方法适用于解决复杂优化问题。它不依赖问题的严格数学模型(如连续性、可微性),而是通过模拟进化、协作、退火等过程,在解空间中高效搜索最优解,尤其擅长处理高维度、非线性、多约束、非结构化的复杂问题。

(4) 梯度下降算法是一种求解无约束优化问题的迭代算法,它通过不断地沿着该函数梯度的反方向更新参数,从而找到一个函数的局部最小值。它在凸函数中可稳定收敛至全局最小值,在非凸函数中易陷入局部最优,其优化效果与函数性质及初始参数密切相关。

(5) 遗传算法作为一种模拟生物进化过程的智能优化方法,具有显著的全局寻优能力:

通过种群迭代中的选择、交叉、变异等操作，能有效跳出局部最优解，适合处理非线性、多峰、高维度及带复杂约束的优化问题，且对问题的数学特性如连续性、可微性要求低，适应性强。但它也存在一定局限，如收敛速度可能较慢尤其在接近最优解阶段，性能受种群规模、交叉率、变异率等参数影响较大，易出现早熟收敛，即过早陷入局部最优等情况，需要结合问题特性进行参数调优以平衡寻优效率与精度。

（6）蚁群算法是一种模拟蚂蚁群体通过信息素协作寻路行为的智能优化方法，其性能特点表现为：具有较强的全局寻优能力和稳健性，通过信息素的正反馈机制能逐步强化优质解，且对初始解依赖较低，适合处理路径规划、调度等组合优化问题，易于与其他算法融合；但同时也存在收敛速度较慢、易陷入局部最优的局限，其性能对信息素挥发系数、启发因子等参数设置较为敏感，在高维度复杂问题中优化效率可能下降。

习题 5

（1）请阐述什么是最优化和最优化问题。
（2）请阐述有哪些常见的智能优化方法。
（3）请阐述梯度的定义及其意义。
（4）请阐述遗传算法常见的编码方法。
（5）请阐述遗传算法常用的交叉方式。
（6）请阐述蚁群算法的基本原理。
（7）请阐述蚁群算法的优缺点。
（8）在遗传算法中，变量 x 取 0～63 的整数值，如何对 x 进行二进制编码？

习题答案