高等院校教材

计 算 智 能

卢才武 唐晓灵 张志霞 顾清华 编著

陕西科学技术出版社

内 容 简 介

计算智能是以模型为基础，以分布并行计算为特征模拟人的智能求解问题的理论与方法。本书系统讲述计算智能的基本内容、基本理论与基本方法。本书包括一个“概述”以及“模拟退火算法”、“人工神经网络”、“进化计算”、“群智能算法”、“人工免疫算法”和“计算智能的未来发展”等七个部分，每章之后均附有应用案例和思考题以拓宽视野、培养自主学习的习惯。

本书叙述深入浅出、简明扼要，先从简单的示例介绍每种算法的原理，然后深入讨论它们的基本理论及应用技术。书中图文并茂，便于自学，可作为研究生教材或参考书，也可供其他有兴趣的读者参考。

图书在版编目（CIP）数据

计算智能/卢才武等编著.－西安：陕西科学技术出版社，2008.10

ISBN 978-7-5369-4161-8

Ⅰ.计… Ⅱ.卢… Ⅲ.人工智能－神经网络－计算 Ⅳ.TP183

中国版本图书馆 CIP 数据核字（2008）第 154386 号

出版者　陕西科学技术出版社
西安北大街 131 号　邮编 710003
电话（029）87211894　传真（029）87218236
http://www.snstp.com
发行者　陕西科学技术出版社
电话（029）87212206　87260001
印　刷　西安建筑科技大学印刷厂
规　格　787mm×1092mm　16 开本
印　张　10.75
字　数　250 千字
印　数　1000 册
版　次　2008 年 10 月第 1 版
2008 年 10 月第 1 次印刷
定　价　25.00 元

目 录

第 1 章 计算智能概论

生命在长期进化过程中，积累了很多新奇的功能，人类很早就从中得到启发而改进自己的工具，如史书中记载“见蓬转而做车辑”，传说鲁班被茅苇划破，而发明锯子……也许早先的发明，只是偶然的模仿和发现，后来人们已有意识地进行这方面的研究，这就是“仿生学”。

仿生学顾名思义就是模仿生物的某些功能的学问。有名的例子很多，如模仿海豚皮而构造的“海豚皮游泳衣”；科学家研究鲸鱼的皮肤时，发现其上有沟槽的结构，于是有个科学家就依照鲸鱼皮构造，造成一个薄膜蒙在飞机的表面。又如有科学家研究蜘蛛，发现蜘蛛的腿上没有肌肉，有脚的动物会走，主要是靠肌肉的收缩，现在蜘蛛没有肌肉为什么会走路？经研究蜘蛛不是靠肌肉的收缩进行走路的，而是靠其中的“液压”的结构进行走路，据此人们发明了液压步行机……总之，从自然界得到启迪，模仿其结构进行发明创造，这就是仿生学。这是我们向自然界学习的一个方面。另一方面，我们还可以从自然的规律中得到启迪，利用其原理进行设计(包括设计算法)，这就是计算智能（computational intelligence）的思想。

我们将着重讲述一些数学建模中常用的算法，包括神经网络算法、遗传算法和模拟退火算法等。用这些算法可以较容易地解决一些很复杂的，常规算法很难解决的问题。由于这些算法都有着很深的理论背景，因此，本书中不可能也没有必要详细地讨论这些算法的理论，我们的目标在于应用，大家只需大概了解这些算法的原理，知道能用这些算法解决一类什么样的问题，并能应用这些算法解决数学建模中的一些问题即可。

1.1 智能理论及其相关术语

1.1.1 智能理论

“智能（intelligence）”一词可以用作名词，也可以用作形容词。如果用作名词，它是指人类所能进行的脑力劳动，包括感觉、认知、记忆、学习、联想、计算、推理、判断、决策、抽象、概括……如果用作形容词，它的意义是：人一样的、聪明的、灵活的、柔性的、自学习的、自组织的、自适应的、自治的……

智能理论的研究也分为两个方面，一方面是研究智能的产生、形成和工作机制；另一方面是研究如何用人工的方法模拟、延伸和扩展智能。前者称为自然智能理论，主要是生理学和心理学研究者所从事的工作；而后者称为人工智能（artificial intelligence）理论，主要是理工学研究者所从事的工作。在前者的领域中，“智能”多取名词的用法，因为研究的是“脑力劳动”本身的机制；在后者的领域中，“智能”多取形容词的用法，因为人们主要考察人工智能的功能与自然智能的功能相比，像不像、高不高、强不强？

按道理讲，人工智能理论应以自然智能理论为基础。如果搞清了各种自然智能的工

作机制及其各个功能部件的结构关系，那么就可以通过已经高度发达的电子的、光学的和生物的器件构筑类似的结构对其进行模拟、延伸和扩展，从而实现人工智能。但遗憾的是，由于人类的头脑结构高度复杂，也由于实验这一现代科学的锐利武器在研究人脑机制和结构时不能随意使用，直到今天，自然智能理论并没有搞清一些基本智能活动的机制和结构，总体进展十分有限。因而人工智能理论的主流已经从结构模拟的道路走向了功能实现的道路。所谓功能实现就是将自然智能的结构看作黑箱，而只控制黑箱的输入输出关系，只要从输入输出关系上看实现了所要模拟的功能即可。功能实现的道路使人工智能理论摆脱了自然智能理论进展缓慢的束缚，通过几十年的发展，已经形成了较为系统的理论体系，包含了极为丰富的内容，并在实际中得到了广泛的应用，发挥了显著的作用。

1.1.2 智能科学的有关术语

1）智能与智能信息处理

从认识论层面上看，信息是认识主体所感受的事物状态及其变化的方式，是用来消除观察者认识上的不确定性的度量。知识则是人类实践经验的总结和提炼，具有抽象性和普遍性，属于认识论范畴的概念。知识是信息加工的产物，是抽象化和广义化的信息。在现有人工智能的文献中知识与信息是同义词，知识的概念完全等同于信息的概念。

从神经生理学、神经解剖学的角度来看，人类智能的核心是思维（thought），思维的器官是大脑。大脑活动的主要内容是处理信息。人工智能则是对人类智能的一种模拟和扩展，其核心是思维模拟。

我们通常把思维定义为人的大脑活动，主要是指处理和再生信息的能力。有时，人们也把思维能力看作是一种狭义的智能。从这种意义上讲，如果没有思维能力，则被认为非智能。智能是指在给定任务或目的下，能根据环境条件制定正确的策略和决策，并能有效地实现其目的的过程或能力。也有人把智能定义为有效地获取、传递、处理、再生和利用信息，使其在任意环境下成功地达到预定目标的能力。可见，对于同样的环境和目标下，具有更强的“获取、处理、再生和利用”信息的能力，就更容易或有效地实现目标，从而表现出具有更高的智能水平。

我们通常所说的智能信息处理是一种利用各种智能手段进行信息变换的过程，这里的各种智能手段包括人工智能、机器智能和计算智能等。

2）计算智能

计算智能，也有人称之为“智能算法”、“智能计算”或“软计算”，虽然至今没有一个统一的定义，但我们可以这样来概括它。智能计算就是借用现代计算工具和自然界（生物界）规律的启迪，根据其原理，模仿设计求解问题（或处理信息）的理论与方法。它是人工智能的深化与发展。如果说人工智能是以知识库（专家规则库）为基础，以顺序离散符号推理为特征的，计算智能则是以模型（计算模型、数学模型）为基础，以分布、并行计算为特征。前者强调规则的作用与形成，而后者强调模型的建立与构成；前者依赖专家个人知识，而后者强调自组织、自学习与自适应。

目前这方面的内容很多，如：人工神经网络技术、遗传算法、进化规划、模拟退火

技术和群集智能技术等。

1.2 传统人工智能

人工智能是研究利用计算机来模拟人的某些思维过程和智能行为（如学习、推理、思考、规划等）的学科，主要包括计算机实现智能的原理、制造类似于人脑智能的计算机，使计算机能实现更高层次的应用。人工智能涉及计算机科学、心理学、哲学和语言学等学科，可以说几乎涵盖了自然科学和社会科学的所有学科，其范围已远远超出了计算机科学的范畴，人工智能与思维科学的关系是实践和理论的关系，人工智能是处于思维科学的技术应用层次，是它的一个应用分支。从思维观点看，人工智能不仅限于逻辑思维，要考虑形象思维、灵感思维才能促进人工智能的突破性的发展。数学常被认为是多种学科的基础科学，人工智能学科也必须借用数学工具，数学不仅在标准逻辑、模糊数学等范围发挥作用，也进入语言、思维领域。数学进入人工智能学科，它们将互相促进而更快地发展。

传统人工智能的研究开始于 1956 年，是由“人工智能之父”McCarthy 及一批数学家、信息学家、心理学家、神经生理学家、计算机科学家在 Dartmouth 大学召开的会议上首次提出。传统人工智能是符号主义，它以 Newell 和 Simon 提出的物理符号系统假设为基础，主要目标是应用符号逻辑的方法模拟人的问题求解、推理、学习等方面的能力。问题求解是传统人工智能的核心问题，当机器有了对某些问题的求解能力以后，在应用场合遇到这类问题时，便会自动找出正确的解决策略。这种问题求解能力是基于规则的，是能够举一反三的。有了问题求解能力的机器就能比普通机器更灵巧地分析问题和处理问题，从而适用于更加复杂多变的应用场合。

推理是人的思维的一个重要方面，推理的三种主要形式是归纳推理、演绎推理和模糊推理。传统人工智能中推理的研究就是要模拟这三种推理形式，实现诸如故障诊断、数学定理证明、模糊问题判断等功能。

在传统人工智能中，“学习”一词有多种含义。在专家系统等应用中，它指的是知识的自动积累；在问题求解中，它指的是根据执行情况修改计划；在数学推理系统中，它指的是根据一些简单的数学概念和公理形成较复杂的概念，作出数学猜想，等等。

人工智能在其发展过程中逐渐形成了三个学派：

（1）符号主义学派（symbolicism）—传统人工智能（认知和逻辑学派），传统人工智能的研究始于 20 世纪 50 年代，以知识为基础，通过推理来进行问题求解。它的研究方法为功能模拟方法（计算机模拟人类认知系统功能）。代表人物有 Simon，Minsky 和 Newell，McCarthy，Nillsson 等。

（2）联接主义学派（connectionism）—仿生学派—计算智能，始于 1943 年的 M-P 模型（McCulloch，Pitts）。到了 1982 年，Hopfiled 又提出了用硬件模拟神经网络。1986 年，鲁梅尔哈特（Rumelhart）等人提出多层网络中的反向传播算法(BP)算法并得到广泛使用。计算智能研究方法为结构－功能模拟方法，代表人物有 McCulloch，Pitts，Hopfield，Rumelhart 等。

（3）行为主义学派（actionism）—进化主义或控制论学派，始于 20 世纪 60～70 年

代，早期的研究工作重点是模拟人在控制过程中的智能行为和作用，如对自寻优、自适应、自镇定、自组织和自学习等控制论系统的研究。到 20 世纪 60～70 年代，上述这些控制论系统的研究取得一定进展，播下智能控制和智能机器人的种子，并在 20 世纪 80 年代诞生了智能控制和智能机器人系统，其智能行为是基于“感知－动作”模式的控制系统。它的研究方法为行为模拟方法，代表人物为 R.A. Brooks。

1.3 计算智能

20 世纪 90 年代以来，在智能信息处理研究的纵深发展过程中，人们特别关注到精确处理与非精确处理的双重性，强调符号物理机制与联接机制的综合，倾向于冲破“物理学式”框架的“进化论”新路，一门称为计算智能的新学科分支被概括地提出来了，并以更加明确的目标蓬勃发展。1994 年 IEEE 为了促进多学科渗透和结合，把模糊系统（fuzzy system）、神经网络（neural network）和进化计算（evolutionary computation）三个年会合并举行，于 1994 年 6 月 25 日至 7 月 3 日在美国佛罗里达州的奥兰多召开全球第一届计算智能大会（WCCI），出版了《计算智能、模仿生命》的论文集。大会决定计算智能会议每三年召开一次。此次会议是计算智能的第一次综合性大会，共收集了来自世界各国学者的约 1600 篇论文，大会的主题是计算智能。人们会提出这样的问题：人工智能和计算智能有什么不同，又有什么关系呢？

首次给出计算智能定义的是美国学者 James C. Bezdek。1992 年，他在近似推理的国际杂志上发表文章指出：计算智能依靠生产者提供的数字材料，而不是依赖于知识，而人工智能使用的是知识精华。Bezdek 还说：人工神经网络应称为计算神经网络，即“人工”两字应改为“计算”。在人工智能 AI 和计算智能 CI 的关系上，Bezdek 认为 CI 是 AI 的子集，即 $CI \in AI$。而这次大会主席 Jacek M. Zurada 却认为 $CI \notin AI$，两者只有部分重合。James C.Bezdek 在题为《什么是计算智能》的报告中讲到：智能有三个层次，第一层是生物智能（biological intelligence，简称 BI），它是由人脑的物理化学过程反映出来的，人脑是有机物，它是智能的物质基础；第二层是人工智能（artificial intelligence，简称 AI），它是非生物的，是人造的，常用符号表示，AI 的来源是人的知识精华和传感器数据。第三层是计算智能（computational intelligence，简称 CI），它是由数学方法和计算机实现的，CI 的来源是数值计算和传感器。以上三者第一个英文字符取出来称之为 ABC。显然，从复杂性看有三个层次，即 B（有机）、A（符号）、C（数值），而且 BI 包含了 AI，AI 又包含了 CI。

按 Bezdek 的看法，AI 是 BI 的中间过渡，因为 AI 中除了计算算法外，还包含符号表示和数值信息处理。模糊集和模糊逻辑是 CI→AI 的平滑过渡，因为它包含了数值信息和语义信息。他还认为：计算神经网络 CNN 是一个最底层最基本的环节，也是 CI 的一个重要基石，主要用于模式识别。CNN 由以下四个点决定：功能、结构（连接拓扑和更新策略）、形式（集成和传递的节点函数式）、数据（用于训练和测试的数据）。按以上几点，CNN 有多种形式，如前馈、自组织以及与 fuzzy 结合的模糊神经网络等。

目前国内外提出的计算智能一般是以人工神经网络为主导，与模糊逻辑系统、进化计算以及信息学科的综合集成。图 1.1 是计算智能与其他学科之间的关系图。

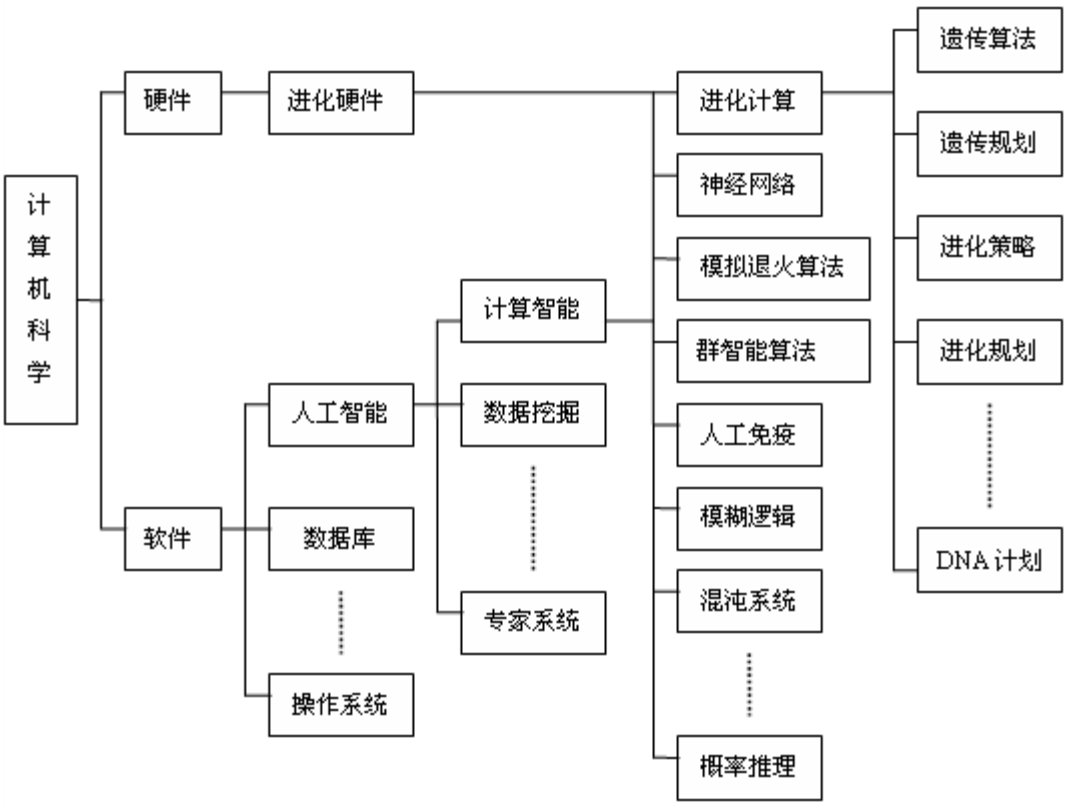


图 1.1　计算智能与其他学科之间的关系

1.4 计算智能的主要内容

计算智能，广义地讲就是借鉴仿生学思想，基于生物体系的生物进化、细胞免疫、神经细胞网络等机制，用数学语言抽象描述的计算方法。是基于数值计算和结构演化的智能，是智能理论发展的高级阶段。计算智能有着传统的人工智能无法比拟的优越性，它的最大特点就是不需要建立问题本身的精确模型，非常适合于解决那些因为难以建立有效的形式化模型而用传统的人工智能技术难以有效解决，甚至无法解决的问题。

计算智能研究的主要问题包括：

- 学习：学习是一个有特定目的的知识获取过程，并通过这一过程逐渐形成、修改新的知识结构或改善行为性能。获取知识的过程包括积累经验、发现规律、改进性能和适应环境。机器学习则是利用机器来完成学习这一过程，从而达到或部分达到学习的目的。
- 搜索：是对问题的一种求解方法、技术和过程。搜索是面向问题的，不同的问题有不同的搜索方法、技术和过程。
- 推理：是人类基于逻辑的一种思维形式，也是计算机基于知识表示的一种知识利用。即根据一定的规则，从已知的断言或知识得出另一个新的断言或知识的过程。

计算智能研究的主要方法包括：

- 模型：具有生物背景知识并描述某一智能行为的数学模型。
- 算法：以计算理论、技术和工具研究对象模型的核心，它具有数值构造性、迭代性、收敛性、稳定性和实效性。
- 实验：对许多复杂问题，难以进行理论分析，数值实验和实验模拟成为越来越重要的研究手段，并获得了很大的成功（分叉、混沌、孤波等）。

从方法论的角度和现在的研究现状来看，计算智能的主要方法有：模拟退火算法、人工神经网络、群智能算法、模糊系统、进化计算、免疫算法、DNA 计算以及交叉融合的模糊神经网络、进化神经网络、模糊进化计算、进化模糊系统、神经模糊系统、进化模糊神经网络和模糊进化神经网络等。

1）模拟退火算法（simulated annealing，简称 SA）

模拟退火算法是一种全局优化方法。早在 1965 年，KhaS 就提出了这一想法，不过并未受到计算机科学与优化应用领域的足够重视。直到 1983 年，Kirk Patlick 提出模拟退火算法，才引起了优化应用领域的重视，成为研究的热点。

SA 算法的特点主要有以下几个方面：

（1）以一定的概率接受恶化解，SA 算法在迭代过程中不仅接受使目标函数变“好”的试探点，而且还能以一定的概率接受使目标函数值变“差”的试探点。迭代中状态是随机产生的，并且不强求后一个状态一定优于前一个状态，即以一定的可能容忍退化状态出现。

（2）引进算法控制参数 T，它将优化过程分成若干个阶段，并决定每个阶段随机状态的取舍标准，接受概率随温度的下降而逐渐减小。

（3）使用对象函数值（即适应度）进行搜索，SA 算法仅使用由目标函数变换来的适应度函数值，就可确定进一步搜索方向和搜索范围，无需其他辅助信息。

2）人工神经网络（artificial neural networks，简称 ANN）

人工神经网络是模仿和延伸人脑智能、思维、意识等功能的非线性自适应动力学系统，神经网络所具有的学习算法能使其对事物和环境具有很强的自学习、自适应和自组织能力，它的知识积累是自动的，无瓶颈效应存在。因此，神经网络信息处理系统是一种全新计算结构的新型智能信息处理系统。它可以模仿人脑处理不完整的、不准确的，甚至处理非常模糊的信息，并能联想记忆，从部分信息中获得全部信息。

神经网络（ANN）通过众多神经元联结，从结构和实现机理方面逼近生物智能，并通过学习、识别和控制等功能模拟弥补对生物智能认识的局限性。其并行和分布式的结构和处理问题的方法，使其在许多实际应用领域取得了显著的成效，解决了一些传统方法无法求解或解决效果较差的问题。

3）进化计算（evolutionary algorithm，简称 EA）

进化计算是遗传算法、遗传规划、进化策略、进化规划的通称，它们都是模拟生物在自然环境中遗传和进化的原理而形成的。

遗传算法是一个群体优化过程，为了得到目标函数的最小（大）值，我们不是从一个初始值出发，而是从一组初始值出发进行优化。这一组初始值好比一个生物群体，优化的过程就是这个群体繁衍、竞争和遗传、变异的过程。

遗传算法主要步骤为：

（1）设置初值。

（2）竞争。选择初始群体中的若干个个体来产生下一代。例如可以根据目标函数值的大小决定个体被选中的概率，并按这个概率选择初始群体中的个体，以体现优生原则。

（3）繁衍。它包括演化、杂交和变异。

（4）以子代代替其父代，反复进行步骤（2）与（3），不断产生后代直至目标函数在整个群体中的最小（大）值不能再继续优化。

事实上，上述优化方法并不能保证达到全局优化，仍然可能陷入局部极值的陷阱，但遗传算法通过引入多个自变量，使得陷入局部极值陷阱的机会相对变小，优化效果会有所改进。此外，人们又提出了在父代和子代间也引入竞争，即不是简单地用子代代替父代，而是选取它们中较优者作为下一步的父代。在遗传算法中还可以引入变异，即在优化过程中以小概率不按局部优化的方向进行。从局部看，这种演化似乎反而“劣化”了目标函数，但正是这种演化才为逃出局部极值的陷阱提供了可能性。由于控制“劣化”的变异只以很小的概率发生，在计算了很多步后，总趋势是向优化发展的。

遗传算法是对每个个体进行评价，按照概率选择高适应度的个体，进行交叉和变异从而产生新的个体；进化策略基于变异与选择的原理，对于每一个父代，通过变异产生一个子代，二者通过竞争获得生存，通过选择以消除低劣解从而进化；进化规划近似进化策略，不同之处在于个体选择采用概率分布选择机制，且每一子代变异次数也满足概率分布，从而产生下一子代。三者的差异通过随机变换类型和选择机制来刻画，研究较多的是遗传算法。它们都是一种全局优化技术，可以解决现实生活中各种优化问题，应用领域主要是 CIMS、CIPS 中的生产调度、规划，机器人、网络通信的路径规划，多参数优化，企业资源规划等；且能够与模糊逻辑、神经网络结合，解决它们中所有参数的快速学习问题。但目前进化算法研究较多是解决“早熟”和提高效率问题，存在的问题是数学理论基础，应用中主要是控制参数的选择和自适应问题，要真正形成基于进化算法的上述领域的真正实用化产品还有许多工作要做。

4）群智能算法（swarm intelligence，简称 SI）

群体智能是在自然界生物群体行为的启发下提出的一种人工智能实现模式，也是计算智能领域的关键技术之一。群体智能的概念源于对蜜蜂、蚂蚁等群居生物群体行为的观察和研究。群体智能是指“简单智能的主体通过合作表现出复杂智能行为的特性”。该智能模式需要以相当数目的智能个体来实现对某类问题的求解功能。作为智能个体本身，在没有得到智能群体的总体信息反馈时，它在解空间中的行进方式完全是没有规律的。只有受到整个智能群体在解空间中行进效果的影响之后，智能个体在解空间中才能体现出具有合理寻优特征的行进模式。

群体智能研究主要是对生物群体协作产生出来的复杂行为进行模拟，并在此基础上，探讨解决和解释一些复杂系统、复杂行为的新思路和新算法。群体智能计算是群体智能研究中的一个分支。

群体智能计算是在群体智能领域中计算智能研究的逐步深入而产生的一种新兴的计算智能模式，它是群体智能研究中的一个重要分支，在对某些群体行为的观察和研究的基础上，运用一定的数学工具和计算机工具，提出相应的群体智能算法，并用来解决那些因为难以建立有效的形式化模型而用传统优化方法又难以有效解决甚至无法解决的问题。

目前，在群体智能计算领域中，蚁群算法和微粒群优化算法作为群体智能计算的两种典型实现模式，受到了普遍关注，两种算法模式都是基于种群寻优的启发式随机搜索算法。在这两种模式的群体智能中，蚁群算法更擅长于离散空间中的优化问题求解，而微粒群算法则更多地用于连续空间内的优化问题求解。

5）免疫算法（immune algorithm，简称 IA）

免疫算法是抽取和反映生物机体免疫系统的特点，结合工程应用而描述的一个计算模型。这里，抗原对应于待求解的问题，而抗体则对应于问题的一个解。

抗原识别模块及初始抗体的产生针对待求解的特征，判别系统是否曾求解过此类问题，若有则从记忆细胞库中搜寻该类问题的记忆抗体，否则随机产生初始抗体群。抗体产生在免疫反应过程中，由于抗原的刺激、抗体产生细胞（B 细胞）不断分化、增殖，形成大量的抗体。这些抗体的来源主要有：①免疫细胞的新陈代谢作用。即抗体维持着一定数量的自然死亡和不断对从骨髓中新产生的 B 细胞中得到补充；②基于交叉与变异的新抗体的产生。在免疫调节机制下，抗体群体不断更新进化，最终排除抗原，即搜索到问题的解。抗体间产生的刺激与抑制采用一种基于浓度的抗体产生调节机制。抗体的浓度 C_1 是指群体中相似抗体所占的比率，即：C_1 = 具有相近适应度的抗体数/抗体总数 N。在群体更新中，适应度高的抗体的浓度不断提高，而浓度达到一定值时，则抑制这种抗体的产生，反之则相应提高浓度低的抗体的产生和选择概率。这种机制保证了抗体群体更新中的抗体多样性，一定程度上避免了未成熟收敛。

1.6 计算智能的应用领域

计算智能理论技术的应用主要可以分为以下几个方面：智能建模、智能控制、智能优化、智能管理、智能仿真、智能设计和制造等。

1）智能建模

智能建模是应用计算智能的理论和方法，针对系统实测输入输出数据、专家知识和操作经验来建模。智能建模可分为智能分类、智能辨识、智能测量、智能预测、智能决策、智能评价、智能诊断、智能信息处理等。

2）智能控制

智能控制包括模糊控制、神经网络控制和混合智能控制。混合智能控制包括：模糊神经网络控制、进化神经网络控制、神经模糊控制、进化模糊控制等。智能控制要求具有一定程度的自适应、自学习、自组织的智能行为，以实现适应环境变化、减少波动、保证高的控制精度。智能控制的核心是高效的控制算法，保证控制的实时性和快速性。目前各类模糊控制技术和应用研究较多，其中，神经控制在前馈控制中应用较好，反馈控制实用化还需深入研究。

3）智能优化

智能优化技术是运用人工智能、思维科学、启发推理、联想识别、学习训练、模糊逻辑、进化算法等技术与运筹学、控制理论、大系统理论中静态优化、动态优化、多级优化等方法相结合，寻求解决现有优化方法存在的人的因素、多目标、局部解、不确定、未确知等问题的新途径。具体有①启发式线性、非线性规划：将专家工作经验、模糊逻

辑思维启发信息引入到线性、非线性问题求解的推理和搜索过程，提高求解的速度和效率；②学习式动态规划：在动态规划中引入机器学习、神经网络自学习机制，解决多步决策、多阶段动态过程优化的动态适应问题；③进化式非线性规划：应用遗传算法、进化策略等进化算法进行非线性问题的优化；④联想式多目标优化：利用 Hopfield 神经网络建立智能优化模型，实现最优选择；⑤模糊多级优化：应用模糊逻辑实现模糊目标分解、模糊约束分解和模糊协调算法，解决大系统的模糊全局优化等。

4）智能管理

应用管理学科、信息技术、运筹学和人工智能等新技术进行科学管理，提高管理系统的“三化”——智能化、集成化、协调化。智能管理是一门综合管理学科，包括智能分析、智能预测、智能规划、智能优化、智能决策、智能指挥、智能信息处理、智能组织、智能评审、智能协调，它们都是在原有方法基础上引入专家系统、模糊逻辑、协同论、多媒体人机接口、定性与定量集成等智能技术，使得管理更有效、更全面、更科学。目前这方面研究深度还不够，应用更少。

5）智能仿真

智能仿真技术是现代计算机管理领域中的重要方法和手段。主要用于①系统仿真：分析系统的动态和稳态特性，系统的定性、定量评价和估算；②方案仿真：对系统待选的决策方案、规划方案、设计方案模拟其实现过程，分析其效果；③预测仿真：对系统未来的发展进行动态分析，预估其发展前景。智能仿真是智能技术（如专家系统、知识工程、模式识别、神经网络等）渗透到仿真技术（如仿真模型、仿真算法、仿真语言、仿真软件等）中，建立智能控制方案，以及发展规划方案的智能仿真平台。目前前者研究较多，后者研究较少。

6）智能设计和制造

在设计系统和产品制造过程中，利用神经网络、模糊系统建立模型，实现虚拟设计和制造；建立智能设计和制造工具包，提高设计和制造的效率，保证产品的性能，降低开发成本。

计算智能作为人工智能的新篇章，是当代高新技术之一，是实现各行各业系统、设备自动化、智能化的核心理论和技术。然而计算智能理论需要不断发展和完善，需要研究更实用的计算智能应用技术，体现真正的智能化，让我们面对这些新的挑战，开创计算智能理论和技术的新篇章。

思考题

1.1 什么是智能？人工智能分为哪几个学派？
1.2 传统人工智能的核心问题是什么？
1.3 简述人工智能与计算智能的区别。
1.4 计算智能的主要研究内容是什么？
1.5 计算智能有哪些应用领域？

参考文献

[1] Andries P. Engelbrecht, Computational Intelligence: An Introduction,Wiley, New York, 2002
[2] 丁永生编著.计算智能—理论、技术与应用.科学出版社,2004
[3] 褚蕾蕾,陈绥阳,周梦编著.计算智能的数学基础.科学出版社,2002
[4] 徐宗本,张讲社,郑亚林编著.计算智能中的仿生学:理论与算法,科学出版社,2003
[5] 徐宗本编著.计算智能（第一册）:模拟进化计算.高等教育出版社,2004
[6] 王国俊编著.计算智能（第二册）:词语计算与 Fuzzy 集.高等教育出版社,2005
[7] 罗四维著.大规模人工神经网络理论基础.清华大学出版社,北方交通大学出版社,2004
[8] Haykin S. Neural Networks: A Comprehensive Foundation, Macmillan, IEEE Press, 1994.
[9] 张乃尧,阎平凡编著.神经网络与模糊控制.清华大学出版社,2002
[10] 王小平,曹立明编著.遗传算法—理论、应用与软件实现,西安交通大学出版社, 2002
[11] 张颖,刘艳秋等编著.软计算方法.科学出版社,2002
[12] 谷荻隆嗣等编著.人工神经网络与模糊信号处理.科学出版社,2003
[13] 李士勇等编著.蚁群算法及其应用.哈尔滨工业大学出版社,2004
[14] 陈国良,王煦法,庄镇权等.遗传算法及其应用.人民邮电出版社,1995
[15] 史忠植编著.知识发现.清华大学出版社,2002
[16] 傅京孙等编著.人工智能及其应用.清华大学出版社,1987
[17] 蔡自兴,徐光佑编著.人工智能及其应用.清华大学出版社,1996
[18] 云庆夏等编著.遗传算法.冶金工业出版社,1999
[19] 云庆夏等编著.进化算法.冶金工业出版社,2002
[20] T. Hastie, R. Tibshirani, J. Friedman, Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer-Verlag, 2001
[21] T. Mitchell, Machine Learning. McGraw Hill, 1997
[22] C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995
[23] C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Knowledge Discovery and Data Mining, 1998, 2(2)
[24] N. Cristianini, J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2002
[25] K. M.Passino, S. Yurkovich. Fuzzy Control. Prentice Hall/Pearson. 2001.11.J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2001

第 2 章 模拟退火算法

2.1 概述

2.1.1 什么是模拟退火算法

模拟退火算法（simulated annealing，SA）是源于固体退火原理的一种拟物智能算法。其出发点是受物理退火过程启发，模拟固体加温、等温、冷却等物理过程，结合 Metropolis 抽样准则进行科学计算的一种启发式随机寻优算法。该算法已得到成功应用，研究成果涉及管理科学、计算机科学、分子物理学、生物学以及图像处理等科技领域。

模拟退火算法的主要特点有：

1）与局部搜索算法相比，模拟退火算法可望在较短时间内求得最优近似。

2）模拟退火算法允许任意选取初始解和随机数序列，又能得出最优近似解。因此该求解优化算法的前期工作量大大减少。

3）模拟退火算法能应用于多种组合优化问题，为一个问题编制的程序可以有效地应用于其他问题的求解。

2.1.2 物理退火过程

在对固体物质进行退火处理时，通常是先对它加温，使其熔化，让其中的粒子可以自由运动，然后随着温度缓慢下降，粒子逐渐形成低能态的晶格。若在凝结点附近的温度下降速率过快，则不能达到这个能量最低态，而是以一种多晶的或者以一种非晶的具有高能量的亚稳态结束。因此，这个过程的本质是慢速冷却，让粒子有充分的时间失去可动性，进行重新分布，确保粒子达到低能态势。

简单而言，物理退火过程由以下三部分组成：

1）加温过程。其目的是增强粒子的热运动，使其偏离平衡位置。当温度足够高时，固体将熔化为液体，从而消除系统原先可能存在的非均匀态，使随后进行的冷却过程以某一平衡态为起点。熔解过程与系统的熵增过程相联系，系统能量也随温度的升高而增大。

2）等温过程。物理学的知识告诉我们，对于与周围环境交换热量而温度不变的封闭系统，系统状态的自发变化总是朝自由能减少的方向进行，当自由能达到最小时，系统达到平衡状态。

3）冷却过程。其目的是使粒子的热运动减弱并逐渐趋于有序，系统能量逐渐下降，从而得到低能的晶体结构。

2.1.3 组合优化与物理退火的相似性

模拟退火算法求解组合优化问题与物理中晶体物质的退火过程具有相似性。 对于组

合优化问题来说，也有类似的过程，组合优化问题解空间的每一点都代表一个具有不同目标函数的解。所谓优化，就是在解空间寻找函数最小解的过程。若把函数看成能量函数，把控制参数视为温度，将解空间作为状态空间，那么模拟退火算法（SA）寻找基态的过程就是求目标函数极小值的优化过程。因此，基于 Metropolis 接受准则的最优化过程与物理退火过程存在一定的相似性。这里将组合优化与物理退火相似性归纳在下表 2-1 中。

表 2-1 组合优化与物理退火的相似性

组合优化	物理退火	组合优化	物理退火
解	粒子状态	Metropolis 抽样过程	等温过程
最优解	能量最低态	控制参数的下降	冷却
设定初始温度	熔解过程	目标函数	能量

这里对模拟退火算法能量下降示意做简单描述。小球 A 在能量函数曲线中的运动过程与物理退火过程相类似。如图 2.1 所示，用小球 A 的运动代表优化问题的智能计算过程。该能量函数曲线上有两个极小点 A 和 B，其中 A 为一个局部极小点，B 为一个全局极小点。如果我们将一小球（代表系统的初态）放在如图所示的 A 点左上方位置，代表初始状态，那么通过系统状态的改变，能量必将到达极小点 A。如果我们在系统中引入噪声，使整个系统产生振动，那么小球可能从 A 的附近被摇到 B 的附近。要使小球 A 达到极小值 B 点，一个比较好的办法是：先让系统剧烈振动，使小球脱离 A，然后轻轻摇，使小球逐步到达 B，这样就寻找到了能量函数的全局极小值。

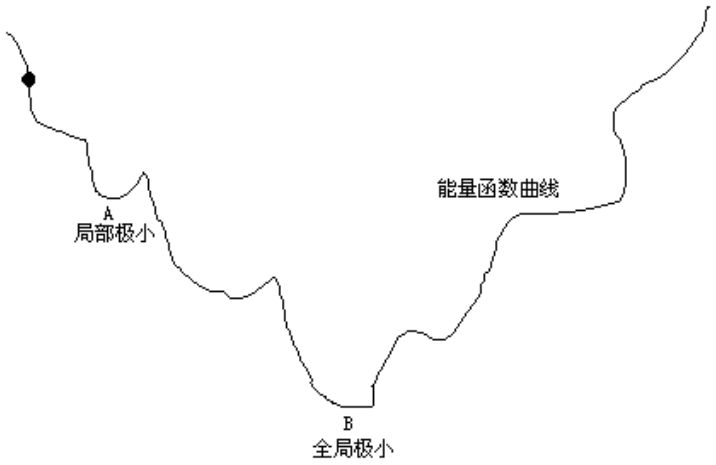


图 2.1 相似于物理退火的优化问题求解示意图

这恰好类似于物理退火过程：将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温度升高变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态。最后在常温时达到基态，内能减为最小。

基于 Metropolis 接受准则的优化过程，可避免搜索过程陷入局部极小，并最终趋于问题的全局最优解。

2.2 模拟退火算法

2.2.1 模拟退火算法的主要思想

作为一种通用的随机搜索算法，模拟退火算法有着更好的渐进行为。该算法的思想最早是由 N. Metoplis 等人于 1953 年提出的，但把它用于组合优化问题却是 S. Kirpatrick（1983）和 V. Cerny（1986）各自独立的工作。就函数最小值问题来说，模拟退火算法的主要思想是：在搜索区间（二维平面中）随机游走（即随机选择点），再以 Metropolis 抽样准则，使随机游走逐渐收敛于局部最优解。而温度即是 Metropolis 算法中的一个重要控制参数，可以认为这个参数的大小控制了随时过程向局部或全局最优解移动的快慢。

事实上，构成算法的三大支柱即为接受准则和随机数产生器（即 Metropolis 算法）、冷却进度表、新解产生器和领域结构。本节将详细讲解模拟退火算法三大支柱、模拟退火算法及其结构以及模拟退火算法应用中的一般要求。

2.2.2 Metropolis 准则

固体在恒定温度下达到热平衡的过程可以用 Monte Carlo 方法加以模拟，虽然该方法简单，但必须大量采样才能得到比较精确的结果，因而计算量很大。基于物理系统倾向于能量较低的状态，而热运动又妨碍它可以准确落到最低状态的考虑，采样时着重取那些有重要贡献的状态则可较快达到较好的结果，因此，Metropolis 等在 1953 年提出了重要性采样法，即以概率接受新状态。

先给定以粒子相对位置表征的初始状态 i ，作为固体的当前状态，该状态的能量为 E_i，然后用摄动装置使随机选取的某个粒子的位移随机地产生一微小变化，得到一个新的状态 j ，新状态的能量为 E_j ，如果 $E_j \prec E_i$ ，则接受新状态 j 为当前状态；否则，考虑到热运动的影响，该新状态是否被“ 接受 ”要依据粒子处于该状态的概率来判断。

粒子在温度 T 时趋于平衡的概率为：

$$p_r = \exp(-\frac{\Delta E}{kT})$$

其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。

p_r 是一个小于 1 的数。用随机数发生器产生一个在[0，1]区间均匀分布的随机数 ξ，若 $p_r \succ \xi$，则接受新状态 j ；反之，则舍弃。

如果新状态 j 可以接受，那么就以 j 取代 i 成为当前状态，重复以上新状态的产生过程。在大量迁移(固体状态的变换称为迁移)后，系统趋于能量较低的平衡状态。

通过对上述物理现象的模拟，可以得到函数优化的 Metropolis 接受准则。

用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f ，温度 T 演化成控制参数 t ，随即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，对当前解重复“ 产生新解→计算目标函数差→接受或舍弃 ”的迭代，并逐步衰减 t 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随

机搜索过程。

由解 i 过渡到解 j 的接受概率用以下的 Metropolis 准则确定：

$$P(t_k)=P(i\Rightarrow j)=\begin{cases}1 & f(j)\le f(i)\\ \exp(\dfrac{f(i)-f(j)}{t_k}) & f(j)>f(i)\end{cases}$$

经过大量迁移后，系统将趋于能量较低的平衡状态，各状态的概率将趋于某种概率分布，同时，需要注意：这种重要性采样过程在高温下可接受与当前状态能量差较大的新状态，而在低温下基本上只接受与当前能量差较小的新状态，当温度趋于零时，就不能接受比当前状态能量高的新状态。这与不同温度下热运动的影响完全一致。

这就是 Metropolis 准则，它比 Monte Carlo 方法的计算量显著减少，是一种有效的重点抽样法。重点抽样时，新状态下如果能量下降，则接受（局部最优）；若能量上升（全局搜索），以一定概率接受。

模拟退火方法从某个初始解出发,经过大量解的变换后,可以求得给定控制参数值时组合优化问题的相对最优解；然后减小控制参数 T 的值，重复执行 Metropolis 算法，就可以在控制参数 T 趋于零时，最终求得组合优化问题的整体最优解。

2.2.3 冷却进度表

我们称调整模拟退火法的一系列重要参数为冷却进度表。一个冷却进度表应当规定下述参数：①控制参数 t 的初值。②控制参数 t 的衰减函数。③马尔可夫链的长度 L_k（即每一次随机游走过程，要迭代多少次，才能趋于一个准平衡分布，即一个局部收敛解位置）。④结束条件的选择。冷却进度表中参数的选取非常重要，决定了模拟退火过程是否达到能量 E 的最小值，一个冷却进度表中参数的选择应当遵循以下原则：

1）控制参数初值 T_0 的选取

温度初始值 T_0 的设置是影响模拟退火算法全局搜索性能的重要因素之一。初始温度高，则搜索到全局最优解的可能性大，但因此要花费大量的计算时间；反之，则可节约计算时间，但全局搜索性能可能受到影响。一般要求初始值 T_0 的值要充分大，即一开始即处于高温状态，且 Metropolis 的接收率约为 1。实际应用过程中，初始温度一般需要依据实验结果进行若干次调整。

2）衰减函数 K 的选择

衰减函数用于控制温度的退火速度，一个常用的函数为：$T(n+1)=K*T(n)$，其中 K 是一个非常接近于 1 的常数。

3）马尔可夫链（Markov）长度 L_k 的选取

模拟退火中每次迭代时产生的变换数与 Markov 链相关，Markov 链是由每个时刻随机状态变量的取值产生的序列。分为时齐算法（固定每一温度 T，算法均计算 Markov 链的变化直至平稳分布）和非时齐算法（无需各温度下算法均达到平稳分布）。Markov 链的长度表示 Metropolis 算法第 k 次迭代时产生的变换数。

Markov 链长度的选取原则是：在控制参数 t 的衰减函数已选定的前提下，对 Markov 链长度的选取，应该满足在控制参数的每一个取值上解的概率分布都趋于平稳分布。即

Markov 链长度的选取，要在给定的温度衰减函数的前提下，保证算法在每个温度上都能够回复到准平衡状态为原则。但由于变换的接受率随控制参数值的递减而减小，接受固定数量的变换需进行的变换数随之增多，最终在 $t_k \to 0$ 时， $L_k \to \infty$ 。为此，可用某些常量 L 限定 L_k 的值，以免在 t 值较小时产生过长的 Markov 链。

4）终止条件 S 的选择

终止条件有很多种选择，选择不同的条件对算法的性能和解的质量有很大影响，本文只介绍一个常用的终止条件。即上一个最优解与最新的一个最优解的之差小于某个容差，即可停止此次马尔可夫链的迭代。

有效的冷却进度表判据为：① 算法的收敛：主要取决于衰减函数和马尔可夫链的长度及停止准则的选择；②算法的实验性能：最终解的质量和 CPU 的时间（即 CPU 全速工作时完成该进程所花费的时间）。

2.2.4 新解的产生和邻域结构

新解的产生是由一个产生装置从当前解出发，在解空间中产生一个新解，以便于下面的计算和接受(这是算法中最耗时的工作)，通常选择由当前解经过简单的变换即可产生新解的方法，如对构成解的全部或部分元素进行置换、互换或反演等，新解的产生方法决定了当前解的邻域结构。在邻域搜索类算法中，邻域结构是最重要的概念之一。邻域结构实质上是一个映射，例如，设 (S, f) 是组合优化问题的一个实例，则一个邻域结构是一个映射 $N: S \to 2^S$,其涵义是，对每一个解 $i \in S$,存在有一个解的集合 $S_i \subset S$ ，这些解在某种意义上是“ 邻近 ” i 的。集合 S_i 称为 i 的邻域，每个 $j \in S_i$ 称为 i 的一个临近解。此外，约定 $j \in S_i \Leftrightarrow i \in S_j$ 。而新解产生器可以理解为：设 (S, f) 是组合优化问题的一个实例，而 N 是一个邻域结构,则一个产生器是从解 i 的邻域 S_i 中选取解的一种方法。

2.2.5 模拟退火算法描述及其结构

有了以上基础，我们可以进行模拟退火算法的描述。设系统所有可能状态为 $V = \{v_1, v_2, ..., v_N\}$ ，与系统相对应有一能量 E ，它是系统状态的函数，即 $E(v)$ 。设控制参数为温度 T ，我们的目标是找到某一个系统状态 v^* ，使 $E(v^*) = \min(v_i, v_i \in V)$ 。模拟退火思想是：让 T 从一个足够高的值慢慢下降，对每个 T ，用 Metropolis 抽样法在计算机上模拟该系统在此 T 下的热平衡状态，即对当前状态 v_i 经过随机扰动产生一个新状态 v_j ，计算系统的能量增加 $\Delta E = E(v_j) - E(v_i)$ ，并以概率 $e^{(-\Delta E / kT)}$ 接受 v_j 作为当前状态。

模拟退火算法（SA）描述如下：

（1）初始化。任给一初始状态 V_0 , $V_1 = V_0$,计算 $E(v_0)$ ，并为参数 T 设置初始温度值。

（2）产生一随机扰动 Δv ,计算 $\Delta E = E(v_i + \Delta v) - E(v_i)$ 。

（3）若 $\Delta E \prec 0$ ，则转（5），否则在（0，1）区间上产生一个均匀分布的随机数 ξ 。

（4）若 $e^{-\Delta E / KT} \le \xi$ ，则转（2）。

（5）用 $v_i + \Delta v$ 取代原来的 v_i ，并令 $E = E + \Delta E$ 。

（6）在该 T 下，检验系统是否稳定，若不稳定则转（2）。

（7）以某一方式取 $T^{i} \prec T$，令 $T = T^{i}$。

（8）退火过程是否基本结束，是就停止，不是则转（2）。

简而言之，模拟退火算法的执行策略是：从一个任意被选择的初始解开始探测整个解空间，并且通过扰动产生一个新解，利用 Metropolis 准则判断是否接受新解，相应的下降控制温度。基本的模拟退火算法（SA）流程图如图 2.2 所示。

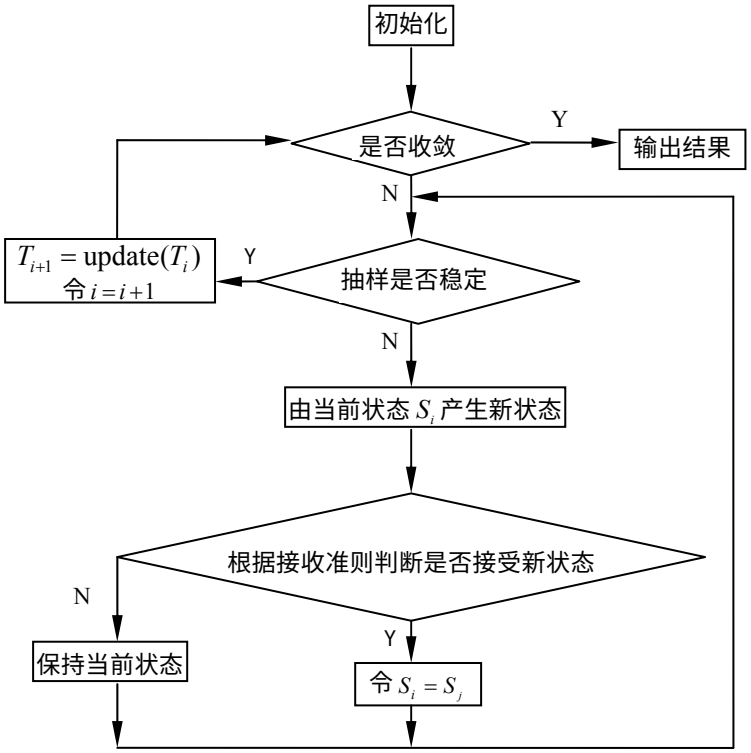


图 2.2　模拟退火算法流程图

归纳起来，模拟退火算法由 4 个步骤构成一轮试验：

（1）按某种随机机制由当前解产生一个新解；

（2）计算新解伴随的目标函数差；

（3）由接受准则，即新解更优，或恶化但满足 Metropolis 准则，判断是否接受新解，对有不可行解而限定解空间仅包含可行解时，还需先判断其可行性；

（4）满足接受准则时进行当前解和目标函数值的迭代，否则舍弃新解。

2.2.6 应用的一般要求

模拟退火算法应用的一般形式是：从选定的初始解开始，在借助于控制参数 t 递减时产生一系列 Markov 链中，利用一个新解产生装置和接受准则（Metropolis 准则），重复进行包括“产生新解—计算目标函数—判断是否接受新解—接受(或舍弃)新解”这四个任务的试验，不断对当前解进行迭代，从而达到目标函数最优的执行过程。因此，算法的应用需满足如下三个方面的要求：

1）对问题的简明形式描述

即数学模型，由解空间、目标函数和初始解三部分构成：

（1）解空间。它为问题的所有可能(可行的或包括不可行的)解的集合，它限定了初始解选取和新解产生时的范围。对无约束的优化问题，任一可能解（possible solution）即为一可行解（feasible solution），因此解空间就是所有可行解的集合；而在许多组合优化问题中，一个解除满足目标函数最优的要求外，还必须满足一组约束（constraint），因此在解集中可能包含一些不可行解（infeasible so1ution）。为此，可以限定解空间仅为所有可行解的集合，即在构造解时就考虑到对解的约束；也可允许解空间包含不可行解，而在目标函数中加上所谓罚函数（penalty function）以“ 惩罚 ”不可行解的出现。

（2）目标函数。目标函数是对问题的优化目标的数学描述，通常表述为若干优化目标的一个和式。目标函数的选取必须正确体现对问题的整体优化要求。例如，如上所述，当解空间包含不可行解时，目标函数中应包含对不可行解的罚函数项，借此将一个有约束的优化问题转化为无约束的优化问题。一般地，目标函数值不一定就是问题的优化目标值，但其对应关系应是显明的。此外，目标函数式应当是易于计算的，这将有利于在优化过程中简化目标函数差的计算，从而提高算法的效率。

（3）初始解。初始解是算法开始迭代的起点。初始解的选取应使得算法导出质量高的最终解，但大量试验结果表明，模拟退火算法是一种“健壮”的算法，即算法的最终解对于初始解的依赖性不大。

2）新解的产生和接受机制

（1）新解的产生见本章 2.2.4。

（2）计算与新解伴随的目标函数差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。

（3）判断新解是否被接受。判断的依据是 Metropolis 准则。此外，在有约束的组合优化问题中，该接受准则还应加上对新解的可行性判定。

（4）新解被确定接受时，用新解代替当前解，同时修正目标函数值。此时，当前解实现了一次迭代。可在此基础上开始下一轮试验。当新解被判定为舍弃时，则在原当前解基础上继续下一轮试验。

（3）冷却进度表

它是控制算法进程的一组参数的集合，包括控制参数初值及其衰减函数。对应的 markov 链（对 T 的每次取值进行的所有迭代过程）的长度和停止准则，是模拟退火算法应用的关键。在上机调试过程中，要求不断修正参数值，以寻求全局最优解。

2.3 实例分析：货郎担问题

2.3.1 货郎担问题的数学模型

货郎担问题（travelling salesman problem，简记为 TSP）可以描述为：设有 n 个城市，用数码 $1,..., n$ 代表。城市 i 和城市 j 之间的距离为 $d(i, j)i, j = 1,...,n$ 。TSP 问题是要找遍访每个域市恰好一次的一条回路，且其路径总长度为最短。

1）解空间

解空间 S 是遍访每个城市恰好一次的所有回路，是 $\{1,...,n\}$ 的所有循环排列的集合，S 中的成员记为 $(p_1, p_2,..., p_n)$，并记 $p_n + 1 = p_1$。初始解可选为 $(1,...,n)$

2）目标函数

此时的目标函数即为访问所有城市的路径总长度或称为代价函数，需求其最小值，选为：$\text{Min } f(p_1, p_2,..., p_n) = \sum_{i=1}^{n} dp_i p_{i+1}$，其中 $p_{n+1} = p_1$。

3）新解的产生

新解可通过分别或交替使用 2 变换法和 3 变换法产生。

以下称简 $p[u]$ 为城市序号，表示被访问的城市，其中 u 为序号，表示被访问城市的次序 $c[0], c[1],..., c[5]$ 表示在下面的函数 generate()中使用的与城市序号相对应的数组成员。

所谓 2 变换法就是任选两个序号 u , v，将这两个及期间所有的城市序号颠倒访问顺序。例如原访问顺序如下：

$$...p[u-1]p[u]p[u+1]...p[v-1]p[v]p[v+1]...$$

$$\begin{array}{cccccc} | & | & | & | & | & | \\ c[0] & c[1] & c[2] & c[3] & c[4] & c[5] \end{array}$$

选中序号 u 和 v， 且 $u \prec v$。 2 变换法就是将 $p[u]$ 和 $p[v]$ 及介于其间的所有城市序号颠倒排列产生新解：

$$...p[u-1]p[v]p[v-1]...p[u+1]p[u]p[v+1]...$$

$$\begin{array}{cccccc} | & | & | & | & | & | \\ c[0] & c[4] & c[3] & c[2] & c[1] & c[5] \end{array}$$

所谓 3 变换法就是任选三个序号 $u \prec v \prec w$,将序号为 u 和 v 及介于其间的所有城市，移动到序号为 w 的城市 $p[w]$ 之后。

例如，原访问顺序如下：

$$...p[u-1]\|p[u]...p[v]\|p[v+1]...p[w]\|p[w+1]...$$

$$\begin{array}{cccccc} | & | & | & | & | & | \\ c[0] & c[1] & c[2] & c[3] & c[4] & c[5] \end{array}$$

3 变换法后，

$$...p[u-1]\|p[v+1]...p[w]\|p[u]...p[v]\|p[w+1]...$$

$$\begin{array}{cccccc} | & | & | & | & | & | \\ c[0] & c[3] & c[4] & c[1] & c[2] & c[5] \end{array}$$

4）代价函数差

伴随新解的代价函数的差可分别由下面的公式计算。

对于2 变换 $(r = 2)$，

$$\Delta f = (d_{p_{u-1}p_v} + d_{p_u p_{v+1}}) - (d_{p_{u-1}p_u} + d_{p_v p_{v-1}})$$

对于3 变换(r=3) ，

$$\Delta f = (d_{p_{u-1}p_{v+1}} + d_{p_w p_u} + d_{p_v p_{w+1}}) - (d_{p_{u-1}p_u} + d_{p_v p_{v+1}} + d_{p_w p_{w+1}})$$

2.3.2 模型评价与分析

在此，取一组城市坐标值，对其进行实验，模拟结果如下：

取城市数$N=30$，坐标如下，运用上一节提到的编码方式，设定初始温度initial_temperature为73150.8，设定冷却率cooling_rate 为0.01，设定阀值threshold为500，Metropolis的步长为3.5，上机调试。

[41 37 54 25 7 2 68 71 54 83 64 18 22 83 91
94 84 67 62 64 99 58 44 62 69 60 54 60 46 38
25 24 58 71 74 87 18 13 82 62 58 45 41 44 4
38 42 69 71 78 76 40 40 7 32 35 21 26 35 50]

模拟结果：

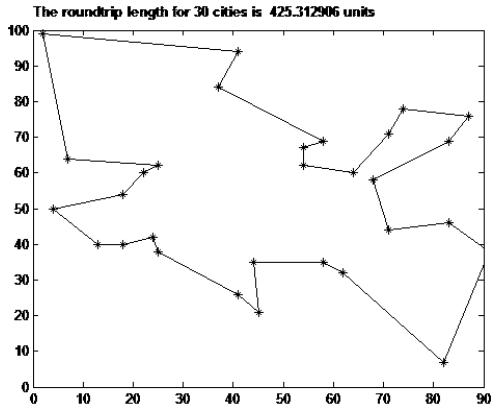


图 2.3 实例模拟结果

上机运行约 10 秒钟，运行结果如图 2.3 所示，获得的回路总长度 425.31（最优值d=423.74）。

由上述实验数据可知，模拟退火算法能够快速的求得最优解，是求解货郎担问题的一种有效算法。

思考题

2.1 什么叫模拟退火算法？组合优化与模拟退火有那些相似性？

2.2 模拟退火算法的主要思想是什么？

2.3 Metropolis 准则与冷却进度表在模拟退火算法中起到什么作用？

2.4 模拟退火算法在参数设置上应注意什么问题？

2.5 结合你的专业，简要概述模拟退火算法的应用领域。

2.6 取城市数 $N=10$,坐标如下。设置初始温度 t 为 168.14, 终止温度不大于 0.01，用模拟退火算法求最优值。

[0.4000 0.2439 0.1707 0.2293 0.5171
0.4439 0.1463 0.2293 0.7610 0.9414
0.8732 0.6878 0.8488 0.6683 0.6195
0.6536 0.5219 0.3609 0.2536 0.2634]

参考文献

[1] 康立山.非数值并行算法-模拟退火算法[M].科学出版社,1997
[2] 孙霞.分形原理及其应用[M].中国科学技术大学出版社,2003
[3] 刘素华,侯惠芳.一种基于模拟退火算法的模糊模式识别及其应用[J].计算机仿真,2004,12：182-184
[4] 张蓉,彭宏.一种快速的模拟退火算法及其在数据聚类中的应用[J].计算机工程与应用,2001(15)：85-87
[5] 龙小琼,郁松年.自适应最优保存的模拟退火遗传调度算法研究及其应用[J].计算机工程与应用.2004,17:65-67,92
[6] 靳利霞,唐焕文.模拟退火算法的一种改进及其在蛋白质结构预测中的应用[J].系统工程理论与实践,2002,09：92-96
[7] Bandyopadhyay, Sanghamitra, Sankar K, etal.Simulated annealing based pattern c1assification[J]. Information Sciences, 1998, 109(1-4)：165-185
[8] Palshikar，Girish Keshav. Simulated annealing：A heuristic optimization algorithm[J]. Software Tools for the Professional Programmer,2001，26(9)：121-124
[9] Mahfoud S.W.,Golberg D.E..A Genetic Algorithm for parallel simulated annealing[J].Proc int Conf in parallel problem solving from nature, Netherland,1992：260,1511-1514
[10] Duvis L. Genetic Algorilhrns and Simulated Annealing Lo ndon [M].London： PiLrnun; Los Altos： Morgan Kauf-mann.1987：32-41
[11] Grefenstette J. Incorporating Problem Specific Knowledge into Genetic Algorithms. In：Davis L Ed. Genetic Algorithms and Simulated Annealing, Pitman,1987,42-60
[12] Basu A., Frazer L. Rapid determination of the critical temperature in simulated annealing inversion[J]. Science, 1990(249): 1409-1412
[13] Mahfound S W, Goldberg D E. A Genetic Algorithm for parallel problem solving from Nature 2, North Holland, 1992,301-310
[14] SZU H, HARTLEY R. Fast simulated annealing. Phys Lett, 1987, A122(3-4)：157-162
[15] Sundharajan S. Optimal selection of capacitor for radial distribution system using a genetic algorithm.IEEE Transactions on Power Systems, 1994, 9：1499-1507
[16] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by Simulated Annealing, Science, 1983, 220：671-680

附录：模拟退火算法的 MATLAB 程序

```
clear all;
inputcities = [];                    %  输入城市坐标值
initial_temperature = 73150.8;       %  设定初始温度
cooling_rate = 0.01;                 %  设定冷却率
threshold = 2000;                    %  设定阀值
numberofcitiestoswap = 0.01;         %  设定 Metropolis 的步长
simulatedannealing(inputcities,initial_temperature,cooling_rate,threshold,numberofcitiestoswap);
%  所有参数都已初始化完毕，调用函数
global iterations;                   % 保持计数全局迭代次数
temperature = initial_temperature; % 插入当前温度作为初始温度
initial_cities_to_swap = numberofcitiestoswap;
%这是 TSP 问题特有的步骤——交换城市旅行的顺序。在这种算法中，城市的交换数目随着温度的降低而减少。这意味着随着温度的降温，搜索是局部进行的无梯度下降
iterations = 1;                      % 初始化迭代次数
complete_temperature_iterations = 0; %冷却温度，不论能量是否减至最低

%目标函数为路线总距离
previous_distance = distance(inputcities);
while iterations < threshold
    temp_cities = swapcities(inputcities,numberofcitiestoswap);
    current_distance = distance(temp_cities);
    diff = abs(current_distance - previous_distance);
    if current_distance < previous_distance
        inputcities = temp_cities;
        if rem(iterations,100) == 0
                plotcities(inputcities);
        end
        if complete_temperature_iterations >= 10
            temperature = cooling_rate*temperature;
            complete_temperature_iterations = 0;
        end
        numberofcitiestoswap = round(numberofcitiestoswap...
            *exp(-diff/(iterations*temperature)));
        if numberofcitiestoswap == 0
            numberofcitiestoswap = 1;
        end
```

```
        previous_distance = current_distance;
        iterations = iterations + 1;
        complete_temperature_iterations = complete_temperature_iterations + 1;
    else
        if rand(1) < exp(-diff/(temperature))
            inputcities = temp_cities;
            if rem(iterations,100) == 0
                plotcities(inputcities);
            end
            numberofcitiestoswap = round(numberofcitiestoswap...
                *exp(-diff/(iterations*temperature)));
            if numberofcitiestoswap == 0
                numberofcitiestoswap = 1;
            end
            previous_distance = current_distance;
            complete_temperature_iterations = complete_temperature_iterations + 1;
            iterations = iterations + 1;
        end
    end
end
```

（注：这里为工具箱下载地址为http：//www.math.org.cn/forums/index.php?showtopic=26860&st=0，运行环境为MATLAB6.5，下载后加入上述程序文件与inputcitie）

第 3 章 人工神经网络

3.1 概述

3.1.1 什么是人工神经网络

人工神经网络（artificial neural network，ANN）是在现代神经生物学研究成果基础上发展起来的一种模拟人脑神经系统的结构及功能，运用大量的处理部件，由人工方式构造的网络系统。它不仅具有处理数值数据的一般计算能力，而且还具有处理知识的思维、学习和记忆能力。

人工神经网络是对脑细胞的模拟，它是由大量处理单元（称人工神经元）经广泛互连而组成的人工网络，能反映出人脑功能的基本特性。在人工神经网络中，信息的处理是由神经元之间的相互作用来实现的，知识与信息的存储表现为网络元件互连间分布式的物理联系，网络的学习和识别取决于各神经元连接权值的动态演化过程。

3.1.2 人工神经网络的发展简史

人工神经网络的发展过程可以分为五个时期:

（1）20 世纪 40 年代信息科学的开创时期是人工神经网络的萌芽时期。1943 年，美国心理学家麦克卡洛克（Me.Culloch）和数学家匹茨（PittS.W.H.）首先从信息处理的观点出发，采用数学建模的方法，对人工神经细胞的工作原理进行了研究，建立了著名的闭值加权和模型(M-P 模型)，开创了神经网络科学研究的时代；1949 年心理学家 D.O.Hebb 提出了神经元之间突触的联系是可变的假说，即通过改变神经元之间的连接强度来实现学习功能，这是人工神经网络学习训练算法的起点和里程碑。

（2）1950－1968 年从单级感知器的构造成功到被否定是人工神经网络的第一次高潮期。这个时期的研究以 Marvin.Minsky，Fra.Rosenblat，Bernard.Windrow 等为代表人物，Rosenblat 成功地构造了单级感知器，它由阐值神经元组成，试图模拟人脑的感知及学习能力。这个模型虽然较为简单，但具备神经网络的一些基本特性，如可学习性、并行处理、分布式存储等，且能识别英文印刷体字母。

（3）1968－1980 年是人工神经网络研究的低潮期。人工智能的创始人 M.Minsky 和 S.Papert 等人对以感知器为代表的神经网络进行了深入系统的理论研究，并于 1969 年发表了很有影响的《Perceptron》一书。他们指出了感知器的局限性：只求解一阶谓词问题，对于较为复杂的高阶谓词的求解却无能为力；并指出如果适当地引入中间隐层，会大大地提高网络的能力，从而能够完成高阶谓词的求解。然而，在当时的技术条件下，对这种神经网络尚缺乏有效的学习算法，因此，M.Minsky 和 S.Papert 认为，要构造多层神经网络学习算法是极其困难的。由于 M.Minsky 和 S.Papert 拥有巨大的声望，人们对神经网络的研究热情大大下降。此外，神经网络研究工作还受到了现代计算机和人工智能快速发展所带来的冲击，基于仿生学的结构主义用硬件来模拟人脑的企图，不可避免地受到

了严重的阻碍。然而世界上仍有一部分学者献身于神经网络的研究，为后期的复兴奠定了基础。如威德罗（Widrow.B.）的自适应线性元件（adeline）模型；安德森（Anderson）的线性联想记忆理论；冯德曼尔斯博格（Von.Der.Makburg）的竞争学习理论，以及格罗斯博格（Grossberg.s）的自适应共振理论（ART）等。

（4）20 世纪 80 年代的第二次高潮期。基于十几年迅速发展起来的以逻辑符号处理为主的人工智能理论和并行分布处理模式的神经网络自身的研究结果，神经网络的研究又一次进入了高潮期。美国加州理工学院的物理学家 Hopfield.J.J.针对感知器的缺陷，将神经网络理论分析与动力学系统稳定性分析方法相结合，引入了“ 计算能量函数 ”的概念，从而明确了神经网络与动力学系统间的关系，提出了网络稳定性判断依据，尤其是该模型所具有的联想记忆、分类与误差自动校正等智能功能，激发了人们的热情，开拓了神经网络用于联想记忆和组合优化等计算的新途径。由此，掀起了神经网络研究的新热潮。比较重要的研究成果还有柯贺尼（kohonen）的特征影射理论，菲尔德曼（Fe1dman）和巴拉得（Ballard）的连接模型等。

（5）20 世纪 90 年代以后的发展时期。这一时期大多数学者的研究集中在三个方面：开发现有模型的应用，并在应用中根据实际运行情况对模型、算法作不同改进。人工神经网络是根据人们对生物神经网络的研究成果设计出来的，由一系列的神经元及相应的连接构成，具有良好的数学描述。其不仅可以用适当的电子线路来实现，而且更可以方便地用计算机程序加以模拟。

我国从 1986 年开始，先后召开了多次非正式的神经网络研讨会。1990 年 12 月，由中国计算机学会、电子学会、人工智能学会、自动化学会、通信学会、物理学会、生物物理学会和心理学会等八个学会联合在北京召开了“ 中国神经网络首届学术会议 ”，从而开创了我国神经网络研究的新纪元。

3.1.3 神经网络的特点

尽管人工神经网络只是人脑的低级近似，但是神经网络不同于一般的计算机和人工智能，它的很多特点和人类的智能相似。单个神经单元的功能很弱，但是大量神经元集体的、并行的行动却使处理功能十分强大。人工神经网络具有以下特点：

（1）人工神经网络是由大量的神经元广泛互连而成的系统，它的这一结构特点决定着人工神经网络具有高速信息处理的能力。人脑的每个神经元大约有 10^3～10^4 个树突及相应的突触，一个人的大脑总计约形成 10^{14}～10^{15} 个突触。用神经网络的术语来说，即是人脑具有 10^{14}～10^{15} 个互相连接的存储潜力。虽然每个神经元的运算功能十分简单，且信号传输速率也较低（大约 100 次/秒），但由于各神经元之间的极度并行互连功能，最终使得一个普通人的大脑在约 1 秒内就能完成现行计算机至少需要数 10 亿次处理步骤才能完成的任务。

（2）知识的分布式存储。与传统计算机不同，在神经网络中，知识不是存储在特定的存储单元中，而是所有记忆的信息都存储在神经元之间互连的权值中，每个神经元都只是整体概念的一部分，从单个权值中看不出其存储信息，每一个神经元都无法决定整体的状态，因而知识的存储是分布式的。

（3）容错性强。人脑具有较强的容错性，每天脑细胞的自动死亡，不会影响我们的

记忆能力和思考能力。同样，人工神经网络也具有很强的容错性，即局部的或部分的神经元损坏后，不会显著影响全局的活动。这是因为信息是分布存储在整个系统中的，因此网络中部分神经元的误差不会显著影响或改变整个系统的行为。

（4）具有自适应性。人工神经网络可以通过学习实现任意复杂的函数关系，而且具有自适应性，即自我调节的能力。把输入与理想输出模式输入到网络中，网络根据给定的学习算法，提取样本中的基本信息，来调整系统各层神经元之间的连接权值，把这些基本信息以神经元之间的连接权的形式存储在网络系统中，直至网络达到稳定状态。

（5）有综合推理能力。该特点又称为泛化能力。对于已经训练好的网络，当输入新的数据时，网络可以根据已有的知识识别新的信息，对信息进行分类，而不需要重新训练网络。

3.1.4 神经网络的研究内容

当前神经网络研究主要包括神经网络理论研究、神经网络实现技术和神经网络应用研究三个方面。

1）神经网络理论研究

神经网络理论研究侧重于寻找合适的神经网络模型和学习算法。其中，模型研究是指构造合适的单个神经元模型，确定神经元之间的连接方式，探讨它所适用的场合；学习算法研究是指在神经网络模型的基础上，找出一种调整神经网络结构和权值的算法，并满足学习样本的要求，同时具有较快的学习速度。

神经网络理论研究的另一个重要内容是从理论上分析常用的神经网络设计方法对泛化能力的影响。

2）神经网络实现技术研究

神经网络实现技术研究主要是探讨利用电子、光学、光电、生物等技术实现神经网络计算机的途径，包括利用传统计算机技术实现模拟神经计算机以及新型神经计算机体系结构的研究等。

3）神经网络应用研究

神经网络应用研究是探讨如何利用神经网络解决实际工程问题。人们可以在几乎所有的领域中发现神经网络应用的影子。当前神经网络的主要应用领域有：模式识别、故障检测、智能机器人、非线性系统辨识及控制、市场分析、决策优化、物资调用、智能接口、知识处理和认知科学等。

3.2 基本的神经元模型

3.2.1 生物神经元的结构

生物的脑神经系统是由巨量神经元构成的大规模神经网络。它通过感觉器官（视觉、听觉、嗅觉、味觉、触觉）接收外界信息，在大脑中进行加工处理，然后通过执行器官向外界输出。该系统无论从功能、结构上来讲都是非常复杂的，而神经元是其基本构成要素。

典型的神经元由细胞体（soma）和突起两部分组成，其中突起又分树突（dendrite）和轴突（axon）两种，如图 3.1 所示。细胞体包括细胞膜、细胞质、细胞核，它负责对输入信号进行处理，相当于 CPU。树突连接到细胞核所在的细胞体，它是由神经纤维组成的树状网络，是神经元中接受信号输入的部分。从细胞体延展出来的一根很长的纤维称为轴突，它是把输出信号传送给下一级的部分。轴突和下一个神经元的树突直接接触，它们的结合部称为突触（synapse），一般来说，一个神经元的轴突与其他神经元之间可以有几千个突触。突触将一个神经元的动作电位传给其他神经元，也就是说，它是神经元之间信息传递的载体。

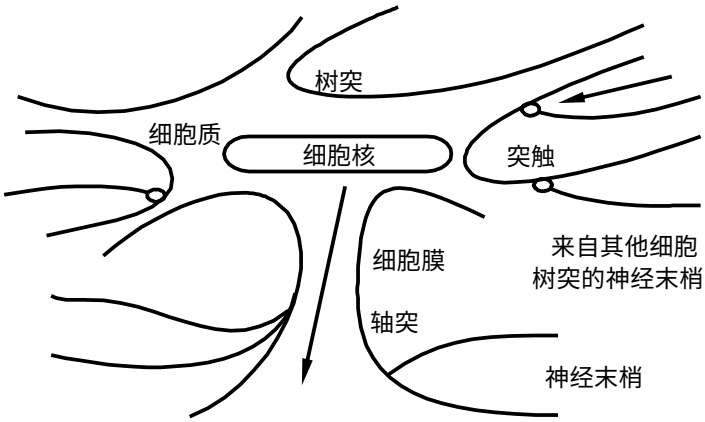


图 3.1　生物神经元的结构

我们可以将图 3.1 简化如图 3.2 所示的神经元功能模型。

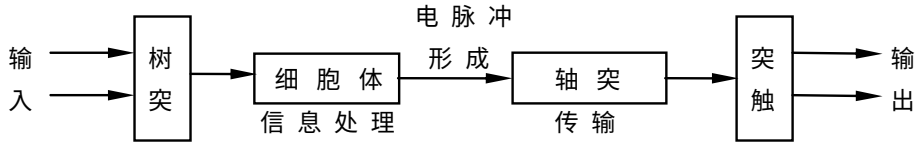


图 3.2　生物神经元功能模型

信号从一个神经元向一个突触上的另一个神经元的传输是一个复杂的化学过程。当神经元兴奋时，这种兴奋被转换为普通脉冲串沿轴突向下传递，直到到达轴突尖端的突触。突触在与下一个神经元接触的边缘领域根据所接收到的脉冲频度而释放化学物质，其作用是提高或降低下一神经元体内的电位。下一个神经元根据这种化学物质的量进行响应，从而发生其兴奋程度的改变。在化学物质越多神经元越兴奋的情况下，这两个神经元之间称为兴奋接合；反之，在化学物质越多、神经元的兴奋越被抑制时则称为抑制性接合。神经元的兴奋程度可以用称为 PSP（突触后电位）的神经元内部电位来说明。如果某神经元的 PSP 达到了一个阈值，该神经元就会沿着轴突发出具有固定强度和周期的脉冲或动作电位。

尽管大脑进行学习和记忆的奥秘仍是一个谜，但有关的认识正在不断增加。从解剖学的观点来看，神经元的可塑性只可能发生在突触部位。学习的形成看来是由于突触效

率的增加，而长期记忆可能与蛋白质的合成有关。人们已经在海马、鼠类动物身上做过大量实验，结果表明，突触的可塑性是学习、记忆的生理基础。

3.2.2 MP 模型

人工神经网络的第一个数学模型是由 Mc.Culloch 和 Pitts 建立的。该模型是基于这样一种思想：神经细胞的工作方式是或者兴奋，或者抑制。基于这个思想，Mc.Culloch 和 Pitts 在神经元模型中引入了硬极限函数，该函数形式后来被其他神经网络（多层感知器、离散 Hopfield 网络）采用。

由于神经元之间的信号连接强度取决于突触状态，因此在 MP 模型中，神经元的每个突触的活动强度用一个固定的实数即权值模拟。于是每个神经元模型都可以从数十个甚至数百个其他神经元接受信息，产生神经兴奋和冲动；同时，在其他条件不变的情况下，不论何种刺激，只要达到阈值以上，就能产生一个动作电位。但如果输入总和低于阈值，则不能引起任何可见的反应。

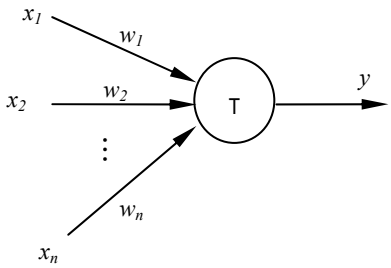


图 3.3　MP 模型

图 3.3 所示为 MP 模型示意图。图中 $x_1, x_2, \cdots, x_n$ 为神经元的输入，$w_1, w_2, \cdots, w_n$ 为相应的连接权值，T 为神经元的兴奋阈值，y 为神经元的输出。神经元的输出取二值函数，即

$$y = \begin{cases} 1, \ \sum_{i=1}^{n} w_i x_i \geq T \\ 0, \ \sum_{i=1}^{n} w_i x_i < T \end{cases} \tag{3-1}$$

式中，x_i 表示神经元的第 i 个输入权值；y 表示神经元的输出；T 表示神经元的阈值；n 为输入个数。

单个 MP 神经元模型可以实现与、或、与非、或非等二值逻辑运算（但不能实现异或运算）。另外，该模型曾因说明了人工神经网络可通过简单的计算产生相当复杂的行为，从而引起极大的轰动，但它是一种静态神经元，即结构固定，权值无法调节，因此缺乏一个关键性的要素，即学习能力。

3.2.3 一般神经元模型

由于 MP 模型过于简单，而且权值不能学习，因此需要更复杂、灵活性更高的神经元模型。图 3.4 所示为一个具有 n 个输入的通用的神经元模型。与 MP 模型一样，

$x=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$ 为神经元输入，$w=(w_1,w_2,\cdots,w_n)^{\mathrm{T}}$ 为可调的输入权值，θ 为偏移信号，用于建模神经元的兴奋阈值。$u(\cdot)$ 和 $f(\cdot)$ 分别表示神经元的基函数和激活函数（也称为神经元函数、挤压函数或活化函数）。基函数 $u(\cdot)$ 是一个多输入单输出函数 $u=u(x,w,\theta)$；激活函数的一般作用是对基函数输出 u 进行“挤压”：$y=f(u)$，即通过非线性函数 $f(\cdot)$ 将 u 变换到指定范围内。

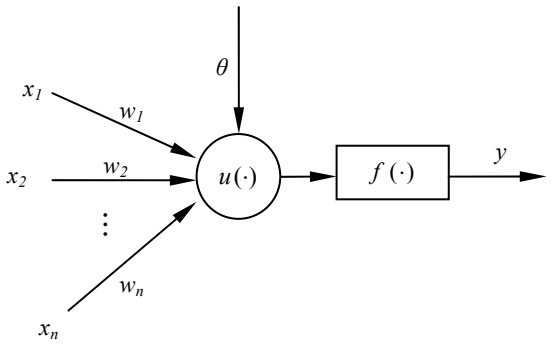


图 3.4　通用神经元模型

下面介绍常用的基函数及激活函数的类型。

1）基函数类型

（1）线性函数。

绝大多数神经网络都采用这种基函数形式，如多层感知器（MLP）、Hopfield 网络等。采用线性函数时，基函数输出 u 为输入和阈值的加权和，即

$$u=\sum_{j=1}^{n}w_jx_j-\theta=x^{\mathrm{T}}w-\theta \tag{3-2}$$

在多维空间中，该基函数形状是一个超平面。

（2）距离函数。

此基函数输出为

$$u=\sqrt{\sum_{j=1}^{n}(x_j-w_j)^2}=\|x-w\| \tag{3-3}$$

式中，w 常被称为基函数的中心。显然，u 表示输入矢量 x 和权矢量 w 之间的欧氏距离。在多维空间中，该基函数形状是一个以 w 为球心的超球。径向基函数主要用于径向基函数神经网络（RBF 网络）。

（3）椭圆基函数。

如基函数采用椭圆基函数，则神经元输入的总和为

$$u=\sqrt{\sum_{j=1}^{n}c_j(x_j-w_j)^2} \tag{3-4}$$

在多维空间中，该基函数形状是一个椭球。

2）激活函数类型

激活函数可以是线性的，也可以是非线性的。常用的激活函数有以下一些类型：

（1）硬极限函数。

硬极限函数的表达式如下：

$$y = f(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \tag{3-5}$$

或

$$y = f(u) = \mathrm{sgn}(u) = \begin{cases} 1, & u \geq 0 \\ -1, & u < 0 \end{cases} \tag{3-6}$$

式中，$\mathrm{sgn}(\cdot)$ 为符号函数。

对应于式（3-5）和式（3-6），硬极限函数的曲线如图 3.5 所示。式（3-5）的硬极限函数也叫单极限函数，式（3-6）的硬极限函数也叫双极限函数。

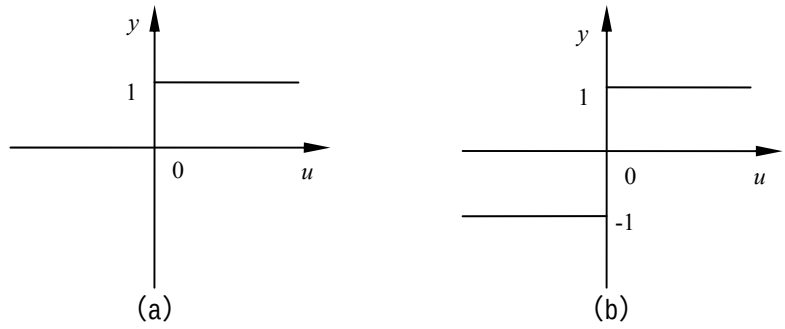


图 3.5　硬极限函数

硬极限函数常用于分类。

（2）线性函数。

如果激活函数采用线性函数，则神经元输出取基函数的输出 u，即

$$y = f(u) = u \tag{3-7}$$

线性函数的曲线如图 3.6 所示。该激活函数常用于实现函数逼近的神经网络的输出层神经元。

（3）饱和线性函数。

饱和线性函数的表达式如下：

$$y = f(u) = \frac{1}{2}(|u+1| - |u-1|) \tag{3-8}$$

饱和线性函数的曲线如图 3.7 所示，该函数也常用于分类。

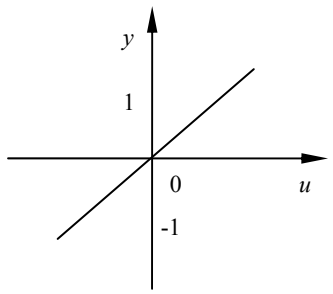


图 3.6　线性函数

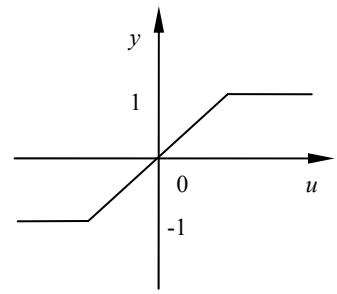


图 3.7　饱和线性函数

（4）Sigmoidal 函数。

Sigmoidal 函数也叫 S 函数，是一类非常重要的激活函数，无论神经网络用于分类、函数逼近或优化，Sigmoidal 函数都是常用的激活函数。Sigmoidal 函数的表达式为：

$$y = f(u) = \frac{1}{1 + e^{-\lambda u}} \tag{3-9}$$

或

$$y = f(u) = \frac{1 - e^{-\lambda u}}{1 + e^{-\lambda u}} \tag{3-10}$$

式中，参数 λ 称为 Sigmoidal 函数的增益，其值决定了函数非饱和段的斜率，λ 越大，曲线越陡。对应于式（3-9）和式（3-10），Sigmoidal 函数的曲线如图 3.8 所示。式（3-9）的函数也称单极限 Sigmoidal 函数，式（3-10）的函数也称双极限 Sigmoidal 函数或双曲正切函数。

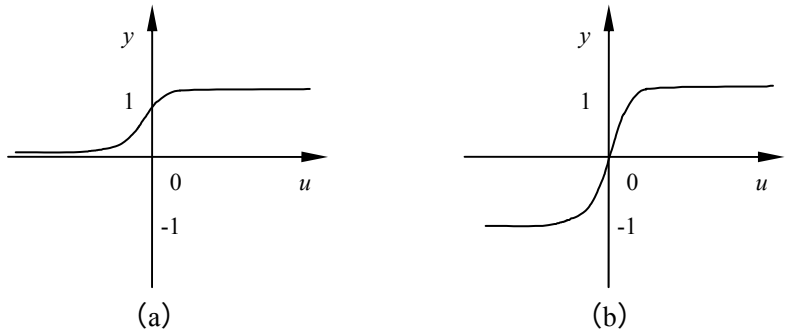


图 3.8　Sigmoidal 函数

与硬极限函数相比，Sigmoidal 函数是连续可微的，使得神经元的权值可用误差反向传播学习算法（BP 算法）调节。

（5）高斯函数。

高斯函数（也称钟形函数）也是极为重要的一类激活函数，常用于径向基函数神经网络（RBF 网络）。高斯函数的表达式为

$$y = f(u) = e^{-\frac{u^2}{\delta^2}} \tag{3-11}$$

式中，参数 δ 称为高斯函数的宽度或扩展常数。δ 越大，函数曲线就越平坦；反之，δ 越小，函数曲线就越陡峭。

高斯函数的曲线如图 3.9 所示。

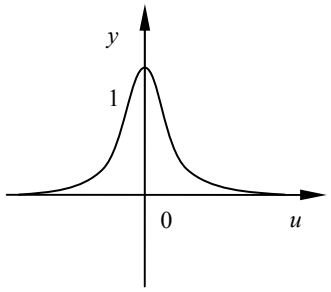


图 3.9　高斯函数

3.3 神经元的学习规则

3.3.1 神经元学习算法

生物之所以能适应环境，是因为生物神经系统具有从周围环境进行学习的能力。对于人工神经网络，学习能力也是其最为重要的特点。神经网络的学习有两种形式：有指导学习和无指导学习。

有指导学习亦称有监督学习（supervised learning）。一般情况下，有指导学习的训练样本是输入输出对 (p_i, d_i)，$i = 1,2,\cdots,N$，其中 p_i 为样本输入，d_i 为样本输出（教师信号）。神经网络训练的目的是：通过调节神经元的自由参数，使网络产生期望的行为，即当输入样本 p_i 时，网络输出尽可能接近 d_i。

无指导学习也称为无监督学习（unsupervised learning）或自组织学习（self-organized learning）。无指导学习不是提供教师信号，而是只规定学习方式或某些规则，具体的学习内容随系统所处环境（即输入信号情况）而异，系统可以自动发现环境特征及规律。

不管是有指导学习还是无指导学习，都需要通过调节神经元的自由参数（权值或阈值）而实现。

下面讲述神经元的学习算法。

对单个神经元，若令权矢量 $w = (w_1, w_2, \cdots, w_n, \theta)^{\mathrm{T}}$，输入样本 $x = (x_1, x_2, \cdots, x_n, -1)^{\mathrm{T}}$，阈值就可以并到权矢量中，于是当前权值 $w(t) = (w_1, w_2, \cdots, w_n, \theta)^{\mathrm{T}}$。对于有指导学习，假定输入 x 对应的期望输出为 d，于是神经元学习算法的内容是确定神经元的权值调整量 $\Delta w(t)$，并得到权值调节公式，即

$$w(t+1) = w(t) + \eta \Delta w(t) \tag{3-12}$$

式中，η 称为学习率，一般取较小的值；$\Delta w(t)$ 的值一般与 x、d 及当前权值 $w(t)$ 有关。

3.3.1.1 Hebb学习规则

Hebb 学习是一种无指导学习算法。1949 年，神经生物学家 Hebb 在其著作《Organization of Behavior》中给出了著名的 Hebb 学习规则，假设了人脑神经元之间突触联接强度的改变是学习和记忆的基础。Hebb 学习规则指出，如果一个突触前活动在时间上紧随以一个突触后活动，那么突触的连接强度会增强。后来其他人将之引申，并把突触连接强度的改变与突触前后电位相关联，即突触连接强度的增加，是与突触前和突触后电位的相关性成比例的。也就是说，如果一个突触有一个正的突触前电位和一个正的突触后电位，则突触的通导性增强；相反的，如果突触前电位为负/正，突触后电位为正/负，则突触的通导性减弱。

Hebb 规则说明：使用频繁，突触联系会变得更紧密，从而可理解为突触的特点是用进废退。实验证据为高频度刺激突触前神经元后，在突触后神经元上记录到的电位会增大，而且会维持相当长的时间。

根据 Hebb 规则，假定神经元的当前的输入为 $x=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$，输出为

$$y=f(w(t)^{\mathrm{T}}x) \tag{3-13}$$

则权矢量 $w(t)$ 的调节量为

$$\Delta w(t)=\eta yx \tag{3-14}$$

故神经元的权值修正公式为：

$$w(t+1)=w(t)+\eta yx \tag{3-15}$$

神经元的初始权值一般取零附近的随机值，激活函数 f 可以取任意形式。式(3-14)中 $\Delta w(t)$ 可理解为样本 x 对当前权值的影响。

除了上述的基本 Hebb 学习规则，根据应用的不同，还有 Oja 和 Karhunenn 的非线性 Hebb 学习算法等。

Hebb 学习规则常用于自组织网络或特征提取网络。

3.3.1.2 离散感知器学习规则

如果神经元的基函数取线性函数，激活函数取硬极限函数，则神经元就成了单神经元感知器。单神经元感知器的学习规则称为离散感知器学习规则，是一种有导学习算法。

对样本输入 x，假定神经元的期望输出为 d，当前神经元的激活函数取符号函数。则离散感知器学习规则中，权值调整量为

$$\Delta w(t)=e(t)x \tag{3-16}$$

其中误差信号为

$$e(t)=d-y=d-\mathrm{sgn}(w^{\mathrm{T}}x) \tag{3-17}$$

神经元的初始权值可以取任意值。

离散感知器学习规则常用于单层及多层离散感知器网络。

3.3.1.3 δ学习规则

δ 学习规则也称梯度法或最速下降法，是最常用的神经网络学习算法。δ 学习规则是一种有导学习算法。

1）梯度法基本原理

假定神经元权值修正的目标是极小化标量函数 $F(w)$。如果神经元的当前权值为 $w(t)$，且假设下一时刻的权值调节公式为：

$$w(t+1)=w(t)+\Delta w(t) \tag{3-18}$$

式中，$\Delta w(t)$ 代表当前时刻的修正方向。显然，期望每次修正均有

$$F(w(t+1))<F(w(t)) \tag{3-19}$$

那么什么样的 $\Delta w(t)$ 才是合适的呢？对 $F(w(t+1))$ 进行一阶泰勒展开，得

$$F(w(t+1))=F(w(t)+\Delta w(t))\approx F(w(t))+g^{\mathrm{T}}(t)\Delta w(t) \tag{3-20}$$

式中

$$g(t)=\nabla F(w(t))\Big|_{w=w(t)} \tag{3-21}$$

表示 $F(w)$ 在 $w=w(t)$ 时的梯度矢量。显然，如果取

$$\Delta w(t)=-cg(t) \tag{3-22}$$

式中，c 取较小的正数（称学习率），即权值修正量沿负梯度方向取最小值，则式（3-20）的右边第二项必然小于零，式（3-19）必然满足。这就是梯度法的基本原理。

下面以寻找 $f(x)=x^2$ 的最小点为例，介绍梯度法的使用过程。在该例子中，取初始点 $x(0)=1$，学习率 $c=0.4$。

由表 3-1 可知，进行 5 次迭代，便可以找到比较满意的近似解。

表 3-1　梯度法的迭代过程

t	$x(t)$	$g(x(t))$	$\Delta x(t)$	$f(x(t))$
0	1.00	2.0	-0.80	1.0
1	0.20	0.4	-0.16	0.04
2	0.04	0.08	-0.032	0.0016
3	0.008	0.016	-0.0064	0.000064
4	0.0016	0.0032	-0.00144	0.00000256
5	0.00016			

应该指出，局部最小点和学习率取值对梯度法的最终解影响很大。

2）神经元的 δ 学习规则

由于梯度法要用到目标函数的梯度值，因此在神经元权值调节的 δ 学习规则中，神经元基函数取线性函数，激活函数取 Sigmoidal 函数，即取

$$f(u)=\frac{1}{1+\mathrm{e}^{-\lambda u}}$$

或

$$f(u)=\frac{1-\mathrm{e}^{-\lambda u}}{1+\mathrm{e}^{-\lambda u}}$$

因为 Sigmoidal 函数是连续可微的。

神经元权值调节 δ 学习规则的目的是：通过训练权值 w，使得训练样本对 (x,d)，神经元的输出误差

$$E=\frac{1}{2}(d-y)^2=\frac{1}{2}(d-f(w^{\mathrm{T}}x))^2$$

达到最小。通过计算梯度矢量

$$\nabla E_w=(d-y)f'(w^{\mathrm{T}}x)x$$

并令 $\Delta w(t)=-\nabla E_w$，即可得到如下权值修正公式：

$$\Delta w(t)=-c(d-y)f'(w^{\mathrm{T}}x)x \tag{3-23}$$

神经元的初始权值一般取零附近的随机值。

δ 学习规则是应用最广泛的学习规则，常用于单层及多层感知器、BP 网络。

3.3.1.4 Widrow-Hoff学习规则

Widrow-Hoff 学习规则也称为 W-H 学习规则，是一种有指导学习算法。

Widrow-Hoff 学习规则可用于神经元激活函数取硬极限函数的情形，如图 3.10 所示。

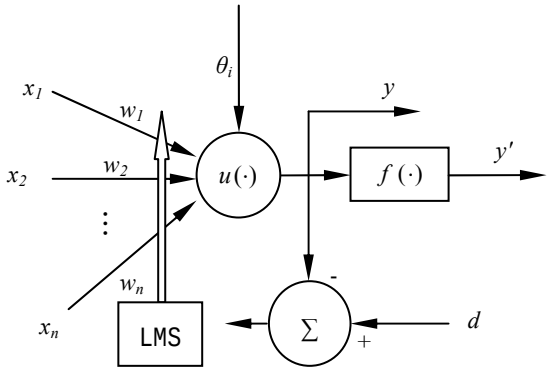


图 3.10 Widrow-Hoff 学习规则原理

通过训练权值 w，使得对于所有训练样本对 (x,d)，神经元的输出误差

$$E=\frac{1}{2}(d-y)^2=\frac{1}{2}(d-f(w^{\mathrm{T}}x))^2$$

达到最小，由于

$$\nabla E_w=(d-w^{\mathrm{T}}x)x$$

故权值修正公式为

$$\Delta w(t)=-c(d-y)x \tag{3-24}$$

Widrow-Hoff 学习规则常用于自适应线性单元（adaline）。

3.3.2 神经网络的拓扑结构

神经元之间的连接可以有任意形式，但最常见的结构是前向神经网络和反馈神经网络。

1）前向神经网络

前向神经网络也称多层前向网络或简称前馈网络，它是指拓扑结构为有向无环图的神经网络。在前向神经网络中，各神经元接受前一层的输入，并将计算结果输出给下一层，没有反馈。除了输入层之外，隐层和输出层神经元都实现一定的运算，因此称计算节点。图 3.11 至图 3.14 所示为常见的前向神经网络。

图 3.11 所示为两层感知器网络，该网络只有输入层和输出层，其中输出层神经元为计算节点，其基函数取线性函数，激活函数取硬极限函数，该网络一般用于线性分类。

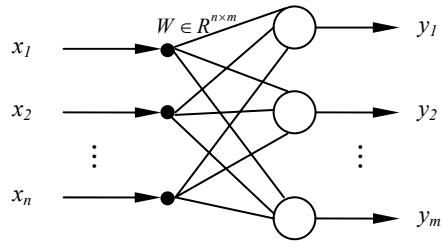


图 3.11　两层感知器网络

图 3.12 所示为多层感知器（MLP）网络，该网络有一个输入层、一个输出层和多个隐层，其中隐层和输出层神经元为计算节点。多层感知器的基函数取线性函数，激活函数可以取多种形式。如果所有的计算节点都取硬极限函数，则网络称为多层离散感知器；如果所有的隐层节点都取 Sigmoidal 函数，则就是所谓的 BP 网络，此时网络权值和阈值可用误差反向传播学习算法（即 BP 算法）学习。BP 网输出节点的激活函数根据应用的不同而异：如果 BP 网络用于分类，则输出层节点一般用 Sigmoidal 函数或硬极限函数；如果 BP 网络用于函数逼近，则输出层节点应该用线性函数。

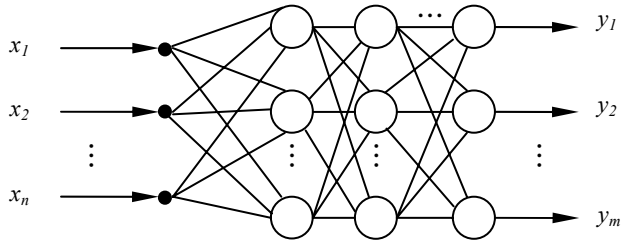


图 3.12　多层感知器网络

图 3.13 所示为径向基函数（RBF）神经网络，该网络有一个输入层、一个隐层和一个输出层，其中隐层和输出层神经元为计算节点。RBF 网的隐层单元的基函数取距离函数，激活函数一般取高斯函数。类似于多层感知器，如果 RBF 网用于分类，则输出层节点一般用 Sigmoidal 函数或硬极限函数；如果多层感知器用于函数逼近，则输出层节点可以用线性函数。

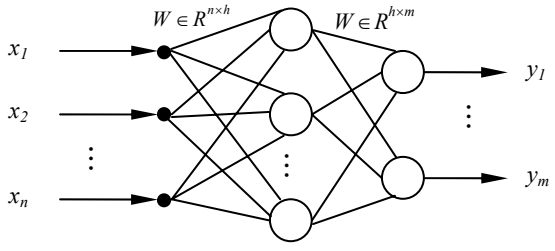


图 3.13　径向基函数神经网络

图 3.14 所示也是一种著名的前向神经网络，称为级连相关神经网络，该网络可以用

级连相关算法实现快速训练。

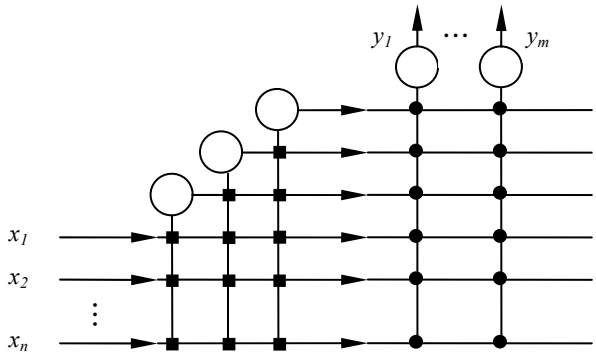


图 3.14　级连相关神经网络

2）反馈神经网络

反馈神经网络是指拓扑结构中有环路的神经网络。最著名的反馈神经网络是 Hopfield 神经网络，如图 3.15 所示。

Hopfield 网络的基函数取线性函数，激活函数可以取 Sigmoidal 函数（构成连续状态 Hopfield 网络）或硬极限函数（构成离散状态 Hopfield 网络）。

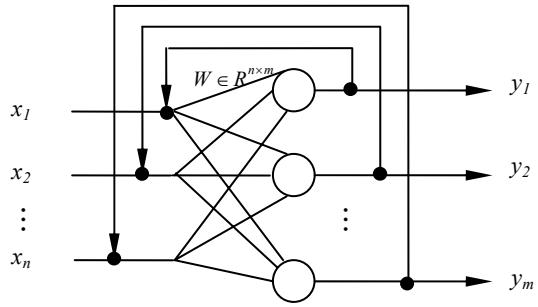


图 3.15　Hopfield 网络

3.4 人工神经网络模型

人工神经网络模型可分为前馈型神经网络模型、反馈型神经网络模型、自组织型神经网络模型和量子神经网络模型等。在此，仅介绍两种常见的网络模型。

3.4.1 BP 误差反向传播神经网络

由于在神经网络中引入了隐层神经元，神经网络就具有更好的分类和记忆等能力，因此相应的学习算法成了研究的焦点。1986 年 Rumelhart 等提出的 EBP（error back propagation，简称 BP 算法）算法，系统地解决了多层神经元网络中隐单元层连接权的学习问题，并在数学上给出了完整的推导。由于 BP 算法克服了简单感知器不能解决的 XOR 和其他一些问题，所以 BP 模型已成为神经网络的重要模型之一，并得以广泛使用。

采用 BP 算法的多层神经网络模型一般称为 BP 网络。它由输入层、中间层和输出层组成。中间层也就是隐含层，可以是一层或多层。BP 网络的学习过程由两部分组成：正向传播和反向传播。当正向传播时，信息从输入层经隐单元层处理后传向输出层，每一层神经元的状态只影响下一层的神经元状态。如果在输出层得不到希望的输出，则转入反向传播，将误差信号沿原来的神经元连接通路返回。返回过程中，逐一修改各层神经元连接的权值。这种过程不断迭代，最后使得误差信号达到允许的范围之内。

由上可以看到在多层前馈网络中有两种信号在流通：

（1）工作信号。它是施加输入信号后向前传播直到在输出端产生实际输出的信号，是输入和权值的函数。

（2）误差信号。网络实际输出与应有输出间的差值即为误差，它由输出端开始逐层向后传播。

下面具体推导用于多层前馈网络学习的 BP 算法。首先，讨论一下 BP 网络误差反向传播学习算法的基本思想。

为了研究神经网络是怎样从经验中学习的，我们应该首先向网络提供一些训练例子，并可以通过下述方法，教会一个三层前馈网络完成某个特定的任务。其方法步骤如下：

（1）向网络提供训练例子，即学习样本，包括输入单元的活性模式和期望的输出单元活性模式；

（2）确定网络的实际输出与期望输出之间允许的误差；

（3）改变网络中所有连接权值，使网络产生的输出更接近于期望的输出，直到满足确定的允许误差。

BP 学习算法是一种有监督的学习过程，它是根据给定的(输入、输出)样本对来进行学习，并通过调整网络连接权来体现学习的效果。就整个神经网络来说，一次学习过程由输入数据的正向传播和误差的反向传播两个子过程组成。设有 N 个学习实例 $(X_k,Y_k),k=1,2,\cdots,N$，对实例 (X_k,Y_k)，在正向传播过程中，实例 k 的输入向量 $X_k=(x_{1k},x_{2k},\cdots,x_{nk})$ 从输入层的 n 个节点输入，经隐层逐层处理，在输出层的 m 个节点的输出端得到实例 k 的网络计算输出向量 $Y_k=(y_{1k},y_{2k},\cdots,y_{mk})$。比较 Y_k 和实例 X_k 的期望输出向量 $Y_k^*=(y_{1k}^*,y_{2k}^*,\cdots,y_{mk}^*)$，若 N 个学习实例的计算输出都达到期望的结果，则学习过程结束；否则，进入误差反向传播过程，把 Y_k 与 Y_k^* 的误差由网络输出层向输入层反向传播，在反向传播过程中，修改各层神经元的连接权值。

1）误差反向传播计算

设 $I_{jk}^{(l)}$ 表示实例 k 的输入向量 X_k 输入后，传播到第 l 层节点 j 的输入；$O_{jk}^{(l)}$ 表示第 l 层节点 j 的输出；$w_{ij}^{(l-1)}$ 为第 $l-1$ 层的节点 i 连接第 l 层节点 j 的权值；$n^{(l-1)}$ 为第 $l-1$ 层的节点数；f 为节点神经元的传递函数，BP 网络的神经元传递函数一般使用可微的 Sigmoid 型函数。由 BP 网络神经元的输入输出关系，有

$$\begin{aligned} I_{jk}^{(l)} &= \sum_{i=1}^{n^{(l-1)}} w_{ij}^{(l-1)} O_{ik}^{(l-1)} \\ O_{ik}^{l} &= f(I_{jk}^{(l)}) \end{aligned} \tag{3-25}$$

实例 k 对节点 j 的期望输出 $O_{jk}^{*(l)}$ 与节点 j 对实例 k 的网络计算输出 $O_{jk}^{(l)}$ 的误差定义为

$$E_{jk}^{(l)}=\frac{1}{2}(O_{jk}^{*(l)}-O_{jk}^{(l)})^2 \tag{3-26}$$

若第 l 层是 BP 网络的输出层，即节点 j 是输出节点，则 $O_{jk}^{*(l)}=y_{jk}^{*}$，$O_{jk}^{(l)}=y_{jk}$，实例 k 的输出误差为

$$E_{jk}^{(l)}=\frac{1}{2}(y_{jk}^{*}-y_{jk})^2 \tag{3-27}$$

若对第 N 个学习实例的任一实例 k 有输出层的 m 个输出节点的计算输出都分别满足实例 k 的 m 个期望输出，即有 $E_{jk}^{(l)}\le\varepsilon(j=1,2,\cdots,m)$，则学习过程结束，$\varepsilon$ 为指定的允许误差；否则，由误差反向传播过程修改权值分布 w。

按误差的负梯度来修改权值，即

$$\begin{cases} w_{ij}^{(l-1)}=w_{ij}^{(l-1)}+\Delta w_{ij}^{(l-1)} \\ \Delta w_{ij}^{(l-1)}=-\eta\dfrac{\partial E_{jk}^{(l)}}{\partial w_{ij}^{(l-1)}} \end{cases} \tag{3-28}$$

其中：η 为学习率，$0<\eta<1$。

由式（3-25）和式（3-26）有

$$\frac{\partial E_{jk}^{(l)}}{\partial w_{ij}^{(l-1)}}=\frac{\partial E_{jk}^{(l)}}{\partial O_{jk}^{(l)}}\frac{\partial O_{jk}^{(l)}}{\partial I_{jk}^{(l)}}\frac{\partial I_{jk}^{(l)}}{\partial w_{ij}^{(l-1)}}=\delta_{jk}^{(l)}\frac{\partial I_{jk}^{(l)}}{\partial w_{ij}^{(l-1)}}=\delta_{jk}^{(l)}O_{ij}^{(l-1)} \tag{3-29}$$

其中：

$$\delta_{jk}^{(l)}=\frac{\partial E_{jk}^{(l)}}{\partial I_{jk}^{(l)}}=\frac{\partial E_{jk}^{(l)}}{\partial O_{jk}^{(l)}}\frac{\partial O_{jk}^{(l)}}{\partial I_{jk}^{(l)}}=\frac{\partial E_{jk}^{(l)}}{\partial O_{jk}^{(l)}}f'(I_{jk}^{(l)}) \tag{3-30}$$

为了得出计算 $\delta_{jk}^{(l)}$ 的表达式，进行下述讨论。

（1）若第 l 层是输出层，则由式（3-27）可得

$$\frac{\partial E_{jk}^{(l)}}{\partial O_{jk}^{(l)}}=\frac{\partial E_{jk}^{(l)}}{\partial y_{jk}}=-(y_{jk}^{*}-y_{jk}) \tag{3-31}$$

由式（3-30）和式（3-31）可得式（3-32）

$$\delta_{jk}^{(l)}=-(y_{jk}^{*}-y_{jk})f'(I_{jk}^{(l)}) \tag{3-32}$$

由式（2-28）和式（2-29）和式（3-32）可得式（3-33）

$$\Delta\omega_{ij}^{(l-1)}=-\eta\delta_{jk}^{(l)}O_{ik}^{(l-1)}=\eta(y_{jk}^{*}-y_{jk})f'(I_{jk}^{(l)}O_{ik}^{(l-1)}) \tag{3-33}$$

（2）若第 l 层不是输出层，则根据 BP 网络的误差反向传播定义，第 l 层节点 j 的误差 $E_{jk}^{(l)}$ 对节点 j 的输出 $O_{jk}^{(l)}$ 的变化率为第 $l+1$ 层的 $n^{(l+1)}$ 个节点的各节点误差对其输出的变化率之和。若 $E_{qk}^{(l+1)}$ 表示第 $l+1$ 层节点 q 的误差，$O_{qk}^{(l+1)}$ 表示节点 q 的计算输出，则有

$$\frac{\partial E_{jk}^{(l)}}{\partial O_{jk}^{(l)}}=\sum_{q=1}^{n^{(l+1)}}\frac{\partial E_{qk}^{(l+1)}}{\partial O_{qk}^{(l+1)}} \tag{3-34}$$

对第 $l+1$ 层节点 q，类似第 l 层的节点 j 的式（3-25）、式（3-26）和式（3-30），有

$$I_{qk}^{(l+1)}=\sum_{q=1}^{n^{(l)}}\omega_{jq}^{(l)}O_{jk}^{(l)} \tag{3-35}$$

$$O_{qk}^{(l+1)}=f(I_{qk}^{(l+1)}) \tag{3-36}$$

$$E_{qk}^{(l+1)}=\frac{1}{2}(O_{qk}^{*(l+1)}-O_{qk}^{(l+1)})^2 \tag{3-37}$$

$$\delta_{qk}^{(l+1)}=\frac{\partial E_{qk}^{(l+1)}}{\partial I_{qk}^{(l+1)}} \tag{3-38}$$

因此，可把式（3-34）表示为

$$\frac{\partial E_{jk}^{(l)}}{\partial O_{jk}^{(l)}}=\sum_{q=1}^{n^{(l+1)}}\frac{\partial E_{qk}^{(l+1)}}{\partial O_{qk}^{(l+1)}}=\sum_{q=1}^{n^{(l+1)}}\frac{\partial E_{qk}^{(l+1)}}{\partial I_{qk}^{(l+1)}}\frac{\partial I_{qk}^{(l+1)}}{\partial O_{qk}^{l}}=\sum_{q=1}^{n^{(l+1)}}\delta_{qk}^{(l+1)}\omega_{jq}^{(l)} \tag{3-39}$$

由式（3-30）和式（3-39）可得

$$\delta_{jk}^{(l)}=f'(I_{qk}^{(l)})\sum_{q=1}^{n^{(l+1)}}\delta_{qk}^{(l+1)}\omega_{jk}^{(l)} \tag{3-40}$$

由式（3-28）、式（3-29）和式（3-40）可得

$$\Delta\omega_{ij}^{(l-1)}=-\eta\delta_{qk}^{(l)}O_{ik}^{(l-1)} \tag{3-41}$$

式（3-32）和式（3-40）给出了误差反向传播时，计算输出层各节点和隐层各节点的 δ 的关系。由式（3-32）计算出输出层的各节点的 δ 值后，就可由式（3-40）反向逐层计算出各隐层的所有节点的 δ 值。由各节点的 δ 值，采用式（3-33）或式（3-41），就可计算出各节点的权值修改量 Δw，从而对权值进行修改。

在实际应用中，学习时要输入训练样本，每输入一次全部训练样本称为一个训练周期，学习要一个一个周期地进行，直到目标函数达到最小值或小于某一给定值。

用 BP 算法训练网络时有两种方式，一种是每输入一个样本修改一次权值，另一种是批处理方式，即将组成一个训练周期的全部样本都依次输入后计算总的平均误差。

2）BP 反向传播算法

经过上面误差反向传播的计算分析，反向传播算法的步骤可归纳如下。

（1）输入 N 个学习实例 $(X_k,Y_k^*),k=1,2,\cdots,N$。

（2）建立 BP 网络结构。确定网络层数 L≥3 和各层节点数，由学习实例输入向量 X_k 的长度 n 确定网络输入层节点数为 n；由学习实例输出向量 Y_k^* 的长度 m 确定网络输出层节点为 m；第 l 层的节点数为 $n^{(l)}$。定义各层间连接权矩阵，第 l 层连接第 $l+1$ 层的连接权矩阵为 $W^{(l)}=[W_{ij}^{(l)}]n^{(l)}\times n^{(l+1)}(l=1,2,\cdots,L-1)$，初始化各连接权矩阵的元素值。

（3）输入允许误差 ε 和学习率 η，初始化迭代计算次数 $t=1$，学习实例序号 $k=1$。

（4）取第 k 个学习实例 (X_k,Y_k^*)，$X_k=(x_{1k},x_{2k},\cdots,x_{nk})$，$Y_k^*=(y_{1k}^*,y_{1k}^*,\cdots,y_{nk}^*)$。

（5）由 X_k 进行正向传播计算。计算输入层各节点的输出为

$$O_{jk}^{(l)}=f(x_{jk})\qquad(j=1,2,\cdots,n)$$

逐层计算各层的各节点的输入和输出为

$$I_{jk}^{(l)}=\sum_{i=1}^{n^{(l-1)}}\omega_{ij}^{(l-1)}O_{ik}^{(l-1)} \qquad (l=2,\cdots,L;\ j=1,2,\cdots,n^{(l)})$$
$$O_{ik}^{(l)}=f(I_{jk}^{(l)})$$

（6）计算输出层（第 L 层）的各输出节点误差为

$$y_{jk}=O_{jk}^{(l)}$$
$$E_{jk}=\frac{1}{2}(y_{jk}^{*}-y_{jk}) \qquad (j=1,2,\cdots,m)$$

（7）若对 N 个学习实例的任一实例 k 有 $E_{jk}\le\varepsilon(j=1,2,\cdots,m)$，则学习过程结束；否则，进行误差反向传播修改各连接权矩阵。

（8）误差反向传播计算。修改第 $L-1$ 层隐层至输出层（第 L 层）的连接权矩阵为

$$\delta_{jk}^{(L)}=-(y_{jk}^{*}-y_{jk})f'(I_{jk}^{(L)})$$
$$\Delta\omega_{ij}^{(L-1)}(t)=\eta\delta_{ji}^{(L+1)}O_{ik}^{(L)}$$
$$\omega_{ij}^{(L-1)}(t+1)=\omega_{ij}^{(L-1)}(t)+\Delta\omega_{ij}^{(L-1)}(t)$$
$$(j=1,2,\cdots,m;i=1,2,\cdots,n^{(L-1)})$$

反向逐层修改连接各隐层的连接权矩阵：

$$\delta_{jk}^{(l)}=f'(I_{jk}^{(l)})\sum_{q=1}^{n^{(l+1)}}\delta_{qk}^{(l+1)}\omega_{jq}^{(l)}$$
$$\Delta\omega_{ij}^{(l-1)}(t)=-\eta\delta_{jk}^{(l)}O_{ik}^{(l-1)}$$
$$\omega_{ij}^{(l-1)}(t+1)=\omega_{ij}^{(l-1)}(t)+\Delta\omega_{ij}^{(l-1)}(t)$$
$$(l=L-1,\cdots,2,1;\ j=1,2,\cdots,n^{(l)};i=1,2,\cdots,n^{(l-1)})$$

（9）$k=k+1(\mathrm{mod}\ N)$，$t=t+1$，转步骤（4）。

3.4.2 Hopfield 模型

Hopfield 模型是霍普菲尔特分别于 1982 年及 1984 年提出的两个神经网络模型，一个是离散的，一个是连续的，它们都属于反馈网络，即它们从输入层至输出层都有反馈存在。在反馈网络中，由于网络的输出要反复地作为输入送人网络中，这就使得网络的状态在不断地改变，因而就提出了网络的稳定性问题。所谓一个网络是稳定的，是指从某一时刻开始，网络的状态不再改变。设用 $X(t)$ 表示网络在时刻 t 的状态，如果从 $t=0$ 的任一初态 $X(t)$ 开始，存在一个有限的时刻 t，使得从此时刻开始神经网络的状态不再发生变化，即

$$X(t+\Delta t)=X(t) \tag{3-42}$$

就称该网络是稳定的。

霍普菲尔特提出的离散网络模型是一个离散时间系统，每个神经元只有两种状态，可用 1 和-1，或者 1 和 0 表示，由连接权值 w_{ij} 所构成的矩阵是一个零对角的对称矩阵，即

$$w_{ij}=\begin{cases}w_{ij}, & 若 i\ne j\\ 0, & 若 i=j\end{cases} \tag{3-43}$$

在该网络中，每当有信息进入输入层时，在输入层不做任何计算，直接将输入信息分布地传递给下一层各有关节点。若用 $X_j(t)$ 表示节点 j 在时刻 t 的状态，则该节点在下一时刻（即 t+1）的状态由下式决定：

$$X_j(t+1)=\operatorname{sgn}(H_j(t))=\begin{cases}1, & \text{若 } H_j(t)\geq 0\\ -1\text{或}0, & \text{若 } H_j(t)<0\end{cases} \tag{3-44}$$

这里

$$H_j(t)=\sum_{i=1}^{n}w_{ij}X_i(t)-\theta_j \tag{3-45}$$

其中，w_{ij} 为从节点 i 到节点 j 的连接权值；θ_j 为节点 j 的阈值。

整个网络的状态用 $X(t)$ 表示，它是由各节点的状态所构成的向量。对于图 3.14，若设输出层只有 2 个节点，并用 1 和 0 分别表示每个节点的状态，则整个网络共有 4 种状态别为：

00，01，10，11

如果假设输出层有 3 个节点，则整个网络共有 8 种状态，每个状态是一个 3 位的二进制数，如图 3.16 所示。

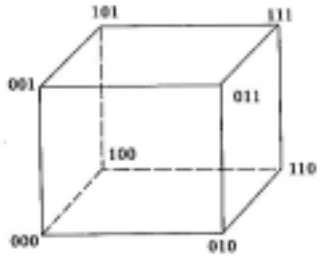


图 3.16　3 个神经元的 8 个状态

在该图中，立方体的每一个顶角代表一个网络状态。一般来说，如果在输出层有 n 个神经元，则网络就有 $2n$ 个状态，它可以与一个 n 维超立体的顶角相联系。当有一个输入向量输入到网络后，网络的迭代过程就不断地从一个顶角转向另一个顶角直至稳定于一个顶角为止。如果网络的输入不完全或只有部分正确，则网络将稳定于所期望顶角附近的一个顶角那里。

霍普菲尔特的离散网络模型有两种工作方式，即串行（异步）方式及并行（同步）方式。所谓串行方式，是指在任一时刻 f 只有一个神经元 i 发生状态变化，而其余 n-1 个神经元保持状态不变，即

$$X_j(t+1)=\operatorname{sgn}(H_j(t))$$

$$X_i(t+1)=X_i(t)\qquad \text{对所有 } i\neq j$$

所谓并行方式，是指在任一时刻 t，都有部分或全体神经元同时改变状态。

关于离散霍普菲尔特网络的稳定性，早在 1983 年就由科恩（Cohen）与葛劳斯伯格（S.Grossberg）给出了稳定性的证明。霍普菲尔特等又进一步证明，只要连接权值构成的矩阵是具有非负对角元的对称矩阵，则该网络就具有串行稳定性；若该矩阵为非负定矩阵，则该网络就具有并行稳定性。

离散霍普菲尔特网络中的神经元与生物神经元的差别较大，因为生物神经元的输入输出是连续的，而且生物神经元存在延时。于是霍普菲尔特于 1984 年又提出了连续时间的神经网络，在这种网络中，节点的状态可取 0 至 1 间任一实数值。

霍普菲尔特网络是一种非线性的动力学网络，它通过反复运算这一动态过程求解问题，这是符号逻辑方法所不具有的特性。在求解某些问题时，它与人们求解的方法很相似。例如对于“旅行推销员问题”就是这样。所谓旅行推销员问题是指：给定 8 个城市，要求找出一条能够到达各个城市但又不重复访问的最短路径。对该问题，若用穷尽搜索的方法来求解，则运算量将随城市数 N 的增加呈指数性增长，但若用霍普菲尔特网络求解这个问题，就会把最短路径问题化为一个网络能量（霍普菲尔特认为网络行为是受网络能量支配的，对于离散及连续的霍普菲尔特网络均有相应的能量函数来表示网络的能量）求极小的问题，这个动态系统的最后运行结果就是问题的解。当然，用这种方法求得的解不一定是最优的，即不一定是最短路径，而是某个较短路径，但这正好与人们凭直觉求解问题的效果是一致的。人们遇到问题时，经常能直观地很快作出决策，但它却不一定是最优的。

下面给出霍普菲尔特模型的算法：

1）设置互连权值

$$w_{ij}=\begin{cases}\sum_{s=1}^{m}x_i^s x_j^s, & i\neq j\\ 0, i=j, i\geq 1, j\leq n\end{cases}$$

其中，x_i^s 为 s 类样例的第 i 个分量，它可以为+1 或-1（0），样例类别数为 m，节点数为 n。

2）未知类别样本初始化

$$y_i(0)=x_i \qquad 1\leq i\leq n$$

其中，$y_i(t)$ 为节点 i 在 t 时刻的输出，当 t=0 时，$y_i(0)$ 就是节点 i 的初始值，x_i 为输入样本的第 i 个分量。

3）迭代直到收敛

$$y_i(t+1)=f(\sum_{i=1}^{n}w_{ij}y_i(t))$$

该过程将一直重复进行，直到进一步的迭代不再改变节点的输出为止。

4）回到步骤 2）继续。

霍普菲尔特模型的主要不足之处为：

（1）很难精确分析网络的性能。

（2）其动力学行为比较简单。

3.5 实例分析：人工神经网络在铁矿石价格预测中的应用

美国的 Mathwork 公司推出的 MATLAB 软件包既是一种非常实用有效的科研编程软件环境，又是一种进行科学和工程计算的交互式程序。MATLAB 本身带有神经网络工具箱，可以大大方便权值训练，减少训练程序工作量，有效地提高工作效率。

下面我们来看一个用三层 BP 神经网络预测铁矿石价格的实例。

3.5.1 MATLAB 中有关 BP 网络的重要函数

MATLAB 的 BP 网络工具箱中包含了进行 BP 网络分析和设计的许多工具函数，表 3-2 给出了这些函数的名称和基本功能（本章是在 MATLAB5.1 版本的基础上进行讨论，有关高版本的这部分内容，在理解本章内容的基础上，结合 help 命令，可顺利解决）。

表 3-2　BP 网络的重要函数和功能

函数名	功　　能
deltalin	Purelin 神经元的 δ（delta）函数
deltalog	Logsig 神经元的 δ 函数
dejtatan	Tansig 神经元的 δ 函数
errsHrf	计算误差曲面
initff	至多三层的前向网络初始化
learnbp	反向传播学习规则
learnbpm	利用冲量规则的改进 BP 算法
learnlm	Levenberg Marquardt 学习规则
logsig	对数 S 型传递函数
nwlog	对 Logsig 神经元产生 Nguyen Midrow 随机数
nwtan	对 Togsig 神经元产生 Nguyen Midrow 随机数
purelin	线性传递函数
simuff	前向网络仿真
tansig	正切 S 型传递函数
trainbp	利用 BP 算法训练前向网络
trainbpx	利用快速 BP 算法训练前向网络
trainlm	利用 Levenberg Marquardt 规则训练前向网络

3.5.2 铁矿石价格预测模型

从系统论的角度看来，铁矿石价格的形成机制是一个非线性系统，具有高度的复杂性。在选取预测方法时，正是考虑到铁矿石价格形成机制的复杂性，所以选择 BP 神经网络这种学习能力强，能够较好地拟合非线性系统的方法进行铁矿石价格的预测。运用神经网络对铁矿石价格进行预测时，组织训练样本有两种策略：一种是训练样本全部由过去的铁矿石价格的历史数据组成；另一种是训练样本由过去的铁矿石价格的历史数据及其影响因素的历史纪录组成，且考虑了影响铁矿石价格变化的因素。在此，选用的是第一种方法，即将铁矿石的历史价格作为输入值，预测未来铁矿石价格。

本例选取了河北某地区 2007 年 1 月 29 至 7 月 27 日 25 个周的价格（日平均值）数据作为铁矿石价格预测的历史数据，见表 3-3。

表 3-3 铁矿石价格表

时间/周	价格/元	时间/周	价格/元	时间/周	价格/元
1	645	10	690	19	690
2	650	11	670	20	700
3	660	12	665	21	700
4	660	13	675	22	705
5	665	14	675	23	708
6	685	15	695	24	715
7	695	16	710	25	715
8	695	17	710		
9	695	18	710		

具体实现步骤如下：

1）样本的选取和预处理

根据研究目标，选取合适的训练样本和检验样本，由于 BP 神经网络各层的激活函数、学习规则、数据的数量级等不同，所以在输入之前需对输入样本作归一化处理。MATLAB 提供了对数据进行归一化处理的函数：

[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t)

p—网络输入，pn—归一化后的网络输入，t—目标输出，tn—归一化后的目标输出。

2）确定 BP 网络的结构

包括与输出变量相关的输入变量的选取、隐层数和隐层节点数的选取、响应函数的确定、训练算法和参数的选取。

（1）网络的层数。

理论上早已证明：具有偏差和至少一个 S 型隐含层加上一个线性输出层的网络，能够逼近任何有理函数。增加层数主要可以更进一步地降低误差，提高精度，但同时也使网络复杂化，从而增加了网络权值的训练时间。而误差精度的提高实际上也可以通过增加隐含层中的神经元数目来获得，其训练效果也比增加层数更容易观察和调整，所以一般情况下，应优先考虑增加隐含层中的神经元数。因此，本模型中只设一个隐含层，设定方法如下：

net=newff(minmax(pn),[n,1],{'tansig','purelin'},'trainlm')

此函数是建立网络函数，n—隐含层的神经元数数目，1—隐含层数目。

（2）隐含层的神经元数数目。

网络训练精度的提高，可以通过采用一个隐含层，而增加其神经元个数的方法来获得，这在结构实现上，要比增加更多的隐含层简单得多。那么究竟选取多少个隐含节点才合适？这在理论上并没有一个明确的规定。一般对于三层前向网络隐含层节点数有如下经验公式：

$$K < \sum_{i=0}^{n} C\begin{pmatrix} j \\ i \end{pmatrix} \tag{3-46}$$

式中，K 为样本数，j 为隐含层节点数，n 为输入层节点数，$C\begin{pmatrix} j \\ i \end{pmatrix}=0$。

$$j=\sqrt{n+m}+\partial \tag{3-47}$$

式中，m 为输出层节点数，n 为输入层节点数，a 为 1～10 的常数。

$$j=2n+1 \tag{3-48}$$

式中，n 为输入层节点数。

上述公式根据不同的应用领域来选，其中公式（3-47）和（3-48）应用较多。根据试验情况，本模型的输入层节点数确定为 3，隐含层节点数根据训练情况最终确定为 7。因此，在建立网络时，n 设定为 7，即

```
net=newff(minmax(pn),[7,1],{'tansig','purelin'},'trainlm')
```

（3）初始权值的选取。

由于系统是非线性的，初始值对于学习是否达到局部最小、是否能够收敛以及训练时间的长短关系很大。如果初始值太大，使得加权后的输入落在激活函数的饱和区，从而导致其导数 $f'(x)$ 非常小，而在计算权值修正公式中，因为 ∂ 正比于 $f'(x)$，当 $f'(x)\to 0$ 时，则有 $\partial\to 0$，使得 $\Delta W_{ij}\to 0$，从而使得调节过程几乎停顿下来。所以，一般总是希望初始加权后的每个神经元的输出值都接近于零，这样可以保证每个神经元的权值都能够在它们的 S 型激活函数变化最大之处进行调节。所以，一般取初始权值在（-1，1）之间的随机数，设定格式为：

```
inputWeights=net.IW{1,1}
```

（4）学习速率的选取。

学习速率决定每一次循环中所产生的权值变化量。大的学习速率可能导致系统的不稳定，但小的学习速率将会导致学习时间较长，可能收敛速度很漫，不过能保证网络的误差值不跳出误差表面的低谷而最终趋于最小误差值。所以在一般情况下，倾向于选取较小的学习速率以保证系统的稳定性。学习速率的范围一般选取在 0.01~0.7 之间。和初始权值的选取过程一样，在一个神经网络设计中，网络要经过几个不同的学习速率的训练，通过观察每一次训练后的误差平方和 $\sum e^2$ 的下降速率来判断所选定的学习速率是否合适。如果 $\sum e^2$ 下降很快，则说明学习速率合适；若 $\sum e^2$ 出现震荡现象，则说明学习速率过大。对于每一个具体的网络都存在一个合适的学习速率。但对于较复杂的网络，在误差曲面的不同部位可能需要不同的学习速率。为了减少寻找学习速率的训练次数以及训练时间，比较合适的是采用变化的自适应学习速率，使网络的训练在不同的阶段自动设置不同学习速率的大小。学习速率的设定格式为：

```
net.trainParam.lr=0.05
```

（5）期望误差的选取。

在设计网络的训练过程中，期望误差值也应当通过对比训练后来确定一个合适的值。这个所谓的“合适”，是相对于所需要的隐含层的节点数来确定的，因为较小的期望误差值是要靠增加隐含层的节点、以及训练时间来获得的。一般情况下，作为对比，可以同时对两个不同期望误差值的网络进行训练，最后通过综合因素的考虑来确定采用其中一个网络。期望误差的设定格式为：

```
net.trainParam.goal=0.00001
```

3）训练网络

把输入样本输入神经网络，计算网络输出值，然后与实际值相比较，使用选定的网络训练算法，以一定的规则修改网络的连接权值。反复计算误差和修改权值，直到误差达到一定的范围以内，输入检验样本，判断检验结果。

网络训练的设计过程：

（1）BP 神经网络的初始化。

在训练前馈神经网络之前，我们必须设计权值和阈值的初始值。当我们使用函数 newff 创建前馈神经网络后，网络会自动地初始化权值和阈值，缺省值都为 0。如果要设置这些初始值，可以使用函数 init（），命令格式为：

net = init(net)

函数 init（）会根据网络的初始化函数以及它的参数值来设置网络权值和阈值的初始值，它们分别由参数 net.initFcn 和 net.initParam 表示。

（2）BP 神经网络的创建。

指令格式：net = newff

net=newff(minmax(pn),[n,1],{'tansig','purelin'},'trainlm')；minmax(pn)—pn 的取值范围；tansig—输入层→隐含层的传递函数；purelin—隐含层→输出层的传递函数；trainlm—BP 网络训练函数的缺省值。

当网络大于三层时，其指令格式为：

net = newff(PR，[sl s2... sN]，{TFl TF2... TFN}，BTF，BLF，PF)

PR—输入向量的取值范围；si—第 i 层的神经元个数，总共 N 层；TFi—第 i 层的传递函数，缺省值为“ tansig ”；BTF—BP 网络训练函数，缺省值为“ trainlm ”；BLF—BP 网络权值和阈值学习函数，缺省值为“ learngdm ”；PF—性能函数，缺省值为“ mse ”。

（3）BP 神经网络的仿真。

BP 神经网络的仿真使用函数 sim（），而且对于高维的多个输入，可以使用该函数方便地得到网络的仿真结果。

指令格式：t = sim(net，p)

P—样本输入，t—网络输出。

（4）BP 神经网络的训练。

当神经网络的权值和阈值初始化以后，我们就可以对网络进行训练。在训练的过程中，网络的权值和阈值被反复地调整，以减少网络性能函数 net.performFcn 的值，直到达到预先的要求。BP 神经网络的 net.performFcn 的缺省值是 mse–网络输出和目标输出的均方误差。

BP 神经网络的训练可以使用函数 train（）或 adapt（）。函数 train（）是通过调用参数 net.trainFcn 设定的训练函数来实现网络训练的，而且训练的方式由参数 net.traintParam 的值来确定。而函数 adapt（）是通过调用参数 net.adaptFcn 设定的训练函数来实现网络训练的，训练的方式由参数 net.adaptParam 的值确定。

本例中使用了 train 函数，其指令格式为：

[net TR]=train(net,pn,tn)

4）还原处理及结果分析。

对样本结果进行还原处理得到实际值。如果训练误差在允许范围内，而且网络泛化

能力较好，就可以利用训练好的神经网络来预测未来铁矿石价格了。MATLAB 提供了数据反归一化处理函数：

A=postmnmx(An,mint,maxt)

3.5.3 预测效果与结论

通过反复试验，本例最终采用 17 组数据作为训练样本，5 组数据作为检测样本，得到最优的 3—7—1 网络结构，即输入层、隐层和输出层的神经元数目分别为 3、7、1。利用 MATLAB 编程，设定网络训练代数为 1000，训练速率为 0.05，训练目标误差为 0.001，当收敛精度达到误差值时结果如表 3-4 所示：

表 3-4　训练样本和测试样本检验结果对照表

时间/周	实际值/元	训练值/元	检验值/元	相对误差/%
1	645			
2	650			
3	660			
4	660	660.126		0.019
5	665	665.091		0.013
6	685	684.954		-0.007
7	695	695.019		0.003
8	695	694.996		-0.001
9	695	695.003		0.000
10	690	690.001		0.000
11	670	670.023		0.003
12	665	664.996		-0.001
13	675	674.998		-0.000
14	675	675.006		0.001
15	695	694.999		-0.000
16	710	710.002		0.000
17	710	710.001		0.000
18	710	709.996		-0.001
19	690	689.999		-0.000
20	700	699.984		-0.002
21	700		699.947	-0.010
22	705		704.247	-0.110
23	708		716.884	1.250
24	715		717.353	0.330
25	715		715.512	0.070

图 3.17 为预测值与实际值的比较图，可以看到网络训练值基本与实际值图形重合，说明训练结果较好；而网络检验值与实际值图像虽然不能很好地重合，但是其误差也相差不大。在本例的五个检验值中，只有两个点偏移较大，说明将网络训练好后，该模型

具有较好的预测效果。

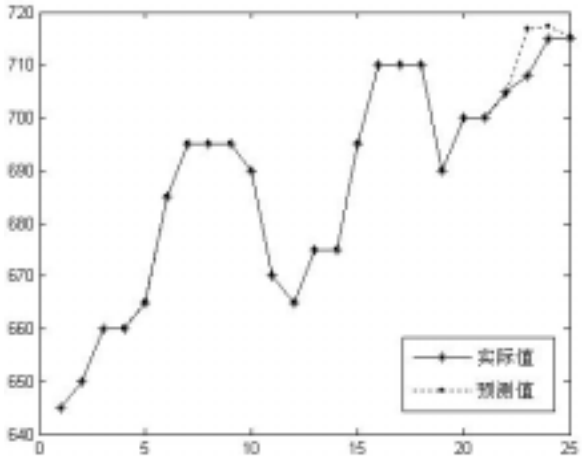


图 3.17　预测值与实际值的比较

思考题

3.1 什么是人工神经网络？它有哪些特征？
3.2 生物神经元由哪几部分构成？每一部分的作用是什么？
3.3 试述 BP 网络训练例子的方法步骤。
3.4 Hopfield 网络模型分为哪几类？
3.5 BP 网络应用设计应考虑哪些问题？
3.6 BP 网络的训练函数分哪几类？
3.7 Trainlm 函数作为训练函数的缺省值有何优缺点？缺点怎么解决？

参考文献

[1] 蔡瑞英,李长河. 人工智能.武汉:武汉理工大学出版社, 2003
[2] 周志华,曹存根. 神经网络及其应用.北京:清华大学出版社, 2004
[3] 钟珞, 饶文碧, 邹承明. 人工神经网络及其融合应用技术.北京:科学出版社, 2007
[4] 魏海坤. 神经网络结构设计的理论与方法.北京:国防工业出版社, 2005
[5] 闻新,周露,李翔,张宝伟. MATLAB 神经网络仿真与应用.北京:科学出版社, 2003
[6] 郭梨,连民杰. 铁矿石价格短期预测的BP神经网络方法.金属矿山,2007, 8
[7] Neural Network Toolbox User's Guide. The Mathworks, Ine, 1994
[8] (USA)Judd,J.S. Neural Network Design & the Complexity of Learning. Cambrige, 2000
[9] Heeht Nielson R. Neuroeom Putting. Addison Wesley,1990
[10] H.瓦格纳.露天和地下采矿技术的发展趋势.矿业工程, 2004(2): 18-24
[11] B.E.Hobbs, S Henley.Computing For Exploration And Mining Industries To the Year 2000.25th APCOM, 1995
[12] Taylor H K.General Background Theory of Cut-off Grades. Trans. Instn. Min. Metal, Sec. A, 81, 1972: 160-179, BRITAIN
[13] Taylor H K. Cutoff Grades-some Further Reflection. Trans. Instn. Min. Metal, Sec. A, 81,1985: 160-179, BRITAIN
[14] Mason F M. Capital and Operational Planning for Open-pits in Modern Economy. APCOM 18, 1985: 791-801, LONDON
[15] Yoshihiro Nakajima. Are Fluctuation in Stock Price Unexpected? Artificial Intelligence and Knowledge Based Processing(AI). 2000, vol(100), no(530): 37-42
[16] YOU Chen. A Study on the Chaos Model of the Liquidity in Stock Markets. Journal of Systems Science and Complexity, 2004, Vol(17), No(2)
[17] Kimoto T, Asakawa K. Stock Market Predication System with Modular Neural Networks. IJCNN, 1990(3): 12-19
[18] Takahira Yamaguchi. A Technical Analysis Expert System in the Stock Market. Future Generation Computer Systems, 1989(5): 21-27
[19] Hotelling H. The Economics of Exhaustible Resources. Journal of political economy, 1931(39): 137-175
[20] Stiglitz J. Monopoly and the Rate of Extraction of Exhaustible Resources. American economic review, 1976(66): 655-661
[21] Lewis T R, Matthews S A, Burness H S. Monopoly and the Rate of Extraction of Exhaustible Resources. American economic review, 1979, 69(1): 227-23
[22] Kurz H D, Salvadori N. Classical Economics and the Problem of Exhaustible resources. Metroeconomica, 2001, 52(3): 282-296

附录：BP 神经网络的 MATLAB 程序

```
p=[];   %训练样本输入
t =[];   %对应的目标输出
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t); %输入数据归一化
n=7; %设置网络隐单元的神经元数(5~30 验证后 7 个最好）
net=newff(minmax(pn),[n,1],{'tansig','purelin'},'trainlm');%建立相应的 BP 网络
inputWeights=net.IW{1,1};
inputbias=net.b{1};
layerWeights=net.IW{1,1};
layerbias=net.b{2};
%  训练网络
net.trainParam.show=50;%两次显示之间的训练次数
net.trainParam.lr=0.05;%训练速度
net.trainParam.mc=0.9;%μ 的初始值
net.trainParam.epochs=2000;%训练代数
net.trainParam.goal=0.00001;%网络目标性能
[net TR]=train(net,pn,tn); %调用 TRAINGDM 算法训练 BP 网络
An=sim(net,pn); %网络仿真
A=postmnmx(An,mint,maxt); %输出数据反归一化
E=A-t
M=sse(E)
N=mse(E)
A
%检验网络
p2=[]; %检验样本输入
p2=p2';
p2n=tramnmx(p2,minp,maxp);
A2n=sim(net,p2n);
A2=postmnmx(A2n,mint,maxt);
A2
```

第 4 章 进化计算

4.1 进化计算概述

进化计算（evolutionary computation， 简称 EC）是一系列启发式的搜索技术，主要包括遗传算法（genetic algorithms，简称 GA）、遗传规划（genetic programming，简称 GP）、进化策略（evolution strategies，简称 ES）和进化规划（evolutionary programming，简称 EP）。尽管进化计算有很多变化，但它们都是基于自然进化过程的基本计算模型，与传统的微积分和穷举法等优化方法相比，进化计算是一种成熟的具有高鲁棒性和广泛适应性的全局优化方法，具有自组织、自适应、自学习的特性，能够不受问题的限制，有效地处理传统优化算法难以解决的复杂问题。

4.1.1 生物的进化和遗传

生物群体的生存过程普遍遵循达尔文的物竞天择、适者生存的进化准则，生物通过个体间的选择、交叉、变异来适应大自然环境。在生物繁殖过程中，大多数生物通过遗传使物种保持相似的后代，部分生物由于变异，后代具有明显的差别，甚至形成新物种。正是由于生物的不断繁殖，生物数目大量增加，而自然界中生物赖以生存的资源却是有限的，因此，为了生存，生物就需要竞争。生物在生存竞争中，根据对环境的适应能力，适者生存，不适者消亡。自然界的生物，就是根据这种优胜劣汰的原则，不断地进行进化。

进化计算就是借鉴生物进化的规律，通过繁殖—竞争—再繁殖—再竞争，实现优胜劣汰，一步一步地逼进问题的最优解。进化计算中的“进化”就由此得名。

在生物繁殖过程中，生物上下代之间通过遗传物质的传递保证物种的延续。绝大多数生物的遗传物质是 DNA。由于细胞里的 DNA 大部分在染色体上，因此，遗传物质的主要载体是染色体。在 DNA 中，控制生物遗传的物质单元称作基因，它是具有遗传效应的 DNA 片断。在进化计算中，为了形成具有遗传物质的染色体，就用不同字符组成的字符串表达所研究的问题，这种字符串相当于染色体，其上的字符相当于基因。

生物在遗传过程中还会发生变异。变异方式有三种：基因重组、基因变异和染色体变异。基因重组是控制物种性状的基因发生重新组合，基因变异是指基因分子结构的改变，染色体变异是指染色体在结构上和数量上的变化。进化计算中，仿效生物的遗传方式，主要采用选择、交叉、变异这三种遗传操作，衍生下一代的个体。

4.1.2 进化计算的产生和发展

20 世纪 60 年代以来，如何模仿生物来建立功能强大的算法，进而将它们运用于复杂的优化问题，越来越成为一个研究热点。进化算法的研究就是在这一背景下孕育而生的。到 20 世纪 90 年代初，提出了“ 进化计算 ”这一术语，它是模拟生物进化过程中“ 优胜劣汰 ”的自然选择机制和遗传信息的传递规律的算法的总称，主要用来解决实际中的复杂优

化问题。目前，进化计算主要由遗传算法、遗传规划、进化策略和进化规划等分支组成，其他的诸如 DNA 计算和分子计算（molecular computing）也开始应用在实际问题中，但还没有形成一个体系，比较零散。上述四个主要分支是由不同的学者独立提出，各个分支都有自己的优缺点。进化计算学科的出现，促进了它的不同分支之间的交流，可以取长补短。

早在 20 世纪 60 年代初，美国 Michigan 大学的 J. H. Holland 教授就意识到了生物进化过程中蕴含着的朴素的优化思想，他借鉴了达尔文的生物进化论和孟德尔的遗传定律的基本思想，并将其进行提取、简化与抽象，提出了第一个进化计算算法——遗传算法。1975 年出版了他的专著《Adaptation in Natural and Artificial Systems》，标志着遗传算法的正式诞生。在这本专著中，他称之为“ genetic plans ”，详细阐述了遗传算法的基本思想和结构框架。“ genetic algorithms ”一词是首先出现在 J.D.Bagley 的博士论文中，他研究了遗传算法在博弈论（六子棋）中的参数搜索，这是遗传算法最早的应用。图 4.1 原理性地描述了自然进化与遗传算法之间的对应关系。

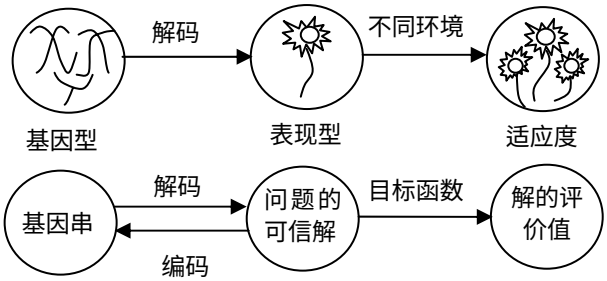


图 4.1　自然进化与遗传算法的对应关系

遗传算法产生后，在 20 世纪 80 年代以前，并没有引起人们的关注，一方面是因为它本身还不成熟；另一方面，当时的计算机容量小，计算速度慢，也使得需要较大计算量的遗传算法难以实际应用。但 Holland 和他的学生一直在进行坚持不懈的努力，进行了理论研究，并开拓其应用领域。直至现在，仍被认为是遗传算法理论基础的模式定理（schema theorem）就是在这个阶段提出的，它揭示了遗传算法的内部机理，解释了遗传算法的优化能力。进入 80 年代，遗传算法迎来了兴盛发展时期，无论是理论还是应用都成了研究热点。尤其是应用研究显得格外活跃，给遗传算法注入了新的活力。研究工作主要在遗传算法的基本理论、生物进化思想的深层利用、遗传算法的并行处理等方面获得了很大突破，目前已经在模式识别、图象处理、人工智能、经济管理、机械工程、电气工程、通讯、分子生物学等领域中获得了较成功的应用。

遗传规划是遗传算法的延伸和扩展。在 20 世纪 90 年代美国斯坦福大学的 J.R.Koza 教授提出了遗传规划。它仿照达尔文进化过程和染色体上自然发生的遗传操作，是基于群体的、具备随机和定向搜索特征的迭代过程。GP 的优点在于不需要给出具体的函数形式，同时，在初始群体足够大而且交叉和变异概率设置合理的情况下，不会陷入局部优化。GP 和遗传算法最大的不同在于个体的形式不同。遗传算法的个体是一个定长的字符串，而 GP 的个体是一个函数表达式，基于函数自身的特点，在分析时把函数表达式用于数据结构中，采用二叉树来表示，进而把基于二进制位串的遗传操作改进为针对二叉树的遗传操作。

进化规划是美国人 I.J.Fogel 于 20 世纪 60 年代提出来的。他提出采用有限字符集上的

符号序列表示模拟的环境，采用有限状态机表示智能系统，这种方法与遗传算法有许多共同之处，但不像遗传算法那样注重父代与子代的遗传细节，而是把侧重点放在父代与子代表现行为的联系上。当时学术界对进化规划持怀疑态度，直到 90 年代才逐渐受到重视并开始解决实际问题。

进化策略是在欧洲独立于遗传算法和进化规划而发展起来的。I.Rechenberg 和 H.P.Schwefel 是德国柏林工业大学的两名学生，1965 年在利用流体工程研究所的风洞做实验时 Rechenberg 提出了按自然变异和自然选择的生物进化思想的进化策略思想，当时的进化策略只有一个个体，而且进化操作也只有变异一种；1975 年，H.P.Schwefel 在他的博士论文中发展了进化策略，采用多个个体组成而不是单个个体参与进化，而且进化的手段包括变异和交叉，使进化策略更加完善。

4.1.3 进化计算的特征

在进化计算中，无论是遗传算法、遗传规划、进化策略或进化规划，都是从一组随机生成的初始个体出发，经过选择（复制）、交叉（重组）、变异（突变）等操作，并根据适应度大小进行个体的优胜劣汰，提高新一代群体的质量，再经过多次反复迭代，逐步逼近最优解。从数学角度讲，进化算法实质上是一种搜索寻优的方法。进化算法的这种搜索寻优不同于传统的方法，它不要求所研究的问题是连续的、可导的，但是却可以很快地得出所要求的最优解。图 4.2 表示进化算法的基本流程。

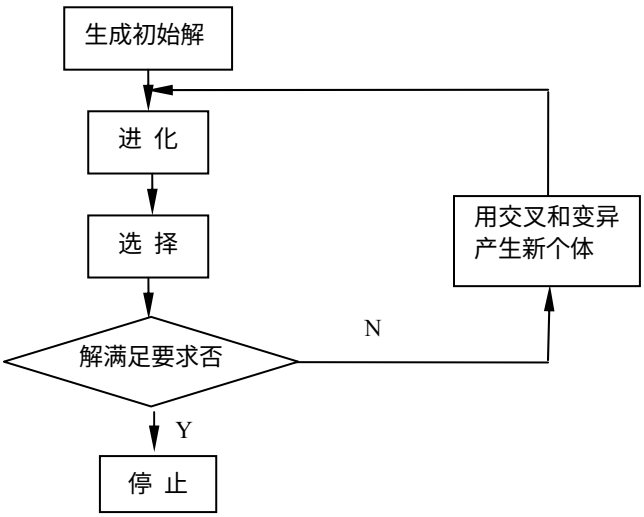


图 4.2　进化算法工作流程

进化计算的搜索方式，具有以下特征：

1）有指导搜索

进化计算的搜索策略，既不是盲目式的乱搜索，也不是穷举式的全面搜索，它是有指导的搜索。指导进化计算执行搜索的依据是适应度，也就是它的目标函数。在适应度的驱动下，使进化计算逐步逼近目标值。

2）自适应搜索

进化计算在搜索过程中，借助选择（复制）、交叉（重组）、变异（突变）等操作，体现“适者生存，劣者消亡”的自然选择规律，无需添加任何额外的作用，就能使群体的品质不断得到改进，具有自动适应环境的能力。

3）渐进式寻优

进化计算从随机产生的初始解出发，一代一代反复迭代，使新一代的结果优于上一代，逐渐得出最优解，这是一个逐渐寻优的过程。

4）并行式搜索

进化计算每一代运算都是针对一组个体同时进行，而不是对单个个体。因此，进化计算是一种多点齐头并进的并行算法，大大提高了进化计算的搜索速度。并行式计算是进化计算的一个重要特征。

5）黑箱式结构

进化计算根据所解决问题的特性，用字符串表达问题及适应度。一旦完成这两项工作，其余的选择、交叉、变异等操作都可按固定方式进行。个体的字符串表达如同输入，适应度计算如同输出。同时，从某种意义讲，进化计算是一种只考虑输入与输出关系的黑箱问题。图 4.3 就是黑箱表达的形象解释，图中 5 个开关代表 5 位 0～1 变量，表示输入的字符串。图中的输出 f(s)是适应度计算结果，代表输出。

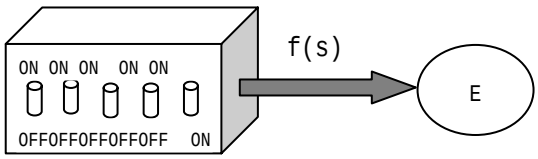


图 4.3　黑箱结构

6）全局最优解

进化计算由于采用多点并行搜索，而且每次迭代借助交叉和变异产生新个体，不断扩大搜索范围，因此进化计算很容易搜索出全局最优解而不是局部最优解。

7）通用性强

传统的优化算法，需要将所解决的问题用数学解析式表达，而且要求该数学函数的一阶导数或二阶导数存在。采用进化计算，只用某种字符表达问题，然后根据适应度区分个体优劣，其余的选择、交叉、变异等操作都是统一的，由计算机自动执行。因此，有人称进化计算是一种框架式算法，它只有一些简单的原则要求，在实施过程中无需额外的干预。

4.1.4 进化计算的应用

进化计算是一种新型的优化技术，它仿效生物界的进化与遗传，弥补传统优化技术的不足。然而进化计算并不是万能的，它并不能代替已有的优化技术，但可以独立或与其他优化算法相结合应用于以下领域：

1）人工智能

进化计算继模糊数学、专家系统、人工神经网络之后，成为处理人工智能的又一个有力工具。作为新颖的智能优化技术，进化计算必将在未来的人工智能中起到核心的优化技术作用，如机器学习中的规则优化，分类器系统中知识的自动获取、筛选，进化计算与人

工神经网络的融合，图像识别等。

2）结构性优化

通常，工程技术的优化包括结构优化和参数优化。对于后者，人们已经成功地使用了许多方法，如运筹学、数理统计、有限元等数值计算方法。然而对于结构性优化，还缺乏成熟、有效的方法。近年来，人们运用进化算法，成功地解决了建筑桁架结构、飞机结构以及管网等结构性问题，充分显示进化计算在这一领域的广泛应用前景。

3）复杂问题的优化

通常，实际要解决的问题具有非线性、多峰值、不确定性时，使用传统的优化方法常常不能奏效。进化计算由于是一种黑箱式的框架型技术，不要求有明确的因果关系数学表达式，因此它是解决这类问题的有力工具，可以解决诸如液体流动、气候变化以及军事战略等非线性动态系统问题。

4）人工生命

人工生命是研究用人工的方法模拟自然生命的特有行为，而基于进化计算的模型是研究人工生命的主要基础理论之一。这里，特有行为是指自组织行为、学习行为等。人工生命研究的意义在于，从科学的角度讲，通过对生命现象基本动力学的抽象，可以研究生命的起源，再现生命的原始进化过程，揭示生命遗传信息的存储与处理原理；从工程的角度讲，可以利用生命计算原理研究进化系统和自适应系统的构造，并应用在实际工程问题中。

5）综合应用

随着科学技术的发展，各种学科不断交叉渗透，相互促进。同样，进化计算也要和其他技术手段相结合，各自发挥特长，综合解决问题。例如，遗传算法用于神经网络自适应 KNO 控制系统是通过遗传算法训练神经网络而构成的自适应控制系统，实现了遗传算法、神经网络、KNO 控制三者的有机结合，获得了优异的控制效果。

进化计算虽然在理论与应用方面都取得了令人瞩目的成就，但仍然存在许多理论和应用技术中的不足与缺陷，值得深入研究，随着时间的推移，进化计算还会不断地开辟新的应用领域。

4.2 遗传算法

遗传算法是由美国的 J. H. Holland 教授于 1975 年在他的专著《Adaptation in Natural and Artificial Systems》中首先提出的，它是一类借鉴生物界自然选择和自然遗传机制的随机化搜索算法。Holland 教授所提出的 GA 通常称为简单遗传算法（SGA）。目前 GA 使用非常广泛。

4.2.1 简单遗传算法的基本原理

下面结合具体实例来介绍简单遗传算法的基本原理。

[例] 求一元函数 $f(x)=x\cos(4\pi x)+x^2$ 的最大值，$x\in[-1,2]$，求解结果精确到 6 位。

此问题是一峰值函数优化问题（见图 4.4），用传统的方法很难求解，现在采用基本遗传算法来解决这一优化问题。

4.2.1.1 编码与解码

遗传算法的工作对象是字符串，因此，编码是一项基础性工作。我们将问题结构变换为位串形式编码表示的过程叫编码；而相反，将位串形式编码表示变换为原问题结构的过程叫解码。这种位串形式编码表示叫染色体，有时也叫个体。

为了适应计算机的处理要求，遗传算法大多常采用二进制编码、格雷编码、实数编码、符号编码、排列编码等编码方式，其中，二进制的 0/1 编码方式是最常用的，本文就以二进制编码为例。

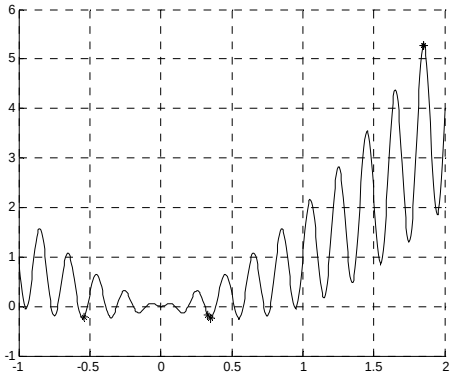


图 4.4 $f(x)$ 的几何图形

对于数值计算问题，一般要涉及将二进制转化为十进制，这个过程称为解码。对于长度（位数）为 L 的 0/1 字符串，按数学排列组合计算，它可以表达 $2\times2\times2\cdots2=2^L$ 个数，除了零以外，为 2^L-1，根据内插计算，十进制与二进制有如下关系：

$$x=x_{\min}+\frac{x_{\max}-x_{\min}}{2^L-1}\mathrm{Dec}(y) \tag{4-1}$$

式中，$x_{\min}$、$x_{\max}$ —— 变量 x 所取的最小及最大的十进制数值；y —— 对应于 x 的二进制数；Dec —— 将二进制数转换为十进制数。

在这种换算关系下，二进制表示法的精度 δ 为：

$$\delta=\frac{x_{\max}-x_{\min}}{2^L-1} \tag{4-2}$$

［例］针对上述优化问题，精度要求 $\delta=10^{-6}$，对于 x，由式（4-2）可得：

$$2^L=\frac{x_{\max}-x_{\min}}{\delta}+1=\frac{2+1}{10^{-6}}+1=3000001$$

即 $221<3000001<222$ ，所以本例的二进制编码长度至少需要 22 位，本例的编码过程实质上是将区间[-1，2]内对应的实数值转化为一个长度为 22 位的二进制字符串。

4.2.1.2 产生初始群体

SGA 采用随机方法生成若干个个体的集合，该集合称为初始种群。初始种群中个体的数量称为种群规模。种群规模 M 越大，搜索的范围越广，但是每代的遗传操作时间也越长。通常，M 取 50～100。

对于本节的示例，目标函数和约束条件并不复杂，所以可假定种群规模 M 取 30，即每次随机产生 22 位 0/1 随机数，作为一个初始个体，共产生 30 次，组成初始种群，例如：

$$v_1=(1001110001011010000110)$$
$$v_2=(1101110001010100001100)$$
$$\cdots\cdots$$
$$v_{30}=(1100010001010100001110)$$
$$M=\{v_1,v_2,\cdots,v_i,\cdots v_{30}\}\quad i=(0,1,\cdots 30)$$

4.2.1.3 计算适应度

遗传算法对一个个体（解）的好坏用适应度函数值来评价，适应度函数值越大，解的质量越好。适应度函数是遗传算法进化过程的驱动力，也是进行自然选择的唯一标准，它的设计应结合求解问题本身的要求而定。通常，遗传算法中个体的适应度也就是所研究问题的目标函数，但是，有时还需要对目标函数进行一定的转换，例如，目标函数为求最小值的情况。

本节的示例经过编码、产生初始种群的处理，形成了初始个体，为了计算适应度，首先应对二进制个体进行解码，然后代入目标函数中进行计算。

根据式（4-1），有

$$\begin{aligned}x_1&=-1+\frac{2+1}{2^{22}-1}\mathrm{Dec}(1001110001011010000110)\\&=-1+\frac{3}{4194303}\times 2561670\\&=0.832250\end{aligned}$$

同理，$x_2=1.488242$

对应的适应度为：

$$f(x_1)=f(0.832250)=0.266742$$
$$f(x_2)=f(1.488242)=3.686890$$
$$\cdots\cdots$$

4.2.1.4 选择操作

选择操作也叫复制操作，根据个体的适应度函数值所度量的优、劣程度决定它在下一代是被淘汰还是被遗传。一般地说，选择将使适应度较大（优良）个体有较大的存在机会，而适应度较小（低劣）的个体继续存在的机会也较小。简单遗传算法采用J.H.Holland教授推荐的赌轮选择法，赌轮选择法的基本思想是个体被选中的概率取决于个体的相对适应度：

$$p_i=f_i/\sum f_i \qquad (4\text{-}3)$$

式中，p_i—— 个体 i 被选中的概率；

f_i——个体 i 的适应度。

近年来，在遗传算法中常常采用择优选择法。这种方法没有明显的选择操作，而是根据个体的相对适应度 p_i 反复地从群体中选择 M 个个体组成下一代群体。很显然，个体的适应度越高，他被重复选中的可能性就越大，而重复选中的就相当于复制。相反的，适应度

小的个体选中的可能性也小，它极有可能被淘汰，这种选择过程可表述为：

1）顺序累计群体中各个体的适应度 f_i，得适应度累加值 S_n：

$$S_n = \sum_{i=1} f_i$$

2）利用公式（4-3），得相对适应度 p_i，即个体被选中的概率：

$$p_i = f_i / S_n$$

3）累计 p_i 得累计概率 g_i：

$$g_i = \sum_{j=1}^{i} p_j$$

4）产生[0，1]均匀分布的随机数 r；

5）将 r 与 g_i 相比较，若 $g_{i-1} < r < g_i$，则选择个体 i 进入下一代新群体。

6）重复执行 4）与 5）的操作，直至新群体的个体数目等于父代群体规模。

在本节示例中，按照上述过程计算个体的适应度累加值 S_n、相对适应度 p_i、累计概率 g_i 及产生随机数 r，重复这个过程，直至产生 30 个个体组成新群体。在这个群体中，必然有一些个体被重复选中，有一些个体被淘汰，这就类似于生态环境中的“优胜劣汰”的过程。

4.2.1.5 交叉操作

交叉是遗传算法区别于其他进化算法的重要特征，它在遗传算法中起关键作用，是产生新个体的主要方法。它类似于生物学中的杂交，通过不同个体的基因交换，从而产生新个体。SGA 中交叉算子采用单点交叉操作。具体而言，就是将随机选择出的两个个体 P_1 和 P_2 作为父代个体，将两者的部分码值进行交换。假设有如下两个体：

P_1	1	0	0	1	0	1	0	0
P_2	1	1	0	0	1	0	1	0

产生一个在 1 到 7 之间的随机数 c，假如现在产生的是 3，将 P_1 和 P_2 的低三位交换，P_1 的高五位与 P_2 的低三位组成数串 10001001，这就是 P_1 和 P_2 的一个后代 Q_1 个体；P_2 的高五位与 P_1 的低三位组成数串 11011110，这就是 P_1 和 P_2 的一个后代 Q_2 个体。其交换过程如图 4.5 所示。

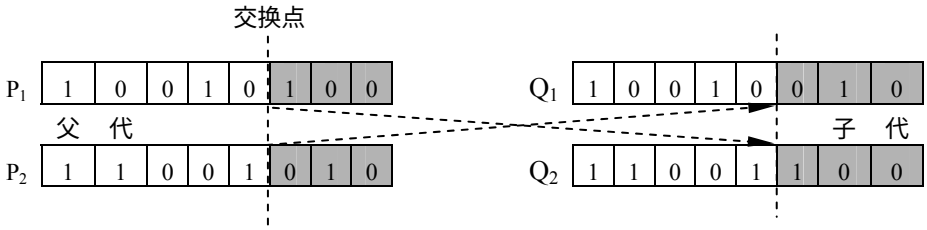


图 4.5 交换原理

通常，进行交叉操作的个体从选择后的新群体中随机选择。选择的方法也是赌轮选择法，采用这种方法可以使优良个体尽可能被选中，劣质个体选中的可能性较小，但也有可

能被“破格”选中。

在整个选择后的新群体中，究竟有多少个体要进行交叉操作需要一个参数来控制，控制被交换个体数目参数 M_c 可表达为：

$$M_c = p_c \cdot M \tag{4-4}$$

式中，M ——群体中个体的数目；p_c ——交换概率，表示群体中被交换个体的比例，常取 0.2～0.6。

若计算所得的 M_c 为奇数，则要增加一个交换个体或者删除一个交换个体，使交换对象成对出现。

通过交换，子代的字符串不同于父代。正是有了交叉操作，群体呈现出的性态才多种多样。

本节示例中，假设 p_c 取 0.3，由于群体规模 $M=30$，故有 $M_c = p_c \cdot M = 9$，即要有 9 个个体要进行交叉操作，然而 9 为奇数，所以需要随机选择增加或删除一个个体，假定选择的结果是删除，那么就有 8 个个体要进行交换。然后利用选择操作中的择优选择法选中 4 组待交换对象，若 v_2 和 v_{30} 是其中的一组，交叉点位 c=8，则其交换过程如表 4-1 所示。

表 4-1　交换结果

交换前	v_2	11011100010101 ¦ 00001100
	v_{30}	11000100010101 ¦ 00001110
交换后	v_2'	11011100010101 ¦ 00001110
	v_{30}'	11000100010101 ¦ 00001100

4.2.1.6 变异操作

所谓变异操作，是指依据变异概率 p_m 将个体编码串中的某些基因值用其他基因值来替换，从而形成一个新的个体。遗传算法中的变异操作是产生新个体的辅助方法，它决定了遗传算法的局部搜索能力，同时保持种群的多样性。交叉操作和变异操作相互配合，共同完成对搜索空间的全局搜索和局部搜索。SGA 中变异算子采用基本位变异算子。

基本位变异算子是指对个体编码串随机指定的某一位或某几位基因作变异运算。对于基本遗传算法中用二进制编码符号串所表示的个体，若需要进行变异操作的某一基因位上的原有基因值为 0，则变异操作将其变为 1；反之，若原有基因值为 1，则变异操作将其变为 0。其变异过程如图 4.6 所示。

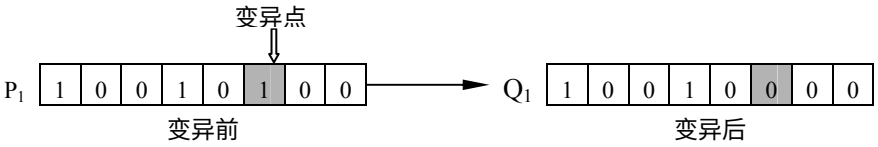


图 4.6　变异过程

在变异操作中，用变异概率 p_m 确定需要变异的个体。因为变异是针对字符进行的，所以变异概率也是针对字符而言，即：

$$p_m = \frac{B}{M \cdot L} \tag{4-5}$$

式中，B——每代中变异的字符数； M——每代中群体拥有的个体数目； L——个体中字符串长度。

通常， p_m 约为 0.005～0.01。

变异字符的位置也是随机确定的，如表 4-2 所示。假定某群体有 3 个个体，每个个体含 4 个字符。针对每个个体的每个字符产生一个[0，1]区间具有 3 位有效数字的均匀随机数。假设变异概率 p_m 取 0.01，则随机数小于 0.01 的对应字符产生变异。途中 3 号个体的第 4 位数字随机数为 0.001，小于 0.01，所以该字符的第四位产生变异，变异的结果为 1011。

在本节示例中，每代有 30 个个体，每个个体有 22 个字符，若变异概率 p_m 取 0.01，根据公式（4-5）可得 $B=6.6$，即每代平均有 6.6 个字符产生变异。为此，针对每个字符产生一个[0，1]区间的有效位为 5 位的随机数。一旦该随机数小于 0.01，则该字符产生变异。

表 4-2 变异示例

个体编号	个体	随机数				新个体
1	1100	0.835	0.421	0.763	0.123	
2	0101	0.451	0.101	0.035	0.074	
3	1010	0.542	0.763	0.024	0.001	1011

在表 4-3 中可看出，个体 v_4、 v_{10}、 v_{17}、 v_{24} 和 v_{30} 五个个体的 5 个字符产生变异。在这里，实际产生的变异字符数为 5，小于 6.6，而且 17 号个体有 2 个字符产生变异，这都是随机选择的结果。

表 4-3 变异结果

序号	字符位置	随机数	个体编号	个体中字符位置	原来字符	变异后的字符
1	82	0.00005	4	16	0	1
2	216	0.00124	10	8	1	0
3	357	0.00845	17	5	1	0
4	368	0.00021	17	16	0	1
5	641	0.00325	30	3	0	1

4.2.1.7 终止条件

遗传算法是一个反复迭代的过程，每次迭代期间，要执行适应度计算、选择、交叉、变异等操作，直至满足终止条件。常用的遗传算法终止条件主要有：

1）规定最大迭代次数 N

当某种操作的迭代次数达到 N，操作停止并输出结果。一般 N 取 200～500 次。由于遗传算法中有许多随机因素的影响，所以，最后一代的个体不一定含有最优个体，所以要记录每代的最优个体以便于最后比较。

2）规定最小的偏差 δ

对于适应度目标已知的遗传算法，如用方差作为适应度计算的曲线拟合问题，可用最小的偏差 δ 作为终止条件，即：

$$\left|f_{\max}-f'\right|\le\delta \tag{4-6}$$

式中，f' 为已知的适应度目标值；$f_{\max}$ 为每代最大的适应度。

3）观察适应度的变化趋势

在遗传算法初期，最优个体的适应度以及群体的平均适应度都较小，以后随着选择、交叉、变异等操作，适应度值增加，到了算法后期，这种增加趋缓甚至停止，在这种情况下，应终止遗传算法。

经过对本节示例的计算机演算，得到本问题的最优解 $\max f(x)=5.2786$，迭代过程中种群的变化和解的变化见图 4.7 所示。

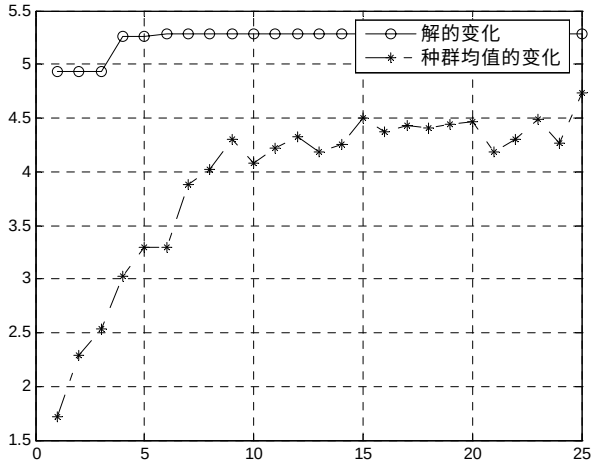


图 4.7　示例解的变化情况

4.2.2 遗传算法的求解步骤

遗传算法是一种基于空间搜索的算法，它通过自然选择、遗传、变异等操作以及达尔文适者生存的理论，模拟自然进化过程来寻找所求问题的解答。遗传算法具有以下特点：

（1）遗传算法是对参数集合的编码而非针对参数本身进行进化；

（2）遗传算法是从问题解的编码组开始而非从单个解开始搜索；

（3）遗传算法利用目标函数适应度而非利用导数或其他辅助信息来指导搜索；

（4）遗传算法利用选择、交叉、变异等算子而不是利用确定性规则进行随机操作。

简单遗传算法框图如图 4.8 所示。概括而言，其主要步骤包括：

（1）初始化群体；

（2）计算群体中每个个体的适应度值；

（3）由个体适应度值所决定的某个规则选择将进入下一代的个体；

（4）按概率 Pc 进行交叉操作；

（5）按概率 Pm 进行变异操作；

（6）没有满足某种停止条件，则转第（2）步，否则进入（7）；

（7）输出种群中适应度值最优的染色体作为问题的满意解或最优解。

遗传算法还可以用形式化语言表达，具体可描述为：

```
Procedure: Genetic Algorithms
begin
    t ← 0；initialize P(t)；evaluate P(t)；
    while (not termination condition ) do
    begin
            recombine P(t) to yield C(t)；
            evaluate C(t)；
            select P(t+1) from P(t) and C(t)；
            t ← t+1；
    end
end
```

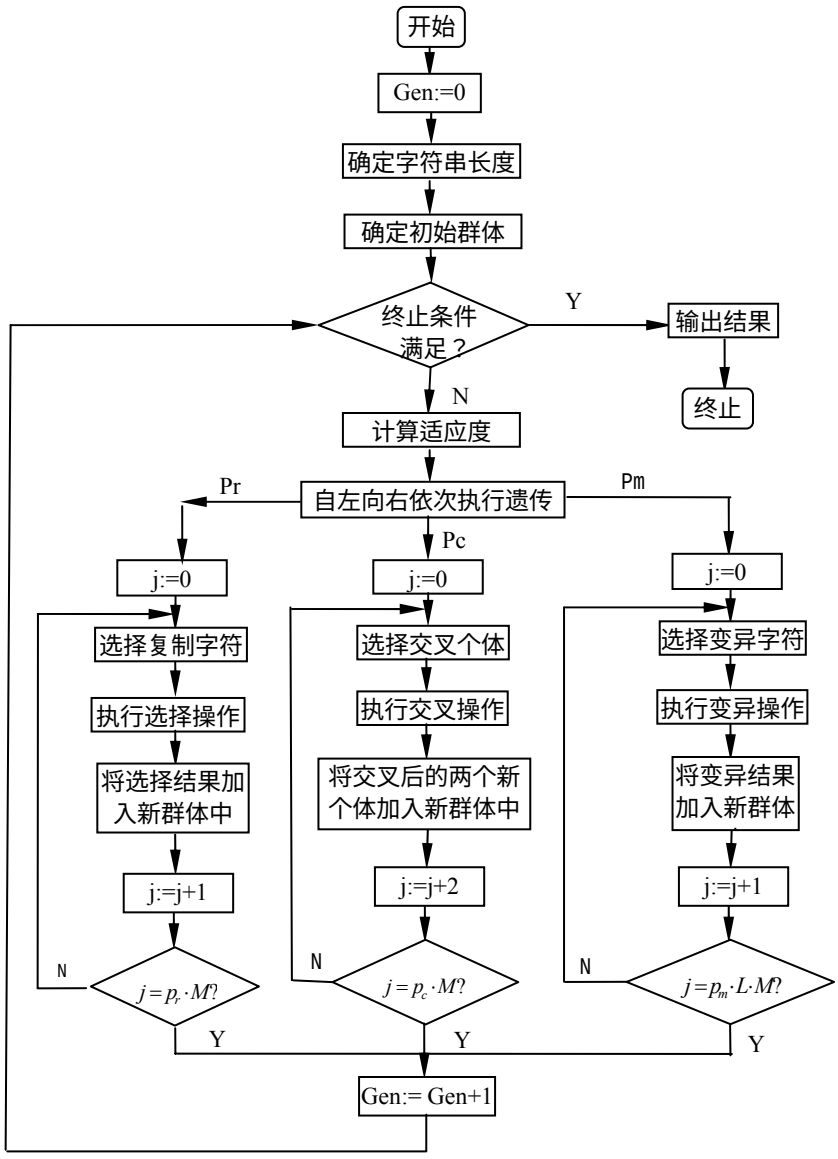


图 4.8　遗传算法流程图

4.2.3 遗传算法的理论基础

对遗传算法的运行机理的解释有两类，传统的模式理论和 1990 年以后发展起来的有限状态马尔可夫链模型。

4.2.3.1 概述

模式理论是由 Holland 教授创建， 主要包括模式定理、隐并行性原理和积木块假设三部分。模式理论认为遗传算法实质上是模式的运算，编码的字符越短，算法处理一代种群的隐含处理的模式就越多。遗传算法这种以计算少量编码适应度而处理大量模式的性质称为隐并行性。模式理论还指出，目标函数还满足积木块假设， 即阶数高、长度长、平均适应度高的模式可以由阶数低、长度短、平均适应度高的模式（积木块）在遗传算子的作用下结合而生成。而不满足积木块假说的优化问题被称为骗问题。模式理论为遗传算法构造了一条通过在种群中不断积累， 拼接积木块达到全局最优解的寻优之路。

由于模式理论的种种缺陷，研究者开始尝试利用有限状态马尔可夫链模型研究遗传算法的运行过程。对于遗传算法可以解决的优化问题的可行域都是由有限个点组成的，即使对于参数可以连续取值的问题，在实际中搜索空间也是特定精度要求的离散空间，因此遗传算法的实际运行过程可以用有限状态马尔可夫链的状态转移过程建模和描述。对于有 m 个可行解的目标函数和群体规模为 N 的遗传算法，实际优化问题的可行解数量 m 和群体规模 N 都十分可观，马尔可夫模型的状态数几乎为天文数字，因此利用精确的马尔可夫模型计算种群的状态分布是不可能的。

为了保证模型的可执行性，必须对实际模型采取近似简化，保持算法的实际形态，通过对目标函数建模，简化目标函数结构实现模型的可执行性。遗传算法优化的过程，可以看作算法在循环过程中不断对可行域进行随机抽样，利用前面抽样的结果对目标点的概率分布进行估计，然后根据估计出的分布推算下一次的抽样点。马尔可夫模型认为遗传算法是通过对搜索空间不同区域的抽样来估计不同区域的适应度，进而估计最优解存在于不同区域的概率，以调整算法对不同区域的抽样密度和搜索力度，进而不断提高对最优解估计的准确程度。可见，以邻域结构为依据划分等价类的马尔可夫模型更符合实际，对问题的抽象更能体现优化问题的本质。

4.2.3.2 模式的概念

模式（schema）指一些相似的模块，它描述了在某些基因位置上具有相似结构特征的个体字符串的一个子集。对二进制编码，把基于三值字符集(01*)所产生的能描述具有某些结构相似性的 0、1 字符串称为模式。模式(01*)可代表两个个体：{010，011}。

遗传算法的本质就是通过对这些模式进行一系列运算，将一些较差的模式逐步淘汰，而将一些较好的模式逐步被遗传和进化，最终得到问题的最优解。为了定量估计模式运算，引入了两个概念：模式的阶次和模式的定义距。

模式的阶次（order）是指模式中已有明确含义（二进制字符是指 0 或 1）的字符个数，记为 $O(H)$，式中 H 代表模式。例如模式(01*1**1)含有 4 个明确含义的字符，其阶次为 4，记作 O(01*1**1)=4。模式(1*****)的阶次是 1。很显然，阶次越低，模式的概括性越强，所

代表的字符串个数越多，反之亦然。当模式的阶次为 0 时，它没有明确含义的字符，其概括性最强。

模式 H 中第一个确定位的位置到最后一个确定位的位置之间的距离称为该模式的定义距（defining length），记为 $\delta(H)$。例如，模式(*01*1**1)的第一个确定位为 0，最后一个确定位为 1，中间有 5 个字符，其定义距为 6，记作 $\delta(*01*1**1)=6$，有一些特殊情况，如模式（1******）的定义距为 0。一般地定义距可表示为：

$$\delta(H)=b-a \tag{4-7}$$

式中，b——模式 H 中最后一个确定位的位置；a——模式 H 中第一个确定位的位置。

模式的定义距代表模式在今后遗传操作（交叉、变异）中被破坏的可能性。模式长度越短，被破坏可能性越小，定义距为 0 的模式最难破坏。

根据模式及阶次的定义，我们可以计算字符串所含有的模式数目。

1）模式总数

以二进制字符串为例，假设字符串的长度为 L，字符串中每一个字符可取{0，1，*}三个符号中任意一个，于是，能组成的模式数目为：

$$3\times3\times3\times\cdots\times3=(2+1)^L$$

一般情况下，假设字符串长度为 L，字符的取值为 k 种，则该字符串组成的模式数目 n_1 最多为：

$$n_1=(k+1)^L \tag{4-8}$$

2）已知阶次的模式所含字符串的总数

对于二进制字符串，若字符串的长度为 L，某种模式的阶次为 $O(H)$，则在 L 个字符串中取 $O(H)$ 个字符的可能组合方式为 $C_L^{O(H)}$，而在 $O(H)$ 的字符串中具体取 0 或 1 的可能性为 $2^{O(H)}$，于是，组成 H 的各种字符串数目最多为 $C_L^{O(H)}\times2^{O(H)}$。

一般情况下，假设字符串的长度为 L，字符的取值为 k 种，模式 H 的阶次为 $O(H)$，则组成 H 的数目 n_2 最多为：

$$n_2=C_L^{O(H)}\times k^{O(H)} \tag{4-9}$$

3）字符串所含模式总数

对于长度为L的某二进制字符串，它含有的模式总数最多为 $2\times2\times2\times\cdots\times2=2^L$ 个。注意，这个数目是指字符串已确定为 0 或 1，每个字符只能在定值（0/1）或*中选取。前面所述的 n_1 是指字符串未确定，每个字符可在{0，1，*}三者中选取。

一般情况下，长度为L、取值有k种的某一字符串，它可能含有的模式数目最多为：

$$n_3=k^L \tag{4-10}$$

模式概念的引入不仅仅是为了描述的方便，而是为了揭示遗传操作的一些基本特征和本质。显然当字符串的长度固定时，模式阶数越高，能与该模式匹配的字符串（即样本）数就越少，因而该模式的确定性也就越高。一个染色体实际上隐含着许多个模式，一个模式可以隐含在多个不同的染色体中，父代与子代染色体通过模式联系着。交叉和变异操作对染色体的作用实质上是对模式的作用。通过分析模式在遗传操作中的变化可以了解染色

体的性质，从而从理论上把握遗传算法的核心和本质。

4.2.3.3 遗传过程的模式数目

上述的字符串数目 n_1、n_2 和 n_3 是静态下的模式数目，下面我们从动态的角度，讨论遗传过程中的模式数目。

假设在进化过程中群体 A 中有 n 个个体，其中 m 个个体属于模式 H，则第 t 代时当前群体 $P(t)$ 能与模式 H 匹配的个体数为 $m(H,t)$，下一代群体 $P(t+1)$ 中能与模式 H 匹配的个体数记为 $m(H,t+1)$。下面分析基本遗传算法在选择、交叉和变异算子的作用下，模式 H 的样本数 $m(H,t)$ 的变化情况。

1）选择算子

在进行选择操作时，个体 a_i 是根据其适应度 f_i 的大小进行复制。从统计意义讲，个体 a_i 被复制的概率为：

$$p_i = f_i / \sum f_i$$

因此选择后在下一代 $P(t+1)$中，群体 A 内属于模式 H 的个体数目 $m(H,t+1)$ 可用平均适应度按下式近似计算：

$$m(H,t+1) = m(H,t) \cdot n \cdot f(H) / \sum f_i$$

式中，$f(H)$ ——第 t 代属于模式 H 的所有个体之平均适应度；n ——群体中拥有的个体数目。

假设第 t 代所有个体（不论它属于何种模式）的平均适应度是：

$$\bar{f} = \sum f_i / n$$

综合上述两式，选择后模式 H 所拥有的个体数目可按下式近似计算：

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}} \tag{4-11}$$

式（4-11）是模式计算的基本公式，它说明选择后下一代群体中属于模式 H 的个体数目，取决于该模式平均适应度 $f(H)$ 与群体的平均适应度 $\bar{f}$ 之比。换句话讲，只有当模式 H 的平均值 $f(H)$ 大于群体的平均值 $\bar{f}$ 时，H 模式的个体数目才能增长。否则 H 模式的个体数目要减少。模式 H 的这种增减规律，正好符合选择操作的“优胜劣汰”原则，这也说明模式的确能描述字符串的内部特征。

进一步，假设某一模式 H 在选择过程中其平均适应度 $f(H)$ 比群体的平均适应度 $\bar{f}$ 高出一个定值 $c\bar{f}$，其中 c 为常数，则式（4-11）改写为：

$$m(H,t+1) = m(H,t)\frac{\bar{f}+c\bar{f}}{\bar{f}} = (1+c)\cdot m(H,t)$$

从第一代开始，若模式 H 以常数 c 繁殖到第 t+1 代，其个体数目为：

$$m(H,t+1) = m(H,t)(1+c)^t \tag{4-12}$$

根据式（4-12），若 C>0，则 $m(H,t)$ 呈指数级增长；若 C<0，则 $m(H,t)$ 呈指数级减少。于是得出结论：在选择操作作用下，对于平均适应度高于群体平均适应度的模式，其个体数将呈指数级增长，而对于适应度低于群体平均适应度的模式，其个体数将呈指数级减少。

2）交叉算子

以单点交叉为例，假设有一模式 H，则隐含在该模式中的样本与其他个体进行单点交

叉操作时，根据交叉点的位置不同，有可能破坏该模式，也有可能不破坏该模式而使其继续生存到下一代群体中，显然当随机设置的交叉点在模式的定义距之内时，将有可能破坏该模式；而当随机设置的交叉点在模式的定义距之外时，肯定不会破坏该模式。假设交换位置用均匀分布的随机数确定，由于在最后一个字符后发生交换没有意义，所以模式 H 在长度为 L 的字符串中任意字符开始发生交换的概率为 $1/(L-1)$，而且交叉操作的概率为 p_c，则模式的生存概率为：

$$p_s = 1 - p_c \cdot \delta(H)/(L-1) \tag{4-13}$$

其中 L 为字符串的长度。这样，在选择算子和交叉算子的作用下，模式 H 的样本数满足下式：

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{L-1} \right] \tag{4-14}$$

由式（4-14）可知，在其他值固定的情况下(C>0)，$\delta(H)$ 越小，则 $m(H,t)$ 越容易呈指数级增长；$\delta(H)$ 越大，则 $m(H,t)$ 越不容易呈指数级增长。

3）变异算子

变异时字符串的每一个字符发生变化的概率是变异率 p_m，相反地，每个字符存活的概率为 $1-p_m$。在模式中有明确含义的字符有 $O(H)$ 个，于是模式 H 存活的概率为：

$$p_{ms} = (1 - p_m)^{O(H)}$$

通常，$p_m \ll 1$，上式用 Taylor 级数展开取一次项，可近似表达为：

$$p_{ms} = 1 - O(H) \cdot p_m \tag{4-15}$$

显然 $O(H)$越小，模式 H 越易于生存；$O(H)$越大，模式 H 越不易于生存。

综合式（4-14）和式（4-15），在选择、交叉和变异算子的连续作用下，群体中模式 H 的子代个体数为：

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{L-1} - O(H) p_m \right] \tag{4-16}$$

4.2.3.4 模式定理

由式（4-16）可得模式定理：具有低阶、短定义距以及平均适应度高于群体平均适应度的模式将在子代中以指数级增长。

模式定理阐述了遗传算法的理论基础，它提供了一种解释遗传算法机理的数学工具，蕴涵着发展编码策略和遗传操作的一些准则，保证了较优模式的样本数呈指数级增长，满足了寻找全局最优解的必要条件，从而给出了遗传算法的理论基础，它说明了模式的增加规律，同时也给遗传算法的应用提供了指导作用。然而，在推导模式定理时，Holland 仅考虑了遗传算子的破坏作用，并未考虑算子的构造作用，模式定理也不能解释遗传算法的早熟现象。

为了说明遗传算法中群体子代个体数的变化，我们就以一个简单的指数函数为例。

[例] 对于 $\max y = x^2, x \in [0.31]$ 这个函数，分析在遗传算子选择、交叉和变异的作用下

群体子代的变化情况。

这是一个简单的指数函数求极值的问题，利用代数方法就很容易求解。选用这样一个示例字符串比较短小，便于计算和观察。

在此示例中，自变量 x 的最大取值为 31，对应的用 5 位二进制就可以表达，所以利用 5 位二进制数组成个体。本例中，采用随机方法产生 4 个初始群体，然后通过复制完成选择操作，表 4-4 列举遗传算法中字符串和模式的变化情况。

表 4-4　字符串及模式的变化情况

字符串变化

序号	初始群体	x_i	适应度 $f(x_i)$	$\frac{f(x_i)}{\sum f(x_i)}$	$\frac{f(x_i)}{\bar{f}}$	实际数目	选择的新群体	交换对象	交换位置	新群体	x_i	适应度 $f(x_i)$
1	01101	13	169	0.14	0.58	1	01101	2#	3	01100	12	144
2	11000	24	576	0.49	1.97	2	11000	1#	3	11001	25	625
3	01000	8	64	0.06	0.22	0	11000	4#	2	11011	27	729
4	10011	19	351	0.31	1.23	1	10011	3#	2	10000	16	256
合计			1170	1.00	4.00	4.0						1754
平均值 $\bar{f}$			293	0.25	1.00	1.0						439
最大值			576	0.49	1.97	2.0						729

模式变化

选择前				选择后			选择及交叉后		
拥有的个体号			模式的平均适应度	预期数目	实际数目	拥有的个体号	预期数目	实际数目	拥有的个体
H_1	1****	2，4	469	3.20	3	2，3，4	3.20	3	2，3，4
H_2	*10**	2，3	320	2.18	2	2，3	1.64	2	2，3
H_3	1***0	3	576	1.97	2	2，3	0.0	1	4

表 4-4 的上半部（字符串变化）说明初始群体经过选择、交叉后变成新群体的过程，下半部（模式变换）表示三种模式 H_1、H_2、H_3 的变化过程。

模式 H_1 的定义距最小（$\delta(H_1)=0$），阶次最低（$O(H_1)=1$）。在初始群体中，H_1 代表 2 号及 4 号个体，其平均适应度 $f(H_1)=(576+361)/2=468.5$，高于初始群体适应度的平均值 $\bar{f}=293$。利用公式（4-12）可得选择后新群体中模式 H_1 所代表的个体数目：

$$m(H_1,t+1)=2\times 468.5/293=3.20$$

实际上复制后的新群体中，2、3、4 号三个个体都属于模式 H_1，与理论计算值一致。在交叉过程中，由于 $\delta(H_1)=0$，$p_c=1$，从公式（4-13）可以得出交换后 H_1 的存活概率为：

$$p_s=1-1\times 0/(5-1)=1$$

即交换操作对 $m(H_1,t+1)$ 没有影响。假设变异率 $p_m=0.004$，由公式（4-15）可计算出变异存活率为：

$$p_{ms}=1-1\times 0.004=0.996$$

H_1 基本上不会破坏。总之，模式 H_1 由于长度小、阶次低、平均适应度高，属于优良

模式，在遗传过程中不断增长。

对于模式 H_3，其定义距最大（$\delta(H_3)=4$），阶次 $O(H_3)=2$，在初始群体中它仅代表 2 号个体。不过它的平均适应度较高（$f(H_3)=576$），超过群体的平均适应度 $\bar{f}=293$，因此选择后拥有的个体数目要增加。利用公式（4-13）可计算存活率：

$$p_s=1-p_c\cdot\delta(H)/(L-1)=1-1\times4/4=0$$

再代入公式（4-16），当不考虑变异时有：

$$m(H_3,t+1)=1\times576/293\times0=0$$

表 4-4 中，H_3 在选择、交叉后的新群体中只代表 4 号个体。总之，模式 H_3 因适应度高在选择中增长，但由于长度大在交叉中衰减。

同理，模式 H_2 的定义距为 1、阶次为 2，在初始群体中代表 2 号、3 号个体。利用公式（4-11）得到：

$$m(H_2,t+1)=2\times320/293=2.18$$

表 4-4 中实际上 H_2 代表选择后新群体中的 2 号、3 号个体，与计算结果相符。利用公式（4-16），当不考虑变异时有：

$$m(H_2,t+1)=2\times320/293\times(1-1\times1/(5-1)=1.64$$

表 4-4 在选择和交叉后的新群体中 H_2 代表 2 号、3 号个体，与计算结果相近。

应该指出，表 4-4 中理论计算值与实际数目稍有误差的原因是由于群体中的个体数目太少。公式（4-11）至公式（4-16）都是从统计意义上推导出来的，一旦个体数目足够大，计算的精确度会明显提高。

4.2.3.5 积木块假设（Building Block Hypothesis）

Holland 在阐述模式定理时，将那些低阶、短定义距、平均适应度高于种群平均适应度的模式称为积木块，并指出可通过积木块重组构造更好的积木块，产生适应值更高的字符串，从而找到更优的可行解。

积木块假设是基于以下两个基本前提：

- 表现型相近的个体具有相近的基因型；
- 遗传算子间相对独立，相关性低。

积木块假设具体表述为：低阶、短距、高平均适应度的模式（积木块）在遗传算子的作用下，相互结合，能生成高阶、长距、高平均适应度的模式，可最终生成全局最优解。

模式定理说明了积木块的样本数呈指数级增长，亦即说明了用遗传算法寻找最优解的可能性，但它并未指明遗传算法一定能够找到最优解。而积木块假设却指出了遗传算法具备寻找最优解的能力，即积木块在遗传算子作用下能最终生成全局最优解。

积木块假设说明了遗传算法求解各类问题的基本思想，但遗憾的是上述结论并没有得到严格的数学证明，至今还没有一种方法可用来判别“对于一个给定的问题，积木块假设是否成立”，正因为如此它才被称为假设而非定理。目前大量的应用实例都表明，积木块假设在许多领域都获得了成功，尽管大量证据并不等于理论证明，但至少可以肯定，对常见问题遗传算法是适用和有效的。

为了形象地说明积木块的作用，我们仍用前述 $\max y=x^2, x\in[0.31]$ 为例，x 用二进制字符串表示。

图 4.9 表示模式 $H_1=1****$ 的分布。图中横坐标代表适应度 $f(x)=x^2$，纵坐标用千分数表示。图中弧线表示适应度曲线，网点区代表所有符合此模式的字符串集合。在此模式下，x 最小值的字符串为 10000（$x=16$），其适应度 $f(x)=x^2=256$；最大值的字符串为 11111（$x=31$），其适应度为 961。若初始群体中含有字符串 10000，则此模式沿此网点区逐步向字符串 11111 逼近。因此，模式 H_1 属于积木块，可产生性能优良的字符串。

图 4.9 的左半部空白区代表模式 $H_2=0****$ 的分布。在此模式下，x 最小值的字符串是 00000（$x=0$），其适应度为 0；最大值的字符串为 01111（$x=15$），其适应度 225。很明显，这种模式不能达到最优解，在后期要消亡。

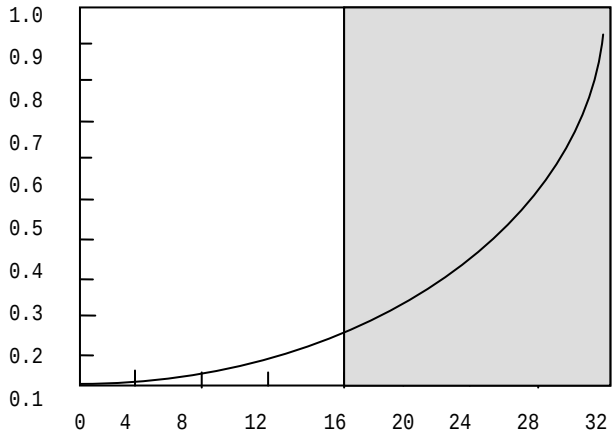


图 4.9　模式 1****的分布

图 4.10 是模式 $H_3=10***$ 的分布图。这时，字符串介于 10000（$x=16$）～10111（$x=24$），分布在中间的宽条带中，很明显，这种模式无法到达最优解。

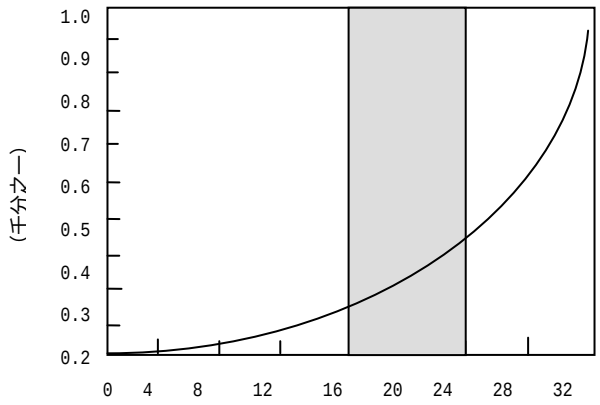


图 4.10　模式 10***的分布

比较模式 H_1 和 H_3，可以看出 H_1 比 H_3 阶次低，定义距短，适应度也高，因此，H_1 为积木块，而 H_3 不是。

4.2.3.6 隐含并行机理（implicit parallelism）

遗传算法的奠基人 Holland 在最早提出遗传算法的理论和模型时就阐述了它所包含的固有的并行性。其适应系统的逻辑理论是后来明确形成的遗传算法理论基础。这一机理表明：在遗传算法中尽管每一代只处理 n 个个体，但实际上却是处理 n^3 以上模式。下面我们将证明这个机理的存在。

从模式定理可以看出，模式在交叉和变异时可能遭破坏。由于变异概率很小，我们先考虑交叉的破坏。由公式（4-13）可知，对于字长为 L、模式定义距为 $\delta(H)$ 的模式 H ，当交叉概率为 p_c ， H 被破坏的概率 p_d 是：

$$p_d = 1 - p_s = p_c \frac{\delta(H)}{(L-1)}$$

即：

$$\frac{p_d}{p_c} = \frac{\delta(H)}{(L-1)} = \varepsilon$$

上式只考虑了交叉，若再兼顾变异的破坏作用，令 ε 为模式 H 被交叉和变异破坏的可能性，则上式可写为：

$$\varepsilon \cdot (L-1) = \delta(H) \tag{4-17}$$

式中，$\delta(H)$ 是指破坏概率为 ε 的模式长度。根据模式定义距的定义，模式不被破坏的最小字长 L_s 为：

$$L_s = \delta(H) + 1$$

代入式（4-17）有：

$$L_s - 1 < \varepsilon(L-1)$$

即：

$$L_s < \varepsilon(L-1) + 1$$

因此，模式存活的最小字长为 L_s 小于 $\varepsilon(L-1)+1$。对于拥有字长为 L_s 的字符串的模式数目又如何确定呢？我们先观察一个引例。

假设有一个字符串，长度 L=10：

1 0 1 1 1 0 0 0 1 0

若模式 H 的存活字长 $L_s = 5$，将它放在字符串的最左侧，则有：

1 0 1 1 1 0 0 0 1 0

写成模式的形式，上述字符串变为：

% % % % 1 * * * * *

%可在{0，1，*}三者中任取一个。也就是说，%可为固定值（0/1）或不固定值（*）两种情况。字符串中 1 表示有一个确定的模式，也可以选其他固定值表示。这时，可以组成的模式是 $2\times2\times2\times\cdots\times2 = 2^{L_s-1}$。

将上述字符串右移一位，有：

1 0 1 1 1 0 0 0 1 0

其模式表达式又可以组成 2^{L_s-1} 个模式。

上述字符串可发生在 6 个不同位置上，即发生次数为 $L - L_s + 1$。于是，长度为 L 的字

符串可组成字长为 L_s 的模式数目 n_{s1} 是：

$$n_{s1} = 2^{(L_s-1)} \times (L - L_s + 1)$$

当群体由 n 个字符串组成时，可能组成的字长为 L_s 的模式数目 n_{s2} 为：

$$n_{s2} = n \cdot n_{s1} = n \times 2^{(L_s-1)} \times (L - L_s + 1)$$

在选择（复制）中产生的个体完全相同。根据模式定理，它们都是一些不易被破坏的低阶模式。因此按上式计算 n_{s2} 显然偏大，我们不妨保守地取 $n = 2^{L_s/2}$。

另一方面，由于遗传操作都是利用平均随机数，模式数目服从二次分布，即模式中有一半的阶次高于 $L_s/2$，另一半小于 $L_s/2$。于是，计算时模式数目应取上述的 1/2。

综合上述各方面，可以存活的模式数目 n_s 为：

$$\begin{aligned} n_s &= n \times 2^{Ls-1} \times (L - L_s + 1) \times 1/2 \\ &= n \times n^2 \times 1/2 \times (L - L_s + 1) \times 1/2 \end{aligned}$$

即：

$$n_s = \frac{(L - L_s + 1)}{4} n^3 = Cn^3$$

上式说明，遗传算法中存活的模式数目 n_s 是群体中个体数目 n 的三次方。表面上遗传算法每一代只处理 n 个个体，但实质上却处理了 Cn^3 个能存活的模式。因此，遗传算法实际上是一种并行算法，隐藏在字符串的背后，故称隐并行算法。正是由于这种隐并行性，遗传算法的搜索效率才很高。

模式理论虽然在一定意义上解释了 SGA 的有效性，但还存在以下缺陷：

（1）仅适用于基于二进制编码的 SGA，对其他编码方式此定理不一定成立。

（2）模式理论仅适用于单峰值函数和线性空间，但事实上线性空间和单峰函数的优化问题可以非常方便地利用梯度算法解决。而需要利用 GA 搜索的空间几乎全部是非线性、多峰值函数，对于后者模式理论并不成立，显然用只适合于线性、单峰值函数的思想去解释非线性、多峰值函数中的现象是不严谨的。

（3）模式定理中仅提供了一个期望值的下界，且由于欺骗问题的广泛存在，利用模式理论并不能证明算法的全局收敛性。

（4）模式理论对算法参数的选取不能提供实用指导，对算法操作也有依赖性。

因此，引入 Markov 链来研究遗传算法的收敛性。

4.2.3.7 Markov 链的基本定义

定义 4.1（Markov 链）：，设 $X = X_k,\ \ k = 1,2,\cdots$ 是定义在概率空间上的离散参数随机过程，其状态空间 Ω 为有限集。如果 X 具有下列定义的 Markov 性（或无后效性），即对任意的非负数 k，及任意的状态，只要

$$P(X_0, X_1, \cdots, X_k) > 0$$

总有

$$P(X_{k+1} | X_0, X_1, \cdots, X_k) = P(X_{k+1} | X_k)$$

成立，则称 X 为有限 Markov 链，简称 Markov 链。

定义 4.2（转移概率）：X 在时刻 k 处于状态 i 的条件下，经 m 步转移，在时刻 $k+m$，到达状态 j 的条件概率称为 X 在时刻 k 的 m 步转移概率，记为 $p_{ij}(k,k+m)$。

$p_{ij}(k,k+m)$ 具有以下性质：

1） $p_{ij}(k,k+m)\geq 0$

2） $\sum_{i} p_{ij}(k,k+m)=1$

定义 4.3（齐次 Markov 链）：对于 Markov 链，如果

$$p_{ij}(m,n+1)=p\{X(m+1)=j \mid X(m)=i\}=p_{ij} \qquad (i,j\in I)$$

即从状态 i 出发转移到状态 j 的转移概率与时间起点 m 无关，则称这类为齐次 Markov 链。

定义 4.4 （随机矩阵）：对于齐次 Markov 链，称 p_{ij} 为一步转移概率，全部 $p_{ij}(i,j\in I)$ 所组成的一个矩阵 $P=(p_{ij})_{mn}$ 称为一步转移概率矩阵，或称为随机矩阵。

4.2.3.8 遗传算法的收敛性分析

简单遗传算法（SGA）可以被描述为一个齐次 Markov 链，因为 SGA 的选择、交叉和变异操作都是独立随机进行的，新群体仅与其父代及遗传算子有关，而与其父代群体之前的各代群体无关，即群体无后效性，并且各代群体之间的转换概率与时间的起点无关。

如果变异概率为 $p_m\in(0,1)$，交叉概率为 $p_c\in(0,1)$，同时按个体适应度占群体适应度的比例进行复制，则由矩阵的性质可知简单遗传算法的变换矩阵 P 是正定的。同时可以证明简单遗传算法并不能收敛到全局最优值。

[证明] 将群体的各种可能状态 I 分为包括最优个体的状态 I_0 和不包括最优个体的状态 I_n，它们具有如下关系：

$$I=I_0\bigcup I_n \qquad (I_0\bigcap I_n=\Phi)$$

用反证法。假设 SGA 能收敛于最优解的概率等于 P，则进入 I_n 状态的稳定概率等于 0，即：$\lim P\{p_t\in I_n\}=0$

在 SGA 的进化过程中，群体从某一状态 $i\in I$，经过选择、交叉和变异算子的连续作用而转变为状态 $j\in I$，这三种遗传算子的转移概率分别为 s_{ij}，c_{ij}，m_{ij}，它们可分别构成相应的随机矩阵 $S=\{s_{ij}\}$，$C=\{c_{ij}\}$，$M=\{m_{ij}\}$，则遗传算法的群体状态变换矩阵为：$R=SCM=\{r_{ij}\}$。

由于 S、C、M 都是随机矩阵，并且 $m_{ij}=P_m^{H(i,j)}(1-p_m)^{1-H(i,j)}>0$（$H_{ij}$ 为状态 i 和 j 之间的海明距离（两个字符串对应位上编码不同的位数称为海明距离）），容易证得 $r_{ij}>0$，即 R 是正定的。

在第 t 时刻，群体是状态 j 的概率 $P_j(t)$ 为：

$$P_j(t)=\sum_{i\in I} p_i(0)r_{ij}^{\,t} \qquad (t=0,1,2,\cdots)$$

由齐次 Markov 链的性质可知，$P_j(t)$ 的稳定概率分布与其初始分布无关，即：

$$P_j(\infty)=\sum_{i\in I} p_i(\infty)r_{ij}>0$$

注意上式中的状态 $j\in I$，即 j 也可能是 I_n 中的一个状态，从而得知：

$$\lim P\{p_j \in I_n\} > 0$$

上式与式（4-17）的假设相矛盾，因此，SGA 只能以小于 1 的概率收敛于最优解，其应用可靠性就值得怀疑。

以上是从概率的意义说明标准遗传算法不能收敛到全局最优解，但是只要对它作一些改动，在选择操作中不按比例进行，而是采用最佳个体保留策略（elitist strategies）的遗传算法（EGA），保留当前个体中的最优值，按交叉、变异和种群选择之后更新当前最优解的循环过程，则遗传算法最终能收敛到全局最优解。

考察个体集合 $P'(t) = \big(A(t), P(t)\big)$，其中 $A(t)$是当前群体中适应度最高的个体，状态转移矩阵 R 由 $P(t)$ 产生 $P(t+1)$，从上一代群体中和本代群体中挑出的一个具有最大适应度的个体 $A(t+1)$，即：

$$A(t+1) = \max\{A(t), A(0)\}$$

其中，$A(0)$是群体 $P(t+1)$中适应度最高的个体。这样所构造出来的随机过程 $\{P'(t), t \geq 0\}$仍然是一个齐次 Markov 链，即有：

$$P'(t) = P'(0)(R')^t$$

假设个体集合状态中包括有最优解的状态为 I_0，则该随机过程状态转移概率为：

$$r_{ij}' > 0 \qquad (\forall i \in I, \forall j \in I_0)$$

$$r_{ij}' = 0 \qquad (\forall i \in I, \forall j \notin I_0)$$

即从任意状态向含有最优解的状态转移概率大于 0，而从含有最优解的状态向不含有最优解的状态转移概率等于 0。

此时，对于 $\forall i \in I, \forall j \notin I_0$，下式都成立：

$$(r')_{ij}^t \to 0 \qquad (t \to \infty) \tag{4-18}$$

$$P_j'(\infty) = 0 \qquad (j \notin I_0) \tag{4-19}$$

由式（4-18）和式（4-19）可以看出：个体集合收敛于不含有最优解的状态转移概率为 0，或者说算法总能以概率为 1 找到最优解。这个结论除了理论上具有重要的意义之外，在实际应用中也为最优解的搜索过程提供了一种保证。

4.2.4 遗传算法的改进技术

从 20 世纪 80 年代以来，遗传算法得到了广泛的应用，在实践过程中，人们对遗传算法的实施提出了许多改进方法，本节分别予以介绍。

4.2.4.1 编码

编码是遗传算法要解决的首要问题。SGA 的编码方法是二进制编码，但对于许多遗传算法的应用，特别是在工业工程中的应用，这种简单的编码方法很难直接描述问题的性质。近十年来，针对特殊问题，人们提出了其他编码方法。

1）二进制编码

它是遗传算法中最常用的一种编码方法。它具有下列一些优点：

- 编码、解码操作简单易行；

- 交叉、变异操作便于实现；
- 符合最小字符集编码原则；
- 便于利用模式定理对算法进行理论分析。

对于长度为 L 的二进制字符串，它可以表达的个体数目为 2^L 个，从表达的模式数目看，二进制字符除了 0、1 外，还有*字符，所以它表达的模式数目为：

$$m_1 = (2+1)^L = 2^L(1+\frac{1}{2})^L \tag{4-20}$$

为了形象地说明二进制的优越性，表 4-5 列举了十进制、二进制及英文字母的字符串的比较。从表中可以明显看出，二进制字符串所表达的模式多于十进制及英文字母，在执行交叉和变异时可以有更多的变化。

表 4-5 英文字母、十进制及二进制的字符串的比较

英文字母	A	B	C	D	E	F	G
十进制	1	2	3	4	5	6	7
二进制	00001	00010	00011	00100	00101	00110	00111
英文字母	H	I	J	K	…	Y	Z
十进制	8	9	10	11	…	25	26
二进制	01000	01001	01010	01011	…	11001	11010

2）格雷码

对于一些连续优化问题，二进制编码由于遗传算法的随机特性而使其局部搜索能力较差。为改进这一特性，人们提出用格雷码进行编码。格雷码编码方法是二进制编码方法的一种变形。

由标准二进制码 a_i 转化到格雷码 b_i 的公式为：

$$b_i = \begin{cases} a_i & \text{若} i = 1 \\ a_{i-1} \oplus a_i & \text{若} i > 1 \end{cases}$$

由格雷码 b_j 转换到标准二进制码 a_i 的公式为：

$$a_i = \overset{i}{\underset{j=1}{\oplus}} b_j$$

式中，$\oplus$ 表示以 2 为模的加运算。

表 4-6 比较了格雷码与十进制以及二进制的关系。以 5 对应的二进制 0101 为例，格雷码第一位不变为 0，第二位 $0 \oplus 1 = 1$，第三位 $1 \oplus 0 = 1$，第四位 $0 \oplus 1 = 1$，所以对应的格雷码为 0111。从表中可以看出，连续的两个整数所对应的格雷码编码值之间仅仅只有一个码位是不相同的，其余码位都完全相同。通常，相邻两个二进制字符串中字符不同的数目称作海明距离。格雷码的海明距离总为 1。这样，在进行变异操作时，格雷码某个字符的变异很有可能是字符串变为相邻的另一个字符串，从而实现顺序搜索，避免无规则的跳跃式搜索。因此，格雷码除了具有二进制编码的优点外，还能提高遗传算法的局部搜索能力。

表 4-6　四位格雷码

十进制	0	1	2	3	4	5	6
二进制	0000	0001	0010	0011	0100	0101	0110
格雷码	0000	0001	0011	0010	0110	0111	0101
十进制	7	8	9	10	11	12	13
二进制	0111	1000	1001	1010	1011	1100	1101
格雷码	0100	1100	1101	1111	1110	1010	1011

3）实数编码

对于一些多维、高精度要求的连续函数优化问题，使用二进制编码来表示个体将会带来一些不利，例如，二进制编码存在着连续函数离散化时的映射误差，同时不便于反映所求问题的特定知识。为了克服这些缺点，人们提出用十进制浮点数（实数）编码来实现遗传算法程序。相比之下，采用实数编码不仅无需转换数值和数据类型使得优化过程更容易理解，而且节省遗传操作时间。另外由于浮点数实数范围大且表示精度高、具有明确的物理意义，对于优化结果有益。概括而言，实数编码方法的优点如下：

- 适合于遗传算法中表示范围较大的数；
- 便于较大空间的遗传搜索；
- 提高了遗传算法的精度要求；
- 改善了遗传算法的计算复杂性，提高了运算效率；
- 便于算法与经典优化方法的混合作用；
- 便于设计专门问题的遗传算子。

4）符号编码方法

是指染色体编码串中的基因值取自一个无数值含义、而只有代码含义的符号集。这些符号可以是字符，也可以是数字。例如，对于旅行商问题，假设有 n 个城市分别记为 C_1，C_2，…，C_n，则［C_1，C_2，…，C_n］就可构成一个表示旅行路线的个体。符号编码的主要优点是便于在遗传算法中利用所求问题的专门知识及相关算法。

4.2.4.2 适应度

首先是遗传算法运行初期阶段，群体中或许会出现少数适应度极好的个体，最终这些个体可能会充斥整个群体，使用于产生新个体作用较大的交叉操作失去作用，从而使得群体多样性降低，遗传算法提前收敛到某个局部最优解。因此，适应度函数的选取应尽量地避免早熟现象，即降低适应度较高的个体与其他个体适应度之间的差异，限制其复制数量以维护群体多样性。

其次是运行后期阶段，群体越来越集中，个体之间的差异减小，相互之间的竞争力也随即减弱。这必然造成个体被选择到下一代中的概率接近，使进化过程失去竞争力，退化为随机选择过程。因此，适应度函数的选取也应该克服这种退化现象，使算法在运行后期阶段能够扩大最佳个体适应度与其他个体适应度之间的差异，提高个体之间的竞争性。

1）适应度的缩放

为了更好地优胜劣汰，有时需要适应度放大或缩小，突出个体的差异，可用下式变化

适应度：

$$f' = af + b \tag{4-21}$$

式中，f'——缩放后的适应度；f——原来的适应度；a、b——系数。

图 4.11 描述适应度缩放原理。图中的缩放斜线采用两点连线的方法确定：一个点是适应度的平均点 A。从统计意义上讲，缩放前后的平均适应度应该相同，即 A 点处有：

$$f_{\text{avg}} = f'_{\text{avg}}$$

式中，f_{avg}——原来的适应度平均值；f'_{avg}——缩放后的适应度平均值。

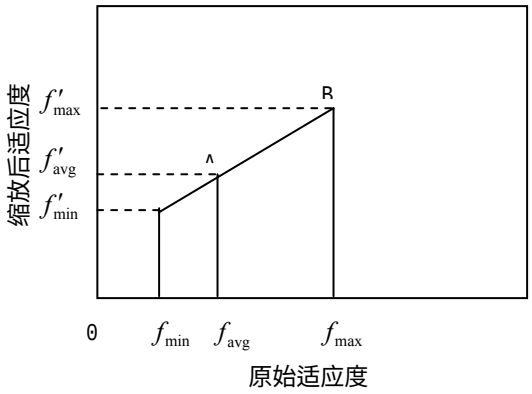


图 4.11　适应度缩放原理

连线的另一点 B 是适应度最大值点，可用下式确定：

$$f'_{\max} = C \cdot f_{\text{avg}}$$

式中，$f'_{\max}$——缩放后的最大适应度；C——系数。当群体中个体数目为 50～100 时，常取 C=1.2～2.0。

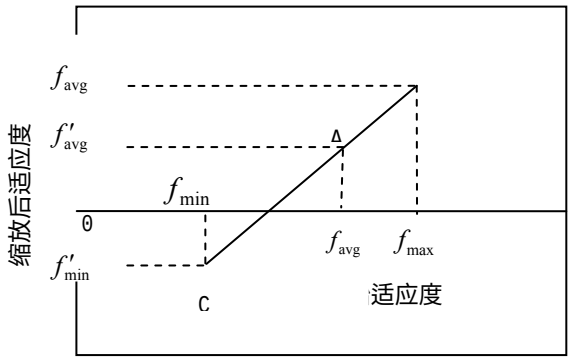


图 4.12　负值点的映射

在遗传算法后期，个别劣质个体的适应度远远小于群体平均适应度及最大适应度，并且后两者比较接近，这样按上述方法缩放会使低适应度变成负值，如图 4.12 所示的 C 点。为此，需要修正上述的两点连线法。平均点 A 的位置仍然同前，即：

$$f'_{\min} = 0$$

综上所述，缩放斜线式（4-21）的参数 a、b 计算方法如下：

（1）计算适应度非负判别式：

$$f_{\min} > \frac{C \cdot f_{\text{avg}} - f_{\max}}{C-1} \tag{4-22}$$

若不等式满足，执行（2），否则执行（3）。

（2）正常情况下的缩放：

$$a = \frac{(C-1)}{f_{\max} - f_{\text{avg}}} f_{\text{avg}} \tag{4-23}$$

$$b = \frac{f_{\max} - C \cdot f_{\text{avg}}}{f_{\max} - f_{\text{avg}}} f_{\text{avg}} \tag{4-24}$$

（3）负适应度时的缩放

$$a = \frac{f_{\text{avg}}}{f_{\text{avg}} - f_{\min}} \tag{4-25}$$

$$b = -\frac{f_{\min} \cdot f_{\text{avg}}}{f_{\text{avg}} - f_{\min}} \tag{4-26}$$

调整适应度的另一种方法是方差缩放技术，它根据适应度的离散情况进行缩放。对于适应度离散的群体，调整量要大一些。反之，调整量减少。具体调整方法如下：

$$f' = f + (\bar{f} - C\sigma) \tag{4-27}$$

式中，$\bar{f}$——适应度的平均值；σ——群体适应度的标准差；C——系数，介于 1～5 间的整数。

若计算出的 f' 为负值，令其为 0。

再者，有人还提出一种指数缩放技术，即：

$$f' = f^k \tag{4-28}$$

式中，k——系数，在 1 附近。

上述适应度的各种方法，其目的都是修改各个体性能的差距，以便体现“优胜劣汰”的原则。如果我们想多选择一些优良的个体进入下一代，则尽力加大适应度之间的差距。

2）适应度的缓存

将每代种群中适应度高的个体进行缓冲，以期能延长这些个体参与遗传操作的代数，进而改善算法运行过程中的收敛特性。模式定理认为具有短的定义距、低阶并且适应度在群体平均适应度以上的模式，在遗传算法迭代过程中将按指数增长率被采样。这种改进策略是通过增加缓冲每代种群适应度高的个体，在遗传过程中被缓冲的个体参与到下一代的遗传操作。由于被缓冲的个体具有较高的适应度，所以也将具有较高的被采样率，提供更多的优良个体，从而延缓算法在吸收性作用下过早收敛的情况发生，利于算法进一步向更优解收敛。

缓冲区是由最近若干代群体中最优个体构成的有限集合，设该集合能容纳的个体数量最大为 N，在算法执行开始时为空。并定义下列函数：

Number(G)：求集合 G 中元素个数的操作；

Repeated(G,Individual)：求集合 G 中指定个体 Individual 重复出现数量操作；

GetMinimum：求集合 G 中适应度最小元素操作。

对每代的最优个体 BestIndividual 按照如下规则进行缓冲：

（1）加入规则：若（Number(G)<N），则 G=G∪{BestIndividual}；

（2）替换规则：若（Number(G)=N）且（GetMinimum(G)的适应度<BestIndividual 的适应度），则 G=(G-{ GetMinimum(G)})∪{BestIndividual}。

当算法处于收敛阶段时，相邻若干代群体中的最优个体重复度比较高，在这个阶段是否缓冲这些最优个体采用以下规则：

（1）无限制规则：G=(G-{ GetMinimum(G)})∪{BestIndividual}。这是由于当算法处于收敛阶段，加入的新个体适应度大于或等于集合 G 中的所有元素。在不考虑遗传代数限制的情况下，采用该规则的算法收敛后期集合中的全部个体相同。也可作为算法参考结束条件，即 Repeated(G，BestIndividual)=N。

（2）限制规则：允许确定数量 k 的重复个体加入库中，即

If

Repeated(G，BestIndividual)<k 且 GetMinimum(G)的适应度< BestIndividual 的适应度

Then

G=(G-{ GetMinimum(G)})∪{BestIndividual}

缓冲区中的个体参与遗传操作的时机选择有如下几种方案：一种方案是在每代遗传操作中引用部分或全部缓冲区中的个体；另外一种方案是，采用固定遗传代数间隔 m，即对遗传代数进行计数，每当遗传代数计数器为 m 的整倍数时，将缓冲区中的部分或全部个体引入到下一次的遗传操作。缓冲区中的个体参与下一代遗传操作的方式主要有以下几种：

- 随机替换当前种群中的除最优个体以外的其他个体；
- 替换适应度最低的若干个体；
- 遗传操作中的全部父代个体均从缓冲区中选取。

改进后的遗传算法可表示为：

（1）GA（）初始化；

（2）缓冲区初始化；

（3）结束条件是否满足，若满足转（8）；

（4）GA（）；

（5）如果 Repeated(G，BestIndividual)<k 且 GetMinimum(G)·Fitness< BestIndividual·Fitness，则 G=(G-{ GetMinimum(G)})∪{BestIndividual}；

（6）从缓冲区选取若干个体替换加入到种群；

（7）转（3）；

（8）输出求得的最优解。

4.2.4.3 选择

遗传算法使用选择运算（或称复制运算）来实现对群体中的个体进行优胜劣汰操作：适应度高的个体被遗传到下一代群体中的概率大；适应度低的个体，被遗传到下一代群体中的概率小。选择操作的任务就是按某种方法从父代群体中选取一些个体，遗传到下一代群体。除了 4.2.1.4 所介绍的赌轮选择法以外，下面介绍其他的几种选择方法。

1）随机联赛选择（stochastic tournament model）

每次选取几个个体之中适应度最高的一个个体遗传到下一代群体中，优点是对个体适应度取正值、负值无要求。可知此方法随机性更强，存在更大的随机误差，但是有较大概

率保证最优个体被选择，最差的个体被淘汰。我们称每次进行适应度大小比较的个体数目为联赛规模 N，一般情况下取 N=2。过程如下：

（1）从群体中随机选择 N 个个体进行适应度大小比较，将其中适应度最高的个体遗传给下一代；

（2）重复上述过程 M 次，就可得到下一代群体中 M 个个体（其中 M 为种群大小）。

2）期望值选择方法（expected value model）

期望值选择方法也叫无回放随机选择，根据每个个体在下一代群体中的生存期望值来进行随机选择运算，步骤如下：

（1）计算每个个体的生存期望数目：

$$n_s = M \cdot \frac{f_i}{\sum_{i=1}^{M} f_i} \tag{4-29}$$

其中，M——群体中个体的数目；f_i——i 个体的适应度。

（2）若某一个体被选中参与交叉运算，则生存期望数目减去 0.5，若某一个体未被选中参与交叉运算，则它在下一代中的生存期望数目减去 1.0。

（3）随着选择过程的进行，若某一个体的生存期望数目小于 0，则该个体就不再有机会被选中。

这种方法能够较大概率保证最优个体别选择，但是操作较复杂。

3）确定式采样选择（deterministic sampling）

针对最优个体保留的不确定性，此方法是按照一种确定的方式来进行选择操作，这种选择方法可以保证适应度较大的一些个体一定能够被保留在下一代群体中，并且操作也比较简单，具体过程如下：

（1）计算个体在下一代中的生存期望数目 $n_s = M \cdot f_i / \sum_{i=1}^{M} f_i$ ；

（2）确定各个对应个体在下一代群体中的确定生存数目 $\lfloor n_s \rfloor$，故可确定下一代群体中个体为 $\sum_{i=1}^{M} \lfloor n_s \rfloor$ 个（其中 $\lfloor n_s \rfloor$ 是对 n_s 的下取整）；按照 n_s 的小数部分对个体进行降序排序，顺序取前 $M - \sum_{i=1}^{M} \lfloor n_s \rfloor$ 个个体加入下一代。

4）无回放余数随机选择（remainder stochastic sampling with replacement）

这是对确定式采样选择的改进，与其区别是余数的新适应度的随机选择方案，步骤如下：

计算生存期望数目 $n_s = M \cdot f_i / \sum_{i=1}^{M} f_i$ ；

取 n_s 的整数部分 $\lfloor n_s \rfloor$ 为对应个体在下一代群体中的生存数目，这样共可确定下一代群体中的 $\sum_{i=1}^{M} \lfloor n_s \rfloor$ 个个体；

以 $f_i - \lfloor n_s \rfloor \cdot \sum_{i=1}^{M} f_i / M$ 为各个个体的新的适应度，用赌轮选择等随机选择性的方法来随机确定下一代群体中还未确定的 $M - \sum_{i=1}^{M} \lfloor n_s \rfloor$ 个个体。

这四种选择算子是遗传算法中最基本的选择算子，它们各有特点，针对不同的具体问题各有优劣，又可针对实际问题将以上几种方法结合处理，这样会产生更好的效果。

在这四种基本算法选择算子的基础上，人们又提出了一些新的选择策略，主要有以下三种：

1）基于上限的确定式采样（deterministic sampling based on upper-limit）

此方法基本思想是按一种确定的方式来进行选择操作，是对确定式采样选择的一种改进，具体过程如下：

计算个体在下一代中的期望生存数目 $n_s = M \cdot f_i / \sum_{i=1}^{M} f_i$ ；

确定各个对应个体在下一代群体中的准生存数目 $\lceil n_s \rceil$，故可确定下一代群体中个体为 $\sum_{i=1}^{M} \lceil n_s \rceil$ 个；

从 $\sum_{i=1}^{M} \lceil n_s \rceil$ 个个体中删除适应度最低的 $\sum_{i=1}^{M} \lceil n_s \rceil - M$ 个个体，故可以得到 M 个新选择的个体。

其中，M 为群体规模，f_i 为第 i 个个体的适应度，$\lceil n_s \rceil$ 是对 n_s 向上取整。

这种选择方法可以保证适应度较大的一些个体一定能够被保留在下一代群体中，使适应度较小的一些个体一定能被淘汰掉，并且操作也比较简单。

2）基于切断的赌轮选择法（roulette wheel selection based on cutting）

此方法基本思想是先用赌轮选择的方式来单独选择某个最优个体，但是在选定某个个体后，则相应地减小该个体比例的大小，此后重复以上操作，直至产生了所有的个体。该方法是对赌轮选择法的一种改进，具体过程如下：

根据适应度计算所有个体的所占比例值，即计算 $p_i = f_i / \sum_{i=1}^{M} f_i$ ，假设种群的大小为 M。

（1）计算比例 $p_s = p_i / \sum_{i=1}^{M} p_i (i = 1,2,\cdots,M)$，根据比例计算所有个体的所占比例值；

（2）根据以上比例，用赌轮选择法选择一个个体 $t(t \in N$ 且 $t \in [1, M])$，则：

$$p_s = \begin{cases} 0 & \text{若} p_s - \dfrac{1}{M} \le 0 \\ p_s - \dfrac{1}{M} & \text{若} p_s > \dfrac{1}{M} \end{cases} \tag{4-30}$$

（3）若选出的个体数达到种群大小，则转（4），否则转（1）；

（4）存储所有新选出的个体，并且返回。

3）无回放最大值选择法（biggest value selection with replacement）

由于赌轮选择法的不确定性，故通过对基于切断的赌轮选择法的改进，将其中的赌轮选择改为直接选取最优个体，则保证一定选到最好个体。此方法基本思想是基于通过选择最大比例来单独选择当前最优个体。但是在选定某个个体后，相应地减少个体比例的大小，此后重复以上操作，直至产生了 M 个个体为止，假设种群大小为 M，具体过程如下：

（1）计算比例 $p_s = f_i / \sum_{i=1}^{M} f_i (i = 1,2,\cdots,M)$，根据适应度计算所有个体的所占比例值；

（2）选择最大比例的一个个体 $t(t \in N$ 且 $t \in [1, M])$，然后判断 p_s，得：

$$p_s=\begin{cases}0 & 若 p_s-\dfrac{1}{M}\le 0\\ p_s-\dfrac{1}{M} & 若 p_s>\dfrac{1}{M}\end{cases}$$

（3）若选出的新个体数目达到种群大小 M，则转（4），否则转（2）；

（4）存储所有新选出的个体，并且返回。

显然这种方法不仅能够保证最优个体能够被选中，而且操作简单。

4.2.4.4 交叉

遗传算法中，在交叉运算之前还必须对群体中的个体进行配对，目前常用的配对策略是随机配对。交叉算子的设计包括两个方面的内容：①如何确定交叉点的位置？②如何进行部分基因的交换？下面介绍几种适用于二进制编码或实数编码的交叉算子。

1）单点交叉

单点交叉又称为简单交叉，它是指在个体字符串中随机设置一个交叉点，然后在该点相互交换两个配对个体的部分字符。考虑父代个体 $p_1=(a_{L-1}a_{L-2}\cdots a_0)$ 是固定的，另外一个父体 $p_2=(b_{L-1}b_{L-2}\cdots b_0)$ 是均匀分布随机选取的情况，使用单点交叉算子生成子代个体 $(a_{L-1}a_{L-2}\cdots a_k b_{k-1}\cdots b_0)$，即 p_1 的 0～k-1 字符位被随机取代，由于父代均匀分布，因此对每一字符位 b_i 来分析，有：

$$P(b_i)=\begin{cases}1/2 & b_i=a_i\\ 1/2 & b_i=\overline{a}_i\end{cases}$$

2）双点交叉

它的具体操作过程是：(1) 在相互配对的两个个体编码串中随机设置两个交叉点；(2) 交换两个交叉点之间的部分基因。当 $p_1=(a_{L-1}a_{L-2}\cdots a_0)$ 是固定的，另外一个父体 $p_2=(b_{L-1}b_{L-2}\cdots b_0)$ 随机均匀选取时，使用双点均匀交叉生成的子代个体 $C_1=(a_{L-1}a_{L-2}\cdots a_k b_{k-1}\cdots b_j a_{j-1}\cdots a_0)$，两交叉点的距离为 $k-j$。

3）均匀交叉

它是指两个配对个体的每一位基因都以相同的概率进行交换，从而形成两个新个体。具体操作过程如下：

（1）随机产生一个与个体字符串长度相同的二进制随机数 $W=w_{L-1}w_{L-2}\cdots w_0$；

（2）按下列规则从 A、B 两个父代个体中产生两个新个体 X、Y：若 $w_i=0$，则 X 的第 i 个字符继承 A 的对应字符，Y 的第 i 个字符继承 B 的对应字符；若 $w_i=1$，则 A、B 的第 i 个字符相互交换，从而生成 X、Y 的第 i 个字符，见表 4-7 所示。

表 4-7　均匀交叉

父代	A：	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
	B：	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
随机数		0	1	0	1	1	0	0	1	0	1
子代	X：	a_0	b_1	a_2	b_3	b_4	a_5	a_6	b_7	a_8	b_9
	Y：	b_0	a_1	b_2	a_3	a_4	b_5	b_6	a_7	b_8	a_9

4）算术交叉

它是指由两个个体的线性组合而产生出新的个体。设在两个个体 A、B 之间进行算术交叉，则交叉运算后生成的两个新个体 X、Y 为：

$$\begin{cases} X = \alpha A + (1-\alpha)B \\ Y = \alpha B + (1-\alpha)A \end{cases} \tag{4-31}$$

4.2.4.5 变异

除了在 4.2.1.6 介绍的传统变异算法外，人们在变异方面还做了许多深入的研究。

1）变异概率取决于字符串长度及种群规模。字符串长度 L 增大后，左侧字符所代表的数值远比右侧大，而字符串中每个字符产生变异的概率相同，一旦变异发生在左侧字符，会使新个体偏离旧个体太远；特别是在遗传算法的后期，会使结果无法收敛。因此，若字符串长度大时，变异概率 P_m 宜取小值。

变异概率还与群体规模 M 有关。当群体所含个体数目较多时，个体间的适应度差别会增大。为了使群体形态不致于过高分散，也希望变异概率随群体规模的增大而减小。

针对上述两种因素的考虑，可将式（4-5）进行一定的处理，变异概率为：

$$p_m = \frac{1.75}{M \cdot \sqrt{L}} \tag{4-32}$$

式中，M——群体中拥有的个体数目；L——个体中字符串的长度。

2）变异概率随时间而变化。尽管交叉和变异都能产生新个体，然而前者的新个体是原有字符的重新组合，后者则添加崭新的字符。特别是临近遗传算法的后期，变异会使结果无法收敛。因此，在遗传算法的各个阶段，变异概率应该发生变动，早期的变异概率大一些，后期的变异概率小一些。具体数值可参考下式选取：

$$p_m(t) = \sqrt{\frac{C_1}{C_2}} \frac{\exp(-C_3 \cdot t/2)}{M \cdot \sqrt{L}} \tag{4-33}$$

式中，$p_m(t)$—— 第 t 遗传代次时的变异概率；M—— 群体规模；L—— 个体字符串长度；C_1、C_2、C_3—— 系数。

3）变异概率是一个敏感参数。由于变异会产生新个体，而新个体的形态可能会变得更好，也会变得更坏，因此变异的效果很难预料。通过人们多次试验，人们发现变异概率对计算结果影响很大，因此，选取时要慎重。

遗传算法的研究归纳起来分为理论与技术研究、应用研究两个方面。理论与技术研究主要从遗传操作、群体大小、参数控制、适应度评价以及并行实现技术等方面来提高遗传算法的性能。应用研究则是遗传算法的主要方向，开发遗传算法的商业软件、开拓更广泛的遗传算法应用领域是今后应用研究的主要任务。

4.3 遗传规划

在遗传算法的发展历史中，将线性编码改进为非线性编码，是近年来提出的一种新的思路。1992 年 Stanford 大学的 J.R.Koza 首先提出了一种在编码方式上与常规遗传算法有着本质不同的仿生进化算法，它采用了层式编码结构，由于在遗传算法的基础上，改进了编码方法，他称之为“ 遗传规划 ”（genetic programming，简称 GP），同遗传算法一样，它们

都属于新兴的进化计算学科；但相对于遗传算法，遗传规划更适合应用于层式结构的优化。

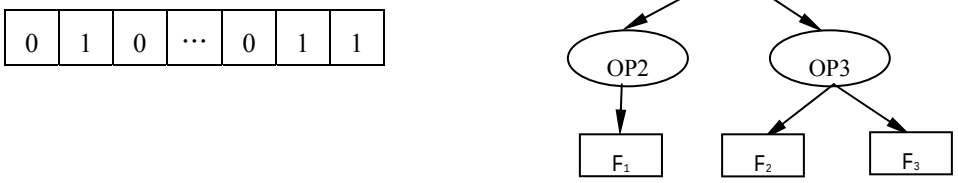
遗传规划 GP 是由遗传算法发展、延伸而来的，传统的遗传算法是用定长的线性字符串表达问题，而工程中许多复杂问题往往不能用简单的字符串表达所有的性质，因此有必要对传统的遗传算法进行改进。遗传规划就是在此背景下产生的，它与遗传算法最大的不同是以层次结构（“ 树 ”型）表达问题，而且其结构与大小都是动态自适应调整；因此，遗传规划更适于表达复杂的结构问题。遗传规划的任务就是从由许多树型可行解组成的搜索空间中寻找出一个具有最佳适应度的“ 树 ”。遗传规划提供了一套寻找具有最好适应度的“ 树 ”的方法。目前，遗传规划的研究已经渗透到工程技术科学、生命科学及社会科学的各个领域中。

4.3.1 遗传规划的基本技术

遗传规划采用层次结构可变的形式表达问题，整个过程与遗传算法比较类似，只是在编码、初始群体及遗传算子的构造方面有所不同。

1）编码

前文已经提到，遗传规划采用层式的编码形式。例如，遗传规划用于特征构造时，代表由简单特征组合而成的复合特征如图 4.13 中的（b）所示，（a）是遗传算法的个体，从（a）、（b）两图可看出二者编码的不同。图中，$OP_i(i=1,2,\cdots,S)$ 表示 S 个运算符（单目或双目），$F_j(j=1,2,\cdots T)$ 表示 T 个初始特征。遗传规划的个体就是通过运算符和初始特征的不同树形组合来表达。



（a）遗传算法的个体　　（b）遗传规划的个体

图 4.13　遗传算法与遗传规划的编码结构

2）构造初始群体

遗传规划初始群体的构造比遗传算法复杂得多，主要是由于层式树形个体的构造比线性基因串的构造复杂。在构造初始群体以前，要做好以下两方面的工作：

（1）终止符集的构造。

终止符是指如图 4-13 中所示的树型结构末端的子节点 $F_j(j=1,2,\cdots T)$ 。终止符集根据求解对象的不同而不同，通常由常量、变量、符号等组成。

（2）运算符集的构造。

运算符 $OP_i(i=1,2,\cdots,S)$ 代表了一个或多个终止符之间执行的操作。运算符集的构造也因具体问题而异。常见的运算符主要有：

- 算术运算符，如 ＋，－，×，÷等；
- 标准数学函数，如 sin，cos，exp，log 等；
- 布尔运算符，如 and，or，not 等；
- 条件运算符，如 if-then-else，case 等。

以上列举的都是计算机程序中经常使用的运算符。另外，也可以根据适用的问题自行设计面向对象的运算符。终止符与运算符皆称为遗传规划的元素。

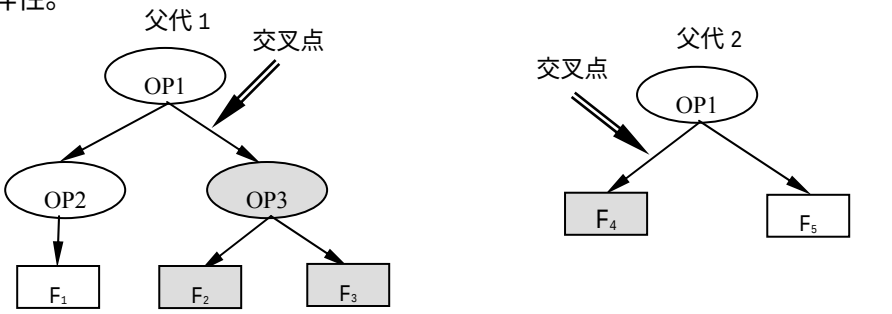
有了以上两步准备，就可以进行初始群体的构造了。首先从运算符集中随机选择一个运算符，没有从终止符集选择是为了避免产生空的个体；然后根据运算符的目数，确定从该运算符引出的线数；其次，依次在每条线的终端，加入随机选出的元素；最后，重复上述过程，直至“生长”为满足要求的个体。实际上，终止符可看作是零目运算符。初始群体由 N 个以同样方式产生的个体组成（N 为预先确定的群体规模）。

3）遗传算子

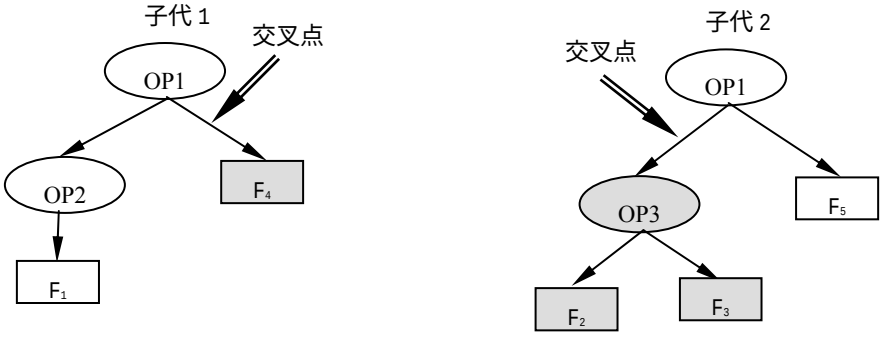
编码方式决定了相应的遗传算子操作方式，因此，遗传规划的遗传算子比遗传算法也要复杂得多。下面，就最常用的交叉和变异算子作一对照。

（1）交叉。

遗传规划的交叉算子如图 4.14 所示。对比遗传算法和遗传规划，可以发现遗传算法的交叉仅需要线性串的复制操作；而遗传规划的交叉操作则涉及到子树结构的拆合。对遗传算法，如果参与交叉操作的两个个体完全相同，则不会产生新个体；而对遗传规划，所产生的子代个体与父代一般是不相同的，除非两个个体的交叉点恰好也相同，但出现这种情况的概率极小，所以遗传规划的交叉施加了一个偏离同一化的反平衡作用，有利于维持群体的多样性。



（a）交叉前

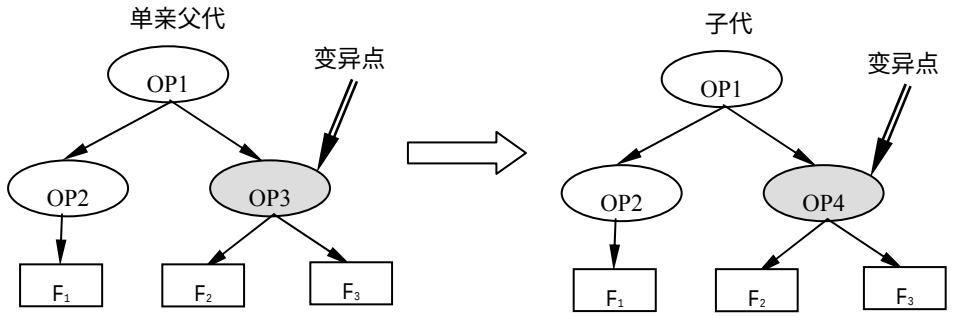


（b）交叉后

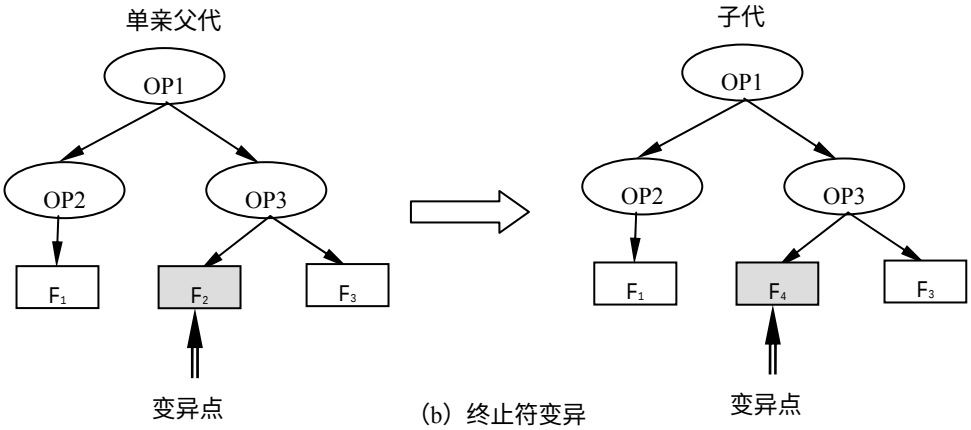
图 4.14 遗传规划的交叉算子

（2）变异。

如图 4.15 所示是遗传规划的变异算子。遗传算法的变异采用位的置反操作；而遗传规划的变异方式有两种：运算符变异和终止符变异。为了保证变异后产生的新个体在语法上的合法性，必须首先判断变异点是运算符还是终止符，然后，根据判断结果，从相应的集合中随机选取一个元素代替原来的元素。如果是运算符，还要注意变异前后的“运算目数”相同。



（a）运算符变异



（b）终止符变异

图 4.15　遗传规划的变异算子

4.3.2 遗传规划的基本流程

遗传规划的基本步骤可归纳如下：

（1）确定个体的表达方式，包括函数集 F 及终止符集 T ；

（2）随机产生初始群体；

（3）计算各个体的适应度；

（4）根据遗传参数，用选择、交叉/变异操作产生新个体；

（5）反复执行（3）及（4），直至取得满意结果。

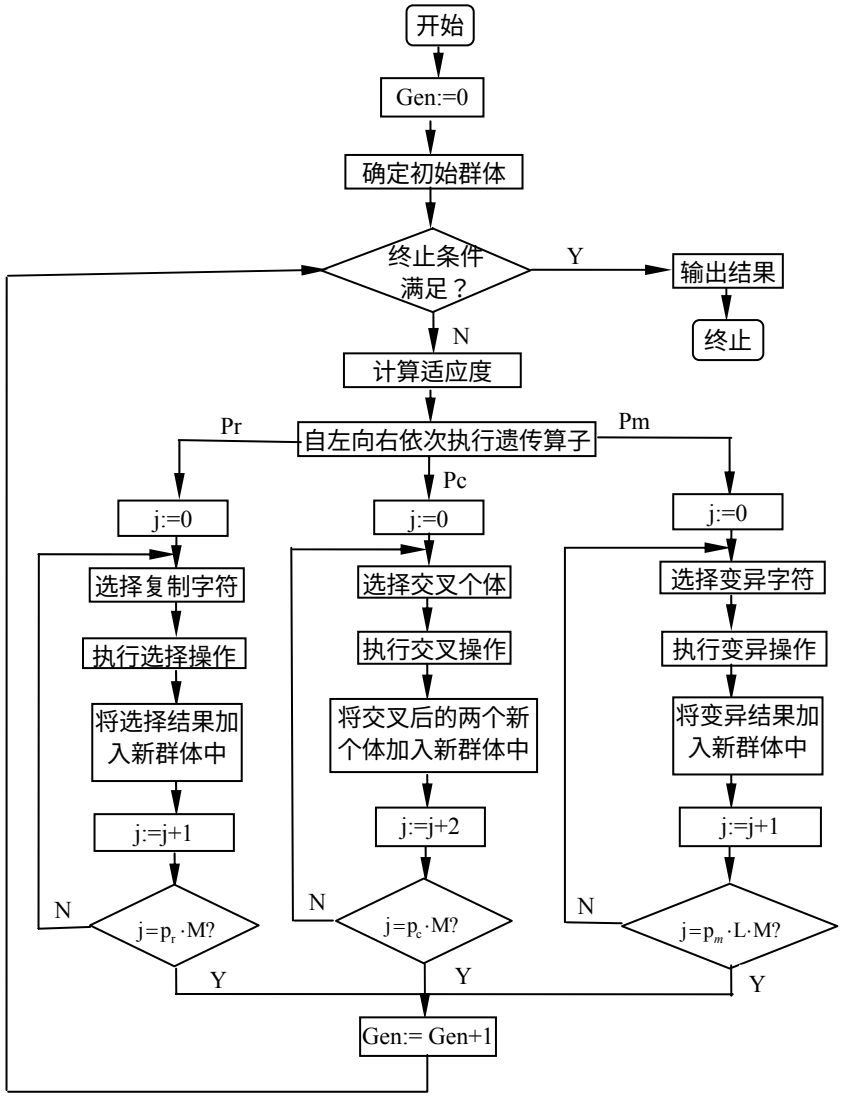


图 4.16 遗传规划的计算流程

图4.16是遗传规划的计算流程，图中Gen表示遗传代数，N表示群体规模，i表示个体计数器。从该图中可以看出，遗传规划与遗传算法的原理基本相同，但在编码、遗传算子等方面与遗传算法有所不同。

遗传规划的理论和应用虽然还只是刚刚起步，在如何优化数值和“回旋体”的避免方面仍然需要继续的研究工作，但这并不能限制遗传规划的广泛应用，特别是在模式识别中的特征构造和特征选择中将大有用武之地。

4.4 进化策略

进化策略的研究始于1964年。当初主要用于试验流体动力学问题，如弯管形状的优化。由于当时现有的一些优化算法不适于解决这类问题，Renchenberg提出按照自然变异和自然

选择的生物进化思想，对物体的外形参数进行随机变化并尝试其结果。后来，Schwefel系统地推广了Renchenberg的二元进化策略（又称1+1-ES），建立了多元进化策略，即(μ+λ)-ES和(μ，λ)-ES。

4.4.1 二元进化策略

进化策略可以分成二元进化策略（two-membered evolution strategies）和多元进化策略（multi-membered evolution strategies）。我们先从二元进化策略开始讨论。

二元进化策略是进化策略中最简单的一种形式。描述如下：

（1）初始化。随机生成1个个体，构成第1代群体的父代。

（2）变异。父代个体将自己的基因经过变异后产生1个子代，种群数为2。子代个体与父代个体在基因上有所不同。

（3）选择。每个个体由于对环境的适应性不同而表现出不同的生命力或适应度，它由遗传基因决定2个个体中只有生命力较强的个体成为下一代的父辈个体。如果终止条件满足，则算法结束。否则回到步骤（2）。

以下用$N(0，1)$表示一个服从期望为0，标准差为1的一维正态分布的随机变量的实现，$N_i(0,1)$ 表示对于下标t的每个随机变量要重新抽样。一个服从期望为0，标准差为1的一维正态分布随机变量的实现就可以由$N(0，\sigma)$给出。

二元进化策略的编译操作是在原个体基础上加上一个满足正态分布随机变量来实现，如式（4-34）所示：

$$X^{t+1} = X^t + N(0,\sigma) \tag{4-34}$$

式中，X^t ——第t代个体的数值；$N(0,\sigma)$ ——服从正态分布的随机数，其均值为零，标准差为σ。

下面仍用4.2.1简单遗传算法的实例：求一元函数 $f(x) = x\cos(4\pi x) + x^2$ 的最大值，$x \in [-1,2]$。

针对此优化问题，假设在t代，有 $(X^t, \sigma) = (1.3,1.0)$ ，则变异后的新个体可利用式（4-34）得：

$$x^{t+1} = x^t + N(0,1.0) = 1.3 + 0.4 = 1.7$$

式中 $N(0,1.0)$ 产生一个服从正态分布的随机数0.4。

由于

$$f^{(t)} = f(1.3) = 0.638278$$
$$f^{(t+1)} = f(1.7) = 1.514671$$

因此，x^{t+1} 被接纳，用新个体 X^{t+1} 代替父代个体 X^t 。

为了提高搜索效率，Rechenberg还提出著名的“1/5成功法则”：

对所有的变异，成功变异的比率应该是1/5。如果大于1/5，增加变异算子的方差；否则，减少其方差。

Schwefel在实际运算中建议如下：

$$\sigma^{t+1}=\begin{cases}C_d\cdot\sigma^t & \text{若}\varphi<1/5\\ \sigma^t & \text{若}\varphi=1/5\\ C_i\cdot\sigma^t & \text{若}\varphi>1/5\end{cases} \tag{4-35}$$

式中，φ——经历k次迭代后变异成功的次数与总变异次数之比，通常$k>10$；C_d——小于1的系数，通常取0.82；C_i——大于1的系数，通常取1/0.82。

4.4.2 多元进化策略

多元进化策略简记为(μ+λ)-*ES*和(μ，λ)-*ES*，这两种进化策略都采用含有μ个个体的原始群体，并通过选择和变异产生λ个新个体。它们的差别仅仅在于下一代群体的组成上。(μ+λ)-*ES*是在原有μ个个体及新产生的λ个新个体中（共μ+λ个个体）再择优选择μ个个体作为下一代群体。(μ，λ)-*ES*则是只在新产生的λ个新个体中择优选择μ个个体作为下一代群体，这时要求$\lambda>\mu$，整个过程描述如下：

（1）初始化。随机生成k个个体，构成第1代群体的父代。其中每个个体的不同是由个体的基因型的不同而决定的。

（2）变异。每个父代个体将自己的基因经过变异后产生λ个子代，此λ个子代个体与其各自的父代个体在基因上有所不同。

（3）选择。每个个体由于对环境的适应性不同而表现出不同的生命力或适应度，它由遗传基因决定。在λ个个体中((μ，λ)-*ES*)或者μ+λ个个体中((μ+λ)-*ES*)选择生命力较强的μ个个体成为下一代的父辈个体。

如果终止条件满足，则算法结束。否则回到步骤2）。

在多元进化策略中，变异算子有了新的发展，标准差σ既不是固定的常数，也不是按1/5成功规则的确定性变化，而是自适应地调整，即：

$$\begin{cases}\sigma'=\sigma\cdot e^{N(0,\Delta\sigma)}\\ X'=X+N(0,\sigma')\end{cases}$$

式中，(X,σ)——父代个体；(X',σ')——子代新个体；$\Delta\sigma$——参数；$N(0,\sigma')$——独立的服从正态分布的随机变量，其均值为0，标准差为σ'。

近年来，(μ,λ)-ES得到广泛的应用，这是由于这种进化策略使每个个体的寿命只有一代，更新进化很快，特别适合于目标函数有噪声感染或优化程度明显受迭代次数影响的课题。

4.4.3 进化策略的基本技术

ES是最早引入自适应机制的算法。它与最广为人知的遗传算法相比，在编码、种群规模、交叉、变异和选择等诸多方面有着许多的不同。

1）编码

在进化策略中，目标参数和策略参数都需要编码到染色体中。目标参数是指直接涉及适应度计算的参数，策略参数为应用于进化算法中的控制参数，例如种群规模、变异步长、变异频率、交叉位置、交叉频率等。一般而言，策略参数的选取对进化算法的性能有着直接的影响。因此，人们希望策略参数和目标参数在进化的过程中可以同时得到优化，这就是“自适应”的由来。

与简单遗传算法采用二进制编码不同，进化策略采用实数编码将目标变量直接编码到

染色体中。它的另一个重要特征是将策略参数也编码到染色体中。因此，一个目标向量X的染色体可以表示为：

$$X=(x_1x_2\cdots x_n\sigma_1\sigma_2\cdots\sigma_n)$$

其中$x_i,i=1,2,\cdots,n$是目标参数，而$\sigma_i,i=1,2,\cdots,n$是策略参数。一般来说，遗传算法的编码方式比进化策略有着更大的灵活性，这是遗传算法应用更为广泛的原因之一。

2）变异算子

与遗传算法不同，变异算子是进化策略的主要算子。在经典进化策略中，变异是通过给目标参数加上一个服从正态分布的随机数来实现的。进化策略目标参数和步长参数的变异公式如式（4-36）所示。

$$\begin{cases}\sigma_i'=\sigma_i\cdot\exp(\tau'\cdot N(0,1)+\tau\cdot N_i(0,1))\\ x_i'=x_i+\sigma_i'\cdot N_i(0,1)\end{cases}\tag{4-36}$$

式中，(x_i,σ_i)——父代个体的第i个分量；(x_i',σ_i')——子代新个体的第i个分量；$N(0,1)$——服从标准正态分布的随机数；$N_i(0,1)$——针对第i分量重新产生一次符合正态分布的随机数；τ'——标准差向量σ的整体步长参数；τ——是σ的每个分量σ_i的步长参数。其中τ'和τ是给定的外部参数，Schwefel建议如下设置：

$$\tau'\propto(\sqrt{2\sqrt{n}})^{-1},\quad \tau\propto(\sqrt{2n})^{-1}$$

其中$\propto$表示成正比。通常τ'和τ的比例因子取为1。

上式表明，新个体是在旧个体基础上随机变化而来。策略参数的数目在不同的进化策略中是不同的。这一特别的变异机制使得ES能够进化本身的策略参数，在搜索的过程中，开发利用了适宜的内在模式和好的适应值之间的一个隐含的关联，由此所导致的根据适应值曲面的拓扑特性而使策略参数得到进化的机制，被称为自适应机制。

3）交叉算子

交叉操作是提供两个不同个体之间交换信息的机制。与遗传算法不同，交叉算子在进化策略中并不是必需的。常用的交叉操作主要有四种，见式（4-37）所示。

$$x_i'=\begin{cases}x_{a,i}\text{或}x_{b,i} & \text{离散性重组}\\ \dfrac{1}{2}(x_{a,i}+x_{b,i}) & \text{中值型重组}\\ x_{a,i}\text{或}x_{b_i,i} & \text{全局离散性重组}\\ \dfrac{1}{2}(x_{a,i}+x_{b_i,i}) & \text{全局中值型重组}\end{cases}\tag{4-37}$$

其中，x_i'——新个体目标变量X得第i个分量；$x_{a,i}$——固定的父代个体的目标变量X的第i个分量；$x_{b,i}$——离散重组时另一固定父代个体的目标变量X的第i个分量；$x_{b_i,i}$——随机选择的另一个父代个体的第i个分量；

通常，在一代群体中，所有的父代个体具有相同的交叉概率，即参加交叉的父代是通过均匀随机数来确定的。在离散交叉情况下，子代的成员向量的每一维都是随机继承某个父代，即$x_{a,i}$或$x_{b,i}$的对应部分，相当于遗传算法中交叉点数量可变的交叉算子。在中间交叉情况下，子代的成员向量是父代成员向量的平均值。在全局交叉情况下，通过选取另外的父代x_{b_i}，使子代继承群体中更多的遗传信息。由此可见，交叉操作可以有效的提高进化

策略的性能，但是出于方便和效率方面的考虑，许多应用中均不采用交叉算子。

4）选择方法

ES中的选择方式是完全确定性的。(μ,λ)-*ES* 是从λ个子代个体中选择μ个最好的个体作为下一代的父代；$(\mu+\lambda)$-*ES*是从λ个子代个体和μ个父代中个体选择μ个最好的个体作为下一代的父代，即精英选择方法，从而保证了性能的改进是单调的，但是该选择方法无法适应变动的环境，不利于实施策略参数的自适应机制。因此，(μ,λ)-*ES*在今天更受到推崇。

4.4.4 进化策略的基本流程

图4.17表示进化策略的工作流程。

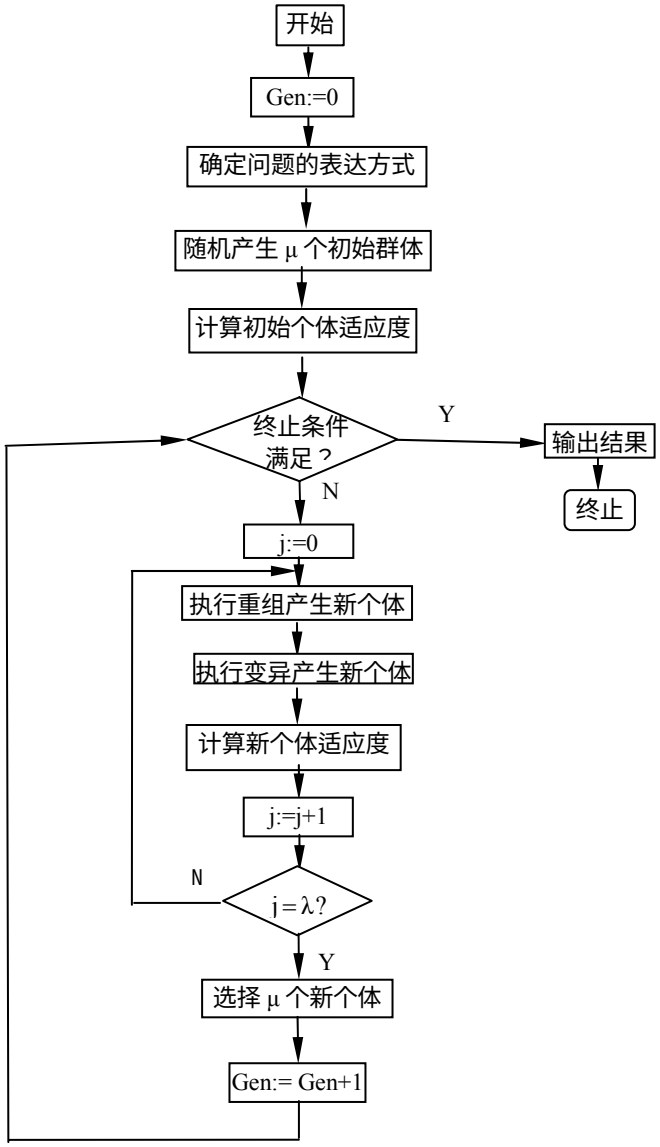


图 4.17　进化策略工作流程

图中Gen表示进化的次数，在第0代，根据问题表达选择二元组或多元组方式，随机产生μ个初始个体，并计算其适应度。然后依次执行重组和变异操作，产生新个体。随后计算新个体的适应度，再根据选择策略，从(μ+λ)个个体中或λ个新个体中选择μ个个体组成新群体，直到满足终止条件为止，ES的基本流程可以描述如下：

```
Procedure: Evolution Strategies
begin
    t ← 0；
    initialize P(0)；evaluate P(0)；
    while (not termination condition ) do
    begin
      recombine P(t) to yield  P′(t) ；
      mutation  P′(t)  to yield  P″(t) ；
      evaluate  P″(t) ；
      select P(t+1)
      if  (μ,λ)  is selcted
      then select P″(t)
      else select P(t+1) from P(t) and P″(t) ；
      t ← t+1；
    end
end
```

4.5 进化规划

进化规划是19世纪60年代中期L.J.Fogel等人为有限状态机的演化而提出的一类进化算法。与遗传算法的不同在于，进化规划注重父代与子代的表现行为而不是遗传细节。1966年，Fogel出版了《Artificial Intelligence through Simulated Evolution》，系统阐述了进化规划的思想。但当时学术界对在人工智能领域采用进化规划表示怀疑，直到20世纪90年代初才逐步被学术界重视。

应该指出，尽管进化规划与进化策略十分相似，但是它们一直是相互独立地平行发展，前者主要是在美国，后者主要在欧洲，双方缺乏交流。一直到1992年，进化规划和进化策略的科技人员才相互交流，促使两者相互渗透。

4.5.1 进化规划的几种表达方式

进化规划算法包括：标准进化规划、元进化规划、旋转进化规划等多种表达方式。其中旋转进化规划比较复杂，目前尚未得到充分认可，最常用的还是元进化规划，本节重点介绍元进化规划算法。

1）标准进化规划（standard EP）

进化规划用传统的十进制实数表达问题。在标准进化规划中，个体的表达形式为：

$$x_i' = x_i + \sqrt{f(X)} \cdot N_i(0,1) \tag{4-38}$$

式中，x_i——旧个体目标变量X的第i个分量；x_i'—— 新个体目标变量 X' 的第i个分量；

$f(X)$——旧个体X的适应度；$N_i(0,1)$—— 针对第i分量发生的随机数，它服从标准正态分布。

根据这种表达方式，进化规划首先产生μ个初始个体，然后进行变异，接着从μ个旧个体及μ个新个体（2μ个个体）中根据适应度挑选出μ个个体组成新群体。如此反复迭代，直至得到满意结果。进化规划的工作流程是：产生初始群体—变异—计算个体适应度—选择一组成新群体，然后反复迭代，一代一代进化，直至达到最优解。

2）元进化规划（meta EP）

为了增加进化规划在进化过程中的自适应调整功能，人们在变异中添加方差的概念。类似于进化策略，在进化规划中个体的表达采用下述公式：

$$\begin{cases} x_i' = x_i + \sqrt{\sigma_i} \cdot N_i(0,1) \\ \sigma_i' = \sigma_i + \sqrt{\sigma_i} \cdot N_i(0,1) \end{cases} \tag{4-39}$$

式中，x_i—— 旧个体目标变量X的第i个分量；x_i'—— 新个体目标变量X'的第i个分量；σ_i—— 旧个体第i个分量的标准差；σ_i' —— 新个体第i个分量的标准差；$N_i(0,1)$——针对第i分量发生的随机数，它服从标准正态分布。

从上式可以看出，新个体也是在旧个体的基础上添加一个随机数，该添加量取决于个体的方差，而方差在每次的进化中又有自适应调整。这种进化方式已成为进化规划的主要手段，因此在进化规划前冠以“元”这个术语以表示它为基本方法。

元进化规划的变异尽管类似于进化策略，但是它们有下述区别：

（1）执行顺序不同。进化规划中首先计算新个体的目标变量x_i'，计算中沿用旧个体的标准差σ_i，其次才计算新个体的标准差σ_i'，新的标准差留待下次进行时采用。与之相反，进化策略是先调整标准差σ，然后再用新的标准差σ'去更改个体的目标函数X。

（2）计算式的不同。进化规划的计算式比进化策略的计算式（4-36）简单。

3）旋转进化规划（rmeta EP）

旋转进化规划进一步扩展进化规划，在表达个体时添加第三个因子——协方差，用三元组表示个体，即(X,σ,ρ)，具体计算如下：

$$\begin{cases} X' = X + N(0,C) \\ \sigma_i' = \sigma_i + \sqrt{\sigma_i} \cdot N_i(0,1) \\ \rho_j' = \rho_j + \sqrt{\rho_j} \cdot N_j(0,1) \end{cases} \tag{4-40}$$

式中，X——旧个体的目标变量，其内含n个分量；X'—— 新个体的目标变量，其内含n个分量；$N(0,C)$——服从正态分布的随机数，其数学期望为0，标准差为与协方差有关；ρ_j——相关系数，$\rho_j = \dfrac{c_{ij}}{\sqrt{\sigma_i \sigma_j}}$。

旋转进化规划在控制因子中除了方差σ外，还添加相关系数ρ，增加算法的自适应能力。

4.5.2 进化规划的基本技术

1）表达方式

进化规划是一种反复迭代、不断进化的过程，采用十进制的实型数表达问题。每个个体的目标变量X可以有n个分量，即

$$X=(x_1,x_2,\cdots,x_n)$$

相应地，每个个体的控制因子σ_i和x_i是一一对应的，n个x_i要有n个σ_i。由X和σ组成的二元组(X,σ)是进化规划最常用的表达形式。

2）产生初始群体

进化规划中初始群体由μ个个体组成，每个个体(X,σ)又可以包含n个x_i和n个σ_i。产生初始个体的方法是随机生成。为了便于和传统的方法比较，可以从某一初始点$(X(0),\sigma(0))$出发，通过多次变异产生μ个初始个体，该初始点从可行域中用随机方法选取。

初始个体的标准差$\sigma(0)$可用下式计算：

$$\sigma(0)=\Delta X/\sqrt{n} \tag{4-41}$$

式中，ΔX——初始点与最优点的距离；n——个体中所含分量个数。

由于ΔX在初始时不便确定，可取$\sigma(0)=3.0$。$\sigma(0)$不易取太大，若$\sigma(0)$太大而且μ也大时选择力度不够，易使群体过于分散。尽管$\sigma(0)$较小，但在进化过程中通过个体的自适应调整仍可使搜索点很快散布在可行域内。

3）适应度计算

适应度是衡量个体优劣的尺度。由于进化规划采用十进制的实数表达问题，因此适应度的计算更加直观、简便、易于执行。

进化规划中对于约束条件的处理，主要采用重复试凑法。每当新个体生成，将其带入约束条件中检验是否满足约束条件。若满足，则接纳新个体；否则，舍弃该个体，借助变异再产生一个新个体。

4）变异

变异是进化规划产生新群体的唯一方法，它不采用选择和交叉算子。由于我们主要用元进化规划，因此，这里只介绍元进化规划的变异表达形式。

对于元进化规划，变异公式是：

$$\begin{cases} x_i' = x_i+\sqrt{\sigma_i}\cdot N_i(0,1) \\ \sigma_i' = \sigma_i+\sqrt{\eta\cdot\sigma_i}\cdot N_i(0,1) \end{cases} \tag{4-42}$$

式中，σ_i——旧个体第i个分量的标准差；σ_i'——新个体第i个分量的标准差；η——系数；

其他符号同式（4-39）。

这种进化规划增加了一个控制因子—方差，它使x_i可以在小范围内变动，有利于算法的收敛。在上式中系数η的目的是增加σ_i的变动范围，若η取1，则式（4-42）与式（4-39）一样。

同时，标准差大于零，为此要经常检查σ_i'，及时予以纠正，即如果$\sigma_i' \le 0$，则令$\sigma_i' = \zeta_0$。

其中 ζ_0 为大于零的小数值。

5）选择

在进化规划中，变异之后便执行选择，采用的是 $(\mu+\mu)$ 型的选择。进化规划的选择采用的是 q －竞争选择法。在这种选择方法中，为了确定某一个个体的优劣，我们从新、旧群体的2 μ 个个体中任选 q 个个体组成测试群体。然后将个体i的适应度与 q 个个体的适应度进行比较，记录个体i优于或等于 q 内各个个体的次数，次数便是个体i的得分 w_i ，即

$$w_i=\sum_{i=1}^{q}w_{ik},\quad \text{其中}w_{ik}=\begin{cases}1 & \text{如果}f_i\text{优于或等于}f_j\\ 0 & \text{其他}\end{cases}$$

式中， f_i —— 个体i的适应度； f_j ——q 测试群体中第j个体的适应度。

上述的得分测试分别对2 μ 个个体进行，每次测试时重新选择 q 个个体组成新的测试群体。最后，按个体的得分选择分值高的 μ 个个体组成下一代新群体。

q －竞争选择法是一种随机选择，总体上讲，优良个体入选的可能性较大。但是由于测试群体 q 每次都是随机选择的，当 q 个个体都不甚好时，有可能使较差的个体因得分高而入选。这正是随机选择的本意。

q －竞争选择法 q 中的大小是一个重要参数。若 q 很大，极端地设 $q=\mu$ ，则选择为确定性选择。反之，若 q 很小，则选择的随机性太大，不能保证优良个体入选。通常 q 在10以上，可取 0.9μ 。

6）终止

进化规划在进化过程中，每代都执行突变—计算适应度—选择等操作，不断反复执行，使群体素质得到改进，直至取得满意的结果。一般来说，进化规划的终止准则一般是根据最大进化代次、最优个体与期望值的偏差、适应度的变化趋势以及最优适应度与最差适应度之差等四个判据。

4.5.3 进化规划的基本流程

图4.18表示进化规划的工作流程。图中Gen表示进化的次数，在第0代，根据问题的表达方式，随机产生初始群体，并计算其适应度。然后进入正常循环。每次迭代中，执行变异操作，产生新个体并计算新个体的适应度。当新个体达到 μ 个后，从 2μ 个新、老个体中择优选出 μ 个个体组成新群体，直到满足终止条件为止，EP的基本流程可以描述如下：

```
Procedure: Evolution Programming
begin
    t ← 0；
    initialize P(0)；evaluate P(0)；
    while (not termination condition ) do
        begin
            mutation P(t) to yield  P′(t) ；
            evaluate  P′(t) ；
            select P(t+1) from P(t) and P′(t) ；
            t ← t+1；
        end
end
```

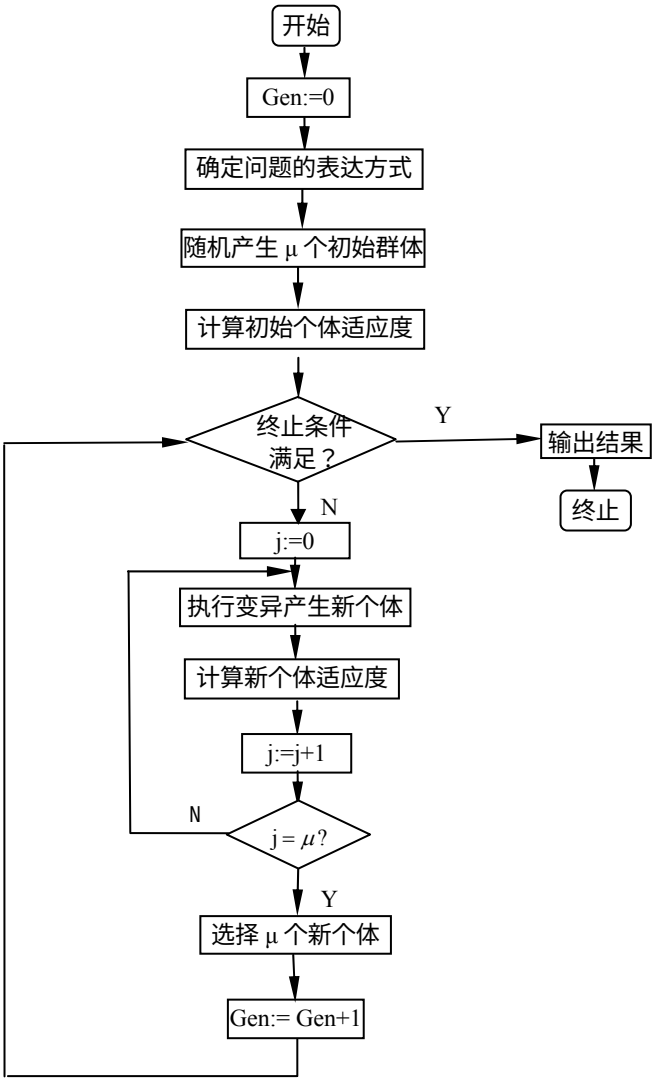


图 4.18　进化规划的基本流程

4.6 案例分析：露天矿床开拓系统结构优化

露天矿床开拓是从地表掘进一系列斜坡道通达矿体，使地表与矿体间形成一条完整的运输、排水和动力供应系统。这些开拓工程在空间的布置体系就构成开拓系统。由于露天矿巷道纵横交错、数目众多，用传统的优化方法难以得到优化的开拓方案，所以选用遗传算法来实现开拓系统的优化，具体而言，首先根据矿体与地表之间空间关系依据已经确定好的各种参数指标，确定各开拓水平及开拓境界，然后依据地表排土场、矿仓的位置布置尽可能多的各种开拓斜坡道，即从不同的地表点通达露天矿坑，然后从中选出最优的开拓系统，如图4.19所示。

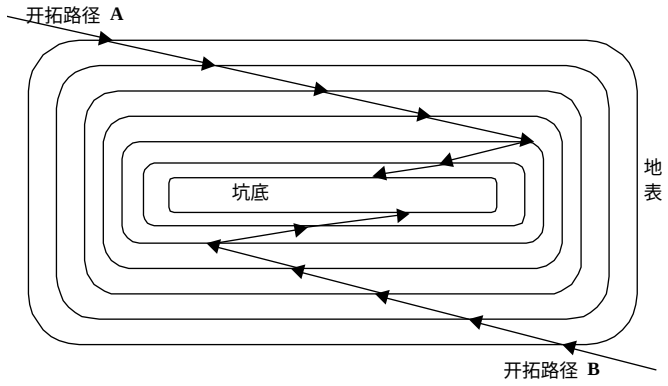
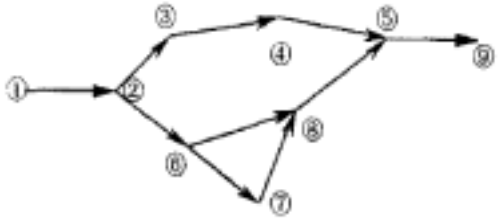


图 4.19　开拓系统各种可能的开拓路径

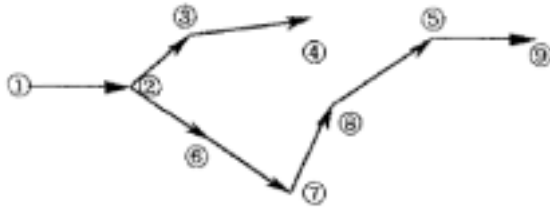
4.6.1 系统分析

1）编码

本系统采用二进制编码方式。在开拓系统确定中，对所有可能存在的联系上下水平的开拓斜坡道顺序排列，假定露天矿开拓设计共有N个水平，则有$N-1$个开拓斜坡道，这样才能从地表通达坑底，依据从地表向坑底的顺序对各开拓斜坡道连续编号，而且每条开拓斜坡道用二进制字符表示。在字符串中，若第i位为1，则表示该方案中存在第i条边；反之，若第i位为0，则表示该方案中不存在第1条边。这样就可用长度固定的B位0/1字符串来表示某一可能的开拓系统方案，如以下示例所示。



（A）



（B）

图 4.20　编码示例

以图4.20（A）为例，按上述规则对图4-20（B）进行编码，字符串及其所对应的分支

和节点如表4-8所示。

表 4-8　开拓系统的编码

边编号	1	2	3	4	5	6	7	8	9	10
始终点	①②	②③	③④	④⑤	②⑥	⑥⑧	⑥⑦	⑦⑧	⑧⑤	⑤⑨
随机字符串	1	1	1	0	1	0	1	1	1	1

2）产生初始群体

本系统初始群体由一组随机生成的个体组成，M取50。为了保证初始群体的多样性，人为地使各种开拓斜坡道出现在部分初始个体中。然后再用交换、突变等操作产生其他新的初始个体。

3）选择

选择（复制）是遗传算法的主要操作。本系统采用Holland教授推荐的赌轮选择法选择复制对象。个体被选择的概率取决于个体的适应度，即：

$$P_a = \frac{f_a}{\sum_{i=1}^{M} f_i} \tag{4-43}$$

式中，p_a——个体a被选中复制的概率；f_a——个体a的适应度；M——群体中个体的数目。

赌轮选择法使优良个体有更多的机会被复制，然而个别劣质个体也有可能偶尔被破格选中，从而增加群体的多样性。

4）交叉

交叉是遗传算法产生新个体的主要手段。尽管表示开拓系统的字符串长度较大，为了保证开拓通道的连贯性，本系统采用单点交叉。交换产生的新个体，从开拓设计角度衡量不一定都是技术上可行的，因此本系统在交换时只要有一个新个体符合要求即予保留，另一个新个体如果不合格则用原旧个体代替。

5）变异

变异是遗传算法产生新个体的另一种方法。本系统为了执行方便，当突变个体确定之后，再用随机方法确定突变字符的位置。

总之，本系统由二进制字符串表示各种开拓方案，在随机产生初始群体的基础上，通过复制、交换、突变等操作不断产生新的群体，从而得出千姿百态的多种开拓方案。

4.6.2 实现技术

通过深入分析可以看出，作为技术上可行的开拓系统，应该满足下述基本要求：

（1）同一开拓路径中存在的各斜坡道应该相互连接，不允许在从地表到坑底方向有“ 断层 ”现象存在。从图论角度讲，即开拓系统中的各开拓路径均是一个连通图。

（2）一个开拓系统，应该至少有1条开拓路径的斜坡道通至坑底。也就是说，保证开拓系统至少满足从地表能通达坑底。

4.6.2.1 编码的技术处理

在文中4.6.1部分用遗传方法产生的开拓方案，有些不满足上述要求，需要删除。为此，采用下述三种技术执行：

1）模板

在开拓系统中，至少要有一条开拓路径通达坑底，因为同一开拓路径中不允许出现“断层”现象，故此路径实际上是从地表通达坑底的，即开拓系统中至少存在一条开拓路径，该开拓路径中联系上下各水平的斜坡道均存在，也就是在开拓系统个体字符串中对应于该开拓途径的所有位为1。为此，可根据开拓系统优化时考虑的开拓路径的数目生成相对应数目的模板。所谓模板，就是一个专门的字符串，字符串中用1标记必须存在的斜坡道，而字符串的其他字符为任意值。每次用遗传算法生成一个新的开拓系统，都用某个模板去校正其字符串，以保证新的开拓系统具备这些模板的特征，即总存在某条完整的开拓路径。

2）顺序编号

按从地表向坑底方向对各条开拓路径联系上下水平的斜坡道顺序编号，并保证在组成个体字符串时仍维持局部相对顺序不变。这样有利于对个体中的各开拓路径进行下一步的连通性判断。

3）路径连通性判断

由于个体中的各开拓路径采用顺序编号，因此可将图论中连通性判断这一较复杂性问题转化为：分别对个体字符串中对应的各开拓路径部分，按从地表向坑底方向进行这样校验：若某位为0（其前全为1），则此个体中对应该开拓路径部分字符串的此位之后部分全为0。符合以上规则则开拓方案中的各开拓路径连通，反之则存在不连通路径，即不符合条件1（也即个体无效）。

4.6.2.2 适应度的计算与终止

在遗传算法中，适应度是衡量个体优劣的依据，它也是驱动群体进化的动力。本系统根据年成本法计算每个开拓系统的年成本C，用它作为各个个体的适应度。即：

$$C = D + P \cdot \frac{(1+i)^n \cdot i}{(1+i)^n - 1} \tag{4-44}$$

式中，D——年经营费用；P——开拓费；i —— 年利率；n——矿山存在年限。

由于适应度用作开拓方案比较，因此计算时只计算各方案不同部分的费用，相同部分尽量不予计算。

1）开拓费用

计算开拓费用时，适应度包括斜坡道工程费和坑外公路建设费，另外还考虑露天排土场的建设费用。

2）经营费用

露天矿适应度内的经营费用仅包括运输费用，它可分为坑内运输费用和坑外运输费用两种，每种又包括矿石运输和排土运输两类。

运输费用按吨·公里运输单价计算。对于坑内及坑外，各有不同的运输单价。至于运输量，按各水平储量计算。具体算法如下：若某水平L储量为：矿石M吨，剥土量N吨，X种开拓路径中有n种开拓路径开拓斜坡道到达此水平，则规定本水平中的M吨矿石、N吨剥土量平均走这n种不同的开拓路径。最后各开拓路径依各水平的运输量及运输距离、运输单价计算出运输费用，开拓方案中的各开拓路径的运输费用之和即本个体方案的运输费用。经营费用中的运输费用按年计划剥采量计算。

4.6.2.3 终止准则

在本系统中，优化的对象为开拓系统结构，其适应度（目标函数）的计算本来就相当复杂（含风网计算），无法确定最优值。单纯用规定遗传迭代次数的终止方法不能确定运算得出结果的优劣性，本文采用两种终止条件联合作用法来决定迭代何时终止：

1）规定最大迭代代次（本系统中，采用对话框的人机界面，在系统开始优化运算前，可人为指定最大迭代次数，或默认为最大迭代次数为100），当迭代次数达到最大迭代次数时，停止迭代运算。

2）控制最大适应度与平均适应度之间的偏差，通过在遗传运算前设定最大偏差δ（也是采用对话框的人机界面），当最大适应度与平均适应度之间的偏差小于或等于最大偏差δ时，则停止迭代运算。

通过以上两种终止条件联合作用，可及时、有效地得出合理的运算结果，避免软件系统无谓的计算开销。

4.6.3 模型评价与分析

本系统以湖北黄石金铜矿业公司一期开拓系统为实例。

1）编码

在实例中，共有两种开拓方案参与优化（竖井开拓和斜井开拓），总共的分支数为33（编号为1～ 33），即字符串长度为33，其中必须存在的分支数为13条（编号为15～27），总节点数为22（编号为1～22）。共有6个开拓水平。依次为-40m、-80m、-120m、-160m、-200m及-240m水平。

2）群体规模的确定

群体中个体数目越大，则搜索范围越广，但是每代的运算时间也加长，同时对计算机的内存容量提出了更高的要求，也体现不了遗传算法真正的优越性。因此个体数目要取得合适，通常为50到100个个体为宜。第0代的初始群体，用随机方法产生，在给定的合理范围内用均匀分布的随机数任意产生初始个体。随着遗传计算的发展，群体中个体的适应度逐渐改善，不断产生新的优良个体。

3）适应度的计算

在本系统中，由于在遗传算法基础上加入了开拓系统结构的系统判别，这使得程序的运算量很大，在适应度的计算中，又考虑了开拓系统的风网计算。故本系统的研究范围较广，计算相当大。因此，对工程系统的基建投资采用静态法计算（基建过程中的施工进度表不在本系统研究范围之内，故不予考虑）；对经营成本中的矿石运输费用，由于涉及到网络运输流问题，亦超出了本文研究范围，故本系统在计算运输费用时仅考虑主要运输工程的运输费用。具体方法为将开拓工程分为四类：（1）中段运输巷道工程（水平）；（2）竖井运输工程（垂直）；（3）斜井运输工程（倾斜）；（4）其他工程。在计算经营成本之运输费用时，仅考虑前三种工程的运输费用，而且不考虑工程具体的运输量，简化的计算方法为：将具体开拓方案中的各类运输工程的长度之和乘以相应的费用系数（可以以某种运输工程为基准，将其他运输工程长度乘以一个系数后换算为相应长度的该工程，原则是换算前后的运输费用相同），再将三者之和累计，按照总运输量计算出这个累计长度相应的运输费用。

基建投资费用大体上分为工程施工费用和设备投资费用两部分。对于工程施工费用，以某一时间为基准按有关定额分别计算出各工程的单独施工费用；对于设备投资部分，如

果能分摊，则分摊到相关的各工程中；如果不能分摊，则将其计入与其相关的某一工程中（如通风系统中的抽风机、竖井提升系统中的提升机等）。

对于经营成本中的通风费用V，先计算出系统的风网总阻力h_z，则通风功耗可近似计算为$N=h_zQ$（风机的实际功率要大于N，它等于h_zQ/ξ，ξ为通风机的功率系数，$0<\xi<1$，但由于多种开拓方案中使用同一种风机，故在比较方案时，可不考虑系数ξ的影响，直接用$N=h_zQ$来计算通风费用）。然后按开拓系统服务年限n累计通风费用。

对于运输费用，也以以上简化计算方法，按开拓系统服务年限n累计运输成本。

在适应度的计算中本系统的实际目标函数为最小值问题，因此需引入一足够大的常数减去费用之和后的值作为适应度函数（最大值问题），这样便于计算。

4）遗传算子的确定

在本系统中，采用简单遗传算法的思想确定遗传算子。用赌轮选择法确定选择算子，采用单点交叉进行个体转换，变异概率按个体计算而不是按字符计算。

通过上述分析，结合遗传算法的基本思想，开发了相应的应用软件，具体的参数设置为：

字符串长度	33	
群体大小	100	
最大迭代次数	50	
选择概率	10%	
交叉概率	65%	
变异概率	15%	（按个体计算）

运行结果如图4.21所示。

最优个体的字符串是：$N=$［000001111111001111111111111011100］

最优个体适应度对应的总投资为：2996万元，最优个体占总群体的76%。

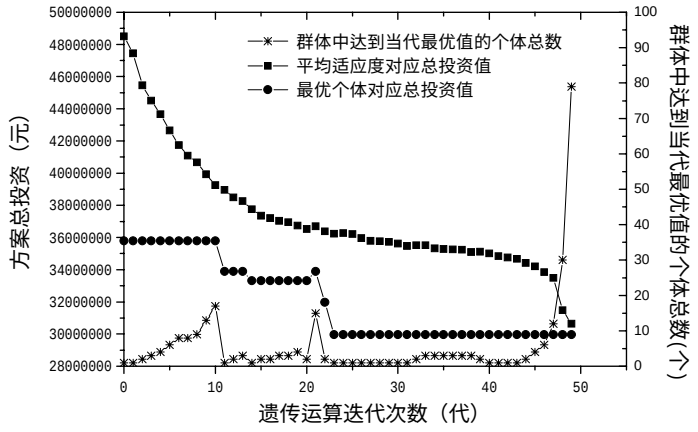


图 4.21 实例运行结果

目前，该系统已成功地应用于湖北黄石金铜矿业公司的开拓设计中，应用中还对遗传算法的计算参数及运行特征进行了深入的研究。实践表明，本系统所使用的方法是成功而有效的，可推广应用于其他矿山系统中。

思考题

4.1 进化计算具有什么特点，它通常应用于哪些领域？

4.2 试论述二进制编码存在的缺点以及简单遗传算法的求解过程。

4.3 试论述遗传算法、进化策略和进化规划的相似之处和不同之处。

4.4 适应度函数如何有效反映每一个染色体与问题的最优解染色体之间的差距？

4.5 试从遗传算法理论基础的角度，分析SGA是否收敛，参数对算法有什么影响。

4.6 遗传算法的改进措施有哪些？请分项列举。

4.7 对于 $\max f(x) = 4x_1^2 + x_1 \cos(x_2)$ 这个3维最大化问题，若要求满足 $7x_1 + 5x_2 \le 14$，且 $x_1, x_2 \ge 0$，试确定其编码及适应度。

4.8 遗传规划和遗传算法有什么区别？

参考文献

[1] 云庆夏．进化算法.北京：冶金工业出版社，2000
[2] 徐宗本．计算智能（第一册）：模拟进化计算．北京：高等教育出版社，2005
[3] 黄凯明．遗传算法在开拓系统结构优化中的应用研究．西安建筑科技大学硕士学位论文，2000
[4] 孙瑞祥，屈梁生．进化计算的过去、现在与未来．“遗传算法/进化计算研究生学术论坛”论文集.2001
[5] 熊军，高敦堂等．遗传算法交叉算子性能对比研究．南京大学学报，2004，40（4）：432-437
[6] 杨平，郑金华．遗传选择算子的比较与研究．计算机工程与应用，2007，43（15）:59-65
[7] 王春水,肖学柱,陈汉明．遗传算法的应用举例．计算机仿真,2005,22(6):155-157
[8] Wolfgang Banzhaf, James Foster. Special Issue on Biological Applications of Genetic and Evolutionary Computation. Genetic Programming and Evolvable Machines, 2004, 5(2): 119-241
[9] Burke E., Gustafson S. , G. Kendall. Diversity in Genetic Programming: An Analysis of Measures and Correlation with Fitness. IEEE Transactions on Evolutionary Computation, 2004, 8(1):47-62
[10] Jian-Ping Li, Marton E, Balazs, et.al. A species-conserving genetic algorithmfor multimodal function optimization. Evolutionary Computation, 2002, 10(3):207-234
[11] O. Hrstka. A competitive comparison of different types of evolutionary algorithms. Computers and Structures, 2003(81):1979-1990
[12] Cindida Ferreira.Gene Expression Programming:A New Adaptive Algorithm for Solving Problems. Complex Systems,2001,13（2):87 -12
[13] Z Li. Genetic algorithm toolbox for SC ILAB. http: / / liama.ia.ac.cn /SC ILAB /works/GATS. html, Feb, 2004
[14] K C Tan, A Tay, J Cai. Design and Imp lementation of a distributed Evolutionary Computing Software. IEEE TransSyst., Man, Cybern. C,2003,33: 325-337
[15] T.Back, et.al. Proceedings of the Seventh International Conference on Genetic Algorithms and Their Applications, San Mateo, CA: Morgan Kaufmann, 1997
[16] Z.Michalewicz et.al. Evolutionary Algorithms in Engineering Applications. Spinger,1997

附录：简单遗传算法的 MATLAB 程序

```
figure;
hold on,grid;
fplot('variable.*sin(10*pi*variable)+variable.^2',[-1,2]);
NIND=30;
MAXGEN=25;
PRECI=22;
GGAP=0.9;
trace=zeros(2,MAXGEN);
FieldD=[22;-1;2;1;0;1;1];
Chrom=crtbp(NIND,PRECI);
gen=0;
variable=bs2rv(Chrom,FieldD);
ObjV=variable.*sin(10*pi*variable)+variable.^2;
while gen<MAXGEN
    FitnV=ranking(-ObjV);
    SelCh=select('sus',Chrom,FitnV,GGAP);
    SelCh=recombin('xovsp',SelCh,0.3);
    SelCh=mut1(SelCh);
    variable=bs2rv(SelCh,FieldD);
    ObjVSel=variable.*sin(10*pi*variable)+variable.^2;
    [Chrom ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel);
    gen=gen+1;
    [Y,I]=max(ObjV),hold on;
    trace(1,gen)=max(ObjV);
    trace(2,gen)=sum(ObjV)/length(ObjV);
end
variable=bs2rv(Chrom,FieldD);
hold on;
plot(variable,ObjV,'b *');
figure;
plot(trace(1,:),'- o');
hold on;
plot(trace(2,:),'-. *');
grid;
legend('解的变化','种群均值的变化');
```

第 5 章 群智能算法

5.1 群智能算法概述

群智能（swarm intelligence，简称 SI）是一种在自然界生物群体行为的启发下提出的人工智能实现模式，是简单智能的主体通过相互合作表现出复杂智能行为的特性。该智能模式需要以相当数目的智能体来实现对某类问题的求解功能。作为智能个体本身，在没有得到智能群体的总体信息反馈时，它在解空间中的行进方式完全是没有规律的。只有受到整个智能群体在解空间中行进效果的影响之后，智能个体在解空间中才能体现出具有合理寻优特征的行进模式。群体智能研究主要是对生物群体协作产生出来的复杂行为进行模拟，并在此基础上，探讨解决和解释一些复杂系统复杂行为的新思路和新算法。

5.1.1 群智能的起源

群智能的概念最早由 Beni，Hack-wood 和 Wang 在分子自动机系统中提出。分子自动机中的主体在一维或二维网格空间中与相邻个体相互作用，从而实现自组织。1999 年，Bonabeau，Dorigo 和 Theraulaz 在《Swarm Intelligence：From Natural to Artificial Systems》中对群智能进行了详细的论述和分析，给出了群智能的一种不严格定义：任何一种由昆虫群体或其他动物社会行为机制而激发设计出的算法或分布式解决问题的策略均属于群智能。

这里，群体可被描述为一些相互作用相邻个体的集合体，蜂群、蚁群、鸟群都是群体的典型例子。一只蜜蜂或蚂蚁的行为能力非常有限，它几乎不可能独立存在于自然世界中，而多个蜜蜂或蚂蚁形成的群体则具有非常强的生存能力，且这种能力不是多个个体之间的能力通过简单叠加所获得的。社会性动物群体所拥有的这种特性能帮助个体很好地适应环境，个体所能获得的信息远比它通过自身感觉器官所取得的多，其根本原因在于个体之间存在着信息交互能力。信息的交互过程不仅仅在群体内传播了信息，而且群内个体还能处理信息，并根据所获得的信息（包括环境信息和附近其他个体的信息）改变自身的一些行为模式和规范，这样就使得群体拥有一些单个个体所不具备的能力和特性，尤其是对环境的适应能力。这种对环境变化所具有的适应能力可以被认为是一种智能，也就是说动物个体通过聚集成群而涌现出了智能。因此，Bonabeau 将群智能的定义进一步推广为：无智能或简单智能的主体通过任何形式的聚集协同而表现出智能行为的特性。这里我们关心的不是个体之间的竞争，而是它们之间的协同。

James Kennedy 和 Russell C.Eberhart 在 2001 年出版的《Swarm Intelligence》是群智能发展的一个重要历程碑，因为此时已有一些群智能理论和方法得到了应用。他们不反对 Bonabeau 关于群智能的定义，赞同其定义的基本精神，但反对定义中使用“主体”一词。其理由是“主体”所带有的自治性和特殊性是许多群体的个体所不具备和拥有的，这将大大限制群体的定义范围。他们认为暂时无法给出合适的定义，赞同由 Mark Millonas（1994）提出的构建一个群智能系统所应满足的五条基本原则：

（1）邻近原则（proximity principle）：群内个体具有能执行简单的时间或空间上的评估和计算的能力。

（2）品质原则（quality principle）：群内个体能对环境（包括群内其他个体）的关键性因素的变化做出响应。

（3）多样性反应原则（principle of diverse response）：群内不同个体对环境中的某一变化所表现出的响应行为具有多样性。

（4）稳定性原则（stability principle）：不是每次环境的变化都会导致整个群体的行为模式的改变。

（5）适应性原则（adaptability principle）：环境所发生的变化中，若出现群体值得付出代价的改变机遇，群体必须能够改变其行为模式。

以上五条原则现在成为了群智能的最基本理论，现有的群智能方法和策略都符合这些原则。

《Swarm Intelligence》最重要的观点是：Mind is social，也就是认为人的智能是源于社会性的相互作用，文化和认知是人类社会性不可分割的重要部分，这一观点成为了群智能发展的基石。

目前，尽管群智能理论还不很成熟，但它已成为有别于传统人工智能中连接主义、行为主义和符号主义的一种新的关于智能的描述方法，并成为人工智能领域新的研究热点。2003 年 IEEE 第一届国际群智能研讨会在美国印第安纳州首府召开，后来每年都举办一次群智能国际研讨会。

5.1.2 群智能算法发展

群智能算法是在群智能领域中计算智能研究的逐步深入而产生的一种新兴的计算智能模式，它是在对某些群体行为的观察和研究的基础上，运用一定的数学工具和计算机工具，提出相应的群智能算法，并用来解决那些因为难以建立有效的形式化模型而用传统优化方法又难以有效解决甚至无法解决的问题。它是群智能研究的一个重要分支。

群智能算法的研究始于意大利学者 Dorigo 开发的蚂蚁算法，受蚁群在觅食过程中总能找到从巢穴到食物源的最短路径这一现象的启发，他提出了用于求解旅行商问题（traveling salesman problem，TSP）的蚂蚁系统（ant system，AS），后来又归纳提炼出蚁群优化（ant colony optimization，ACO）的元启发方法。后来，又有其他学者根据蚁群的劳动分工和墓地构造等现象提出了一系列模型和算法。Kennedy 和 Eberhart 则通过观察鸟群的协作觅食活动，于 1995 年提出了粒子群优化方法（particle swarm optimization，PSO）。因此，就优化而言，群智能已包括蚁群算法（ACO）和粒子群优化方法（PSO）两个大类。

在群智能算法领域中，蚁群算法和粒子群优化算法作为群体智能计算的两种典型实现模式，受到了普遍关注，两种算法模式都是基于种群寻优的启发式随机搜索算法。在这两种模式的群体智能中，蚁群算法更擅长于离散空间中的优化问题求解，而粒子群算法则被更多地用于连续空间内的优化问题求解。

自从算法提出后，很多研究人员分别对蚁群算法和粒子群优化算法进行了改进，在蚁群算法方面，提出了如最大最小 AS、具有变异特征的 AS、自适应 AS 和动态 AS 等改

进算法。粒子群算法研究人员对粒子群体组织和协作模式以及算法参数等进行了研究，提出了如模糊 PSO、带选择的 PSO、具有高斯变异的 PSO、具有繁殖和子种群的混合 PSO、簇分析 PSO、协同 PSO 以及多阶段 PSO 等。

另外，随着群智能算法研究的不断深入，还有一些专家学者提出了新的群智能算法：2003 年李晓磊、邵之江等提出的鱼群算法，它利用自上而下的寻优模式模仿自然界鱼群觅食行为，通过鱼群中各个体的局部寻优，达到全局最优值在群体中凸现出来的目的。随后，李晓磊等又采用分解协调的思想构造一种改进的人工鱼群算法，并以换热器系统为例，验证了该算法，结果表明该算法具有较好的收敛性。不过，由于该算法刚刚诞生且是建立在仿真基础上，其性能及理论基础还有待继续研究。

5.1.3 群智能算法与进化计算

群智能理论目前最成功的应用是优化算法，将其与以遗传算法为代表的进化计算（evolutionary computation，EC）方法进行对比分析是非常有必要的。

群智能算法和进化计算都是基于群体迭代的启发式随机优化算法，有着很多相似之处，它们都是对自然界中随机系统的仿真，都具有本质并行性。另外，与进化计算还一样的是，群智能算法的目的并不是为了忠实地模拟自然现象，而是利用它们的某些特点去解决实际问题。ACO 和 PSO 也一度曾被归类于进化计算之中，但它们与进化计算之间的区别也是明显的。ACO、PSO 在本质上不能用广义进化计算算法的流程进行描述。

首先，进化计算和群智能所模拟的自然随机系统不同。进化计算是模拟生物系统进化过程，其最基本单位是基因（gene），它在生物体的每一代之间传播；已有的基于群智能的优化算法都是源于对动物社会通过协作解决问题行为的模拟，它主要强调对社会系统中个体之间相互协同作用的模拟，其最基本单位是敏因（meme），这一词由 Dawkin 在《The Selfish Gene》中提出。

其次，进化计算中强调“适者生存”，不好的个体在竞争中被淘汰；群智能强调“协同合作”，不好的个体通过学习向好的方向转变，不好的个体被保留还可以增强群体的多样性。进化计算中最好的个体通过产生更多的后代来传播自己的基因，而群智能中的优秀个体通过吸引其他个体向它靠近来传播自己的敏因。

最后，进化计算的迭代由选择、变异和交叉重组操作组成，而群智能的迭代中的操作是“跟随”，ACO 中蚂蚁跟随信息素浓度爬行，PSO 中粒子跟随最优粒子飞行。在某种程度上看，群智能的跟随操作中隐含了选择、变异和交叉重组操作。例如，PSO 中在群体所有粒子经历过的最好位置的索引号 gbest 和它经历过的最好位置 pbest 的更新可以类似一种弱选择；而粒子位置更新则类似于 3 个父代：Xi（第 i 个粒子）、gbest 和 pbest 的之间重组，其中还包含了变异的成分。群智能中所隐含的变异是有偏好的，而并非通常进化计算中的完全随机变异，这与最近对实际生物系统变异行为的新研究成果相符。

Kennedy 认为，进化计算和群智能计算所分别模拟的两个自然随机系统 Evolution 和 Mind 之间存在着显著的差异，尽管它们都是基于群体的，且都是由其中的随机成分带来创新，但其本质是不同的。

5.1.4 群智能算法的典型应用

目前，群智能算法主要用于复杂问题求解，现有的群智能模型和算法已经广泛应用于许多自然科学与工程科学领域，显示出了强大的优势和潜力。

5.1.4.1 智能优化及其工程应用

群智能的研究源于对蚂蚁和鸟等生物群体的“趋优”行为的模拟，在探索过程中发现这类群体趋优模型特别适于求解复杂优化问题。目前，ACO 和 PSO 的一系列实现算法已在组合优化和函数优化方面取得了较好的效果。蚁群优化算法在求解多种组合优化问题时性能表现突出，其中主要包括旅行商问题（traveling salesman problem，TSP）、一次分配问题（quadratic assignment problem）、调度问题（scheduling problem）、图着色问题（graph coloring）、多重背包问题（multiple knapsack）、约束满足问题（constraint satisfaction problem）、离散多目标优化问题。对于上述某些问题，ACO 的求解性能甚至超过了遗传算法和人工神经网络等计算智能方法。同样，粒子群优化算法 PSO 已被确认是一种高效、简单的全局优化算法，尤其是引入了函数扩展技术（function stretching technique）的 PSO 算法对于各类复杂优化问题，均能有效地收敛到其全局最优解。目前，PSO 算法已在多峰值函数优化、多目标优化和约束优化等方面取得了很好的应用效果。

由于通信网络的分布式信息结构、非稳定随机动态特性以及网络状态的异步演化过程与蚁群算法的本质和特性非常相似，相应的应用研究持续高涨。围绕通信网络的特点，通过对蚁群算法进行细化处理，已得到一些性能优良的算法，如解决分组交换网络路由问题的 AntNet 算法和解决连通有向图路由问题的 AntNet- FA 算法。实际上，对于动态网络的网络保持，ATM 网上 VC 路由选择、电信网动态路由优化、QoS 路由调度、多波长全光传输网的虚拟波长路径路由和波长分配等问题，蚁群算法的求解结果都是令人鼓舞的。法国电信公司、英国电信公司和 MCIWorldCom 在 20 世纪 90 年代中后期都开展了蚁群算法用于电信路由的研究，并在实际电信网中得到成功应用。

PSO 在人工神经网络优化方面取得了良好的效果，其优化对象不仅包括网络的权重，而且可以包括网络的结构。高海滨等人提出的基于连接结构优化的粒子群优化算法（SPSO）与 BP 神经网络及遗传算法比较，在提高分类误差精度的同时加快了训练收敛速度。Van den Bergh 和 Engelbrecht 提出的协同 PSO 算法在训练求解分类与函数逼近问题的神经网络时，所达到的精度及泛化能力要优于基于梯度的学习方法及遗传算法。这方面的成果已应用于医学中震颤行为的分析和设计电力变压器的智能保护机制等。

在化工方面，PSO 算法已被美国一家公司用于将各种生物化学成分进行优化组合，进而人工合成微生物。与传统的工业优化方法比较，PSO 产生合成结果的适应度是传统方法的两倍，实验的优化过程充分显示了 PSO 算法的优越性。

群智能在工程设计中的应用也有若干成功的实例，如数字电路的进化设计、机构运动链的同构判定、复杂设计过程的重组与优化、公差优化设计、复杂产品设计中的方案组合优化、卫星舱的布局优化设计、空间桁架设计、几何元胞线的生成等。

5.1.4.2 生产管理

群智能在生产管理中的应用研究已不鲜见，例如具有群智能特性的先进组织模式和

隐式协调控制机制，为分布式自治制造系统提供了绝好的样板。Cicirello 和 Smith 提出了一套蚁群控制系统（ant colony control，AC^2）。Gao 等人利用基于信息激素的机床选择算法实现制造系统中的资源-任务分配，实现了在制造系统中模仿蚁群的隐式协调机制。Bonabeau 等人在蚁群觅食算法的基础上开发了一套能够处理动态变化条件的软件，帮助 Unilever 公司实现工厂设备的快速、自动和有效配置。Bonabeau 等人还在蜂群模型的基础上开发了一套汽车厂油漆刷调度系统。群集智能方法在 JSP、JIT、FOP 等生产调度问题求解方面也体现出了良好的性能，在多孔类零件的孔群加工路径规划方面取得了优于 Hopfield 神经网络的结果。

群智能理论与方法在管理领域进行推广应用还存在一些障碍：首先是管理者在对自组织系统和群集智能的运行机制的理解上存在一定的困难，从而导致即使通过大量的实际数据证明这些理论确实有用，也难以有效说服管理者实施群集智能解决方案。其次就是目前还无法用相同的数学框架来描述昆虫与人这两类关键的对象，并且难以将人在企业中的受限行为规则直接映射为群集智能中的个体行为规则，一旦这些问题得到解决，将促使企业管理从“命令-控制”模式到“自组织涌现”模式的深层次变革。

5.1.4.3 机器人学

群智能理论与方法在机器人学中的应用主要有以下 4 个方面：

（1）机器人路径规划。利用蚁群算法可以有效地解决自由飞行空间机器人避障问题。

（2）群体机器人的任务协调和协作规划。已有学者在蚁群算法思想的基础上设计了一种多机器人协作策略，使得多机器人系统可以在未知环境中自主进行协作规划，并能防止任务死锁。

（3）群体机器人自组织行为模拟。Holland 考察了异构的真实机器人群体中基于 Stigmergy 的自组织操作，完成对两种不同类型的目标对象进行有效的聚类和排序的任务。

（4）具有生命特征的机器人的自复制、自装配、自修复。将群集智能的理论与方法应用于可重构群体机器人系统，每个机器人个体相对较小，且具有简单行为规则集（如连接、断开、爬到临近的个体之上）和规则的外形（三角形、正方形、六边形等）。通过这些简单的模块，能够生成所需要的二维或者三维结构，这些结构是可重构的，且具有自修复功能。

值得注意的是，这样的群体机器人系统已经逐步具备生命的特征，是“人工生命”的一个重要研究领域。

5.1.4.4 数据分析与模式识别

基于蚁群聚类的模型已经被应用于数据分析与模式识别之中。群智能聚类算法起源于对蚁群蚁卵的分类研究，Lumer 和 Faieta 将 Deneubourg 提出的蚁巢分类模型应用于数据聚类分析。Kuntz 等人进一步扩展了 LF 模型并将其应用于各种不同的图像生成和图像分割的问题。Ramos 等人在 LF 模型的基础上，提出了一种蚁群聚类系统 ACLUSTER，并应用于文本文档聚类。他们还从类似脑结构的神经元的自组织与蚁群的自组织的相似性出发，尝试构建通过简单局部规则的交互而产生涌现的模式识别系统。他们通过利用人工蚂蚁对环境的反应和感知，提出了一种处理数字图像聚集的扩展模型，并指出人工

蚂蚁可以对任何类型的数字聚集（digital habitat）作出适当的反映和适应。国内吴斌等人对简化分类模型进行了扩展，并应用于 Web 文档聚类和移动通信客户关系管理中的客户行为分析。

5.2 蚁群算法

“蚁群算法”（ant colony optimization，ACO），又称蚂蚁算法，是一种受到自然界蚂蚁觅食行为的启发而提出的通过正反馈与分布式协作来寻找最优路径的随机优化算法。该算法能将问题求解的快速性、全局优化特征以及有限时间内答案的合理性结合起来，通过正反馈式的信息传递和积累、分布式计算以及贪婪启发式搜索保证求解的快速性与合理性。

蚁群算法是对自然界蚂蚁的寻径方式进行模拟而得出的一种仿生算法。蚂蚁在运动过程中，能够在它所经过的路径上留下一种称之为外激素（pheromone）的物质进行信息传递，而且蚂蚁在运动过程中能够感知这种物质，并以此指导自己的运动方向，因此由大量蚂蚁组成的蚁群集体行为便表现出一种信息正反馈现象：某一路径上走过的蚂蚁越多，则后来者选择该路径的概率就越大。

5.2.1 蚁群算法的发展简史

蚂蚁是地球上最常见、数量最多的昆虫种类之一，常常成群结队地出现于人类的日常生活环境中。这些昆虫的群体生物智能特征，引起了一些学者的注意。意大利学者 M.Dorigo，V.Maniezzo 等人在观察蚂蚁的觅食习性时发现，蚂蚁总能找到巢穴与食物源之间的最短路径。经研究发现，蚂蚁的这种群体协作功能是通过一种遗留在其来往路径上的叫做信息素（pheromone）的挥发性化学物质来进行通信和协调的。化学通信是蚂蚁采取的基本信息交流方式之一，在蚂蚁的生活习性中起着重要的作用。通过对蚂蚁觅食行为的研究，他们发现，整个蚁群就是通过这种信息素进行相互协作，形成正反馈，使多个路径上的蚂蚁逐渐聚集到最短的那条路径上来的。

M.Dorigo 等人于 1991 年首先提出了蚁群算法。其主要特点就是：通过正反馈、分布式协作来寻找最优路径。这是一种基于种群寻优的启发式搜索算法。它充分利用了生物蚁群能通过个体间简单的信息传递，搜索从蚁穴至食物间最短路径的集体寻优特征，以及该过程与旅行商问题求解之间的相似性，得到了具有 NP 难度（non-deterministic polynomical completeness）的旅行商问题的最优解答。同时，该算法还被用于求解 Job-Shop 调度问题、二次指派问题以及背包问题等，显示了其适用于组合优化类问题求解的优越特征。

1992 年，M.Dorigo 在他的博士论文中进一步提出了蚁群系统（ant system，AS）。在这篇论文中，根据信息素增量的不同计算方法，M.Dorigo 给出了三种不同的模型，分别称之为蚁周、蚁量和蚁密模型，同时通过大量实验，讨论了不同参数对算法性能的影响，确定了算法主要参数的有效区间。

由于基本蚁群算法进化收敛速度慢，且易陷入局部最优或者出现停滞现象等缺陷。许多学者相继对蚁群算法进行改进，提出了改进的蚁群算法。具有代表性的主要有：1996

年 Gambardella 和 M.Dorigo 提出的自适应蚁群算法（adaptive AS），该算法采用伪随机选择机制对 AS 的蚂蚁选择策略进行了改进，即将确定性选择和随机选择策略有机结合在一起；随后，Stutzle 等人提出了最大最小蚁群算法（max-min AS），该算法就信息素更新机制进行了改进，即只增加最佳路径的信息素浓度，且将各路径可能的信息素浓度限制在[Tmin， Tmax]，从而更好地利用历史信息加快收敛速度，避免较早地出现停滞现象。MMAS 是到目前为止解决 TSP、QAP 等问题最好的 ACO 类算法。

1998 年 ACO 的第一届蚁群优化国际学术会议（ANT'98）召开，更引起了研究者的广泛关注。各种改进方案不断涌现逐步完善了蚁群算法的性能。其中包括 Ant-Q 算法、带禁忌搜索策略的蚁群算法、多群体蚁群算法、具有变异特征的蚁群算法、自适应蚁群算法、与其他智能算法相结合的蚁群算法等。这些改进的蚁群算法在实际应用中都取得了很好的效果。但应当指出，现阶段对蚁群算法的研究还只是停留在仿真阶段，尚未能提出一个完善的理论分析，对它的有效性也没有给出严格的数学解释。

5.2.2 简化的蚂蚁寻食模型

为了说明蚁群算法的原理，先简要介绍一下蚂蚁搜寻食物的具体过程。

在蚁群寻找食物时，它们总能找到一条从食物到巢穴之间的最优路径。这是因为蚂蚁在寻找路径时会在路径上释放出一种特殊的激素。当它们碰到一个还没有走过的路口时，就随机地挑选一条路径前行，与此同时释放出相应的激素，路径越长，相同时间内经过的蚂蚁数越少，释放的激素浓度越低。当后来的蚂蚁再次碰到这个路口的时候，选择激素浓度较高路径概率就会相对较大，这样形成一个正反馈。最优路径上的激素浓度越来越大，而其他的路径上激素浓度却会随着时间的流逝而消减，最终整个蚁群会找出最优路径。

图 5.1 中的（a）、（b）、（c）、（d）依次为蚂蚁搜寻食物的具体过程。

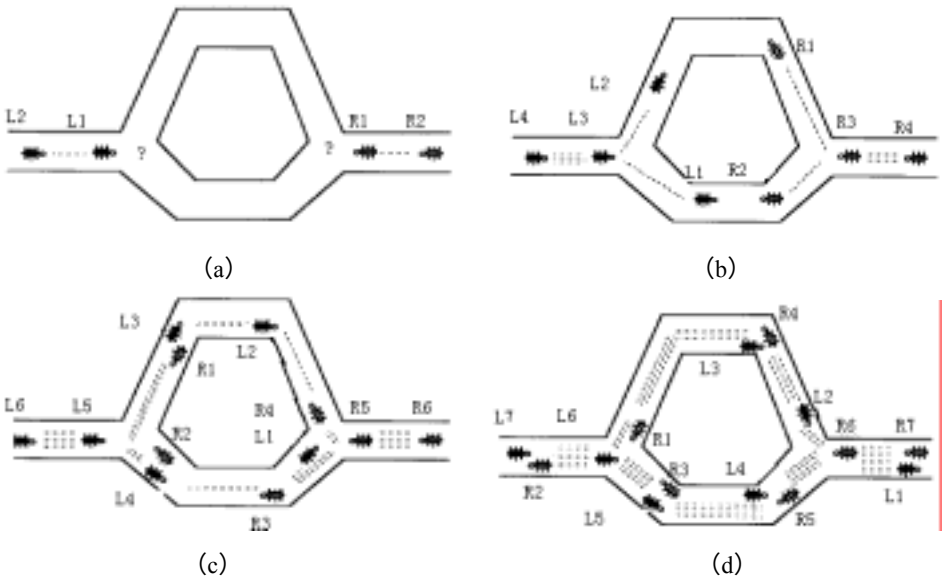


图 5.1　蚂蚁搜寻食物的具体过程

（1）蚂蚁到达决策点，准备选择路径；

（2）蚂蚁随机地选择路径，某些蚂蚁选择上面路径，某些选择下面路径；

（3）因为蚂蚁以近似的速度爬行，所以选择下面、较短路径的蚂蚁比选择上面、较长路径的蚂蚁更快达到对面的决策点；

（4）蚂蚁在较短路径上以更高的速率累积信息素，图中虚线的数目大致正比于蚂蚁留下的信息素总量。

定量分析：蚂蚁从 A 点出发，速度相同，食物在 D 点，可能随机选择路线 ABD 或 ACD，如图 5.2 所示。假设初始时每条分配路线有一只蚂蚁，每个时间单位行走一步，本图为经过 9 个时间单位时的情形：走 ABD 的蚂蚁到达终点，而走 ACD 的蚂蚁刚好走到 C 点，为一半路程。

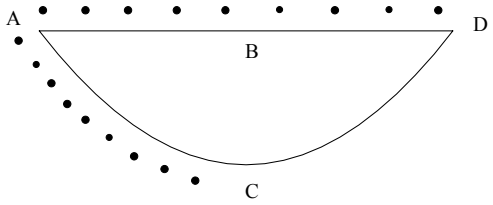


图 5.2　蚂蚁寻食路线图

图 5.3 为从开始算起，经过 18 个时间单位时的情形：走 ABD 的蚂蚁到达终点后得到食物又返回了起点 A，而走 ACD 的蚂蚁刚好走到 D 点。

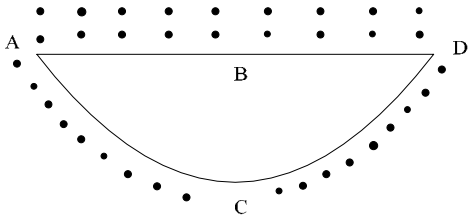


图 5.3　蚂蚁寻食路线图

假设蚂蚁每经过一处所留下的信息素为一个单位，则经过 36 个时间单位后，所有开始一起出发的蚂蚁都经过不同路径从 D 点取得了食物，此时 ABD 的路线往返了 2 趟，每一处的信息素为 4 个单位，而 ACD 的路线往返了一趟，每一处的信息素为 2 个单位，其比值为 2:1。

寻找食物的过程继续进行，则按信息素的指导，蚁群在 ABD 路线上增派 1 只蚂蚁（共 2 只），而 ACD 路线上仍然为 1 只蚂蚁。再经过 36 个时间单位后，两条线路上的信息素单位积累为 12 和 4，比值为 3:1。

若按以上规则继续，蚁群在 ABD 路线上再增派一只蚂蚁（共 3 只），而 ACD 路线上仍然为 1 只蚂蚁。再经过 36 个时间单位后，两条线路上的信息素单位积累为 24 和 6，比值为 4:1。

若继续进行，则按信息素的指导，最终所有的蚂蚁会放弃 ACD 路线，而都选择 ABD

路线。这也就是前面所提到的正反馈效应。

基于以上蚁群寻找食物时的最优路径选择问题，可以构造人工蚁群，来解决最优化问题，如旅行商（TSP）问题。作为蚁群算法的蚂蚁个体，其运动和通信的简单规则，只需包含以下几个主要方面：

（1）搜索范围：可具体设定蚁群个体的搜索参数半径，这样就限制了其运动过程中的观察能力和移动距离。

（2）局部环境：蚂蚁个体仅需要感知它周围的局部环境信息，并且该局部环境中的信息素是按一定速度消失的。

（3）觅食规则：每只蚂蚁只在其能感知的范围内进行信息探索和留存，在局部环境中，哪一点的信息素最多，就以较大的概率决定了它的运动方向。这样，虽然在其运动过程中，会出现小概率的搜索错误，但从总体上说，其搜寻的效率和正确性会通过其他蚂蚁的行为反馈加以调整。

（4）移动规则：每只蚂蚁都朝信息素最多的方向移动，当周围没有信息素指引的时候，蚂蚁会按照自己原来运动的方向惯性地运动下去，并且，在运动的方向上有一个随机的小的扰动，以保留原来的运动记忆。如果发现有其已经经过的地点，则以较大概率进行避让。

（5）避障规则：如果在蚂蚁即将移动的方向上存在障碍物，则它会随机选择另一个方向。或者按照信息素的指引继续其觅食行为。

（6）通信规则：实际上，每只蚂蚁是通过其信息素的播撒和感知来进行通信的。其具体规则是多元化的，它可以在找到相对最优解的时候散发最多的信息素，并且随着它走的距离越来越远，播撒的信息素越来越少。

人工构造的蚁群中把具有简单功能的工作单元看作蚂蚁，其与自然蚁群的相似之处在于都是优先选择信息素浓度大的路径。较短路径的信息素浓度高，所以能够最终被所有蚂蚁选择，也就是最终的优化结果；两者的区别在于人工蚁群有一定的记忆能力，能够记忆已经访问过的节点。同时，人工蚁群再选择下一条路径的时候是按一定算法规律有意识地寻找最短路径，而不是盲目的。例如在 TSP 问题中，可以预先知道当前城市到下一个目的地的距离。

5.2.3 基本蚁群算法的原理

基本的蚁群算法是与旅行商问题的求解联系在一起的。这里，我们使用通用的旅行商（TSP）问题作基准进行论述，这样可以很容易地与其他启发式寻优算法进行比较。虽然该模型定义要受所求解问题结构的影响，但从后面的论述中，我们将看出该方法可以很容易地拓展到其他优化问题的求解中。

1）基本定义

TSP 问题是一种典型的组合优化问题。其定义是：给定 n 个城市的集合，TSP 问题等价于寻找一条只经过各城一次的具有最短长度的闭合路径。

设 d_{ij} 为城市 i 和 j 之间的距离，

欧几里得空间中，$d_{ij}=\{(x_i-x_j)^2+(y_i-y_j)^2\}^{\frac{1}{2}}$

一个 TSP 问题可由图（N，E）给定，其中 N 是城市集合，E 是城市之间的支路集合

（欧几里得空间中 TSP 意义下的一个全连接图），令 $b_i(t)(i=1,2,3,...,n)$ 为在 t 时刻 i 城市上的蚂蚁数，则 $m=\sum_{i=1}^{n}b_i(t)$ 为总的蚂蚁数。

每只蚂蚁都可以是具有下列特征的简单智能体：

- 其选择城市的概率是城市之间的距离和连接支路上所包含的当前信息素余量的函数；
- 为了强制蚂蚁进行合法的周游，直到一次周游完成时，才允许蚂蚁游走已访问过的城市（这可由禁忌表来加以控制）；
- 当完成一次周游，它在每条访问过的支路 (i,j) 上留下一种叫信息素的物质；

令 $\tau_{ij}(t)$ 为支路 (i,j) 在时刻 t 上的信息素强度。每只蚂蚁在 t 时刻选择下一个城市，并在 $t+1$ 时刻到达那里。因此，若我们称由 m 只蚂蚁在区间 $(t,t+1)$ 内做的 m 次移动为 AS 算法的一次迭代，则算法每迭代 n 次（我们称为一个周期），每只蚂蚁就完成了一次周游。在这个时间点，信息素强度可根据下面公式进行更新：

$$\tau_{ij}(t+n)=\rho\tau_{ij}(t)+\Delta\tau_{ij} \tag{5-1}$$

其中 $1-\rho$ 代表了 t 时刻和 $(t+n)$ 时刻之间信息素的挥发程度，

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{5-2}$$

$\Delta\tau_{ij}^{k}$ 为第 k 只蚂蚁在 t 时刻与 $(t+n)$ 时刻之间留在支路 (i,j) 上的单位长度的信息素量（对真实妈蚁而言为气息）。其可按下式进行计算：

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{若第}k\text{只蚂蚁在本次循环过的支路}\\ 0, & \text{其他}\end{cases} \tag{5-3}$$

式中，Q 为常数，L_k 是第 k 只蚂蚁的周游长度。

系数 ρ 必须设为一个小于 1 的数，以避免信息素无限制地积累。在具体仿真实验中，为避免计算机出错，我们把 0 时刻的信息素强度 $\tau_{ij}(0)$ 设为一个小的正常数 c。

为满足一只蚂蚁访问所有 n 个城市的约束，我们将每只蚂蚁与一个称为禁忌表的数据结构联系起来，该表存储了直至时刻 t 的所有已访问城市，并禁止蚂蚁在 n 次迭代（一个周游）结束之前再次访问它们。一次周游结束时，该禁忌表可被用来计算蚂蚁的当前解（也即蚂蚁所走路线的长度）。计算之后可清空禁忌表，这时蚂蚁又可以自由选择路径了。我们定义 Tabu_k 为包含第 k 只蚂蚁禁忌表的动态增长矢量，tabu_k 为从 Tabu_k 的元素得到的集合，$\mathrm{tabu}_k(s)$ 为表中的第 s 个元素（也即第 k 只蚂蚁在当前游走中访问的第 s 个城市）。

蚂蚁要根据其周围的能见度大小和信息素的浓度多少来确定蚂蚁下一步要到达的城市。这里我们定义蚂蚁 k 从 i 城到 j 的转移概率为：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum k\in\text{allowed}_k[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}}, & j\in\text{allowed}_k\\ 0, & \text{otherwise}\end{cases} \tag{5-4}$$

其中 η_{ij} 称为能见度。

$$\eta_{ij}=\frac{1}{d_{ij}}$$

这个量在蚁群算法运行过程中不作修改。这与刚才按式（5-1）计算的信息素不同。$\text{allowed}_k=(n-\text{tabu}_k)$，$\alpha$ 和 β 为控制信息素和能见度之间相对重要性的参数。因此，转移概率是能见度（意味着近的城市被选中的概率大，这样就执行了贪婪性试探法）和 t 时刻信息素浓度之间的折衷方案（意味着若在此支路 (i,j) 上有很大交通量时，那么这条支路是很有吸引力的，这样就执行了自催化过程）。

2）算法描述

根据信息素更新方式的不同，M.Dorigo 提出了三种不同的模型：蚁周（ant-cycle system），蚁量（ant-quantity system），蚁密（ant-density system）模型。其中蚁周算法具有代表性，根据前面的描述，蚁周算法可以表述如下：

在零时刻，首先进行的是初始化步骤，将蚂蚁安置在不同的城市，同时设置支路上信息素的初始值 $\tau_{ij}(0)$，所有蚂蚁禁忌表内第一个元素被设成它的出发点。此后，每只蚂蚁从城 i 向城 j 游走，它选择城市的概率是两个满意度指标的函数。其中，$\tau_{ij}(t)$ 表明过去有多少蚂蚁选择了相同的支路 (i,j)，而能见度 η_{ij} 意味着城市越近就越可取。若设 $\alpha=0$，就意味着不再考虑信息素水平，这样就得到了有多重起点的随机贪婪算法。

在经过 N 次迭代后，所有的蚂蚁都完成了一次周游，它们的禁忌表将全满。此时已算得每只蚂蚁 k 的 L_k 值，这样，$\Delta\tau_{ij}^k$ 的值也得到了更新。同时，可以存储蚁群所找到的最短路径（即 $\min L_k$，k =1，2，3，…），清空所有的禁忌表。这个过程可一直迭代到周游计数器达到最大值（由用户定义）$NC_{\max}$，或所有蚂蚁都在做相同的周游。

我们将后一种情况称为停滞行为，因为它表示算法停止了可选解的搜索。

在蚁群算法的另两个版本：蚁密和蚁量算法模式下，信息素更新的方式上是有差异的。在这两种模型中，每只蚂蚁在每一步后都留下了它的信息素，而不必等到周游结束。

在蚁密算法中，蚂蚁每次从 i 到 j 都会在支路 (i,j) 上留下数量为 Q 的痕迹；

在蚁量算法中，一只从 i 到 j 的蚂蚁在支路 (i,j) 上留下数量为 Q/d_{ij} 的痕迹。

故而在蚁密算法中，我们有：

$$\Delta\tau_{ij}^k=\begin{cases}Q, \text{若第}k\text{只蚂蚁在其本次循环过的支路}\\ 0, \text{其他}\end{cases} \tag{5-5}$$

在蚁量算法中，我们有：

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{Q}{d_{ij}}, \text{若第}k\text{只蚂蚁在本次循环过的支路}\\ 0, \text{其他}\end{cases} \tag{5-6}$$

从这些定义可以很清楚地看到，在蚁密算法中，当一只蚂蚁从 i 到 j 时，支路 (i,j) 上的信息素的增强与 d_{ij} 无关；而在蚁量算法中，信息素的增强与 d_{ij} 成反比，即短路径对蚁群而言更有吸引力。蚁周模型采用了全局信息，而蚁量-蚁密模型采用的是局部信息。仿真结果证明蚁周模型优于蚁量、蚁密模型。

3）算法特点

在以旅行商问题为代表的组合优化类问题求解中，基本蚁群算法体现的优点如下：

（1）通用性。它可用于解同一类型的优化问题，从 TSP 问题到 ATSP 只需作直接扩展即可。

（2）鲁棒性。只要作很小的改动，就可将蚁群算法用于求解其他组合优化问题，如二次分配问题和作业调度问题。

（3）群体性。这是一种基于群体的方法，它允许采用正反馈作为搜索机制。

（4）并行性。该算法适用于并行操作，在求解大规模的优化问题时，不仅可以从算法本身的优化出发来提高求解效率，也可以从算法的执行模式出发来进行研究。

基本蚁群算法的不足之处：

（1）需要较长的计算时间，容易出现停滞现象。

（2）根据式（5-3），所有通过路段（i，j）的搜索路径对应的候选解均会对该路段带来信息素的增量。而实际上，候选解并非都是最优解，这样计算信息素的增量会导致错误的引导信息，从而造成大量的无效搜索，使系统出现停滞现象。

（3）式（5-3）中，采用了信息素均匀分配策略，即对已搜索路径中的所有路段采用同样的信息素增量，与路段的重要性无关。没有考虑当连续空间优化问题转换到有向图搜索问题时，信息素分配给可行解带来的尺度变化对于连续解空间搜索效率的影响。

5.2.4 蚁群算法求解步骤

如图 5.4 所示，利用蚁周算法求解 TSP 问题的步骤如下：

（1）初始化过程：设 t:=0;　/* t 时间计数器 */

Nc:=0;　/* Nc 为循环次数计数器 */

$\tau_{ij}(0)$：=C;　/* 为每条路径（i， j）设一个信息素强度的初始值 */

$\Delta\tau_{ij}$=0;　/* 信息素强度增量的初值设为 0 */

η_{ij} 由某种启发式算法确定；/* 在 TSP 中，$\eta_{ij}=1/d_{ij}$ */

$tabu_k = \Phi$；/* 在初始阶段，禁忌表为空 */

将 m 只蚂蚁随机置于 n 个节点（城市）上。

（2）设置 s:=1;　/* s 为禁忌表索引，将各只蚂蚁的初始城市置于当前禁忌表中 */

for i:=1 to n　do;

for k:=1 to m　do;

$tabu_k(s)$=i；/* 把第 k 只蚂蚁的起始城市位置存入禁忌表 */。

（3）循环执行以下步骤，直到禁忌表满为止　/* 这一步要循环（n-1）次 */

设置 s:= s +1;

for i:=1 to n　do;

for k:=1 to m　do;

以概率 $p_{ij}^{k}(t)$ 选择下一个城市 j，其概率具体由式（5-4）给定；

在 t 时刻，蚂蚁 k 在城市 $i = tabu_k(s-1)$ 将蚂蚁 k 移到城市 j；

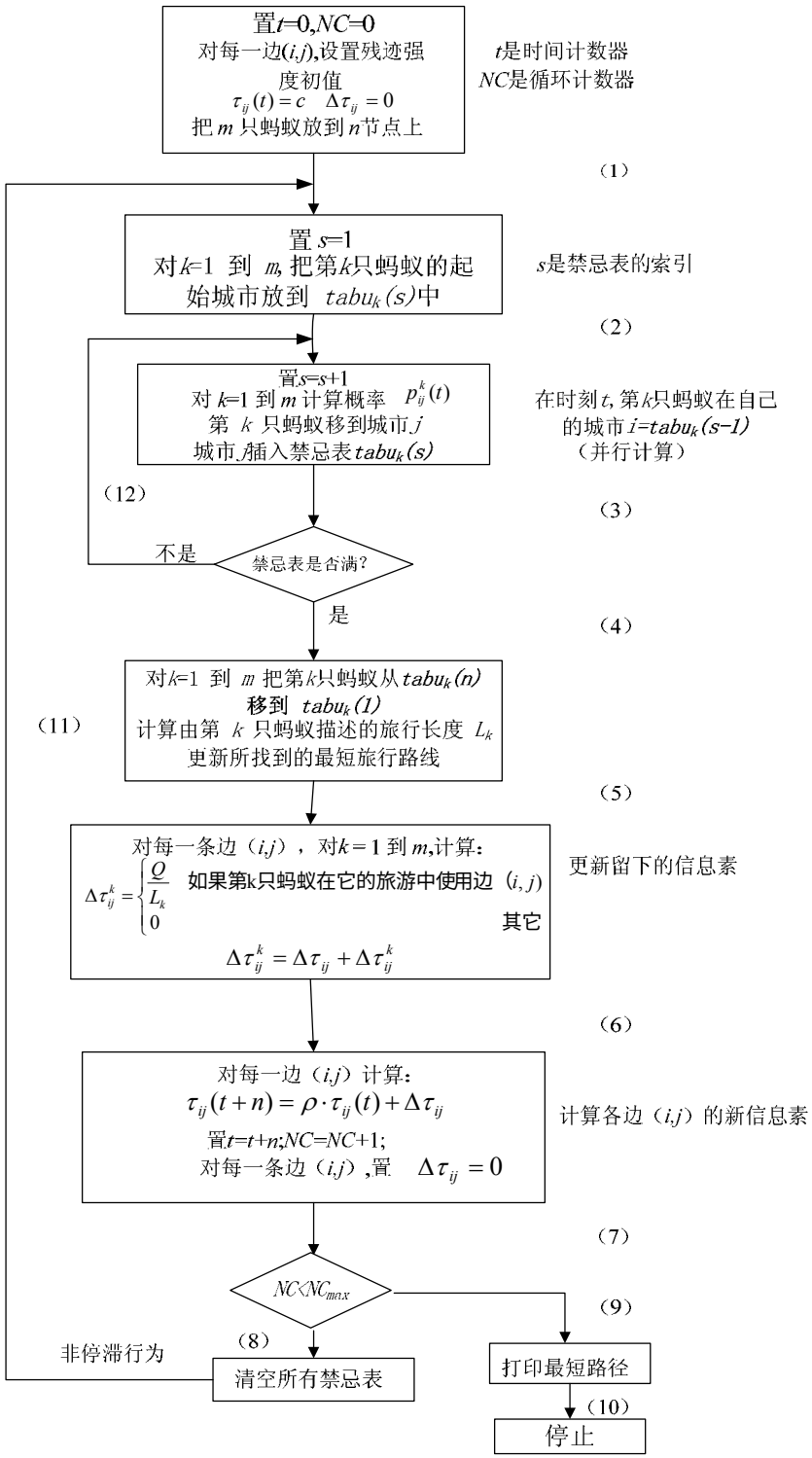


图 5.4 蚁群算法求解 TSP 问题流程图

将刚刚选择的城市 j 加到 $tabu_k(s)$ 中。

（4）for k:=1 to m do

把蚂蚁 k 从 $tabu_k(n)$ 移回到 $tabu_k(1)$，一次循环后蚂蚁重新回到初始位置；

计算蚂蚁 k 所周游的长度；

更新所找到的最短周游路线；

for 每条支路（i，j）

for k:=1 to m　do

根据式（5-3）计算 $\Delta\tau_{ij}^{k}$。

（5）for 每条支路（i，j）

根据式（5-1）计算信息素强度 $\tau_{ij}(t+n)$

设 t:=t+n

NC:=NC+1

for 每条支路（i，j）设 $\Delta\tau_{ij}=0$。

（6）if　$NC < NC_{max}$ 且无停滞行为

清空所有的禁忌表

返回步骤（2）；

else

输出最短路径。

5.2.5 蚁群算法算子分析

蚁群算法的主要依据是信息正反馈原理和某种启发式算法的有机结合，转移概率公式揭示了这一原理。蚁群算法同样不包含直接的操作算子。

蚁群算法的优化过程主要包括选择、更新以及协调三个过程。在选择过程中信息素越多的路径被选择的概率越大；在更新过程中路径上的信息素随蚂蚁的经过而增加，同时也随时间的推移挥发消失；在协调过程中蚂蚁之间通过信息素进行信息交流相互协作。在选择和更新过程中较好的解（较短的路径）通过路径上的信息素得到加强，从而引导下一代蚂蚁向较优解邻域搜索使算法收敛；但同时更新过程的信息素挥发又使得算法具有探索能力增加解的多样性，使得算法不易陷入局部最优。

同遗传算法（GA）比较来看路径上的信息素量相当于 GA 中适应度函数即评价路径的优劣；而蚂蚁按概率选择路径的操作则相当于 GA 中的选择交叉算子，它以信息素为媒介实现较优解之间的优化重组，从而保证了算法的收敛性；信息素挥发起到 GA 中变异算子的作用，保证了群体的多样性。

在蚁群算法中各个参数对算法性能影响极大，起主要作用的参数有控制信息素相对重要性的启发式因子 α，控制能见度相对重要性的期望启发式因子 β 和信息素残留常数 ρ 等 3 个参数。它们是影响蚁群算法利用/探索关系（exploitation/exploration relationship，简称 EER）平衡的主要因素，下面就这几个参数详细讨论它们对算法性能的影响。

信息启发式因子 α 反映蚂蚁在运动过程中所积累的信息量（即残留信息量 $\tau_{ij}(t)$ 在指导蚁群搜索中的相对重要程度，期望值启发式因子 β 反映蚂蚁在运动过程中启发信息（即期望值 η_{ij}）在指导蚁群搜索中的相对重要程度。

α 的大小体现了蚁群在路径搜索中随机性因素作用的强度，其值越大，蚂蚁选择以前走过的路径的可能性越大，搜索的随机性减弱。但当 α 值过大将会使蚁群的搜索过早陷于局部最优。β 的大小则体现了蚁群在路径搜索中先验性、确定性因素作用的强度，其值越大，蚂蚁在某个局部点上选择局部最短路径的可能性越大。虽然搜索的收敛速度得以加快，但减弱了蚁群搜索过程中的随机性，使算法易于陷入局部最优。

蚁群算法的全局寻优性能，要求蚁群的搜索过程必须具有很强的随机性；而蚁群算法性能的影响和作用是相互配合、密切相关的。

在搜索开始阶段，各条路径上的信息素相差不明显，主要由启发因子引导搜索，属于局部信息的搜索过程。通过信息正反馈，较好路径上的信息素逐渐增大，故在搜索后期路径上残留的信息素对搜索起主导作用属于全局信息的搜索过程。且当 α =0 时蚁群算法就等同于传统的贪婪算法，仅由路径信息决策；当 β =0 时算法就变成纯粹的正反馈的启发式算法。因此，应恰当地选择 α 、β 使算法收敛全局最优解。

在蚁群算法中，人工蚂蚁是具有人类记忆功能的，随着时间的推移，以前留下的信息将逐渐消逝。在算法模型中用参数 $1-\rho$ 表示信息消逝程度，而 ρ 就是信息素残留系数。

信息素挥发度 $1-\rho$ 的大小直接关系到蚁群算法的全局搜索能力及其收敛速度：由于信息素挥发度 $1-\rho$ 的存在，当要处理的问题规模比较大时，会使那些从来未被搜索到的路径（可行解）上的信息量减小到接近于 0，因而降低了算法的全局搜索能力。而且当 $1-\rho$ 过大时以前搜索过的路径被再次选择的可能性过大，将会减弱算法的随机性能使之陷入局部极值，反之，通过减小信息素挥发度 $1-\rho$ 虽然可以提高算法的随机性能和全局搜索能力，但算法的收敛速度会随之降低。特别是当 $1-\rho$ 很小时，由于路径上的残留信息占主导地位，信息正反馈的作用相对较弱，搜索的随机性增强，因而蚁群算法收敛速度很慢；而在 $1-\rho$ 比较大时，由于信息正反馈的作用占主导地位，搜索的随机性减弱，虽然算法收敛速度加快，但易于陷入局部最优状态。总之，应恰当地选择参数 α 、β 、ρ 使算法最终取得 ERR 平衡收敛到全局最优点。

5.2.6 蚁群算法的改进

针对蚁群算法的基本特点和蚁群算法的算子分析，作为一种体现群体协作寻优特征的启发式优化方法，蚁群算法的改进模式可以从以下两方面进行考虑：

（1）算法的总体结构：在这里，需要考虑的是蚁群的总体结构和组织模式。当然，各子系统间的信息联系模式也需要考虑。如多群体蚁群算法和混合型蚁群算法等，就是由几个蚁群来协同解决 TSP 问题的算法，而传统的蚁群算法中只有一个蚁群。而且，在这些算法的信息联系模式中，蚁群群体层的交互还使用了正负两种信息素效应。由于引入了群体层的交互作用，蚁群能更好地交换问题解决过程中的规划信息，并保持它们在搜索过程中的多样性。另外，还可考虑不同信息交换方式的多重蚁群算法和具有分工特征的蚁群算法等改进方向。

（2）算法的具体参数设定和调整策略：在这里，可以对参数的选择和变化模式进行设定，如最大最小蚁群算法的使用就形成了比传统蚁群算法更贪婪的搜索模式。另外，带变异特征的蚁群算法、Ant-Q 算法、带禁忌搜索策略的蚁群算法等，也在具体的算法参

数优化方面取得了一定的效果。如果在基本蚁群算法中引入变异机制，对参数进行变异式改进，则可使算法既有较快的求解速度又有较高的求解精度。而自适应蚁群算法、动态蚁群算法则是一种分布平衡的蚁群算法，可以在加速收敛和防止早熟、停滞现象之间取得很好的平衡。如果将随机特征引入算法参数的修正模式，则所形成的随机扰动蚁群算法必然具有很强的全局搜索能力。如果对算法关键参数进行分配策略和修正策略研究，采用基于目标函数值的信息素分配策略等，则可根据目标函数值来自适应调整蚁群的搜索行为，从而保证算法快速找到全局最优解。当然，也可将其他的启发式算法用于信息素分配，以及搜索过程中最优解的筛选等过程，并且与其他智能算法相结合，形成一些效果较好的改进型蚁群算法。

下面我们将介绍几种比较典型的改进蚁群算法：

5.2.6.1 最大最小蚁群算法（max-min ant system）

最大最小蚁群算法（MMAS）是由德国学者 T.Stuctzle 和 H.Hoos 提出的，这种改进型蚁群算法的性能与传统蚁群算法的共同点是：在算法的每次迭代中只允许表现最好的蚂蚁更新路径上的信息素，该算法目的主要在于防止过早的算法停滞现象。它们的基本做法是：限定信息素浓度允许值的上下限，并且采用了平滑机制。

算法过早的停滞（早熟）现象出现是因为信息素过分集中到几条较好的路径上，而其他路径则由于长时间没有蚂蚁经过，信息素逐渐挥发，致使路径上的信息素浓度趋于 0。这样，蚂蚁就几乎不会再选择这些信息素浓度极低的路径，从而丧失了探索新路径的可能，算法就表现出停滞的特征。为了减少在搜索早期算法出现停滞的可能，可以对算法中信息素浓度引入最大值和最小值限制，将每条路径上的信息素 $\tau_{ij}(t)$ 限定在 $[\tau_{\min},\tau_{\max}]$ 之间，即 $\tau_{\min} \le \tau_{ij}(t) \le \tau_{\max}$。在每次迭代之后，都保证信息素浓度满足该式。如果 $\tau_{ij}(t) > \tau_{\max}$，则 $\tau_{ij}(t) = \tau_{\max}$；同样，如果 $\tau_{ij}(t) < \tau_{\min}$，则 $\tau_{ij}(t) = \tau_{\min}$。当然，同时还应该保证 $\tau_{\min} > 0$，并且，如果对于所有的解元素有 $\eta_{ij} < \infty$，那么选择一特定解元素的概率不为 0。

在搜索过程中，可以由下式设置信息素 $\tau_{\max}$ 的最大值为一个最大极限值的估计，其中，s^{gb} 为全局最优解：

$$\lim_{t\to\infty} \tau_{ij}(t) = \tau_{ij} \le \frac{1}{(1-\rho) f(s^{gb})} \tag{5-7}$$

当得到最优解时，按照上式更新 $\tau_{\max}$，这样便可得到一个动态的变化值 $\tau_{\max}(t)$。

为了决定 $\tau_{\min}$，可以使用下面的假设：

（1）最优解是在搜索停滞前得到的。在这种情况下，算法的一次迭代中找到全局最优解的概率就远远大于 0，因为较好的解可能就处于接近最优解的地方。

（2）影响解的一个主要原因取决于信息素最大和最小值限制的差异，而不是启发式信息。

根据上面的假设，可以选择一个合适的 $\tau_{\min}$ 值来提高算法的收敛性。这里先给定一个 P_{best}（$P_{dec} = \sqrt[n]{P_{best}}$），然后根据下式设置 $\tau_{\min}$：

$$P_{dec} = \frac{\tau_{\min}}{\tau_{\max} + (\overline{\tau} - 1)\tau_{\min}} \tag{5-8}$$

式中，$\tau_{\min}=\dfrac{\tau_{\max}(1-P_{dec})}{(\overline{\tau}-1)P_{dec}}=\dfrac{\tau_{\max}(1-\sqrt[n]{P_{best}})}{(\overline{\tau}-1)\sqrt[n]{P_{best}}}$

如果 $P_{best}=1$，那么 $\tau_{\min}=0$。如果 P_{best} 太小，通过上式计算得到的值可能会出现 $\tau_{\min}>\tau_{\max}$，这时可设置 $\tau_{\min}=\tau_{\max}$。

信息素浓度最大值和最小值的选择取决于平均路径的长度。通过设定信息素的上下限，不会使某条路径的信息素浓度过高，从而导致过早停滞，也不会因为某些路径的信息素浓度过低而导致算法发现新路径可能性的降低。这样，通过对信息素浓度的限制，就降低了算法搜索中的早期收敛（停滞）问题。特别是，对于要求迭代次数较多的问题求解过程，这样的寻优模式将具有更好的效果。

在最大最小蚁群算法中，只允许其中的一个路径更新信息素。该路径通常是最优路径，它可以是在所有周游过程中已经找到的最优路径，也可以是在当前周游中找到的最优路径。信息素可按照下面的公式进行更新：

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}^{best} \tag{5-9}$$

式中，$\Delta\tau_{ij}^{best}=1/f(s^{best})$，并且 $f(s^{best})$ 表示该次迭代中的最优路径（s^{ib}），或者是全局最优路径（s^{gb}）的代价函数。仅使用一只蚂蚁来更新所找到路径的方法在传统蚁群算法中也可以应用。不过在传统算法中，使用的仅是（s^{gb}），当然也有有限的试验使用了（s^{ib}）。当仅使用（s^{gb}）对信息素进行更新时，搜索可能会过快地收敛，对可能的较好解的范围进行开发的过程也将受到限制，这将会导致算法的局部收敛。如果只使用（s^{ib}），则可避免上述情况。因为，该次迭代的最优路径（解）会随着迭代次数的不同而显著变化，并且会有较大数量的路径（解）得到增强。当然，这里还可以使用混合策略，比如使用（s^{ib}）来作为信息素更新的默认值，而仅在固定的迭代次数间隔时使用（s^{gb}）。

事实上，当我们使用最大最小蚁群算法，并结合局部搜索算法来解决大规模的 TSP 问题或者二次配置经典实例时，最好的策略应该是使用一种动态的混合策略，即：在搜索的过程中，增加使用（s^{gb}）的频率来对信息素进行更新。这样，在初始化时，每条边上的信息素 $\tau_{ij}(0)=\tau_{\max}$；在每次迭代后，信息素浓度以 ρ 挥发，并且只有找到最优路径的蚂蚁才能增加它的信息素浓度，使其保持较高的水平。因此，较差路径上的信息素浓度就会增加较慢，而只有最优路径上的信息素浓度才会维持一个较高的水平，并且被更多的蚂蚁所选择。

这样，最大最小蚁群算法相对于基本的蚁群算法，其性能必然有相当大的提高。但是这里需要指出的是，尽管在算法中对信息素浓度进行了最大和最小值的限制，但最大最小蚁群算法仍然可能出现停滞现象。也就是说，只采用最大最小痕迹浓度的限制还不足以在较长的运行时间里持久消除停滞现象。因此，这里可采用信息素的平滑机制：按照比例更新的方法来调整信息素的浓度，使信息素浓度的增加值正比于 $\tau_{\max}$ 与边（i，j）上的信息素浓度 $\tau_{ij}(t)$ 的差值，即：

$$increase\propto\tau_{\max}-\tau_{ij}(t) \tag{5-10}$$

在最大最小蚁群算法中，还可结合局部搜索算法。局部搜索算法是从一个初始解

$i \in S$ 开始的，然后运用一个产生器，持续在解 i（当前解）的邻域 S_i 中搜索比 i 更优的解。若找到比 i 更优的解，就用这个解取代 i 成为当前解，再继续计算；否则算法终止，当前解就是算法的最终解。在该算法中可使用局部搜索算法的原因有两点：一方面，通过结合局部搜索模式，可以提高算法的性能，从而可以在早期找到较好的解，并且能更直接地引导学习机制；另一方面，最大最小蚁群算法还可以为随后的局部搜索阶段构造一条好的初始路径，这样就可以逐步找到一个接近最优的路径。蚁群算法与局部搜索方法相结合，就提高了局部搜索的效率。实验也表明最大最小蚁群算法在寻找解的有效性方面和防止算法的过早停滞方面有了较大改进。

该算法的一个不足之处在于它运行时间较长。运行时间会随着问题规模的增加而大大增加。为了降低运行时间，可以考虑增加候选解集合的选择概率。这样，除了可以降低算法的运行时间之外，还可以增加该算法的性能。这些改进模式可以通过对一些较大规模的寻优问题求解加以证明。

5.2.6.2 具有随机扰动特征的蚁群算法

蚁群算法的主要依据是信息正反馈原理和某种启发式算法的有机结合。但是，根据式（5-3），所有通过路段（i，j）的搜索路径对应的候选解均会对该路段带来信息素的增量。而实际上，候选解并非都是最优解，这样计算信息素的增量会导致错误的引导信息，从而造成大量的无效搜索，使系统出现停滞现象。故可以采用可变的扰动因子，使系统跳出局部最优。策略如下：

（1）在最初的几次迭代中，为加速算法的收敛，应取较大的转移概率，但概率如果一直不变必将导致随后的搜索出现停滞现象；

（2）在随后的搜索过程中仅在最优值路径上保持原来的转移概率，其他路径上则适当减小转移概率，这样一方面可以提高路径选择的多样性（即起到一定的扰动作用），另一方面可以使收敛趋于平缓。

（3）为了防止最优的一条路径可能被漏选，故设计如下具有随机扰动特性的转移系数：

$$C_{ij}(k)=\begin{cases}(\tau_{ij}(k)\eta_{ij}(k))^{\gamma}, & \tau_{ij}(k)=\max\left(\tau_{is}(k)\right), s\notin \text{tabu}(k)\\ (\tau_{ij}(k)^{\alpha}\eta_{ij}(k)), & \tau_{ij}(k)=\tau_{is}(k)-\max\left(\tau_{is}(k)\right),\\ & \text{且 } p\le\tau_0, s\notin \text{tabu}(k)\\ (\tau_{ij}(k)\eta_{ij}(k))^{\gamma}, & \tau_{ij}(k)=\tau_{is}(k)-\max\left(\tau_{is}(k)\right),\\ & \text{且 } p>\tau_0, s\notin \text{tabu}(k)\\ 0, & \text{其他}\end{cases} \tag{5-11}$$

s 为 $\max(C_{ij}(k))$ 中所对应的城市 j。上式中，γ 为具有倒指数的扰动因子；$\tau_0\in(0,1)$ 称为随机变异率；p 是（0，1）中均匀分布的随机数。

公式（5-11）表明：某次迭代过程中某只蚂蚁有若干条路径可选，对于信息素密度最大的那一条路径，应用转移概率公式（5-4），而对于其他的可选路径，采用随机选择方式。该公式是确定性选择与随机选择相结合的产物。确定性选择导致蚂蚁总是选择转移系数最大的路径，随机选择导致计算转移系数时具有较强的随机性，正是两者的共同作用才

使改进的蚁群算法具有更强的全局搜索能力。

5.2.6.3 具有自适应特征的蚁群算法

鉴于基本蚁群算法利用随机选择策略，使得进化速度较慢，正反馈原理旨在强化性能较好的解，却容易出现停滞现象。通过采用确定性选择和随机选择相结合的选择策略，并且在搜索过程中动态地调整确定性选择的概率，当进化到一定代数后，进化方向已经基本确定，这时对路径上信息量作动态调整，缩小最优和最差路径上的信息量的差距，并且适当加大随机选择的概率，以利于对解空间的更完全搜索，从而可以有效地克服基本蚁群算法的两个不足。算法按照下式确定蚂蚁 k 由 i 转移到的下一城市 j：

$$j=\begin{cases}\arg\max_{u\notin \text{allowedk}}\left(\tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)\right), & \gamma\le\rho_0\\ \text{依概率}\ \ p_{ij}^{k}(t)\text{选择}\ j & \text{其 他}\end{cases} \tag{5-12}$$

上式中 $\rho_0\in(0,1)$，γ 为（0，1）中均匀分布的随机数。当进化方向基本确定后用简单地放大（或缩小）方法调整每一路径上的信息量。实验表明，由于采用自适应选择和动态调整策略，算法的性能明显得到改善。该方法不仅能够加快收敛速度，节省搜索时间，而且能够克服停滞行为的过早出现，有利于发现更好的解，这对于求解大规模优化问题是十分有利的。

5.2.7 蚁群算法的应用领域

在蚁群优化问题的求解研究中，所需要做的工作主要是合理运用其答案选择、信息更新和群体协调机制，尽量避免其算法缺陷，同时针对各种不同的优化问题求解领域，设计不同的蚁群算法来加以解决。对于蚁群算法的优化问题求解过程，其本质在于：

- 合理的选择机制。信息量越大的路径，被选择的概率越大。
- 合理的更新机制。路径上面的信息量会随蚂蚁的经过而增长，同时也会随着时间的推移逐渐减少。
- 合理的协调机制。蚂蚁之间实际上是通过信息量来进行互相通信和协同工作的，这样的自组织机制使得蚁群算法具有很强的发现较好解的能力。

蚁群算法在解决很多组合问题（combinatorial problems）上都取得比较理想的效果。其中有两个比较著名的组合问题： TSP 问题（traveling salesman problem）和 JSP 问题（job-shop scheduling problem）。改进的蚁群算法可以比较好地解决这两个组合问题。另外，将蚁群算法应用于其他实际问题的解决也取得一定的进展，如大规模集成电路中的综合布线以及电信网络中的路由等方面的应用。

1）TSP 问题

TSP 问题（旅行商问题）是一种复杂的组合优化问题，其算法是具有 NP（non-deterministic polynomical completeness）难度的。目前尚无很好的求解方法。旅行商问题可简单描述为：求一条通过全部 N 个城市一次且仅一次的最短旅行路线。研究表明，蚁群算法可以有效地求解这类问题。

2）job-shop 调度问题

生产调度问题可以描述为：在一定的时间范围内为完成特定的生产任务而分配共享资源，并使得预定的某些生产指标最优。生产调度问题的解决过程本质是对一个资源限制问题的寻优过程。其最大的困难是计算的复杂性，因此需要找出有效的调度规则和方法，从而减少寻优问题的求解空间，缩短寻优过程。

3）大规模集成电路综合布线

大规模集成电路中的综合布线问题可以采用蚁群算法的思想来解决。在布线过程中，各个引脚对蚂蚁的引力可根据引力函数来计算。各个线网 Agent 根据启发策略，像蚁群一样在开关盒网格上爬行，所经之处便布上一条金属线，历经一个线网的所有引脚之后，线网便布通了。给定一个开关盒布线问题，问题的计算量是固定不变的，主要由算法的迭代次数决定，而迭代次数由 Agents 的智能和开关盒问题本身的性质确定。蚁群算法本身的并行法，使之比较适合于解决布线问题。

4）电信网络路由

电信网络中的路由是通过路由表进行路由选择。在每个节点的路由表中，对每个目的节点都列出了与该节点相连的节点，当有数据包到达时，通过查询路由表可知道下一个将要到达的节点。首先对路由表中的信息素强度进行初始化。在节点 x，以节点 i 为目的地址，邻节点为 j 处的信息素强度为 $\tau_{ih}=1/d_{ih}$，d_{ih} 为从 x 经节点 j 到节点 i 路径的最小费用值。然后周期性地释放蚂蚁来进行路由，并修改相应的信息素的值。仿真结果表明，无论呼叫是均匀分布还是集中分布，利用蚁群算法所得呼叫拒绝率和平均路径长度均小于最小负载法结果；在呼叫符合集中分布时，蚁群算法所得呼叫拒绝率低于最短路径法。

总之，蚁群算法能将问题求解的快速性、全局优化特征以及有限时间内解的合理性结合起来，并且能够与旅行商问题的求解进行直接的对应，因此对于能够直接转化为路径优化问题的组合类寻优问题求解，应用蚁群算法是十分有效的，如交通、机器人路径寻优、电力系统路径优化配置、制造路径优化、通信路由选择、计算机网络管理等。如果将其进行一定程度的拓展，蚁群算法也可被用于约束优化和二次配置问题的求解。另外，许多类型的流程调度问题也是典型的组合优化类寻优问题，也是适合用蚁群算法来进行求解的。

5.3 实例分析：改进蚁群算法在运输调度规划中的应用

运输调度问题（vehicle routing and scheduling problems，VRSP） 是指在给定的约束条件下，为了减少运输成本，满足客户要求，尽可能给出最优的车辆行车路线（vehicle routing problems，VRP ） 设计和最好的出行时间（vehicle scheduling problem，VSP） 安排。在实际调度问题中，它的约束条件包括车辆的装载量、运输服务的时间窗、运输工

人的工作时间以及运输路线的长度等。

5.3.1 改进的蚁群算法

针对基本蚁群算法容易陷入局部最优及收敛速度慢等缺陷，为了解决运输调度问题，提高蚁群算法的搜索速度和全局寻优能力是算法改进的主要目标。

1）改进收敛速度

挥发系数的自适应调节在任务区域较大的情况下，由于信息量的挥发系数ρ存在，那些从未被搜索到的解上信息量会减小到接近于0，从而降低了算法的全局搜索能力；反之当ρ过大时，随着解信息量的增大，以前搜索过的路径被选择的可能性极大，同样会影响到算法的全局搜索能力。若减小ρ，算法的全局搜索能力会随之提高，但收敛速度会变慢。针对这种情况，可以对ρ值采取自适应控制策略，见式（5-13）。在迭代的初始，ρ的初始值可以取较大值，保证收敛速度，随着迭代次数的增加，ρ逐渐减小，保证算法的全局寻优。同时应有下限（最小挥发系数）$\rho_{\min}$，以避免收敛过慢，影响算法性能。

$$\rho(i+1)=\begin{cases}\mu\rho(i) & \rho(i+1)\geq\rho_{\min}\\ \rho_{\min} & \rho(i+1)<\rho_{\min}\end{cases} \tag{5-13}$$

式中，μ为挥发约束系数，为了保证一定的收敛速度，μ一般取值为0.5~0.9，$\rho_{\min}$的值一般取为0.1。

2）改进全局寻优能力

当采用蚁群算法得到一组局部极小值x_k后，采用遗传算法将所得的值进行遗传变异，即以一个很小的概率随机地改变二进制表达式中的某些位，使得相应的位从1 变为0 或从0 变为1，从而α、β和Q的值在其整个取值域内发生改变。在遗传变异过程中，舍去比x_k劣的解，保存比x_k优的解，再执行蚁群算法。经此遗传变异后，可以跳出局部极小的区域，使解的质量得到提高。

3）算法步骤

步骤1：初始化α，β等参数，置$\tau_{ij}\leftarrow 0$，迭代次数nc $\leftarrow$0。

步骤2：将m 只蚂蚁置于初始点，并将初始点置于当前解集$\mathrm{tabu_k(s)}$中。

步骤3：对每个蚂蚁k（ k = 1， …， m ），按概率$\mathrm{p_{ij}^{\ k}}$ 移至下一城市j；将城市j置于$\mathrm{tabu_k(s)}$中。

步骤4：依式$\tau_{ij}=(1-\varepsilon)\tau_{ij}+\varepsilon\tau_0$进行信息素局部更新。

步骤5：应用3-opt法完善局部搜索，计算各蚂蚁的目标函数，并记录局部最优解。

步骤6：对一组局部极小值$\mathrm{x_k}$进行小概率的遗传变异，舍去比$\mathrm{x_k}$劣的解，保存比x_k优的解。

步骤7：利用式（5-13）计算自适应的挥发系数。

步骤8：依式$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\rho\cdot\Delta\tau_{ij}^{gb}$ 进行信息素全局更新。

步骤9：置$\tau_{ij}\leftarrow 0$ ，nc $\leftarrow$nc + 1。

步骤10：若nc < $\mathrm{nc_{max}}$ ，则转步骤2。

5.3.2 运输调度问题数学模型

VRSP问题描述为：某中心仓库，拥有容量为Q的车辆m辆，$V=\{k\}$，$k=1$，2，…，m，其中m为待定车辆数，它负责对n个客户进行货物派送，客户集为$C=\{i\}$。客户i的需求量为q_i，车辆必须在一定时间范围内$[a_i,b_i]$内到达，从客户i 到客户j 的费用为c_{ij}，车辆到达客户i 的时间为s_{ik}，则$s_{ik}\in[a_i,b_i]$。如何确定车辆数m 和规划运输路线，使总费用最少？

经上述描述，VRSP的数学模型可以表示为：

$$\min Z=\sum_{i\in C}\sum_{j\in C}\sum_{k\in V}c_{ijk}x_{ijk} \tag{5-14}$$

$$\sum_{i\in C}q_i\sum_{j\in C}x_{ijk}\le Q \tag{5-15}$$

$$\sum_{k\in V}\sum_{j\in C}x_{ijk}=1 \quad i\in V \tag{5-16}$$

$$\sum_{j\in C}x_{ijk}=y_{ki} \quad j\in V,\forall k\in V \tag{5-17}$$

$$\sum_{i\in C}x_{ijk}=y_{kj} \quad i\in V,\forall k\in V \tag{5-18}$$

$$\sum_{j\in C}x_{0jk}=1 \quad \forall k\in V \tag{5-19}$$

$$\sum_{i\in C}x_{ihk}-\sum_{j\in C}x_{hjk}=o \quad \forall h\in C,\forall k\in V \tag{5-20}$$

$$\sum_{i\in C}x_{i0k}=1 \quad \forall k\in V \tag{5-21}$$

$$a_i\le s_{ik}\le b_i \quad \forall i\in C,\forall k\in V \tag{5-22}$$

$$x_{ijk}=\begin{cases}1 & \text{车辆}k\text{从客户}i\text{行驶到客户}j\\ 0 & \text{车辆}k\text{不从客户}i\text{行驶到客户}j\end{cases} \tag{5-23}$$

$$y_{ki}=\begin{cases}1 & \text{表示}i\text{的任务由车辆}k\text{完成}\\ 0 & \text{客户}i\text{的任务不由车辆}k\text{完成}\end{cases} \tag{5-24}$$

其中，约束条件（5-15）表示车辆不超载；约束条件（5-16）表示每个客户被访问且被访问一次；约束条件（5-17）和（5-18）表示两个变量之间的约束关系；约束条件（5-19）、（5-20）、（5-21）保证了每辆车始于出发点，访问客户后，最后又回到出发点；约束条件（5-22）表达了时间窗限制。

5.3.3 模型效果检验与结论

某物流公司有 5 台配送车辆，它们的行驶速度均为 50km/h，车辆的最大载重量均为 1t，需要向 11 个客户送货。物流中心的坐标为（99.3km，39.7km），11 个客户的坐标、货物需求量及任务时间窗[允许最早开始时间，允许最迟开始时间]见表 5-1。

表 5-1 实例数据

客户编号	横坐标 x/ km	纵坐标 y/ km	货物需求量 q/t	卸货时间 T/ min	时间窗 [ET，LT]
1	39.2	70.4	0.102	50	[74，124]
2	22.4	45.6	0.113	30	[58，108]
3	97.1	79.3	0.095	10	[15，65]
4	6.0	28.7	0.131	50	[96，146]
5	62.3	79.6	0.029	25	[47，97]
6	42.8	88.5	0.306	10	[85，135]
7	86.7	3.4	0.532	46	[21，71]
8	87.4	61.2	0.617	48	[9，59]
9	88.8	91.3	0.232	30	[37，87]
10	14.6	13.2	0.459	45	[35，85]
11	46.7	77.6	0.121	23	[12，62]

采用改进的蚁群算法，蚂蚁数为8，最大循环次数NC为50，其他参数α=0.5，β=2，μ=0.7，ρ = 0.5，$\rho_{\min}$ = 0.1。

随机求解10次，得到最优调度路线（如图5.5），需要车辆3辆，每辆车的具体路线为：

第一辆车：0 →8 →3 →9 →5 →0；

第二辆车：0 →11 →1 →6 →2 →4 →0；

第三辆车：0 →7 → 10→0；

总里程：601.7km。

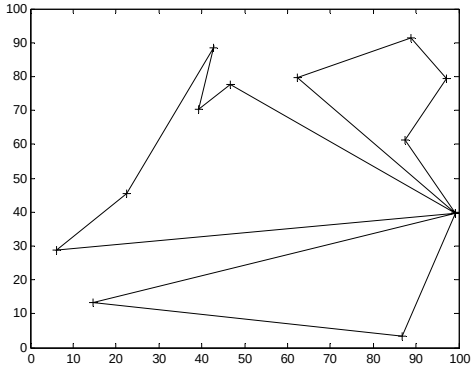


图5.5 最优调度路线

5.4 粒子群算法

粒子群算法（particle swarm optimization，PSO）最早是由 Russell Eberhart 和 James Kennedy 于 1995 年提出的，它的基本概念源于对人工生命（artificial life）和鸟群捕食行为的研究。研究者发现鸟群在飞行过程中经常会突然改变方向、散开、聚集，其行为不可预测，但其整体总保持一致性，个体与个体间也保持着最适宜的距离。通过对类似生物群体行为的研究，发现生物群体中存在着一种社会信息共享机制，它为群体的进化提

供了一种优势，这也是 PSO 算法形成的基础。PSO 算法的运行机理不是依靠个体的自然进化规律，而是对生物群体的社会行为进行模拟。在生物群体中存在着个体与个体、个体与群体之间的相互作用、相互影响的行为，这种行为体现的是一种存在于生物群体中的信息共享机制。PSO 算法就是对这种社会行为的模拟，即利用信息共享机制，使得个体之间可以相互借鉴经验，从而促进整个群体的发展。

PSO 同遗传算法类似，是一种基于群体的优化工具，系统初始化为一组随机解，通过某种方式迭代寻找最优解，是基于个体的协作与竞争来完成在复杂搜索空间中最优解的搜索，也是一种基于群智能方法的演化计算技术。但 PSO 并没有遗传算法用的交叉（crossover）、变异（mutation）等算子，编码方式也相对简单。因此 PSO 具有算法简单、容易实现，并且无许多参数需要调整的优点。

PSO 的提出至今不到 15 年，但它已经得到了广泛的关注。在基本 PSO 算法的基础上，已经出现了各种有意义的改进 PSO 算法，例如，自适应 PSO 算法、协同 PSO 算法、混合 PSO 算法，等等。PSO 算法非常适于求解连续函数的优化问题，主要应用于神经网络训练、多目标优化等应用领域；也有将 PSO 算法用于解决一些离散型优化问题的相关研究，例如用于求解 TSP 问题、任务分配问题等组合优化问题。

5.4.1 PSO 算法的基本原理

自然界中一些生物的行为特征呈现出群体特征，可以用简单的几条规则将这种群体行为（swarm behavior）在计算机中建立个体的运动模型，但这个群体的行为可能很复杂。例如，著名的 Boid 模型使用了下列三个规则作为简单的行为规则：

（1）向背离最近的同伴的方向运动；

（2）向目的运动；

（3）向群体的中心运动。

在这个群体中每个个体的运动都遵循这三条规则，通过 Boid 模型来模拟整个群体的运动。PSO 算法的基本概念也是如此，每个粒子的运动都可用几条规则来描述，因此 PSO 算法简单，容易实现。

设想这样一个场景：一群鸟在随机搜寻事物，在这个区域里只有一块食物，且所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢？最简单有效的方法就是搜寻目前离食物最近的鸟的周围区域。

PSO 算法就从这种生物种群行为特性中得到启发并用于求解优化问题。在 PSO 中，每个优化问题的潜在解都可以想象成 d 维搜索空间上的一个点，我们称为“粒子”（particle）。粒子在搜索空间中以一定的速度飞行，这个速度根据它本身的飞行经验和同伴的飞行经验来动态调整。所有的粒子都有一个被目标函数决定的适应值（Fitness Value），并且知道自己到目前为止发现的最好位置（personal best，记为 pbest，也称为个体极值）和当前的位置。这个可以看作是粒子自己的飞行经验。除此之外，每个粒子还知道到目前为止整个群体中所有粒子所发现的最好位置（global best，记为 gbest，也称为全局极值），实际上，gbest 即是在 pbest 中的最好值，这个可以看作是粒子的同伴的经验。每个粒子使用下列信息来改变自己的当前位置：

（1）当前位置；
（2）当前速度；
（3）当前位置与自己最好位置之间的距离；
（4）当前位置与群体最好位置之间的距离。

优化搜索正是在由这样一群随机初始化形成的粒子而组成的一个种群中，以迭代的方式进行的。PSO 算法中每个粒子即解空间中的一个解。每个粒子都通过上述两个极值不断更新自己，从而产生新一代群体。实际操作中通过由优化问题所决定的适应值来评价粒子的“好坏”程度。显然，每个粒子的行为就是追随着当前的最优粒子在解空间中进行搜索。

以数学形式来描述 PSO 算法则是：假设粒子的群体规模为 N，则第 i（$i=1, 2, \cdots, N$）个粒子的位置可表示为 X_i^k，它所经历过的“最好”位置记为 pbest_i,它的速度用 V_i^k 表示，而在这个种群中至少有一个粒子是最好的，其编号记为 g。则粒子 i 将根据下面的公式来更新自己的速度和位置：

$$V_i^{k+1} = V_i^k + c_1 * \text{rand}_1(...) * \left(\text{pbest}_i - X_i^k\right) + c_2 * \text{rand}_2(...) * \left(\text{gbest} - X_i^k\right) \tag{5-25a}$$

$$X_i^{k+1} = X_i^k + V_i^k \tag{5-25b}$$

其中，k 为迭代次数；c_1, c_2 为正常数，称为学习因子，一般取值为 2；$\text{rand}_1(\cdots)$和 $\text{rand}_2(\cdots)$是均匀分布于[0， 1]上的随机数。为了控制 V_i^k 和 X_i^k 的值在合理的区域内，需要指定 $V_{\max}$ 和 $X_{\max}$ 来限制。

公式（5-25a）主要通过三部分来计算粒子 i 的新速度：一是粒子 i 前一时刻的速度，说明了粒子目前的状态；二是粒子 i 当前位置与自己最好位置之间的距离，表示粒子本身的思考；三是粒子 i 当前位置与群体最好位置之间的距离，表示群体的影响。三个部分共同决定了粒子的空间搜索能力 粒子 i 通过公式(5-25b)计算新位置的坐标 通过式(5-25a)和式（5-25b）粒子决定下一步的运动位置。

在图 5.6 中，以二维空间为例描述了粒子根据公式（5-25a）和（5-25b）的移动原理。

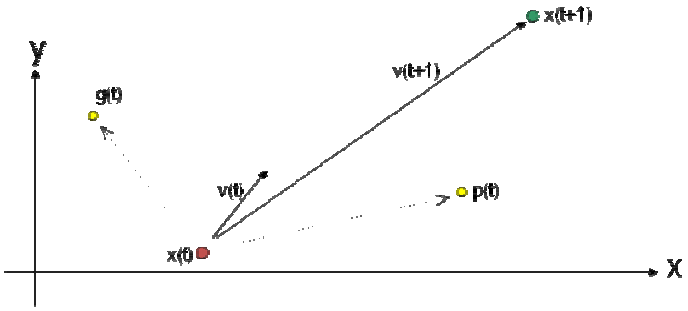


图 5.6　粒子移动原理的示意图

从社会学的角度看，公式（5-25a）的第一部分称为记忆项，表示上次速度大小和方向的影响；第二部分成为自身认知项，是从当前点指向此粒子自身最好点的一个矢量，表示粒子的动作来源于自己经验的部分；第三部分成为群体认知项，是一个从当前点指

向群体最好点的一个矢量，反映了粒子间的协同合作和知识共享。第一部分起到了平衡全局和局部搜索的能力；第二部分使粒子有了足够强的全局搜索能力，避免局部极小；第三部分体现了粒子间的信息共享。在这三部分的共同作用下粒子才能有效地到达最好位置。粒子通过自己的经验和同伴中最好的经验来决定下一步的运动，这与人类的决策极其相似，人们通常也是通过综合自身已有的信息和从外界的信息来作出决策的。

粒子群优化算法的框架如图 5.7 所示。

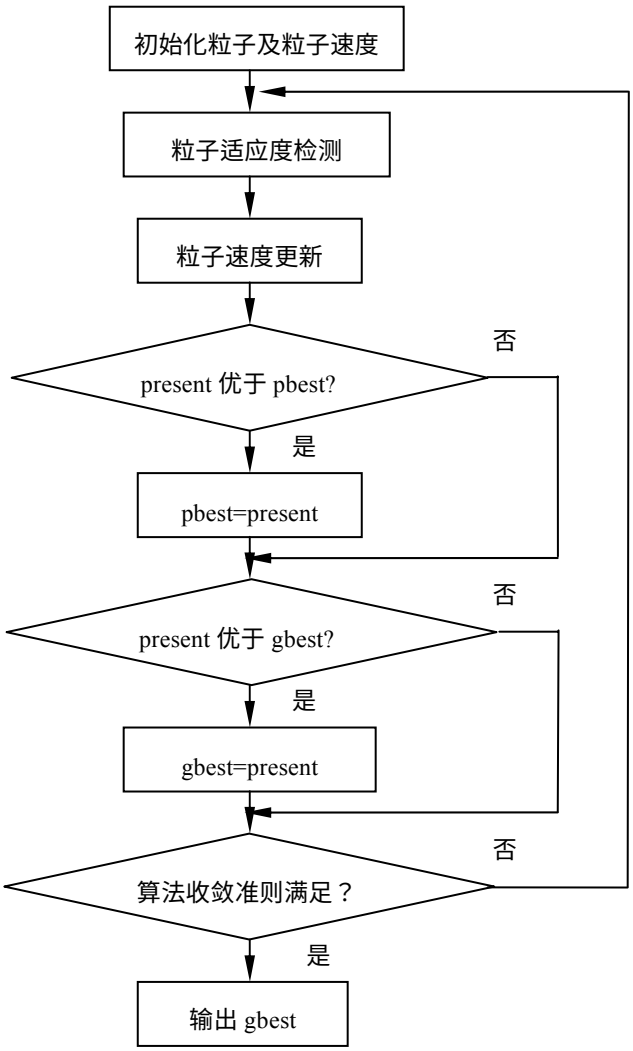


图 5.7　粒子群优化算法框架图

5.4.2 加入惯性权重因子的 PSO 算法

公式（5-25a）由三部分组成,第一部分是粒子先前的速度项，第二和第三部分是导致粒子速度变化的两项。

如果没有后两项，粒子将保持在相同的方向上以当前的速度“ 飞翔 ”至下一个点，

直至达到边界值。这样的 PSO 将不能找到一个可接受的解，除非这样的飞行轨迹是一种合理的方案，而这在现实中是非常少见的。

而如果没有第一项，那么“飞翔”粒子的速度仅仅取决于粒子的当前位置和它们最好的历史位置。速度自身没有记忆性。假设在开始的时候，粒子 i 正好位于整体最好的所在点，那么粒子 i 将以速度 0“飞翔”，也就是保持粒子此次的位置不变，直到出现新的一个最好点替代粒子 i。同时，每一个粒子将向自身最好点和整体最好点的质心方向“飞翔”。在这种状况下，各个粒子逐渐地向当前最好的位置处收缩，直到出现新的最好值。因此可以想象，在没有第一项的 PSO 搜索过程中，其搜索空间将在迭代中逐渐衰退，这类似于局部搜索算法。只有当全局最好点包含在出事搜索空间内的时候，才有可能找到满意解，这样，最终解在很大程度上要依赖于初始化种群。也就是说，没有第一项的 PSO 算法更多地表现的是局部搜索能力。因此也可以说，第一项的内容使得粒子有一种扩展搜索空间的能力，即开拓能力，是一种全局搜索能力的表现。

对于实际中的优化问题的解决，局部搜索和全局搜索都起着重要的作用。为了权衡优化问题中局部搜索与全局搜索能力的贡献，可以考虑在公式（5-25a）中引入一个惯性权重因子 w（inertia weight）。w 可以是个正整数常数，也可以是以时间为变量的线性或非线性正整数。这样公式（5-25a）和公式（5-25b）就变为：

$$V_i^{k+1} = w * V_i^k + c_1 * \text{rand}_1(...) * \left(\text{pbest}_i - X_i^k\right) + c_2 * \text{rand}_2(...) * \left(\text{gbest} - X_i^k\right) \quad (5\text{-}26a)$$

$$X_i^{k+1} = X_i^k + V_i^k \quad (5\text{-}26b)$$

加入惯性权重因子 w 的 PSO 算法为：

（1）初始化粒子群，包括群体规模，每个粒子的位置和速度；

（2）计算每个粒子的适应值；

（3）对每个粒子，用它的适应值和个体极值 pbest 比较，如果较好，则替换 pbest；

（4）对每个粒子，用它的适应值和全局极值 gbest 比较，如果较好，则替换 gbest；

（5）根据公式（5-26a）和公式（5-26b）更新粒子的速度和位置；

（6）如果满足结束条件（误差足够好或者到达最大循环次数）退出，否则回到（2）。

程序的伪代码如下：

```
For each particle
    Initialize particle
End

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pbest) in history
        set current value as the new pbest
End

    Choose the particle with the best fitness value of all the particles as the gbest
    For each particle
        Calculate particle velocity according equation (5-26a)
        Update particle position according equation (5-26b)
```

End
While maximum iterations or minimum error criteria is not attained

5.4.3 PSO 算法关键参数控制

PSO 算法需要调节的主要参数有最大速度 V_{max}、惯性权重 w，学习因子、群体规模和粒子的维度等。

1）最大速度 V_{max} 与惯性权重 w

惯性权重 w 是用来控制粒子以前速度对当前速度的影响，它将影响粒子的全局和局部搜索能力。较大的 w 值有利于全局搜索，较小的 w 值有利于局部搜索。选择一个合适的 w 可以平衡全局和局部搜索的能力，这样可以通过最少的迭代次数来找到最优解。对 PSO 全局和局部搜索能力的平衡主要被惯性权重控制，研究发现，当 V_{max}=2 时，惯性权重一般在[0.9, 1.2]之间的 PSO 算法较好，也就是说，既有很大可能搜索到全局最优而且迭代的次数也较合理。进一步的研究还发现使用每次惯性权重从 1.4 递减到 0 要比用固定的惯性权重更好，因为初始较大的惯性权重有助于找到比较好的范围，而后期较小的惯性权重则保证具有较好的局部搜索能力。

更深入地分析公式（5-26a），能够看出最大速度 V_{max} 对于 PSO 的全局搜索能力也有限制和影响。如果最大速度 V_{max} 设定得很小，全局搜索能力就受到限制，此时无论惯性权重是多少，PSO 算法都主要呈现为局部搜索。设定较大的 V_{max} 则意味着 PSO 可以通过选择惯性权重的大小来取得较大的搜索范围。由于 V_{max} 对于全局搜索能力的影响是间接的而惯性权重的影响是直接的，因此较好的做法是去掉 V_{max} 的约束而仅以惯性权重来控制搜索能力。但是这样做需要非常谨慎，因为总是进行全局搜索有可能使得 PSO 不断开拓新的搜索范围，结果却因为局部搜索能力的不足而使得搜索失败。实验分析得到的结论是，当 V_{max} 较小（2）时，w 近似为 1 是比较好的选择，当 V_{max} 较大（≥3）时，取 0.8 较合适。事实上在 PSO 算法的改进研究中，对惯性权重因子的研究是一个主要热点，认为 w 取值依据迭代过程变化有利于算法的改进，而且也有学者提出了模糊规则动态地修改 w 的取值，来优化真正的实际问题。

2）学习因子

c_1, c_2 用来控制粒子自身的记忆和同伴的记忆之间的相对影响。合适的选择可以提高算法速度、避免局部极小。实验研究[29]认为 $c_1=c_2=2$ 是较好的选择，或者 $c_1=c_2=0.5$ 也能得到较好的结果。也有的研究认为 c_1, c_2 可以选择的大一些，但是最好 $c_1+c_2\leq 4$。

3）群体规模和粒子的维度

群体规模一般来说不用取得太大，几十个粒子一般就足够用了。对于多目标优化等比较困难或某些特定的问题，粒子数可以取到 100～200。粒子的维度则由优化问题所决定，也就是解空间的维度。

4）中止条件

中止条件就是到达最大循环数或者满足最小误差要求，一般由具体的优化问题来决定。

5.4.4 基于 Matlab 的 PSO 程序设计

1）参数的编码

在 MATLAB 环境中，种群中粒子及其速度均采用实数编码[30]，dimsize 表示参数维度，粒子参数编码格式如下：

粒子位置各维的表示：$X_1, X_2, X_3, ..., X_{\text{dimsize}}$

粒子速度各维的表示：$V_1, V_2, V_3, ..., V_{\text{dimsize}}$

适应度：$F(X)$

popsize 表示种群大小，粒子群编码格式如下：

$$\text{POP}=\begin{bmatrix} X_{11} & X_{12} & X_{13} & \cdots & X_{1\text{dimsize}} & V_{11} & V_{12} & V_{13} & V_{1\text{dimsize}} & F(X_1) \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2\text{dimsize}} & V_{21} & V_{22} & V_{23} & V_{2\text{dimsize}} & F(X_2) \\ \cdots & & & & & \cdots & & & & \cdots \\ X_{\text{popsize1}} & & \cdots & & X_{\text{popsizedimsize}} & V_{\text{popsize1}} & & \cdots & V_{\text{popsizedimsize}} & F(X_{\text{popsize}}) \end{bmatrix}$$

2）粒子群的初始化

粒子群的初始化就是要产生一个随机矩阵，其中粒子的编码满足上述要求。用 x 表示粒子，objectfun 表示适应度函数。粒子群初始化伪码为：

```
for dimindex=1:dimsize
  x(dimindex)=粒子取值区间内的随机值；
  x(dimsize+dimindex)=速度取值区间内的随机值；
  x(2*dimsize+1)=objectfun(x(1:dimsize));
end
```

3） 粒子速度的更新

粒子速度更新的算法是基于式（5-26a），但由于粒子群算法在解空间内搜索时，有时会出现粒子在全局最优解附近“ 振荡 ”的现象，为了避免这个问题，可以将惯性权重 w 随着迭代的进行，按下式计算：

$$w = w_{\max} - iter \times \frac{w_{\max} - w_{\min}}{iter_{\max}} \tag{5-27}$$

其中，iter 为当前迭代数，而 $\text{iter}_{\max}$ 为总迭代数。

粒子速度更新伪码如下：

```
for dimindex=1:dimsize
  w=wmax -（wmax-wmin）*iter/itermax；
  subtract1=pbest-x(1:dimsize);
  subtract2=gbest-x(1:dimsize);
  tempV=w*x(dimsize+dimindex)+2*subtract1+2*subtract2;
  if tempV>Vmax
    x(dimsize+dimindex)=Vmax;
```

```
  elseif tempV<-Vmax
    x(dimsize+dimindex)=-Vmax;
  else
    x(dimsize+dimindex)=tempV;
  end
 end
```

4） 粒子位置的更新

粒子群中，在算法运行的每一代，粒子都根据式（5-26b）更新自己的位置，粒子位置更新的程序伪码如下：

```
for dimindex=1:dimsize
  pos=x(dimindex)+ x(dimsize+dimindex);
  if pos>粒子该维最大取值
     pos=粒子该维最大取值;
  elseif pos<粒子该维最小取值
     pos=粒子该维最小取值;
  end
  x(dimindex)=pos
end
```

5） 算法性能的评价

通常使用 DeJong 提出的在线性能（online performance）和离线性能（offline performance）来评价算法性能的优劣，其中在线性能测试动态特性，而离线性能则测试收敛特性。

在环境 e 下策略 s 的在线性能 $X_e(s)$ 定义为：

$$X_e(s)=\frac{1}{T}\sum_{i=1}^{r} f_e(t)$$

其中，$f_e(t)$ 为各代平均适应度。上式表明，如果在线性能用平均适应度表示，则通过简单计算从第一代到当前代的各代平均适应度值对世代数的平均值即可获得在线性能。

在环境 e 下策略 s 的离线性能 $X_e^*(s)$ 定义为：

$$X_e^*(s)=\frac{1}{T}\sum_{i=1}^{r} f_e^*(t)$$

其中，$f_e^*(t)$ 是在环境 e 下，在[0,1]时段内最好的目标函数值或最大适应度。上式表明，离线性能是特定时刻最佳性能的累积平均。

在 MATLAB 环境中，粒子群优化算法主程序运行后，将返回最优解，以及最优解对应的适应度。另外还应该返回各代跟踪信息。跟踪信息采用如下编码结构：

$$\text{traceInfo}=\begin{bmatrix} 1 & X_e(s)_{T=1} & X_e^*(s)_{T=1} \\ 2 & X_e(s)_{T=2} & X_e^*(s)_{T=2} \\ \cdots & & \\ \text{iterMax} & X_e(s)_{T=\text{iterMax}} & X_e^*(s)_{T=\text{iterMax}} \end{bmatrix}$$

PSO 算法的主程序如下：

```
for iter=1:iterMax
   for popindex=1:popsize
      评价各粒子的适应度；
   if 粒子适应度>objectfun(pbest)
      pbest=粒子当前位置；
   end
   if 粒子适应度>objectfun(gbest)
      gbest=粒子当前位置；
   end
   粒子速度更新；粒子位置更新；并计算粒子适应值；
end
   计算 traceInfo(iter);
end
```

5.5 实例分析：POS 算法在函数优化中的应用

PSO 的优势在于算法的简洁性，易于实现，没有很多参数需要调整，且不需要梯度信息。因而 PSO 成为非线性连续优化问题、组合优化问题和混合整数非线性优化问题的有效工具，并广泛应用于函数优化、神经网络训练、模糊系统控制以及其他遗传算法的应用领域，比较有潜力的应用则包括系统设计、多目标优化、分类、模式识别、调度、信号处理、决策、机器人应用等。

MATLAB 具有强大的矩阵运算功能，用其编写离子群算法程序具有很好的优势。本节将以 J. D. Schaffer 提出的函数

$$\begin{cases} \min\ f(x_1,x_2)=0.5+\dfrac{\sin^2\sqrt{X_1^2+X_2^2}-0.5}{\left[1.0+0.001\left(X_1^2+X_2^2\right)\right]^2} \\ s.t.-100\le x_i\le 100, i=1,2 \end{cases}$$

作为一个示例仿真，说明利用 MATLAB 实现粒子群算法在函数优化中的应用。

1） 粒子群的初始化

initpop 函数的功能是实现群体的初始化，xmax 表示变量的上限，xmin 表示变量的下限，其余符号同上节。

```
function pop=initpop(popsize,dimsize)
pop=unifmd(xmin,xmax,popsize,2*dimsize);
```

2）计算粒子的适应度和确定群体的 pbest 和 gbest

calobjvalue 函数的功能是计算目标函数的适应度，其公式是采用本节中的示例仿真，用户可以根据不同的优化问题予以修改。

```
function objvalue=calobjvalue(pop)
   for i=1:popsize
```

```
        obfuct1=sin(sqrt(pop(i,1)^2+pop(i,2)^2))^2-0.5;
        obfuct2=(1.0+0.001*(sqrt(pop(i,1)^2+pop(i,2)^2))^2;
        objvalue(i,1)=0.5+obfuct1/obfuct2;
    end
    pbest=pop(xindex,1:dimsize);
    [gbest,xindex]=max(objvalue);
    xtemp=pop(xindex,1:dimsize);
    gbest=xtemp;
```

3）粒子速度和位置的更新

基于公式（5-26a）和公式（5-26b）进行粒子速度和位置的更新，并生成新的粒子群体。

```
function pop2=renew(pop)
  for i=1:popsize
    for dimindex=1:dimsize
      w=wmax -（wmax-wmin）*iter/itermax；
      sub1=pbest(i,dimindex)-pop(i,dimindex);
      sub2=gbest(i,dimindex)-pop(i,dimindex);
      tempV=w*pop(i,dimsize+dimindex)+c1*unifmd(0,1)*sub1+c2*unifmd(0,1)*sub2;
      if tempV>Vmax
        pop(i, dimsize+dimindex)=Vmax;
      elseif tempV<(-Vmax)
        pop(i, dimsize+dimindex)=-Vmax;
      else
        pop(i,dimsize+dimindex)=tempV;
      end
      if temppos>xmax
        pop(i, dimindex)=xmax;
      elseif tempos<xmin
        pop(i, dimindex)=xmin;
      else
        pop(i, dimindex)-temppos;
      end
    end
  end
```

4）粒子 pbest 和 gbest 的更新

粒子在进化过程中依据其适应度，调节个体的最好位置 pbest 和群体的最好位置 gbest。

```
function[pbest, gbest]=regulate(pop)
  for i=1:popsize
    obfuct1=sin(sqrt(pop(i,1)^2+pop(i,2)^2))^2-0.5;
    obfuct2=(1.0+0.001*(sqrt(pop(i,1)^2+pop(i,2)^2))^2;
    objvalue(i,1)=0.5+obfuct1/obfuct2;
```

```
    obfuct3=sin(sqrt(pbest(i,1)^2+pbest(i,2)^2))^2-0.5;
    obfuct4=(1.0+0.001*(sqrt(pbest(i,1)^2+pbest(i,2)^2))^2;
    pvalue(i,1)=0.5+obfuct3/obfuct4;
  end
    obfuct1=sin(sqrt(gbest(i,1)^2+gbest(i,2)^2))^2-0.5;
    obfuct2=(1.0+0.001*(sqrt(gbest(i,1)^2+gbest(i,2)^2))^2;
    objvaluetemp=0.5+obfuct1/obfuct2;
  for i=1:popsize
    if objvalue(i,1)<pvalue(i,1)
      pbest(i,1:dimsize)=pop(i,1:dimsize)
    end
    if objvalue(i,1)<objvaluetemp
      gbest=pop(i, 1:dimsize);
      xtemp=pop(i,1:dimsize);
    end
  end
```

基于上述思路，取种群个数为 100，进化代数为 100，学习因子为 2，惯性权重的上限为 0.9、下限为 0.1，位置的上限为 100、下限为-100，速度设置与位置相同，求解 J. D. Schaffer 函数。该函数在距全局最优解大约 3.14 范围内存在无穷多个局部极小解将其包围，并且该函数强烈振荡，一般的算法难以得到最优解。而利用 MATLAB 开发 PSO 工具箱，则算法运行 10 次，其在线值和离线值全部收敛于最优解 $\min\left(f\left(X^*\right)\right)=f(0,0)=0$。

事实上，本节所给出的 J. D. Schaffer 函数用遗传算法来求解是比较困难的，一般来说需要设计一些特殊的操作或编码方式，但使用 PSO 算法就显然要简单得多，结果也比较好。而且，基于 MATLAB 的粒子群优化算法工具箱，具有很强的通用性，使用者只要给出新的适应度函数，未知参数集以及参数维度，就可以直接利用该工具箱进行函数优化问题的求解了。

思考题

5.1 群智能算法中包含哪些具体算法？它们通常应用于哪些领域？
5.2 群智能算法与进化计算的异同点有哪些？
5.3 基本蚁群算法根据信息素的更新方式分为哪几类？
5.4 基本蚁群算法的优缺点有哪些？
5.5 试论述蚁群算法求解 TSP 问题的基本流程。
5.6 试论述蚁群算法中各个算子在算法优化过程中的影响。
5.7 蚁群算法的改进措施有哪些？并举例说明。
5.8 蚁群算法主要用来解决哪几类问题？
5.9 粒子群算法的基本原理和特点是什么？
5.10 粒子群算法中加入惯性权重因子的目的是什么？
5.11 粒子群算法主要用于解决哪些类型的问题？

参考文献

[1] 肖人彬,陶振武.群集智能研究进展[J].管理科学学报,2007,10(6):80-96
[2] 冯静,舒宁.群智能理论及应用研究[J].计算机工程及应用,2006 (17):31-34
[3] 彭喜元,彭宇,戴毓丰.群智能理论及应用[J].电子学报,2003,31(12A):1982-1988
[4] 吴启迪等.智能蚁群算法及应用.上海:上海科技教育出版社,2004
[5] 李士勇等.蚁群算法及其应用.哈尔滨:哈尔滨工业大学出版社,2004
[6] 王剑,李平,杨春节.蚁群算法的理论与应用[J].机电工程,2003,20(5):126-130
[7] 金飞虎,洪炳熔,高庆吉.基于蚁群算法的自由飞行空间机器人路径规划[J].机器人，2002, 24(6): 526-529
[8] 侯志荣,吕振肃.基于 MATLAB 的粒子群优化算法及其应用[J].计算机仿真,2003,20（10）: 68-70
[9] 丁滢颍,何衍,蒋静坪.基于蚁群算法的多机器人协作策略[J].机器人,2003,25(5): 414-418
[10] 张志霞,邵必林.基于改进蚁群算法的运输调度规划.公路交通科技,2008,25(4): 137-140
[11] 李宁,邹彤,孙德宝.带时间窗车辆路径问题的粒子群算法[J].系统工程理论与实践, 2004, 24(4)
[12] Dorigo Metal.Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, 1996, 26(1):29-41
[13] Dorigo Metal.Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1997,1(1):53-66
[14] Thomas Stutzle, Holger H Hoos. Max-Min Ant System[J]. Future Generation Computer System,2000,16:889-914
[15] Gambardella, Luca Maria, Dorigo M. Solving symmetric and asymmetric TSPs by and colonies[R]. Proceedings of the IEEE Conference on Evolutionary Computation, 1996,137-142
[16] Kennedy J，Eberhart R C. Swarm Intelligence[M].San Francisco：Morgan Kaufmann division of Academic Press, 2001
[17] Ramos V, Almeida F. Artificial Ant colonies in digital image habitats-A mass behavior effect study on pattern recognition[A]. In Proceedings of the 2nd International Workshop on Ant Algorithms[C],2000:113-116
[18] Holland O,Melhuish C.Stigmergy,self-organization and sorting in collective robotics[J]. Artificial Life，1999,5:173-202
[29] Yoshida E，Murata S，Tomita K et al. Distributed formation control for a modular mechanical system[A]. In ： Proceedings IEEE/RSJ International Conference on Intelligent Robot and Systems[C],1997

[20] Kuntz P，Snyers D. New results on an ant2based heuristic for highlighting the organization of large graphs[A]. In ： Proceedings ofthe 1999 Congress on Evolutionary Computation[C].I Piscataway：IEEE Press，1999:1451-1458

[21] Gao Q L，Luo X，Yang S Z. Stigmergic cooperation mechanism for shop floor control system[J].International Journal of AdvancedManufacturing Technology，2004，25：743-753

[22] Cicirello V A，Smith S F. Ant colony control for autonomous decentralized shop floor routing[A]. In：Proceedings of the 5th International Symposium on Autonomous Decentralized Systems[C],2001

[23] Bonabeau E，Meyer C. Swarm intelligence：A whole new way to think about business[J].Harvard Business Review，May，2001

[24] Kennedy J, Eberhart R C. Particle swarm optimization. Proc. IEEE int'l conf. on neural networks (Perth, Australia), IEEE service center, Piscataway, NJ, 1995, IV:1942-1948

[25] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory. Proc.6th Int'l Symposium on Micro Machine and Human Science (Nagoya, Japan).IEEE service center, Piscataway, NJ, 1995:39-43

[26] Shi Y, Eberhart R. A modified particle swarm optimizer, IEEE int'l conf. on evolutionary computation, Anchorage, Alaska, 1998: 4-9

[27] Shi Y, Eberhart R. A modified particle swarm optimizer, Proc. of IEEE int'l conf. on computational intelligence, 1998: 69-73

[28] Kennedy, J. The behavior of particles. Evolutionary Programming VII, Springer, 1998: 581-590

附录：求解旅行商问题的简单蚁群算法 MATLAB 程序

程序中主要符号说明：C 为 n 个城市的坐标，n×2 的矩阵；NC_max 为最大迭代次数；M 为蚂蚁个数；Alpha 为表征信息素重要程度的参数；Beta 为表征启发式因子重要程度的参数；Rho 信息素挥发系数；Q 为信息素增加强度系数；R_best 为各代最佳路线；L_best 为各代最佳路线的长度。

```
function[R_best，L_best，L_ave，Shortest_Route，Shortest_Length]=ACATSP（C，NC_max，m，Alpha，Beta，Rho，Q）
%%第一步：变量初始化
n=size（C，1）；
D=zeros（n，n）；
for i=1：n
  for j=1：n
    if i~=j
      D（i，j）=（（C（i，1）-C（j，1））^2+（C（i，2）-C（j，2））^2）^0.5;
    else
      D（i，j）=eps;
    end
    D（j，i）=D（i，j）；
  end
end
Eta=1./D;
Tau=ones（n，n）；
Tabu=zeros（m，n）；
NC=1;
R_best=zeros（NC_max，n）；
L_best=inf.*ones（NC_max，1）；
L_ave=zeros（NC_max，1）；
while NC<=NC_max
%%第二步：将 m 只蚂蚁放到 n 个城市上
Randpos=[];
for i=1：(ceil（m/n))
  Randpos=[Randpos，randperm（n）];
end
Tabu（：，1）=（Randpos（1，1：m））';
%%第三步：m 只蚂蚁按概率函数选择下一座城市，完成各自的周游
for j=2：n
  for i=1：m
    visited=Tabu（i，1：(j-1)）；
    J=zeros（1，(n-j+1)）；
    P=J;
    Jc=1;
    for k=1：n
      if length（find（visited==k））==0
        J（Jc）=k;
        Jc=Jc+1;
      end
    end
    %下面计算待选城市的概率分布
    for k=1：length（J）
```

```
        P（k）=（Tau（visited（end），J（k））^Alpha）*（Eta（visited（end），J（k））^Beta）；
    end
    P=P/（sum（P））；
    %按概率原则选取下一个城市
    Pcum=cumsum（P）；
    Select=find（Pcum>=rand）；
    to_visit=J（Select（1））；
    Tabu（i，j）=to_visit;
  end
end
if NC>=2
  Tabu（1，：）=R_best（NC-1，：）；
end
%%第四步：记录本次迭代最佳路线
L=zeros（m，1）；
for i=1：m
  R=Tabu（i，：）；
  for j=1：（n-1）
    L（i）=L（i）+D（R（j），R（j+1））；
  end
  L（i）=L（i）+D（R（1），R（n））；
end
L_best（NC）=min（L）；
pos=find（L==L_best（NC））；
R_best（NC，：）=Tabu（pos（1），：）；
L_ave（NC）=mean（L）；
NC=NC+1
%%第五步：更新信息素
Delta_Tau=zeros（n，n）；
for i=1：m
  for j=1：（n-1）
    Delta_Tau（Tabu（i，j），Tabu（i，j+1））=Delta_Tau（Tabu（i，j），Tabu（i，j+1））+Q/L（i）；
  end
  Delta_Tau（Tabu（i，n），Tabu（i，1））=Delta_Tau（Tabu（i，n），Tabu（i，1））+Q/L（i）；
end
Tau=（1-Rho）.*Tau+Delta_Tau;
%%第六步：禁忌表清零
Tabu=zeros（m，n）；
end
%%第七步：输出结果
Pos=find（L_best==min（L_best））；
Shortest_Route=R_best（Pos（1），：）
Shortest_Length=L_best（Pos（1））
subplot（1，2，1）
DrawRoute（C，Shortest_Route）
subplot（1，2，2）
plot（L_best）
hold on
plot（L_ave，'r'）
title（'平均距离和最短距离'）
```

```
function DrawRoute（C，R）
N=length（R）;
scatter（C（:，1），C（:，2））;
hold on
plot（[C（R（1），1），C（R（N），1）]，[C（R（1），2），C（R（N），2）]，'g'）
hold on
for ii=2：N
   plot（[C（R（ii-1），1），C（R（ii），1）]，[C（R（ii-1），2），C（R（ii），2）]，'g'）
   hold on
end
title（'旅行商问题优化结果'）
```

第 6 章 人工免疫计算

6.1 概述

6.1.1 什么是人工免疫系统

人工免疫系统（artificial immune system，AIS）是模仿自然免疫系统功能的一种智能方法，它实现一种受生物免疫系统启发，通过学习外界物质自然防御机理的学习技术，提供噪声忍耐、无教师学习、自组织、记忆等进化学习机理，结合了分类器、神经网络和机器推理等系统的一些优点，因此具有提供新颖的解决问题方法的潜力。其研究成果涉及控制、数据处理、优化学习和故障诊断等许多领域，已经成为继神经网络、模糊逻辑和进化计算后人工智能的又一研究热点。

6.1.2 人工免疫系统的发展

在生物学领域中，免疫学是一门相对年轻的学科，然而，人类对自然免疫的认识可以追述到 300 年以前。早在 17 世纪，我国医学家就创造性地发明了人痘以预防天花。1796 年英国医生 Edward Jenner 的“牛痘”的发现，取代了人痘疫苗，是公认的现代免疫学开端。法国免疫学家 Pasteur 发明了减毒细菌疫苗，奠定了经典免疫疫苗的基础。经过 300 多年的发展，免疫学已经从微生物学的一章发展成了一门独立的学科，并派生出若干分支。例如，细胞免疫学、分子免疫学、神经与内分泌免疫学、生殖免疫学和行为免疫学等。

表 6-1 总结了免疫学历史上（16 世纪至 20 世纪 90 年代初）比较重要的思想、理论和研究成果。

事实上，人们对自然免疫系统的认识还不是十分充分。就现有的免疫系统理论而言，为学术界所接受，并为工程应用尤其是人工智能领域所借鉴的主要是 Burnet 的克隆选择学说和 Jenner 的免疫网络学说。

Farmer 等人（1986）率先基于免疫网络学说给出了免疫系统的动态模型，并探讨了免疫系统与其他人工智能方法的联系，开始了人工免疫系统的研究。在我国，靳蕃等人在 1990 年前后就已经指出“免疫系统所具有的信息处理与肌体防卫功能，从工程角度来看，具有非常深远的意义”。但是，这以后的研究成果就比较少见了。

直到 1996 年 12 月，在日本首次举行了基于免疫性系统的国际专题讨论会，首次提出了“人工免疫系统”的概念。随后，人工免疫系统进入了兴盛发展期，Dasgupta（1997）和丁永生、任立红（2000）认为人工免疫系统已经成为人工智能领域的理论和应用研究热点，相关论文和研究成果逐年增加。1997 年和 1998 年 IEEE Systems，Man and Cybernetics 国际会议还组织了相关专题讨论，并成立了“人工免疫系统及应用分会”。

虽然人工免疫系统已经被广大研究者逐渐重视，然而无论是免疫机理的认识、免疫算法的构造还是工程应用都还处在一个比较低的水平。

表 6-1　自然免疫学的主要思想、理论及研究成果

主要阶段	时间	代表人物	思想、理论及研究成果
经验免疫时期	16 世纪起	中国民间	“人痘”的发明和应用
	1796 年－1870 年	Jenner	“牛痘”的发明和应用
		Koch	病理学
	1870 年－1890 年	Pasteur	疫苗接种
		Besedovskv	神经内分泌—免疫网络学说
		Metchinikoff	噬菌作用
科学免疫时期	1890 年－1910 年	Von Behring、Kitasato	发现抗体
		Ehrlich	发现细胞受体
	1910 年－1930 年	Bordet	免疫特性
		Landstelner	半抗原
现代免疫学时期	1930 年－1950 年	Breinl、Haurowitz	抗体合成
		Linus Pauling	抗原模型
	1950 年－1980 年	Burllet	克隆选择
		Niels Jerne	免疫网络与协作理论
分子水平研究	1980 年－1990 年	Susumu Tonegawa	受体的结构和多样性

6.1.3 人工免疫系统的研究内容和范围

目前，人工免疫系统的具体研究内容和范围主要包括以下三个方面：

第一方面：免疫计算智能系统。从计算的观点看，自然免疫系统是一种并行和分布式的自适应系统，用学习、记忆和联想恢复完成识别和分类任务，特别是能学习识别相关模式，记住以前见到的模式，用组合学高效地建立模式检测器。自然免疫系统是发展智能技术的启发源泉，研究人员已经开发了许多基于免疫系统的计算技术，包括各种基于免疫原理的免疫算法、人工免疫网络和免疫计算系统等。

（1）根据生物免疫系统原理发展新的算法，主要有阴性选择算法、克隆选择算法、免疫遗传算法、免疫优化算法，以及为完成各种特定任务而设计的基于免疫原理的算法等，可统称为免疫算法。目前这方面的研究和发展较活跃，也较为迅速。

（2）根据生物免疫系统原理建立人工模型，包括人工免疫网络模型和人工免疫系统模型两种形式。各种免疫网络学说，如独特型网络、互联耦合免疫网络、免疫反应网络和对称网络等，可借鉴用于建立人工免疫网络（artificial immune network，AIN）认知模型（比如机器人系统），目前应用最广的是独特型网络。

（3）建立混合智能系统。一方面利用免疫、神经网络、模糊技术建立智能计算系统，利用免疫系统抗体多样性的遗传机制改进遗传算法的搜索优化。这方面整体发展比较缓慢，目前研究主要集中在与神经网络的混合应用。

免疫计算智能系统的另一方面是利用免疫系统自身特性建立新型智能系统。免疫系统具有动态保持自组织记忆能力，并允许信息遗忘和记忆内容可访的特点。这些进化学习机制和学习外界物质的自然防御机制可用于建立解决机器学习等问题的新型机器学习

系统，还可发展用于解决数据分析等人工免疫系统。这方面发展得较为普遍。

第二方面：免疫工程应用研究，包括各种免疫计算智能技术在工程中的应用研究，建立利用生物免疫系统某一特性或某些特性解决特定工程问题的人工智能系统。还可以基于免疫学原理发展各种保安系统、疾病诊断系统、各种计算机安全和网络入侵诊断和检测系统、各种工业生产中的故障诊断、异常检测系统等。

第三方面：人工免疫系统理论研究，比如借助数学模型、非线性、复杂、混沌、计算智能等理论深入研究人工免疫系统的机制，但这方面发展相对滞后。

以上三方面是互相渗透的。就目前研究和应用情况来看，又可以大致归为两种情况：一种是解决特殊问题的计算方法，比如解决谱分析问题的免疫算法，只能用于解决谱分析，没有普遍的应用意义；另一种则是具有普遍意义的方法和系统，比如二进制免疫系统模型，就已经得到了广泛应用。计算机免疫系统和计算机与网络入侵检测已经作为一种有效的计算机安全系统得到了多个研究组织的研究和应用。这两种情况是由于处理问题的角度不同而出现的。值得指出的是，无论哪种情况所依据的系统都是免疫系统，其原理和机制是通用的。

目前人工免疫系统研究手段、内容及应用涉及多个领域，包括医学免疫学、计算机科学技术、计算智能、人工智能、模式识别、智能系统、控制理论与控制工程等。这些领域是比较典型的交叉学科，其理论极为广泛和丰富，发展出的人工免疫系统形式也是多种多样的。国外的专家一般在以上三个领域都有所涉猎，先从事计算机科学研究，而后从事人工免疫系统研究的专家居多。人工免疫系统与其他学科和领域的交叉关系及其应用如图 6.1 所示。

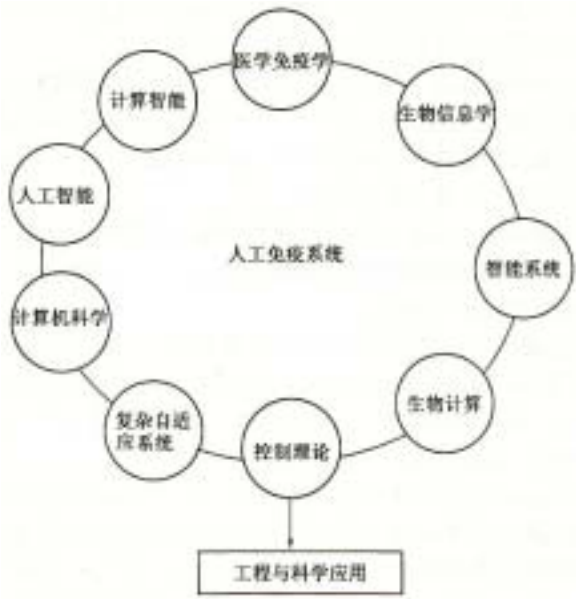


图 6.1　人工免疫系统与其他学科和领域的交叉关系及其应用

6.2 生物免疫系统

自然免疫系统是一个复杂的自适应系统，可保护人体不受外部病原体侵害，并把体内所有的细胞或分子分成或者属于自己的种类(自体细胞)，或者属于外部来源的非自体分子种类。免疫系统不依靠任何中心控制，具有分布式任务处理能力，具有在局部采取行动的智能，也通过起交流作用的化学信息构成网络，进而形成全局观念。

免疫系统多种多样，具有独特性。同样是人，一个人和另外一个人的免疫系统除了本质构成一样外，具体的状态、功能则千差万别。免疫系统是生物，特别是脊椎动物和人类所具有且必备的防御机制。人的免疫系统最为复杂，它由免疫效应分子、有关的基因及具有免疫功能的细胞、组织、器官等组成，可以保护机体，抗御病原体、有害异物等致病因素的侵害。

免疫系统的主要功能是：免疫防御、免疫稳定、免疫监督。免疫系统基本元素包括巨噬细胞、淋巴细胞及其抗体，抗体识别特定抗原并清除抗原。生物系统具有大量发达的抗体系统，能够适应不断变化的环境。

免疫系统分为两个主要部分：固有免疫系统和自适应免疫系统。固有免疫系统是抵抗抗原感染的第一道防线，抗原多数在这里被阻止。如果固有免疫系统被攻破，则自适应免疫系统针对特定感染病原体开始发挥作用。自适应免疫系统能够记住入侵的抗原特征，预防下一次袭击。适应性免疫调节有两个分支：①体液免疫，由 B 细胞及其产物介导；②细胞免疫，由 T 细胞介导。两个分支都对防御遵循类似的步骤顺序—扩增、活化、感应、分化、分泌、袭击、抑制、记忆。但是，它们以不同方式完成任务。

免疫系统具有两种应答方法：初次应答和二次应答。初次应答发生在免疫系统遭遇第一次遇到过的抗原并对其反应的时候。免疫系统能够学习抗原，该机制产生免疫记忆，这样为身体再次遇到同样的抗原时产生二次应答。当抗体结合一个抗原时，B 细胞受刺激产生自体克隆，成长的克隆体现一种变异机制使免疫系统具有适应性。

图 6.2 表明当一种抗原侵入免疫系统后，系统有一个产生抗击抗原感染的抗体的初

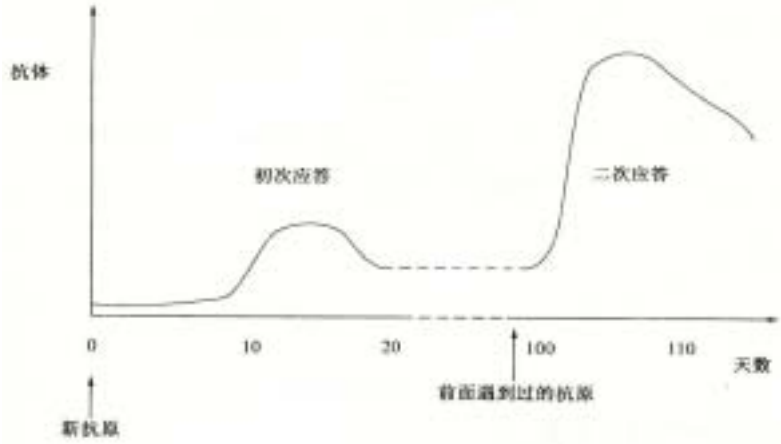


图 6.2　免疫应答过程

始化过程。但是，几天之后，抗体的浓度水平开始下降，直到再次遇到抗原。自适应免疫系统能够学习和记忆特异种类的病原体。初次应答是对以前未见过的病原体的应答过程。初次应答学习过程很慢，发生在初次感染的前几天，要用几周的时间清除感染，而二次应答则对遇到的同样抗原的反应非常迅速。y 轴是免疫应答的强度测量，由抗体浓度表示。

生物进化和免疫系统进化之间在时间范围和目的上有显著区别：在生物进化系统中，进化需要长时间进行才能改善一个物种种群的性能；在免疫系统中，寻找合适的抗体种群成员的目标所用时间可以短到几天。免疫系统的进化可分为两个不同模式：①缓慢进化模式，体现在 DNA 分子的进化（全局优化）；②快速进化模式，体现在免疫系统在与外部抗原斗争时的自适应性（局部优化）。

6.3 人工免疫系统

6.3.1 人工免疫机理

为了适应环境的复杂性和异敌的多样性，生物免疫系统采用了单纯冗余策略。这是一个具有高稳定性和可靠性的方法。免疫系统是由 10^7 个免疫子网络构成的一个大规模网络，机理很复杂，尤其是其所具有的信息处理与机体防御功能，为工程应用提供了新的概念、理论和方法。对这些可借鉴的相关机理扼要阐述如下：

1）记忆学习

免疫系统的记忆作用是众所周知的，如患了一次麻疹后，第二次感染了同样的病毒也不致发病。这种记忆作用是由记忆 T 细胞和记忆 B 细胞所承担的。这是因为在一次免疫响应后，如果同类抗原再刺激时，在短时间内，免疫系统会产生比上一次多得多的抗体，同时与该抗原的亲和力也提高了。免疫系统具有识别各种抗原并将特定抗原排斥掉的学习记忆机制，这是与神经网络不同的记忆机制。

2）反馈机制

图 6.3 反映了细胞免疫和体液免疫之间的关系，以及抗原（Ag）、抗体（Ab）、B 细胞（B）、辅助 T 细胞（T_H）和抑制 T 细胞（T_S）之间的反应，体现了免疫反馈机理。其中，IL^+ 表示 T_H 细胞分泌白细胞介素，IL^- 表示 T_S 细胞分泌白细胞介素。由图 6.3 可见，当抗原进入机体并经周围细胞消化后，将信息传递给 T 细胞，即传递给 T_H 细胞和 T_S 细胞，

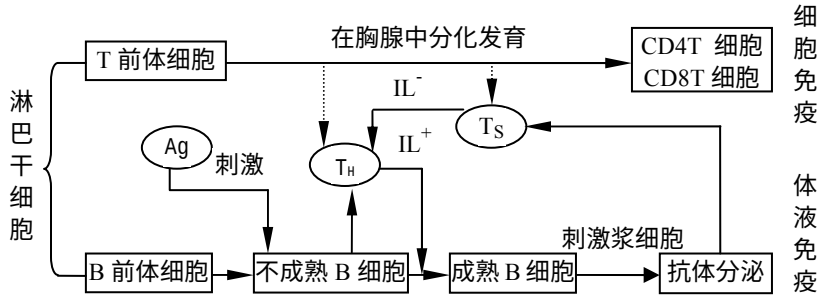


图 6.3 细胞免疫和体液免疫

T_S 细胞用于抑制 T_H 细胞的产生。然后共同刺激 B 细胞，经过一段时间后，B 细胞产生抗体以清除抗原。当抗原较多时，机体内的 T_H 细胞也较多，而 T_S 细胞却较少，从而产生的 B 细胞会多些。随着抗原的减少，体内 T_S 细胞增多，它抑制了 T_H 细胞的产生，则 B 细胞也随之减少。经过一段时间后，免疫反馈系统便趋于平衡。利用这一机理可以提高进化算法的局部搜索能力，突生出具有特异行为的网络，从而提高个体适应环境的能力。

对上述反馈机理进行简化，定义在第 k 代的抗原数量为 $\varepsilon(k)$，由抗原刺激的 T_H 细胞的输出为 $T_H(k)$，T_S 细胞对 B 细胞的影响为 $T_S(k)$，则 B 细胞接收的总刺激为：

$$S(k) = T_H(k) - T_S(k) \tag{6-1}$$

式中，$T_H(k) = k_1\varepsilon(k)$；$T_s(k) = k_2 f[\Delta S(k)]\varepsilon(k)$。$f[\cdot]$是一个选定的非线性函数。特别地，对于控制系统，若将抗原的数量 $\varepsilon(k)$ 作为偏差，B 细胞接收的总刺激 $S(k)$ 作为控制器输出 $u(k)$，则有以下反馈控制规律：

$$u(k) = \{k_1 - k_2 f[\Delta u(k)]\}\varepsilon(k) \tag{6-2}$$

显然，构成了一个参数可变的比例调节器。

3）多样性遗传机理

在免疫系统中，抗体的种类要远大于已知抗原的种类。解释抗体的多样性有种系学说和体细胞突变学说。其主要原因是受基因片段多样性的连接以及重组和轻链配对等复杂机制所控制。该机理可以用于搜索的优化，进化地处理不同抗原的抗体，提高全局搜索能力，避免陷入局部最优。

4）克隆选择机理

由于遗传和免疫细胞在增殖中的基因突变，形成了免疫细胞的多样性，这些细胞的不断增殖形成无性繁殖系。细胞的无性繁殖称为克隆。当每一种抗原侵入机体都能在机体内选择出能识别和消灭相应抗原的免疫细胞克隆，使之激活、分化和增殖，进行免疫应答以最终清除抗原，这就是克隆选择。但是，克隆—无性繁殖中父代与子代间只有信息的简单复制，而没有不同信息的交流，不能促使进化。因此，需要对克隆后的子代进行进一步处理。

5）其他机理

免疫系统所具有的无中心控制的分布自治机理、自组织存储机理、免疫耐受诱导和维持机理以及非线性机理均可用于建立人工免疫系统。

6.3.2 人工免疫算法

正是因为对免疫机理的认识还不是十分系统、深入，所以，有关于人工免疫系统算法（以下简称免疫算法）的研究成果并不多。本文对目前人工免疫系统领域中几类比较有代表性的算法总结如下：

1）基于免疫网络学说的人工免疫网络模型

目前，两个比较有影响的人工免疫网络模型是 Timmis 等人（2001）的资源受限人工免疫系统（resource 1imited artificial immune system，RLAIS）和 Casto 等人（2000；2001）的 aiNet。

资源受限人工免疫系统是 Timmis 在 Cook 和 Hunt 研究的基础上提出的，他还给出了

人工识别球（artificial recognition ball，ARB）的概念。Timmis 认为 ARB 的作用与 B 细胞的功能是类似的，人工免疫系统是由固定数量的 ARB 组成。进一步地，类比自然免疫系统，认为 ARB 受到的刺激包括：抗原的主要刺激 ps；邻近抗体的刺激 nn 以及邻近抗体的抑制 ns；而且，抗体的克隆水平可以 ARB 受到的刺激来确定。

Casto 的 aiNet 算法模拟了免疫网络对抗原刺激的刺激过程，主要包括抗体抗原识别、免疫克隆增殖、亲合度成熟以及网络抑制，免疫网络被认为是一个付权无向图，且不是全连接的。但是，目前的免疫网络模型普遍存在自适应能力比较差、参数比较多、而且过分依赖对网络节点的增减来保持网络动态，缺乏对免疫网络非线性信息处理能力认识等缺陷；同时，算法设计的出发点一般都集中在数据压缩上，因此，限制了算法的应用范围。

2）基于免疫特异性的否定（负）选择算法

Forrest 等人（1994）根据免疫系统的自己／非己的区别原则，研究了一种检测变化的否定选择算法。提出的算法与 T 细胞成熟过程中经历的“否定选择”过程有着相似的原理：随机产生检测器，删除那些测到自己的检测器，以使那些测到非己的检测器保留下来。算法简述如下：

（1）定义自己长度为 L 的有限个字母串的类集 S，S 是一个需要保护监视的集合。例如，S 可以是程序、数据文件(任何软件)。

（2）产生一个检测器的集合 R，每一个都不能与 S 集合中的任一个相匹配，采用部分匹配的原则，而不是精确或完美匹配，即 2 个串至少在 r 个位置上能区别出来。这里，r 是可选择的参数。

（3）监视，不断地将检测器 R 与受监视集合 S 进行匹配，一旦发生匹配，就表示已发生一个变化，检测器是按照不能与 S 中的任一串相匹配设计的。

否定选择算法为免疫在计算机网络安全领域的应用奠定了理论基础。近年来，随着计算机网络的飞速发展，安全问题也从简单的计算机病毒检测，扩展到基于主机的入侵检测和网络安全，这些都可以从免疫系统的信息处理机制中获得启示。

3）克隆算法

在 http://www.google.com／搜索的 1330 个与克隆选择算法相关的英文搜索结果中，最具代表性的算法是 Castro（2000）以及 Kim（2002）等人提出的；而 4 个相关的中文结果都是与计算机编程相关的概念。中国期刊网中出现“克隆”与“算法”的文章共 443 篇，但是几乎没有真正涉及克隆算法本身的文章。

如图 6.4 所示的是 Castro 克隆选择算法（CSA）的流程框图。具体算法如下：

（1）生成候选解集 P，P 是由记忆单元（M）和保留种群（只）组成，即：P=Pr+M。

（2）根据亲合度测量，选择 n 个个体（Pn）。

（3）复制（克隆）种群中这 n 个最好的个体，生成一个克隆临时种群（C），克隆规模和抗体抗原的亲合度成正比。

（4）对克隆临时种群进行高频变异，这里，高频变异和抗体-抗原的亲合度相对应，由此获得一个变异后的抗体群（C^*）。

（5）从 C^*中重新选择改进的个体组成记忆单元 M。P 中的一些个体也被 C^*中其他改进的个体所取代。

（6）利用新产生的抗体代替 d 个旧抗体（引入多样性）。亲合度低的抗体更容易被

取代。

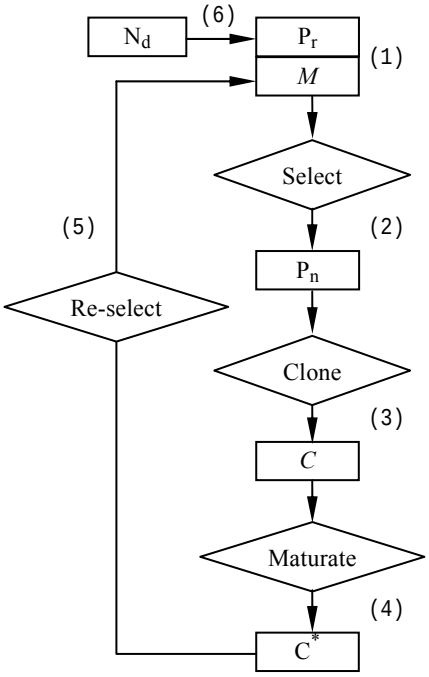


图 6.4　CSA 的算法流程

该算法与一般遗传算法相比的不同点在于，首先，将基于概率的轮赌选择改变为基于抗体—抗原亲合度（适应度）的比例选择；其次，构造了记忆单元，从而将遗传算法记忆单个最优个体变为记忆一个最优解的群体；另外，通过新旧抗体的替代，增加了种群多样性。

Kim 等人在 Hofmeyr（1999）的基础上提出了一种动态克隆选择算法 DynamiCS（dnamic clonal selection algorithm），并用于解决连续变化环境中异常探测的问题。DynamiCS 试图提取使系统产生适应性的关键变量（即减少系统参数的数量以保证算法可用）。DynamiCS 算法的流程用伪码表示，仅从算法流程来看，该算法更多地模拟了免疫系统的外物识别功能。

国内的研究者虽然也已经开始重视对人工免疫系统的研究，但是将更多的注意力还是放在利用免疫机理对进化算法的改进上，虽然这些算法多数都被冠以免疫算法的名字，但本质上可以说只是利用了免疫系统的相关机理对遗传算法改良策略的重新命名。相比较而言，对免疫网络的研究比对克隆选择的研究更加深入。

总之，目前对克隆选择的研究还比较少，已有的算法也多仅给出了算法框架和有关应用，多是静态的、非自适应的；也缺乏对克隆选择机理的深入认识以及与其他算法(主要是进化算法)的深入对比研究，有关收敛性分析等的理论研究就更少。

4）免疫进化算法

进化算法作为一种有向随机搜索的优化方法得到了广泛应用。然而在实际应用中也存在一些需要改进之处：无法保证收敛到全局最优解、群体中最好个体的丢失、进化过程的过早收敛等。将进化与免疫结合起来考虑，能得到更有效的优化算法。在进化算法

的进化大框架上，引入免疫系统的诸多特性，发展起来了很多免疫优化算法，本文称这类算法为免疫进化算法。主要包括：引入疫苗接种理论的免疫优化算法、引入自我调节机制的免疫算法、基于免疫应答的免疫优化算法、基于免疫抗体记忆的免疫算法、借鉴生物独特性网络调节的进化算法、具有免疫体亲近性特征的遗传算法等。这些改进算法可以快速求出满足一定精确度要求的最优解，对解决工程应用问题具有应用价值。

另外，还出现了与其他智能策略相结合的混合免疫方法，如：KrishnaKumar（1997）等人将神经网络和免疫系统机理相结合提出了“免疫神经控制（INC）”的结构；Sasaki（1999）等人提出了一种基于免疫系统反馈机理的自适应学习的神经网络控制器；李亭鹤等人（2001）针对凹形重叠区难以精确寻点的问题，提出了一种新的洞点搜索方法—感染免疫法，实际应用表明，该方法无论是在寻点能力方面，还是在通用性方面都优于传统模式。

6.4 人工免疫系统的应用领域

人工免疫系统的主要应用见领域表 6-2。

表 6-2 人工免疫系统主要应用领域

应用领域	示 例
控制	电压调节器的控制，复杂动力学系统自适应控制
规划	电网规划
设计	设计人工神经网络
组合优化	TSP 问题，CDMA 多用户检测
图像处理	图像分割，立体匹配
数据处理	多组分混合色谱信号的解析
知识发掘	数据库知识发现
机器人	多智能体决策系统，分布式自动机器人系统等
故障监测和诊断	加工工具破损监测，旋转机械在线故障诊断

（1）控制。KrishnaKumar 等人（1997）将“免疫神经控制（INC）”用于复杂动力学系统的模型自适应控制，效果良好。Sasaki 等人（1999）提出了一种基于免疫系统反馈机理的自适应学习的神经网络控制器，避免了神经网络学习在最小值附近的摆动，提高了收敛速度。丁永生等人（2000）针对低阶或高阶对象，提出一种新颖的基于生物免疫系统反馈机理的通用控制器结构，该控制器包括一个基本的 P 型免疫反馈控制器和一个增量模块，P 型免疫反馈规律由模糊控制器自动调整，控制增量模块可以由常规控制或神经网络来实现。激光热疗法中组织温度控制的计算机仿真结果表明，该控制器的控制性能优于常规控制器（丁永生与唐明浩，2001）。李海峰等人（2001）提出了以电力系统电压调节为应用目的的免疫系统的基本模型，演示了应用于 STATCOM 的细胞免疫电压调节器的控制作用。

（2）B 规划。高洁（2001）将一种新的随机优化方法—免疫算法应用于电网规划，

利用 IEEE-6 节点系统作为样本网络进行分析计算，并将该方法与基于遗传算法的电网规划方法进行比较。结果表明，免疫算法在全局寻优的性能方面要优越于遗传算法。

（3）设计。张军等人（2000）利用共生进化原理设计人工神经网络，创造性地融入了免疫调节原理中的浓度抑制调节机制以保持个体的多样性，提出了基于免疫调节的共生进化网络设计方法。周伟良等人（1999）结合遗传算法的随机全局搜索能力和生物免疫中抗体通过浓度的相互作用机制，构造了免疫遗传算法，并利用实验验证了其在设计神经网络时的有效性。

（4）组合优化。曹先彬等人（2000）用一种免疫遗传算法有效地解决了装箱问题的求解。王煦法等人（1999）、刘克胜等人（2000）提出的免疫遗传算法（iramune genetic algorithm，IGA）成功实现了 TSP 优化。牛志强等人（2001）用免疫算法解决 CDMA 中的多用户检测问题。曹先彬等人（2000）构造的免疫进化策略在求解二次布局问题时取得了完美的结果。

（5）图像处理。McCoy（1997）等人将人工免疫系统用于图像分割。王肇捷等人（2001）为了得到最佳视差图，将免疫算法用于解决计算机视觉中的立体匹配；与基于像素点灰度匹配相比，免疫算法的匹配效果要好；与模拟退火匹配相比，虽然都能得到全局最优的视差图，但免疫算法的匹配速度要快。

（6）数据处理。邵学广（2000，2001)将免疫机理用于信号拟合，实现了多组分混合色谱信号的解析；利用免疫—遗传算法实现了二维色谱数据的快速解析；通过对免疫系统中抗体对外来抗原的识别、消除等过程的模拟，建立了一种新型的免疫算法模型，为利用数据库解析混合物或生物大分子等物质的复杂 NMR 谱图开辟了一条全新的途径。杜海峰等人（2001）基于智能互补融合观点，提出了一种新的数据浓缩方法 ART 人工免疫网络，并用于 R2 空间分类和 Fisher 花瓣问题的实验。

（7）知识发掘。J.Timmis 等人（1999）将人工免疫系统用于数据库知识发现，与单一联结聚类分析和 Kononen 网络作了比较，认为人工免疫系统作为数据分析工具是适合的。

（8）机器人。Dasgupta（1998）基于人工免疫系统建立了多智能体决策系统。Meshref（2000）等人探讨了自然免疫系统的行为，并利用其对外部环境变化敏感的特性改进了 DNA 算法，用于“狗-羊”问题的结果表明，改进的 DNA 算法适用于解决分布式自动机器人系统问题。Jin-Hyung Jun 等人（1999）的人工免疫系统在分布式自动机器人系统实现了协作和群行为。R.L.King（2001）等人提出了一个用于智能体的人工免疫系统模型，并总结了人类免疫系统可用于人工免疫系统智能体的主要功能。刘克胜等人（2000）基于免疫学的细胞克隆学说和网络调节理论，提出了能有效增强自律移动机器人在动态环境中自适应能力的新算法。

（9）故障监测和诊断。Dasgupta（1999）等人将人工免疫系统用于工业中，进行加工工具破损监测。刘树林等人（2001）受生物免疫系统自己非己识别过程的启发提出了反面选择算法，在故障诊断应用领域中改进了反面选择算法，提出了对旋转机械在线故障诊断的新方法。杜海峰等人（2002）还将 ART-人工免疫网络用于解决多级往复式压缩机故障诊断，效果良好。

（10）其他。人工免疫系统的理论和方法还广泛应用于计算机安全和密码学等领域。如杨晓宇等人（2001）对 AIS 与网络安全相结合的基因计算机进行了全面的描述，并认

为智能模拟在网络安全方面的应用前景广阔。

6.5 实例分析：人工免疫算法在车辆路径优化中的应用

6.5.1 车辆路径问题的数学模型

车辆路径问题可以描述为：

有一个中心仓库和 n 个顾客（1，2，…，n），第 k 个顾客需要运输的货物量为 d_k（k=1，2，…，n），中心仓库派出若干货车，为 n 个顾客运输商品，求满足货运需求的最低成本车辆运输行程路线，并满足下述条件：

（1）每个顾客只能由一辆汽车来负责运货；

（2）一辆汽车经过的配送路径上各个顾客的商品需求量之和不能超过该汽车的载重量；

（3）要求每个顾客的商品需求都必须得到满足。

假设中心仓库有 K 辆车，负责为 L 个顾客进行运输配送，每个车辆载重为 g_k（k=1，2，…，K），每个顾客的需求为 d_i（i=1，2，…，L），顾客 i 到顾客 j 的运距为 c_{ij}（i=0，1，…，L-1，j=1，2，…，L，当 i=0 时，c_{0j} 表示中心仓库和各个顾客之间的距离）。设 n_k 为第 k 辆车所运输的顾客数量（若 n_k=0 表示没有使用第 k 辆车），用集合 R_k 表示第 k 条路径，其中的元素 r_{ki} 表示 r_{ki} 在路径 k 中的顺序为 i，令 r_{k0}=0 表示中心仓库，则车辆路径问题的数学模型为：

$$\text{Minimize}\ \left(\sum_{k=1}^{k}\sum_{i=1}^{n_k}\left(C_{r_k(i-1)}r_{ki}+C_{\mathrm{r_{ko}}\mathrm{r_{knk}}}*\operatorname{sign}(n_k-1)\right)\right)$$

$$\sum_{i=1}^{n_k} dr_{ki}\le g_k \qquad k=1,2,\cdots,k \tag{6-3}$$

$$0\le n_k\le L \qquad k=1,2,\cdots,k \tag{6-4}$$

$$\sum_{k=1}^{k} n_k = L \tag{6-5}$$

$$R_k\left\{r_{ki}\middle| r_{ki}\in\{1,2,\cdots,L\}, i=1,2,\cdots,n_k\right\} \tag{6-6}$$

$$R_{k1}\cap R_{k2}=\varphi \qquad \forall k_1\ne k_2 \tag{6-7}$$

$$\operatorname{sign}(n_k-1)=\begin{cases}1, & n_k\ge 1\\ 0, & \text{otherwise}\end{cases}$$

上式中不等式（6-3）保证每条路径上的各个顾客的总需求量必须小于运输车辆的载重量，不等式（6-4）表明每条路径上的顾客数量不能超过总顾客数量，等式（6-5）要求每个顾客都必须有车辆为其服务，等式（6-6）表示每辆车的行车路线，等式（6-7）则限制每个顾客的商品需求只能由一辆车来完成。

6.5.2 车辆路径问题的免疫算法实现

本例中，设有 1 个中心仓库和 8 个顾客，顾客之间的距离、顾客和中心仓库之间的距离以及各个顾客的需求量如表 6-3 所示。每个顾客提出的商品需求不可能正好将所有车

辆都使用，因此可以将车辆总数设为一个较大值 10，车的载重量为 8 吨。

表 6-3　顾客间距离及各顾客的需求量

$C_{i,j}$	0	1	2	3	4	5	6	7	8
0	0	4	6	7.5	9	20	10	16	8
1	4	0	6.5	4	10	5	7.5	11	10
2	6	6.5	0	7.5	10	10	7.5	7.5	7.5
3	7.5	4	7.5	0	10	5	9	9	15
4	9	10	10	10	0	10	7.5	7.5	10
5	20	5	10	5	10	0	7	9	7.5
6	10	7.5	7.5	9	7.5	7	0	7	10
7	16	11	7.5	9	7.5	9	7	0	10
8	8	10	7.5	15	10	7.5	10	10	0
需求量		1	2	1	2	1	4	2	2

用免疫算法解决车辆路径问题的步骤如图 6.5 所示。用车辆路径问题的数学模型中目标函数以及各种约束条件作为求解该问题的抗原。

1）构造解的编码，产生初始抗体

根据该问题的求解特性，我们这里采用整数编码方法，且以顾客数量作为编码数量。本例中顾客的数量为 8，车辆的数量为 10，那么我们用 1～8 之间的 8 个互不相同的自然数编码表示各辆车为顾客运输的先后次序，这 8 个自然数的 n（$0<n\leq 10$）个小段，分别表示由 n 辆车运输的顾客。假设产生的编码为：12834657，这个编码表示了各个顾客的运输先后次序和运输路线。具体地说，该编码的一个小段，如“ 128 ”表示由一辆车对编号为 1、2、8 的顾客运输，但实际上有多少辆车，或者说具体需要多少辆车，则需要根据汽车的载重量和顾客的需求量通过计算确定。而且编码中的顾客顺序不同，最终可能导致本次运输所需车辆数也是不同的。

初始抗体的产生是利用随机函数生成符合上述编码规则的群体。命令函数为：

chromGroup =initchromGroup (chromGroupsun,chromlength)

chromGroupsun—群体大小；chromlength—字符串长度（个体长度），即染色体的二进制编码长度。

2）疫苗提取与接种

在实际的操作过程中，对所求解的问题（这里视为抗原）进行具体分析，从中提取出的最基本的特征信息，叫做疫苗。对提取出来的疫苗进行处理，将其转化为求解问题的一种方案（根据该方案而得到的各种解的集合统称为基于上述疫苗所产生的抗体）。疫苗的正确选择对算法的运行效率具有十分重要的意义。

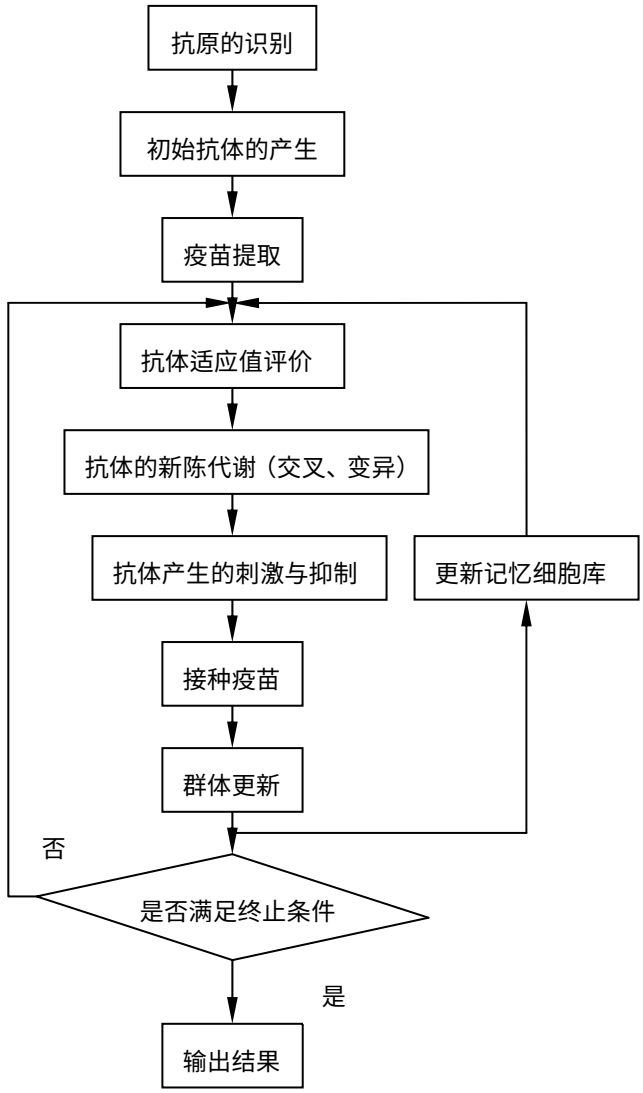


图 6.5　免疫算法流程图

在此，根据所要解决的实际问题，考虑如果将相互之间距离比较近的顾客由一辆车运输，可以节约运输路径。因此疫苗提取与接种策略就是依次比较顾客之间的相互距离，抗体编码时将距离较近的顾客放在一起，用一辆车运输，节省运输费用。疫苗提取的命令函数为：

[bestChrom,bestfit]=best_worstChrom(fatherChromGroup,fitchrom)

bestChrom—最优个体；bestfit—最优个体的适应值。

疫苗接种的命令函数为：

inoculateChrom=immunity(sonchromGroup,bacterinChrom,parameter)

parameter—— 1，随机制取染色体接种；2，每个染色体都接种；3，每个染色体都接种，但接种的位置是随机的。

3）抗体的适应值计算

对于每一个抗体 Ag，其抗体对应的的适应值为 F，F 越大，表示该抗体对应的路径越短，也就是说解越接近于最优解。适应值计算的命令函数为：

fitchrom=calfitchrom (fatherchromGroup,d)

chromGroup—染色体群；d—两个体之间的距离。

4）抗体的交叉、变异

抗体交叉的命令函数为：

sonChromGroup=crossChrom (slecteChromGroup,2)

slecteChromGroup—按概率选择的需要进行交叉的染色体组。

抗体交叉后变异的命令函数为：

sonChromGroup=varianceCh(sonChromGroup,0.8)

抗体不交叉直接变异的命令函数为：

sonChromGroup=varianceCh(fatherChromGroup,0.1)

5）群体更新

求得每代群体中的个体概率，好的染色体概率高，坏的染色体概率低，按概率选择交叉变异函数，产生下一代个体。这种方式抑制了抗体的浓度和确保了抗体的多样性。定义概率函数为：

p=chromProbability(fitchrom)

按概率选择染色体函数为：

slecteChromGroup=selecteChrom(fatherChromGroup,p)

6）记忆库的设计

抗体记忆机制的引入是免疫算法的一个重要特点，系统在每一代抗体的求解后，能保留一定数量的较优抗体，放入记忆库中，同时替换掉较差的若干个抗体。这样可以避免进化的过程中随着交叉变异操作的进行，失去最优解的可能性，从而提高了问题求解的收敛速度，体现了免疫算法自适应环境的能力。保留最优个体的命令函数为：

[holdBestChrom, holdbestfit] = compareBestChrom (holdBestChrom, holdbestfit, bestChromo, bestfit)

7）中止条件

在此，我们采用进化的代数 N 作为循环迭代过程的结束，如果等于 N，则算法结束，输出适应值最高的解；否则，继续重复上述过程。

6.5.3 模型效果检验与结论

运用上一节提到的编码方式，得到本模型的编码的长度为 8 位，设定种群规模为 50，进化代数为 100，上机运算 10 次，运行结果如表 6-4 所示。

由表 6-4 中数据可知，免疫算法的运行结果大多数接近最优解（67.5），其平均路径长度为 68.05。而对于同样的数据，用遗传算法求解该问题的运行结果如表 6-5 所示，其平均路径为 70.05。

表 6-4 免疫算法运行结果

运行次数	1	2	3	4	5
路径长度	67.5	68.5	69.5	67.5	69
运行次数	6	7	8	9	10
路径长度	67.5	68.5	67.5	67.5	67.5

表 6-5 遗传算法运行结果

运行次数	1	2	3	4	5
路径长度	71.5	68.5	69.5	73.5	69
运行次数	6	7	8	9	10
路径长度	70	72	67.5	69.5	69.5

由上述实验数据可知，免疫算法搜索到全局最优解的效率要比遗传算法高，能够快速地求得最优解，是求解车辆路径问题的一种有效算法。

思考题

6.1 什么是人工免疫系统？免疫学的发展经历了哪些时期？
6.2 人工免疫系统的研究内容和范围主要包括哪几个方面？
6.3 生物免疫系统的两种应答方式各是怎样的？
6.4 目前常用的人工免疫算法有哪些？
6.5 简要叙述人工免疫系统的应用领域。

参考文献

[1] 莫宏伟.人工免疫系统原理与应用. 哈尔滨:哈尔滨工业大学出版社,2002
[2] 周志华,曹存根.神经网络及其应用.北京:清华大学出版社,2004
[3] 丁永生,任立红.人工免疫系统:原理与应用.模式识别与人工智能,2000,13(1):52-59
[4] 刘克胜,曹先彬,郑浩然.基于免疫算法的 TSP 问题求解.计算机工程,2000,26(1): 12-16
[5] 樊建华,王秀峰.基于免疫算法的车辆路径优化问题.计算机工程与应用,2006(4): 210-217
[6] 姜大立,杨西龙,杜文等.车辆路径问题的遗传算法研究.系统工程理论与实践,1999, 19(6): 40-44
[7] 王磊,潘进,焦李成.免疫算法.电子学报,2000,28(7):74-78
[8] 朗茂祥.基于遗传算法的物流配送路径优化问题研究.中国公路学报,2002,15(3)：76-79
[9] Danting,Ramser.The truck dispatching problem.MgtSci,1959(6),81-89
[10] Jerne N K.Towards a Network Theory of the Immune System.Annual Immunology, 125c, 1974: 373-389
[11] Perelson A S.Immune Theory. Immunological review,1986(10):5-36
[12] Dasgupta D. Artificial Immune Systems and Their ApplicationsM.Bedin Heidelberg. Springer-Verlang,1999
[13] ChunJang Sung,JangHyun Kyo,HahnSong Yop. A Studyon Comparison Optimization Performances between Immune Algorithm and Other Heuristic Algorithms. IEEE Trans on Magnetics,1998,34(5):2972-2975
[14] Forrest S,Hofmeyr S A.Immunology as Information Processing.In:Segel L A and Cohen I R,Design Principles for Immune System & Other Distributed Autonomous Systems. Oxford Univ.Press,2000:361-387
[15] J H Holland.Genetic algorithems and classifier systems:foundations and their applications J.Proceedings of the Second International Conference on Genetic Algorithms,1987:82-89
[16] Ishida YAdachi N.Active Noise Control by an Immune Algorithm: Adapation in Immune System as an EvolutionJ.Proc,ICEC,1996(96):150-153
[17] J D Farmer,N H Packard A S Perelson.The immune system. Adaptation and Machine Learning J.Physics 22D, 1987
[18] Booker,L B Goldberg,J H Holland.Classifier systems and genetic algorithms. Artificial Interlligence,1989 40:235-283
[19] Huan g S. Enhancement of thermal unit commitment using immune algorithms basedoptimization approachesJ . Electrical Power and Energy Systems,1999(21):245-252

附录：免疫算法的 MATLAB 程序

```
clear all;
N=8;                          % 每个染色体段数（十进制编码位数）
M=100;                        % 进化代数
chromGroupsun =50;          %设置初始参数，群体大小
length=10;                    % length 为每段基因的二进制编码位数
chromlength=N*length;   %字符串长度（个体长度），染色体的二进制编码长度
fatherchromGroup =initchromGroup (chromGroupsun,chromlength); %产生初始群体
holdbestfit =Inf;           %可能最优路径的适应度的初值
bestfit=Inf;
holdBestChrom=0;            %对应最优路径适应度的染色体的初值
for gen = 1: M
    for j = 1: chromGroupsun
        fitchrom(j)=calfitchrom (fatherchromGroup,d);     % 计算初始适应度
        if bestfit >= fitchrom(j)
            exit
        else
            bestfit= fitchrom(j)
        end
    end
    %选择最大适应度对应的最优染色体
    [bestChrom,bestfit]=best_worstChrom(fatherChromGroup, fitchrom);
    %保留最大适应度对应的最优染色体
     [holdBestChrom,holdbestfit]=compareBestChrom(holdBestChrom,holdbestfit, bestChromo,bestfit)
     %把保留的最好的染色体 holdBestChrom 加入染色体群中
    order=round(rand(1)*chromGroupSum);
    if order==0
        order=1;
    end
    fatherChromGroup(order,:)=holdBestChrom;
    fitchrom(order)=holdbestfit;
    for j = 1: chromGroupsun
        p(j)=chromProbability(fitchrom);       %为每一条染色体定义一个概率
        %从父染体中选择优秀染色体
        selecteChromGroup(j)=selecteChrom (fatherChromGroup,p);
        sonChromGroup(j)=crossChrom(selecteChromoGroup,2);              %杂交
        sonChromGroup(j)=immunity(sonChromGroup,holdBestChrom,3);%接种疫苗
        sonChromGroup(j)=varianceCh(sonChromGroup,0.5);              %杂交后变异
        % 解码并计算初始适应度
        fitchrom(j)=calfitchrom (fatherchromGroup,d);
        sonChromGroup(j)          %输出新个体
    end
        %统计种群的适应度
        [sumfitChrom,bestfit] = statistics(sonChromGroup,bestfit,gen);
        fatherChromGroup= sonChromGroup;
end
```

第 7 章 计算智能的未来发展

7.1 计算智能的主要研究成果

自从美国学者 JamesC.Bezedek 1992 年首次提出了“计算智能（computational intelligence，CI）”的说法以来，作为一个极有前途且颇有成效的研究方向， 使 AI（人工智能）真正从理论走向了应用。早在 1943 年，心理学家 Mclulloch 与数学家 Pitts 就总结了生物神经元的一些基本生理特征，并提出了著名的 MP 数学模型（形式神经元数学模型）。1949 年 Hebb 根据神经元联接强度的改变代表生物学习过程的假设提出了 Hebb 学习规则。虽然此后神经网络（nerual networks）的研究经历了几起几伏，但最终还是取得了许多丰硕成果，典型的有 1982 年产生的 Hopfield 网络、1986 年的 BP 网络和 1977 年的 Kohonen 无导师自组织竞争网络等。从上世纪 90 年代以来，一些学者开始注意到某些生物诸如蚂蚁、蜜蜂、鸟群及鱼群等群居生物依靠整个集体的行为能够完成觅食、清扫、搬运、御敌等高效的协同工作，能够建立起坚固、漂亮和精确的巢穴；能够在高速运动过程中保持和变换优美有序的队形等许多令人匪夷所思的事情。这就是一种智能行为。而人们正是由于受到这些生物行为的启发，将得到的这些方法同计算机科学相结合，由此来解决一些传统问题和实际应用中出现的新问题，这就是现在我们所说的仿生智能算法。智能优化算法发展至今，已经提出了许多各种各样的智能优化算法。

目前计算智能主要的研究成果有：

1）神经网络

神经网络由于其模型、拓扑关系、学习与训练算法等都建立在对生物神经元系统的研究之上，虽然离人们设想的程度还很远，但它仍是目前模拟人脑模式识别、联想、判断、决策及直觉的理想工具。它具有高度的并行性、非线性以及良好的容错性与联想记忆能力，同时它还拥有强大的学习能力和很好的自适应性。60 多年来，神经网络技术取得了巨大的发展：前向神经网络模型及其仿真算法、模糊神经网络、径向基函数网络、反馈型神经网络、支持向量机等理论与技术已经相当成熟，在专家系统、知识获取、智能控制、自适应系统中也同样具有良好表现。另外小波神经网络是近几年国际上的研究热点。小波分析在信号去噪、信号奇异点检测及信号发展趋势检测等方面具有其他分析方法无可比拟的优势。神经网络对任意线性和非线性函数具有良好的逼近性，且又具有很强的模式识别、自适应预报和故障诊断能力。小波神经网络综合了小波与神经网络的优点，成为目前众多科技工作者关心的课题。

2）模拟退火

模拟退火（simulated annealing）是基于 Monte Carlo 迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性而设计的一种智能优化算法。1982 年 kirkpatrick 等将退火思想引入组合优化领域，提出了一种解大规模组合优化问题，特别是ＮＰ完全组合优化问题的有效近似算法。模拟退火算法在求解大多数组合优化和连续变量优化等问题上都取得了良好的应用效果。例如：旅

行商问题、图着色问题、设备布局问题和布线问题等。

3）进化计算

早在上个世纪 40 年代, 就有学者开始研究如何利用计算机进行生物模拟的技术, 他们从生物学的角度进行了生物的进化过程模拟、遗传过程模拟等研究工作。进入 60 年代后, 美国密歇根大学的 Holland 教授及其学生们受到这种模拟技术的启发,创造出了一种基于生物遗传和进化机制的，并适合于复杂系统优化计算的自适应概率优化技术——遗传算法。进入 90 年代, 进化计算（genetic algorithm, 缩写为 GA）迎来了兴盛发展时期, 无论是理论研究还是应用研究都成了十分热门的课题。尤其是遗传算法的应用研究显得格外活跃, 不但它的应用领域被扩大, 而且利用遗传算法进行优化和规则学习的能力也显著提高, 同时产业应用方面的研究也在摸索之中。

近年来，基于自然选择和遗传学的进化计算方法在人工智能的应用研究中取得了引人注目的成果。GA 与其他智能计算方法相结合，交叉形成了许多新的研究方向和应用，在组合优化、机器学习、自适应控制、规划设计、智能机器人、人工生命等领域都发挥了巨大的作用。

4）蚁群优化

受到蚂蚁觅食现象的启发，Colorni、Dorigo等学者于1991年首次提出了蚁群算法，并用这一算法解决了一系列组合优化问题。在蚁群算法的基本原理研究方面，针对蚁群算法收敛性、算法参数设置等，提出了带精英策略的蚁群算法、最大最小蚂蚁系统等，改进后的算法提高了搜索速度，避免了算法早熟。将蚁群算法特点与其他智能算法相结合则形成了混合智能优化算法，如将遗传算法和蚁群算法相结合、蚁群算法同爬山法相结合、蚁群算法与混沌算法相结合等。这些算法互相融合，提高了算法的性能。

在研究算法理论的同时，蚁群算法的应用领域也在不断拓展。在静态组合优化中蚁群算法用来解决旅行商（TSP）问题、二次分配问题（QAP）、车间作业调度问题（JSP）、车辆路线问题（VRP）等。在上述问题中，蚁群算法都是有效的，并且在多数情况下优于其他智能优化算法。在动态组合优化中，1997年Schoonderwoerd.R, Holland.O,Brute.J等人将ACO算法应用于路由问题；1998年Di Caro.G, Dorigo等人又成功地将蚁群算法用于高速有向连接网络系统中，达到了公平分配效果最好的路由；蚁群算法还用于电力系统故障诊断、模糊系统、数据挖掘、聚类分析、系统辨识、图像处理以及化学工程等方面。

5）粒子群优化

动物学家 Reynolds 对鸟群的飞翔和群舞行为很感兴趣，而动物学家 Heppner 则对于大数目的鸟群为什么能如此一致地朝一个方向飞行、突然同时转向、分散、再聚集很感兴趣。他们都猜测这些变幻莫测的群体行为仅仅是由于单个个体某种简单的行为规则所导致的。因此，他们猜测鸟群之所以能在快速飞行中保持各种美妙的队型，仅仅是因为每只鸟为了避免和其他的鸟发生碰撞而努力在飞行中与其他鸟保持着合适的最优距离。为了理解其中的奥妙，这些研究者通过对每个个体的行为建立简单的数学模型，然后在计算机上模拟和再现这些群体行为。这种现象，在其他动物群体中，比如畜群、鱼群以至于人类群体也同样存在。社会生物学家 Wilson 认为“ 至少从理论上说，群体中的单个成员在搜寻食物的过程中能够利用其他成员曾经勘测和发现的关于食物位置的信息，当事先不确定食物位于什么地方时，这种信息的利用是至关重要的，这种信息分享的机制

远远超过了由于群体成员之间的竞争而导致的不利之处 ”。以上对于动物群体的观察说明了群体成员之间的信息分享是非常重要的，粒子群优化算法就是基于这样的基本原理而设计出的一种优化算法。该算法认为：在群体搜寻最优目标时，每个个体将参照当前群体中曾有的最优个体和自身曾经达到的最优位置来调整下一步的搜寻方向。

粒子群算法主要应用于非线性复杂约束规划、作业调度优化、经济分配和数据挖掘等。在工程应用方面，微粒群算法在化学工程中用于化学过程的动态分析；在生物工程中用于蛋白质序列 HMMs 模型训练；环境工程中用于大气中臭氧层的预测；农业工程中的温室环境温度预测控制；以及用于光纤通信、电力系统优化和控制参数优化等。

6）人工免疫系统

人工免疫系统（artificial immune systems）是 20 世纪 80 年代，Farmer 等人率先基于免疫网络学说所给出的免疫系统的动态模型，并探讨了免疫系统与其他人工智能方法的联系，开始了人工免疫系统的研究。直到 1996 年 12 月，在日本首次举行了基于免疫性系统的国际专题讨论会，首次提出了 “ 人工免疫系统 ”（AIS）的概念。随后，人工免疫系统进入了兴盛发展时期，D. Dasgupta 和焦李成等认为人工免疫系统已经成为人工智能领域的理论和应用研究热点，相关论文和研究成果也正在逐年增加。1997 年和 1998 年 IEEE 国际会议还组织了相关专题讨论，并成立了“ 人工免疫系统及应用分会 ”。D. Dasgupta 系统分析了人工免疫系统和人工神经网络的异同，认为在组成单元及数目、交互作用、模式识别、任务执行、记忆学习、系统鲁棒性等方面是相似的，而在系统分布、组成单元间的通信、系统控制等方面是不同的，并指出自然免疫系统是人工智能方法灵感的重要源泉。Gasper 等认为多样性是自适应动态的基本特征，而 AIS 是比 GA 更好地维护这种多样性的优化方法。

由于免疫系统本身的复杂性，有关算法机理的描述还不多见，相关算子还比较少。Castro L. D.、Kim J、杜海峰、焦李成等基于抗体克隆选择机理相继提出了克隆选择算法。Nohara 等基于抗体单元的功能提出了一种非网络的人工免疫系统模型。而目前两个比较有影响的人工免疫网络模型是 Timmis 等基于人工识别球（artificial recognition ball， AR）概念提出的资源受限人工免疫系统（resource limited artificial Immune system， RLAIS）和 Leandro 等模拟免疫网络响应抗原刺激过程提出的 aiNet 算法。

7）模糊计算

模糊计算是计算智能的另一个重要方面。1965 年美国加州伯克利大学扎德教授发表论文，首先提出 “ 模糊集 ” 概念，用以解决多值逻辑及推理问题。从此以后，模糊集理论成为了模糊推理、模糊逻辑等模糊计算系统的理论基础。模糊计算方法的重要性在于它可以表现事物本身性质的内在不确定性，因此可以模拟人脑，认识客观世界的非精确、非线性的信息处理能力。模糊集合的出现是数学适应描述复杂事物的需要，扎德教授用模糊集合的理论找到解决模糊性对象加以确切化，从而使研究确定性对象的数学与不确定性对象的数学得以沟通起来；过去精确数学、随机数学描述所感到的不足之处得到了弥补。在模糊数学中，目前已有模糊拓扑学、模糊群论、模糊图论、模糊概率、模糊语言学、模糊逻辑学等分支。模糊数学是一门新兴学科，它已初步应用于模糊控制、模糊识别、模糊聚类分析、模糊决策、模糊评判、系统理论、信息检索、医学、生物学等各个方面。在气象、结构力学、控制、心理学等方面已有具体的研究成果。然而模糊数学

最重要的应用领域是计算机智能，不少人认为它与新一代计算机的研制有着密切的联系。

目前，世界上发达国家正积极研究、试制具有智能化的模糊计算机。1986 年日本山川烈博士首次试制成功模糊推理机，它的推理速度是 1000 万次／ｓ。1988 年，我国汪培庄教授指导的几位博士也研制成功一台模糊推理机——分立元件样机，它的推理速度为 1500 万次／ｓ。另外，模糊计算还可以对人的自然语言进行量化及模糊处理，在社会学方面也颇有建树。但是模糊计算理论还远没有成熟，对它也还存在着不同的意见和看法，有待实践去检验。

8）混沌理论

在大自然中存在的确定性现象和随机性现象之间，还存在一类可由确定性方程描述的非确定性现象——混沌现象。作为科学术语的“ 混沌 ”，指的是貌似随机的事件背后存在着的内在联系。混沌科学着眼于发现隐藏的模式、细微的差别、事物的敏感性，以及不可预测的事物千变万化的“ 规则 ”。

自 1963 年洛伦兹发表 《决定论非周期流 》论文以来，非线性科学获得了迅猛的发展，进一步深刻地揭示了非线性系统的共同性质、基本特征和运动规律。随着对混沌理论的深入研究，近几年来，它在科学和工程技术领域中的应用研究也迅速发展起来。在非线性电路系统中，混沌信号由于具有内在的随机行为，它的非周期、连续宽带频谱、类似的噪声等特性是很适用于保密通信系统的。特别是混沌同步的发现，使得以混沌理论为基础的保密通信领域步入了实际的应用阶段。20 世纪 90 年代，人们开始了对神经网络与混沌相互交融的研究，即研究大脑中的混沌现象。研究表明，混沌理论可用来理解脑中某些不规则的活动，从而使混沌动力学为人们深入研究神经网络提供了新的机遇。因此用神经网络研究或产生混沌及构造混沌神经网络就成为了人们极为关注的新课题。

9）混合逻辑

混合逻辑（hybrid logic）的历史可以追溯到 20 世纪 60 年代 ARTHUR.N 教授的研究工作，是指扩展多个模态逻辑命题使其具有更好的表现力，是自然语言及计算方程之间的纽带。在模型理论化、理论证明和逻辑分析等领域的应用十分广泛。1970 年 Robert Bull 的论文第一次正式定义了混合逻辑的概念。到了 20 世纪混合逻辑的研究成果已被广泛应用到知识发现和知识工程中。在混合逻辑的理论证明方面，2004 年 Braüner 的关于自然演义逻辑证明研究的论文给出了 Gentzen 系统模型，为理论证明和逻辑分析提供了强有力的工具。

另外在工业过程中也存在一些现象，如多子模切换、离散或逻辑输入以及输出不连续等，这些现象的存在使得传统的建模方法不能够很恰当地描述这类工业过程，而且在工业过程中还存在着很多定性的知识。这些知识在传统的建模框架中无法被集成。1999 年 Alberto Bemporad 等人研究了命题逻辑和受约束的混合整数线性之间的转换关系，并在此基础上提出了一种新的针对工业过程的混杂系统建模方法，即基于混合逻辑动态 MLD（mixed logic dynamic）的混杂系统建模。该方法在统一的框架底下描述了连续事件、离散事件以及它们之间的相互作用。在此框架下可描述混杂系统的建模、验证、控制与优化、性能分析以及在故障诊断和其他方面的应用。

计算智能的研究方向和成果还有很多，而且它们之间的结合与渗透又往往形成新的研究领域，对未来计算智能的发展方向有待我们进一步进行研究和探索。

7.2 计算智能的发展动力

计算智能具有多学科交叉的特点，应用数学、统计学、非线性科学、人工智能、自动控制、人工生命、信息论、仿生学、认知科学、脑科学、计算机科学、运筹学等领域的研究进展是 CI 发展的主要动力。

应用系统的复杂性和不确定性日益增加，对机器智能和智能机器的需求日益强烈。随着计算机技术的高速发展，CI 有着 AI 不可比拟的优势：事情可以符号化、可以精确量化、可以在串行的 Von Neumann 型计算机上运行；相反，对于人类在日常生活中辨认人物、听懂语音等这些具有 common-sense 性质的事情，传统的计算机却很难做到。研究表明，这些事情涉及复杂的计算概念及过程，虽然计算机运算速度高于人脑，但是人脑是由 1000 亿神经元互连后并行计算的，其效率和质量远远高于计算机。因此对于超高性能计算机的需求使计算智能成为当代最高新的技术之一，是实现各行各业系统、设备自动化、智能化的核心理论和技术。

脑科学、认知科学、复杂性科学和逻辑学正在迅速发展，可提供坚实的理论基础。计算智能已经在模式识别、优化计算、经济预测、金融分析、智能控制、机器人学、数据挖掘、信息安全、医疗诊断等领域的应用中都取得了显著的成果，然而计算智能理论尚需要不断地发展和完善，需要研究更实用的计算智能应用技术，体现出真正的智能化。未来随着人类思维机制的破译，以及基因技术、生物技术、计算机技术、电子技术、通讯技术等学科的发展，将会出现完全不同于现有计算智能的方法和装置，这种方法和装置的理论及技术核心是生物技术、仿生技术。这些相关交叉学科对于推动计算智能的发展作用十分明显，例如：生物信息学在现代控制理论和进化计算中的应用、生物心理学在模式识别中的应用等均作用显著。

应用数学研究的进展也将在极大程度上推动计算智能的发展。对于模糊集、粗糙集、可扩展的聚类技术等方向的理论研究也是 CI 十分重要的组成部分，这些技术在数据挖掘、模式识别以及混合动力系统中的应用已日趋成熟。

7.3 未来的发展方向

经过近些年的发展，智能优化算法凭借其简单的算法结构和突出的问题求解能力，吸引了众多研究者的目光，并取得了令人注目的成果。大量的研究成果证明了智能优化算法在工程上的适用性，显示了其给各个领域带来的效益。但是，由于其理论依据来源于对生物群落社会性的模拟，因此其相关数学分析还比较薄弱，这就导致了现有研究还存在很多问题。如：智能算法的数学理论基础相对薄弱；缺乏具备普遍意义的理论性分析；算法中涉及的各种参数设置一直没有确切的理论依据，通常都是按照经验型方法确定；对具体问题和应用环境的依赖性比较大。因此，智能算法的研究与应用将一直是非常具有实际意义的研究课题。

计算智能包含了许多与生物和智能有关的算法，通常我们称之为仿生算法。生物系统中“存在的必有其合理的地方”，这种信仰促使人们向生物系统学习，并进而创立了多种有效的算法。预计在未来会有更多的基于仿生算法的计算智能出现，比如基于病毒、寄生等生物机制的仿生计算智能会有所发展和应用。但是人类尚不满足于模仿生物进化行为，还希望能够建立起具有自然生命特征的人造生命和人造生命系统。对人工生命（artificial life，ALife）的研究，自 1987 年起取得了重要进展，这是计算智能的一个新的研究热点。

另外，混合智能计算将是必然趋势，神经网络和进化算法、免疫算法、DNA 计算的融合及大规模的并行智能计算的应用领域会越来越宽；同时，数学证明亦将结合具体的算法有目的地进行。模糊集、粗糙集、可扩展的聚类技术等方向的理论研究将会是近几年的热点。2008 年 IEEE 世界计算智能大会 WCCI2008（2008 IEEE World Congress on Computational Intelligence）在香港会议展览中心召开，日本大阪府立大学工学部的四位学者（Hidetomo Ichihashi，atsuhiro Honda，Akira Notsu，Eri Miyamoto）凭借《Fuzzy c-Means Classifier for High Dimensional Data》一文获得了大会最优秀论文奖。C 类模糊均值在各种数据集中的分类研究已经成为当前计算智能研究的热点和最前沿。

未来随着人类思维机制的破译，以及基因技术、生物技术、计算机技术、电子技术、通讯技术等学科的发展，将会出现更多更先进的计算智能的方法及装置。

参考文献

[1] 史忠植.知识发现.北京.清华大学出版社,2002
[2] 周志华,曹存根.神经网络及其应用.北京:清华大学出版社 2004
[3] 丁永生,任立红.人工免疫系统:原理与应用.模式识别与人工智能,2000,13(1)
[4] 谭营.机器学习研究及最新进展.北京大学智能科学系视觉与听觉信息处理国家重点实验室, 2005.12-1
[5] 鲁斌.智能混合系统研究进展.微型机与应用,2005,(7)
[6] 莫愿斌,刘贺同,王勤.智能优化算法的综述教学研究,科技创新导报 NO13，2008
[7] 田春光,姚锡凡,张毅. 计算智能及其在机械制造中的应用研究.组合机床与自动化加工技术,2003,(06)
[8] 鲍捷.计算智能综述.合肥工业大学图像信息处理实验室,2000
[9] Ying TAN.Artificial Immune System and Its Applications,National Laboratory on Machine Perception Department of Intelligence SciencePeking University, Beijing 100871, P.R.China, 2008.7.29
[10] T.Hastie, R.Tibshirani, J. Friedman. Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer-Verlag, 2001
[11] T. Mitchell. Machine Learning. McGraw Hill, 1997
[12] N. Cristianini, J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2002.7
[13] C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Knowledge Discovery and Data Mining, 1998, 2(2)
[14] Forrest S,Hofmeyr S A.Immunology as Information Processing.In:Segel L A and Cohen I R, Design Principles for Immune System & Other Distributed Autonomous Systems. Oxford Univ.Press,2000:361-387
[15] J H Holland.Genetic algorithems and classifier systems:foundations and their applications J.Proceedings of the Second International Conference on Genetic Algorithms，1987: 82-89
[16] Ishida YAdachi N.Active Noise Control by an Immune Algorithm: Adapation in Immune System as an EvolutionJ.Proc,ICEC 1996(96):150-153
[17] K.M.Passino, S.Yurkovich. Fuzzy Control. Prentice Hall/Pearson. 2001.11.J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2001
[18] C.M.Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995

后　记

《计算智能》这本教材由西安建筑科技大学管理学院卢才武教授主编，陕西科学技术出版社出版发行。卢才武教授在总结多年教学和科研工作的基础上，根据人工智能学科发展的前沿方向，以及硕士、博士研究生培养的要求编写了本书，为研究生教学以及相关领域的科研提供参考。

在本书编写过程中，除西安建筑科技大学管理学院唐晓灵、张志霞、顾清华等教师参与编写之外，西安建筑科技大学博士研究生郭梨、李慧、杨震、李发本等同学也参与了资料整理、校对与程序测试等工作。在此一并表示感谢！

编者

2008 年 9 月