第 5 章

群智能计算

在自然界中，有很多生物以群居的形式生活，并且表现出令人惊叹的群体智慧行为，如鸟群觅食、蚂蚁觅食、菌群觅食等，这些现象吸引着计算机科学领域的学者对这些群体行为展开研究。受这类生物群体行为的启发而提出的人工智能实现模式，称为群智能(Swarm Intelligence，SI)。群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。常见群智能算法有粒子群算法、蚁群算法、菌群算法、人工鱼群算法、狼群算法等，其中以粒子群算法和蚁群算法最具代表性，这些算法已被广泛应用于各个领域。

本章首先对群智能计算进行简要概述，之后重点介绍群智能两大代表性算法——粒子群算法和蚁群算法，让读者深入了解群智能算法原理及解决问题的整个过程，为读者学习更复杂的其他群智能算法做铺垫。本章还介绍了一种新兴的群智能算法——菌群算法的背景、原理和应用。最后简单介绍了人工鱼群算法和狼群算法的基本原理作为本章的附加知识，扩展读者对群智能领域的了解。本章作为读者了解群智能计算领域的参考内容，由于篇幅原因，内容有限，如果读者对群智能计算领域有兴趣，可以查阅更多参考资料。

5.1 群智能概述

本节简要介绍群智能的概念及相关内容，是本章后续内容的先导知识。本节从自然界的生物群体行为引出群智能的基本概念，并总结群智能的特点，为读者更好地理解群智能算法做铺垫。基于基本概念的理解，重点介绍群智能算法的基本思想，并简单介绍常见的群智能算法、与进化算法的异同以及群智能算法的应用。

5.1.1 群智能基本概念

自然界有很多以集群方式生存的生物，常见的集群有鸟群、蚁群、鱼群乃至人类等。这些集群中每个个体的行为比较简单，但是整个集群在整体上呈现出高度的组织性，体现出一些个体不

具备的智能行为。例如鸟群觅食，单只鸟的行为简单，但是鸟群在空中却能一致地朝一个方向飞，有时还能转向、分散和再聚集，最终整个鸟群快速地找到食物，到底是谁主宰着它们？到底是谁维持它们的秩序、预测它们的未来、维持它们的平衡？事实上，研究社会性昆虫的科学家发现昆虫在群体中的协作是高度自组织的，它们的协调行为可以通过个体之间的交互行为直接实现，或者通过个体与环境的交互行为间接实现[1]。

“群智能”是指由简单个体组成的群落与环境以及个体之间的互动行为[2]。在上述的鸟群中，单只鸟和鸟群的互动行为指导这只鸟接下来的飞行方向，因此从整体上看，鸟群能一致地朝一个方向飞行并快速找到食物。虽然个体与群落或环境的交互行为非常简单，但是它们聚在一起却能解决一些难题，这种潜在方式的集群智能为人类提供了解决难题的灵感和方法。“群智能”的概念最早由 G. Beni 和王静[3]在分子自动机系统中提出。群智能中的群，可被定义为“一组相互之间可以进行直接或间接通信的主体”。群的个体组织包括在结构上很简单的鸟群、蚁群、鱼群、蜂群等，而它们的集体行为却可能变得相当复杂。群智能在没有集中且不提供全局模型的前提下，为寻找复杂分布式问题解决方案提供了基础[4]。

群智能行为引起了广大学者们的注意，通过模拟群智能行为过程，学者发现群智能有以下几个特点。

(1) 群智能中的个体比较低智能，个体行为一般比较简单，个体只与局部个体进行信息交互，无法和全局进行信息交流，容易实现模拟个体的算法部分并且执行的时间复杂度也较小，同时，整个算法的实现对计算机的配置要求也不高。另外，该方法只需计算目标函数值，不需要梯度信息，容易实现。因此，当系统中的个体数量增加时，对于系统所增加的信息量比较小，使整个系统具有简单性。

(2) 在群智能系统中，相互协作的个体是分布式存在的，其初始分布可以是均匀或非均匀随机分布。这个系统是没有中心的，个体间完全自组织，从而体现出群体的智能特征。这个特点恰好适应网络环境，也符合大多数实际复杂问题的演变模式。

(3) 由于群智能系统中个体是分布式存在的，没有控制中心，整体的智慧是通过个体间以及个体与环境间的相互作用而体现出来的，所以单个个体对整体的影响比较小，整个系统也不会因为其中一个个体的因素而受到影响，所以群智能系统具有鲁棒性。

(4) 群体智能系统中的个体不仅可以进行相互之间的直接通信，还可以通过环境进行非直接通信，即个体之间通过所处的小环境作为媒介进行交互，具有自组织性。这样就使得整个系统具备良好的可扩展性。

(5) 群体智能算法对要解决的问题是否连续并无要求，这就使得该类算法既适合具有连续性的数值优化，也适合离散性的组合优化。在处理问题的规模上也没有要求，规模越大越能体现出群体智能算法的优越性[1]。

5.1.2 群智能算法基本思想

基于对群智能行为过程的模拟，对群智能特点的归纳总结，学者提出用群智能算法解决实际问题，运用自然界的智慧解决人类社会中的问题。群智能算法的基本思想是模拟自然界生物的群体行为来构造随机优化算法。它将搜索和优化过程模拟成个体的进化或觅食过程，用搜索空间中的点模拟自然界中的个体，将求解问题的目标函

数度量成个体对环境的适应能力，将个体的优胜劣汰过程或觅食过程类比为搜索和优化过程中用好的可行解取代较差可行解的迭代过程。因此，群智能算法是一种具有“生成＋检验”特征的迭代搜索算法。

目前群智能算法已包括三十余种算法，部分主流的群智能算法有粒子群优化(Particle Swarm Optimization，PSO)算法、蚁群优化(Ant Colony Optimization，ACO)算法、菌群算法、人工鱼群(Artificial Fish Swarm Algorithm，AFSA)算法、狼群算法、人工蜂群算法、灰狼优化算法、萤火虫算法、布谷鸟搜索算法、鸡群优化算法、混合蛙跳算法、狮子优化算法、猴群算法、雁群优化算法、蟑螂优化算法、捕食搜索算法、自由搜索算法、食物链算法、共生生物搜索算法等。

与第 4 章介绍的进化算法相比，群智能算法和进化算法的相同之处如下：

(1) 两者都是受自然现象的启发，基于抽取出的简单自然规则而发展出的计算模型；

(2) 两者都是基于种群的方法，且种群的个体之间、个体与环境之间存在相互作用；

(3) 两者都是元启发式随机搜索方法。

二者不同之处在于进化算法强调种群的达尔文主义的进化模型，而群智能优化方法则注重对群体中个体之间的相互作用与分布式协同的模拟[2]。

群智能算法的出现为求解复杂优化问题的最优解或近似最优解提供了高效的方法，它的高速发展对人们的生活产生了很大的影响，为电气、网络、医疗、图像识别、深度学习、生产调度、背包问题、分配问题以及车辆调度等复杂优化问题提供了新的解决方法。

5.2　PSO 算法

PSO 算法是群智能算法的代表性算法之一，读者如果能通过本节的学习深入了解 PSO 算法的过程，就可以加深对群智能及应用群智能思想解决实际问题的理解，也便于以后接受、理解和应用其他群智能算法。PSO 算法是基于群智能的随机优化算法，其基本思想是模拟自然界中鸟类觅食行为来迭代搜索找到全局最优解。本节将从 PSO 算法的生物学背景出发，介绍 PSO 算法的基本模型、求解优化问题的实例、PSO 算法的各种改进模型和应用。

5.2.1　PSO 算法背景

在鸟类觅食过程中，微观上的鸟类个体的行为是简单的，但宏观上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，并且能快速完成寻找食物这样的复杂行为？

为了回答上述问题，先简要介绍两个理论：人工生命(Artificial Life，AL)和复杂适应系统(Complex Adaptive System，CAS)理论。人工生命[5]是借助计算机以及其他非生物媒介，实现一个具有生物系统特征的过程或系统。这些可实现的生物系统具有的特征包括：繁殖、进化、信息交换和决策能力。人工生命主要是指属于计算机科学领域的虚拟生命系统，涉及计算机软件工程与人工智能技术。复杂适应系统理论[6]是 J. H. Holland 教授于 1994 年提出的。复杂适应系统理论包括微观和宏观两

个方面。在微观方面，复杂适应系统理论最基本的概念是具有适应能力的、主动的个体，简称主体。这种主体在与环境的交互作用中遵循一般的刺激——反应模型，所谓适应能力表现在它能够根据行为的效果修改自己的行为规则，以便更好地在客观环境中生存。在宏观方面，由这样的主体组成的系统，将在主体之间以及主体与环境的相互作用中发展，表现出宏观系统中的分化、涌现等种种复杂的演化过程。

为了理解鸟群行为的奥妙，1987 年，C. W. Reynold[7] 在人工生命和复杂适应系统理论的基础上提出 Boid 模型，用以模拟鸟类聚集飞行的行为。在这个模型中，每个个体只需遵循以下 3 条规则。

（1）避免碰撞：远离最近的邻居，避免和邻近的个体相互碰撞。

（2）向个体目标靠近：和邻近个体的平均速度保持一致。

（3）向群体中心群集：飞向群体的中心，向邻近个体的平均位置移动。

1990 年，在 C. W. Reynold 的鸟群复杂群体模型基础上，生物学家 F. Heppner 和 U. Grenander 通过加入鸟群受到栖息地的吸引的特点，进一步提出了鸟群聚集模型[8]。在该模型中，刚开始每只鸟都没有特定的飞行目标，只是使用简单的规则确定自己的飞行方向和飞行速度，当有一只鸟飞到栖息地附近时，它周围的鸟也会跟着飞向栖息地，最终整个鸟群都会落在栖息地。利用上面几条简单的规则，就可以非常接近地模拟出鸟群飞行的现象。

1995 年，美国社会心理学家 J. Kenney 和电气工程师 R. C. Eberhart 通过对鸟群觅食过程的分析和模拟，提出了 PSO 算法[9]，起初只是设想模拟鸟群觅食的过程，但后来发现 PSO 算法也是一种很好的优化工具。PSO 算法的基本思想源于对鸟类觅食过程中迁徙和聚集的模拟，通过鸟之间的集体协作和竞争达到目的。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，在事先不确定食物的方位时，这种信息的利用是至关重要的，这种信息分享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。这一点也是 PSO 算法得以建立的基本原理之一。

在寻找食物过程中，鸟类个体是如何利用其他成员曾经勘测和发现的关于食物位置的信息来引导自身的飞行方向的呢？设想这么一个场景：一群鸟进行觅食，而远处有一片地有最多的食物，所有的鸟都不知道这片地到底在哪里，但是它们知道自己当前的位置距离那里有多远。那么找到拥有最多食物所在地的最佳策略，也是最简单有效的策略就是通过不断和其他鸟进行交流，判断哪只鸟找到的区域中包含的食物量最多，具体如图 5.1 所示。

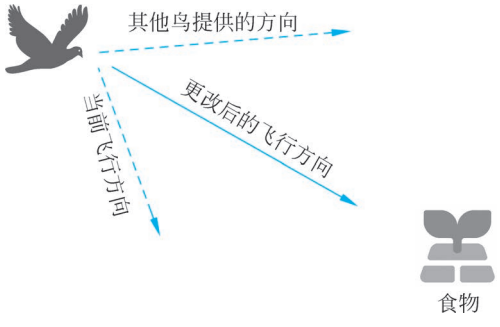


图 5.1 鸟群觅食过程

PSO 算法将鸟群运动模型中的栖息地类比于所求问题空间中可能解的位置，通过个体间的信息传递，导引整个群体向可能解的方向移动，在求解过程中逐步增加发现较好解的可能性。PSO 算法群体中的鸟被抽象为没有质量和体积的“粒子”，鸟群中寻找的食物最多的地方被抽象为“问题的最优解”，鸟群飞向的森林被抽象为“求解空间”，鸟类个体当前飞行到的区域内的食物量被抽象为“目标函数值”，每只鸟所处的位置被抽象为“空间中的一个解”，食物最多的位置被抽象为“全局最优解”。所有的粒子都有一个位置向量（粒子在解空间的位置）和一个速度向量（决定下次飞行的方向和速度），并可以根据目标函数计算当前所在位置的适应值，可以将其理解为每只鸟当前飞行到的区域内的食物量。通过这些“粒子”间的相互协作和信息共享，使其运动速度受到自身和群体的历史运动状态信息的影响。以自身和群体的历史最优位置对粒子当前的运动方向和运动速度加以影响，较好地协调粒子本身和群体之间的关系，以利于群体在复杂的解空间中进行寻优操作。

由于 PSO 算法概念简单，实现容易，短短几年时间，便获得了很大的发展，取得了丰硕的成果，并在许多领域得到应用。现已被国际遗传与演化计算（World Genetic and Evolution Computing，WGEC）会议列为主要的讨论专题之一。

5.2.2　用于连续优化问题的 PSO 算法模型

PSO 算法采用速度-位置搜索模型，一只鸟称为一个“粒子”，每个粒子代表解空间的一个候选解，解的优劣程度由适应度函数决定，适应度函数由优化目标定义；每个粒子还有一个速度决定它们飞行的方向和距离；每个粒子通过动态跟踪两个极值来更新其速度和位置，第一个是粒子本身从初始到当前迭代搜索产生的最优解，第二个是粒子种群目前的最优解。

在介绍具体的速度和位置更新方式前，先给出几个重要的符号表示。假设在 D 维搜索空间中，有 N 个粒子，其中第 i 个粒子的位置向量表示为 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{iD})$，它的飞行速度表示为 $\boldsymbol{v}_i=(v_{i1},v_{i2},\cdots,v_{iD})$，这个粒子搜索到的最优位置为 $\boldsymbol{P}_i=(p_{i1},p_{i2},\cdots,p_{iD})$，对整个粒子群来说，其搜索到的最优位置记为 $\boldsymbol{P}_{\text{gbest}}=(\boldsymbol{p}_{\text{gbest1}},\boldsymbol{p}_{\text{gbest2}},\cdots,\boldsymbol{p}_{\text{gbest}D})$。

1. 基本 PSO 算法

基本 PSO 算法中的第 i 个粒子在第 $k+1$ 代的速度由第 i 个粒子在第 k 代的速度和位置、第 i 个粒子历史最优位置以及整个粒子群的历史最优位置决定。第 i 个粒子在第 d 维上的速度更新公式为

$$v_{id}^{k+1}=v_{id}^{k}+c_1\cdot\text{rand}_1(p_{id}-x_{id}^{k})+c_2\cdot\text{rand}_2(p_{\text{gbestd}}-x_{id}^{k}) \tag{5.1}$$

其中，c_1 和 c_2 是两个正常数，称为加速因子，它们取均匀分布于（0，1）内的随机数。当第 $k+1$ 代的速度确定好后，其位置可根据上一时刻的位置和该时刻的速度决定，位置更新公式为

$$x_{id}^{k+1}=x_{id}^{k}+v_{id}^{k+1},\quad d=1,2,\cdots,D \tag{5.2}$$

在 PSO 算法的迭代过程中，每个粒子总是先更新自身的速度，然后再进行位置更新。式（5.1）中，速度的更新主要受到三部分因素影响。

（1）自身速度——v_{id}^{k}，由粒子自身的速度构成，表示粒子对自身运动状态的信任。

（2）认知部分——$c_1\text{rand}_1(p_{id}-x_{id}^{k})$，这里 p_{id} 表示第 i 个粒子的历史最优位

置，认知部分表示对粒子本身的思考，即来源于自己经验的部分。

(3) 社会部分——$c_2\text{rand}_2(p_{\text{gbestd}}-x_{id}^k)$，这里 p_{gbestd} 表示整个粒子群搜索到的历史最优位置，社会部分代表粒子间的信息共享，来源于群体中的其他优秀粒子的经验。如果 $c_1=0$，则粒子没有认知能力，在粒子的相互作用下，虽然能到达新的搜索空间，但是也容易陷入局部极值点；如果 $c_2=0$，粒子间没有社会信息共享，其算法变成一个多起点的随机搜索；如果 $c_1=c_2=0$，粒子将一直以当前的速度飞行，直到到达边界。通常 c_1、c_2 取值范围为[0,4]，一般取 $c_1=c_2=2$。需要注意的是粒子在进行速度更新和位置更新时，是针对每个粒子的每维变量分别进行更新，式(5.1)和式(5.2)中的 d 表示第 d 维。

2. 标准 PSO 算法

基本 PSO 算法的提出引起了学者的广泛研究，鉴于基本 PSO 算法的不足，研究人员在该算法的基础上提出了很多改进方法。

为了使粒子保持运动惯性，使其有扩展搜索空间的趋势，有能力探索新的区域，史玉回与 R. C. Eberhart[10] 提出了一种改进算法，即在式(5.1)的 v_{id}^k 前乘以一个惯性权重 w，通过调整惯性权重的大小平衡算法全局搜索和局部搜索之间的矛盾。改进后的算法称为“标准 PSO 算法”，其速度更新公式如式(5.3)所示，位置更新公式同式(5.2)。

$$v_{id}^{k+1}=w\cdot v_{id}^k+c_1\cdot\text{rand}_1(p_{id}-x_{id}^k)+c_2\cdot\text{rand}_2(p_{\text{gbestd}}-x_{id}^k) \tag{5.3}$$

其中，速度的更新主要受到三部分因素影响，分别是：惯性部分——$w\cdot v_{id}^k$，认知部分——$c_1\text{rand}_1(p_{id}-x_{id}^k)$，社会部分——$c_2\text{rand}_2(p_{\text{gbestd}}-x_{id}^k)$，其示意如图 5.2 所示。由此可以得出每个粒子在飞行中，受到自身惯性、个体经验以及群体最优粒子经验的影响，不断变换飞行速度，从而使整个粒子群能够向着搜索空间中的最佳位置去飞行，并有望搜索到最优解。

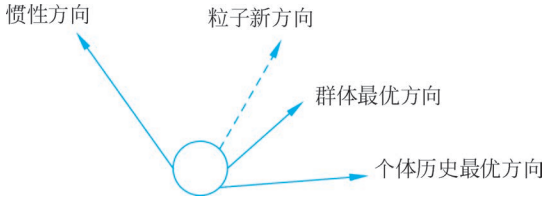


图 5.2 粒子更新示意图

由于标准 PSO 算法改进了基本 PSO 算法易陷入局部最优的不足，它的性能也大大得以提升，在此后常常将标准 PSO 算法简称为 PSO 算法，本书中接下来的章节也是以标准 PSO 算法为例进行介绍。以上过程中介绍了 PSO 算法的核心部分——个体更新公式。对计算机而言，以求最小值为例，PSO 算法的实现过程如算法 5.1 所示。PSO 算法的流程图如图 5.3 所示。

算法 5.1：PSO 算法

Step 1：设置参数(粒子数量 N，速度范围 $V_{\max}$，惯性权重 w，加速因子 c_1 和 c_2)并初始化粒子群；

Step 2：计算每个粒子的适应度值，将其适应度值与其经过的最好位置 P_{best} 进行比较，如果较好，则将其作为该粒子当前的最好位置 P_i；否则，输出当前全局最优粒子；

Step 3：根据各个粒子的历史最优位置 P_i 找出群体历史最优位置 P_{gbest}；

Step 4：按式(5.3)更新每个粒子的速度，按式(5.2)更新当前位置；

Step 5：判断是否满足终止条件。若满足，输出当前历史最优位置，否则返回 Step 2。

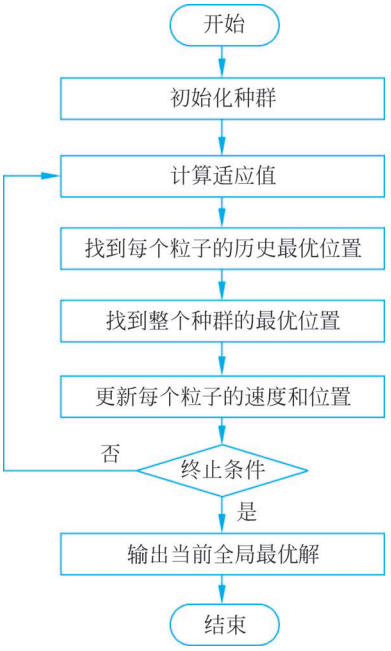


图 5.3　PSO 算法流程图

5.2.3　PSO 算法求解实例

PSO 算法起初只是设想模拟鸟群觅食的过程，研究并解释复杂的社会行为，但后来发现该算法是一种很好的优化工具，因此，PSO 算法也常被应用于求解优化问题。

1. 求解连续优化实例

以求解无约束最小化优化问题 $\min f(x)=x_1^2+2x_2^2$ 为例，为了观察基本 PSO 算法与标准 PSO 算法的区别，这里给出了粒子初始化及两种 PSO 算法的前两代的搜索过程(以种群规模为 3 为例进行说明)。

首先，初始化种群中各粒子的位置及速度，如表 5.1 所示。

表 5.1　初始化种群中各粒子的位置和速度

初始化粒子编号	位　　置	速　　度
S_1	(8,−5)	(3,2)
S_2	(−5,9)	(−2,−3)
S_3	(−7,−8)	(5,3)

对于基本 PSO 算法，设置参数 $c_1=2$，$c_2=2$(基本 PSO 算法无参数 w)。在第一代搜索过程中，初始粒子的适应度值分别为 114、187、177。由于该问题是最小化问题，当前种群中全局历史最优位置即为粒子 S_1 所在的位置(8，−5)，当前各个粒子的自身历史最优位置即为初始位置，分别是(8，−5)，(−5，9)，(−7，−8)。根据式(5.3)和式(5.2)中的更新公式，计算可得这 3 个粒子更新后的速度和位置，具体见表 5.2。

表 5.2 基本 PSO 算法第一代搜索时粒子的更新过程

粒子编号	适应度值	p_{best}	p_i	更新后的速度	更新后的位置
S_1	114	(8,−5)	(8,−5)	(3,2) ($r_1=0.5, r_2=0.5$)	(11,−3)
S_2	187		(−5,9)	(13.6,−19.8) ($r_1=0.5, r_2=0.6$)	(8.6,−10.8)
S_3	177		(−7,−8)	(14,4.8) ($r_1=0.1, r_2=0.3$)	(7,−3.2)

对于标准 PSO 算法，设置参数 $w=0.7, c_1=2, c_2=2$。同基本 PSO 算法一样，初始粒子的适应度值分别为 114、187、177，当前种群中全局历史最优位置为粒子 S_1 所在的位置(8,−5)，当前各个粒子的自身历史最优位置即为初始位置。根据式(5.3)和式(5.2)的更新公式，计算可得这 3 个粒子更新后的速度和位置，具体见表 5.3。

表 5.3 标准 PSO 算法第一代搜索时粒子的更新过程

粒子编号	适应度值	p_{best}	p_i	更新后的速度	更新后的位置
S_1	114	(8,−5)	(8,−5)	(2.1,1.4) ($r_1=0.5, r_2=0.5$)	(10.1,−3.6)
S_2	187		(−5,9)	(14.2,−18.9) ($r_1=0.5, r_2=0.6$)	(9.2,−9.9)
S_3	177		(−7,−8)	(12.5,3.9) ($r_1=0.1, r_2=0.3$)	(5.5,−4.1)

接下来进行第二代搜索，与第一代搜索过程类似，在基本 PSO 算法中，计算当前各粒子的适应度值为 139、307.24、69.48。在最小化问题中，此时全局历史最优位置为粒子 S_3 所在的位置(7,−3.2)。再计算各粒子的自身历史最优位置，会发现只有 S_3 的适应度值变小了，由 177 变为 69.48，因此其历史最优位置更新为第二代的位置(7,−3.2)，而 S_1 和 S_2 的历史最优位置仍为第一代的位置(8,−5) 和(−5,9)。接下来根据式(5.3)和式(5.2)分别计算每个粒子更新后的速度和位置，计算结果见表 5.4。

表 5.4 基本 PSO 算法第二代搜索时粒子的更新过程

粒子编号	适应度值	p_{best}	p_i	更新后的速度	更新后的位置
S_1	139	(7,−3.2)	(8,−5)	(−1,0.32) ($r_1=0.4, r_2=0.2$)	(10,−2.68)
S_2	307.24		(−5,9)	(8.96,−6.72) ($r_1=0.1, r_2=0.6$)	(17.56,−17.52)
S_3	69.48		(7,−3.2)	(14,4.8) ($r_1=0.5, r_2=0.7$)	(21,1.6)

在标准 PSO 算法的第二代搜索中，粒子的适应度值为 127.93,280.66,63.87，粒子 S_3 所在位置为全局最优位置，且粒子 S_3 当前位置取代了上一代的位置，成为自身和全局最优位置，其他粒子的适应度值也都下降了，因此自身没有位置需要更新，结果见表 5.5。

表 5.5　标准 PSO 算法第二代搜索时粒子的更新过程

粒子编号	适应度值	p_{best}	p_i	更新后的速度	更新后的位置
S_1	127.93	(5.5,−4.1)	(8,−5)	(−1.21,0.22) (r_1=0.4,r_2=0.2)	(8.89,−3.38)
S_2	280.66		(−5,9)	(4.08,−6.29) (r_1=0.1,r_2=0.6)	(13.28,−16.19)
S_3	63.87		(5.5,−4.1)	(8.75,2.73) (r_1=0.5,r_2=0.7)	(14.25,−1.37)

按照这样的过程，不断迭代直至达到规定的最大运行代数，算法终止。可以看到，相比于基本 PSO 算法，标准 PSO 算法的粒子适应度值下降得更快，即更靠近最小值，收敛速度更快。关于 w、c_1 和 c_2 对算法性能的影响，下面具体说明。

2. 参数分析

PSO 算法中需要调节的参数有：粒子数量 N、速度范围 V_{max}、惯性权重 w、加速因子 c_1 和 c_2。其中，w、c_1 和 c_2 对算法性能的影响较大，目前有很多学者对其设定和调节方式进行了研究。绝大多数文献研究的参数区域[11]是：$w\in[-1,1]$，$c_1+c_2\in[0,8]$。

1）惯性权重

惯性权重 w 使粒子保持运动惯性，使其有搜索扩展空间的趋势，有能力探索新的区域。惯性权重也表示粒子对当前自身运动状态的信任，依据自身的速度进行惯性运动。惯性权重大的粒子更趋向于探索未知的空间，保证算法的探索能力；惯性权重小的粒子更趋向于跟随种群最优方向，保证算法的收敛能力。

在优化实际优化问题时，往往希望先采用全局搜索，使搜索空间快速收敛于某一区域，然后采用局部精细搜索以获得高精度的解。因此学者提出了惯性权重 w 自适应调整的策略，已有的调整策略有：线性调整、模糊调整、非线性权重递减策略、随机调整和自适应动量因子等，下面简要介绍前两种调整策略。

线性调整策略也称为惯性权重线性递减算法，即随着迭代的进行，线性地减小 w 的值，w 随迭代次数增加线性递减的公式为

$$w = w_{max} - \frac{w_{max} - w_{min}}{k_{max}} \times k \tag{5.4}$$

其中，w_{max} 和 w_{min} 分别是 w 的最大值和最小值；k 和 k_{max} 分别是当前迭代次数和最大迭代次数。该策略是目前粒子群算法中调整惯性权重的常用方法。

模糊调整策略：现实问题的解空间搜索过程是非线性且高度复杂的，惯性权重线性递减的策略往往不能反映实际的优化搜索过程。例如，对于目标跟踪问题，就需要优化算法拥有非线性搜索的能力以适应动态环境的变化。

因此，有学者使用模糊推理机预测合适的惯性权重，动态地平衡全局和局部搜索能力，但是其参数比较多，增加了算法的复杂度，使得其实现较为困难，因此该策略并不常用。

2）加速因子

加速因子 c_1、c_2 代表将粒子推向个体最优位置和全局最优位置的加速项的权重，表示粒子的动作来源于自己经验的部分和其他粒子经验的部分。加速因子过低会使得粒子在目标区域外徘徊，而过高会导致粒子越过目标区域。目前，通常将 c_1、c_2 统

一为一个控制参数 φ，$\varphi=c_1+c_2$。如果 φ 很小，粒子群运动轨迹将非常缓慢；如果 φ 很大，则粒子位置变化非常快。根据实验结果可以获得 φ 的经验值，当 $\varphi=4.0(c_1=2.0, c_2=2.0)$时，粒子群算法具有很好的收敛效果。

3）粒子规模

从经验上看，粒子数量 N 通常取 20～40，对较难的问题可以取 100～200。

4）最大速度

最大速度 $V_{\max}$ 决定当前位置与最好位置之间的区域分辨率(或精度)。如果 $V_{\max}$ 太高，粒子可能会越过好的解；如果 $V_{\max}$ 太小，粒子容易陷入局部优值。$V_{\max}$ 决定了粒子在一个循环中的最大移动距离，通常设定为粒子每维变化范围的 10%～20%[2]。

总之，种群的搜索能力和算法的性能是依靠 w、c_1 和 c_2 的相互配合来调节的，并不是仅仅靠一个参数可以决定的。

3. 求解组合优化实例

旅行商问题(Travelling Salesman Problem，TSP)是运筹学、图论和组合优化中的经典 NP 难题，常被用来验证智能启发式算法的有效性。旅行商问题描述如下：给定 n 个城市及两两城市之间的距离，求一条经过各城市一次且仅一次再回到原出发城市的最短路线。其图论描述为：给定图 $G=(V,E)$，其中 V 为顶点集，E 为各顶点相互连接组成的弧集，已知各顶点间连接距离要求确定一条长度最短的 Hamilton 回路，即遍历所有顶点一次且仅一次的最短回路。设 d_{ij} 为城市 i 与 j 之间的距离，即弧 (i,j)的长度。引入决策变量

$$x_{ij}=\begin{cases}1, & \text{旅行商访问城市 } i \text{ 后访问城市 } j\\ 0, & \text{其他}\end{cases} \tag{5.5}$$

TSP 的目标函数为

$$\min Z=\sum_{i,j=1}^{n} x_{ij}d_{ij} \tag{5.6}$$

TSP 的问题描述非常简单，但最优化求解很困难，若用穷举法搜索，则要考虑所有可能情况，并两两对比，找出最优，其算法复杂性呈指数增长，即所谓的“组合爆炸”。所以，寻求有效的启发式算法是研究 TSP 问题的关键。

PSO 算法的速度公式难以表达诸如 TSP 等离散域问题。下面通过引入交换子和交换序的概念[12]对 PSO 算法进行改造，并将其应用于求解 TSP 中。

(1) **交换子**。设 n 维 TSP 的解序列为 $S=(a_i)$，$i=1,2,\cdots,n$。定义交换子 $SO(i_1,i_2)$为交换 S 中的点 a_{i_1} 和 a_{i_2}，则 $S'=S+SO(i_1,i_2)$为解 S 经算子 $SO(i_1,i_2)$操作后的新解。这里，“+”被赋予了新的含义，例如，$S'=S+SO(2,3)=(14523)+SO(2,3)=(15423)$。

(2) **交换序**。一个或多个交换子的有序队列为一个交换序，记为 $SS=(SO_1,SO_2,\cdots,SO_n)$，其中 $SO_1,SO_2,\cdots,SO_n$ 是交换子，它们之间的顺序是有意义的。交换序作用于一个 TSP 解上，意味着此交换序中的所有交换子依次作用于该解上，即 $S'=S+SS=S+(SO_1,SO_2,\cdots,SO_n)=[(S+SO_1)+SO_2]+\cdots+SO_n$。

(3) **合并算子**。若干个交换序可以合并为一个新的交换序，⊕定义为两个交换序的合并算子。

（4）**交换序的等价集**。不同的交换序作用在同一解上可能产生相同的新解，所有具有相同效果的交换序的集合称为交换序的等价集。

（5）**基本交换序**。在交换序等价集中，拥有最少交换子的交换序称为该等价集的基本交换序。

设给定两个解路径 A(2　3　1　4　5)和 B(1　2　3　4　5)，欲构造一个基本交换序 SS，使得 $B+\mathrm{SS}=A$。可以看出，$A(1)=B(2)$，故第一个交换子是 SO(1,2)，$B_1=B+\mathrm{SO}(1,2)$，得到 B_1(2　1　3　4　5)。此时 $A(2)=B(3)$，故第二个交换子是 SO(2,3)，$B_2=B_1+\mathrm{SO}(2,3)$得到 B_2(2　3　1　4　5)。这样就得到一个基本交换序 $\mathrm{SS}=A-B=(\mathrm{SO}(1,2),\mathrm{SO}(2,3))$。

引入交换子和交换序等概念后，重新构造 PSO 算法的速度更新公式，如式(5.7)所示，位置更新公式同式(5.2)。

$$v_{id}^{k+1}=wv_{id}^{k}\oplus\alpha(p_{id}-x_{id}^{k})\oplus\beta(p_{\mathrm{gbest}d}-x_{id}^{k}) \tag{5.7}$$

其中，$\alpha,\beta\in[0,1]$为随机数，$\alpha(p_{id}-x_{id}^{k})$表示基本交换序$(p_{id}-x_{id}^{k})$中所有交换子以概率 α 保留，同理，$\beta(p_{\mathrm{gbest}d}-x_{id}^{k})$表示基本交换序$(p_{\mathrm{gbest}d}-x_{id}^{k})$中的所有交换子以概率 β 保留。α、β 分别表示个体极值和全局极值对粒子的影响程度。可以看出，α 的值越大，$(p_{id}-x_{id}^{k})$保留的交换子越多，p_{id} 的影响越大；β 的值越大，$(p_{\mathrm{gbest}d}-x_{id}^{k})$保留的交换子越多，$p_{\mathrm{gbest}d}$ 的影响就越大。

以上过程中介绍了 PSO 算法求解 TSP 的核心部分——个体更新公式。对计算机而言，PSO 算法求解 TSP 的实现过程如算法 5.2 所示。

算法 5.2：PSO 算法求解 TSP

Step 1：设置参数(粒子数量 N，惯性权重 w)并初始化粒子群，即给群体中的每个粒子赋一个随机的初始解和一个随机的交换序；

Step 2：计算每个粒子的适应度值，将其适应度值与其经过的最好位置 P_{best} 进行比较，如果较好，则将其作为该粒子当前的最好位置 P_i；否则，输出当前全局最优粒子；

Step 3：根据各个粒子的历史最优位置 P_i 找出群体历史最优位置 P_{gbest}；

Step 4：按式(5.7)更新每个粒子的速度；按式(5.2)更新当前的位置；

(1) 计算 p_{id} 和 x_{id}^{k} 的差 A，$A=p_{id}-x_{id}^{k}$，其中 A 是一个基本交换序，表示 A 作用于 x_{id}^{k} 得到 p_{id}；

(2) 计算 $B=p_{\mathrm{gbest}d}-x_{id}^{k}$，其中 B 也是一个基本交换序，表示 B 作用于 x_{id}^{k} 得到 $p_{\mathrm{gbest}d}$；

(3) 根据式(5.7)计算新的粒子速度 v_{id}^{k+1}，并将交换序 v_{id}^{k+1} 转换为一个基本交换序；

Step 5：判断是否满足终止条件。若满足，输出当前历史最优位置，否则返回 Step 2。

对于一个 10 点 TSP 问题，其 TSP 描述见表 5.6，城市分布如图 5.4 所示。

表 5.6　10 点 TSP 问题

序号	1	2	3	4	5
X	0.0000	5.2100	4.8900	6.2400	6.7900
Y	0.0000	8.8500	7.9600	0.9900	2.6200
序号	**6**	**7**	**8**	**9**	**10**
X	3.9600	3.6700	9.8800	0.3800	2.3200
Y	3.3500	6.8000	1.3700	7.2100	9.1300

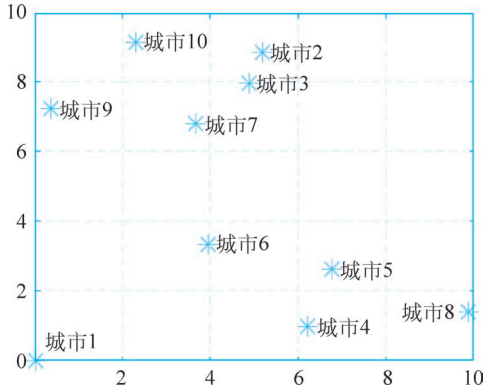


图 5.4 10 个城市分布图

(1) 粒子数量为 80,迭代次数为 2000,最短距离为 35.18,算法结果见图 5.5 和图 5.6。

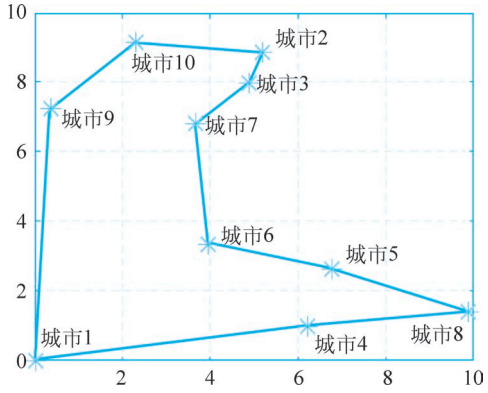


图 5.5 10 城市 TSP 问题最优路径(一)

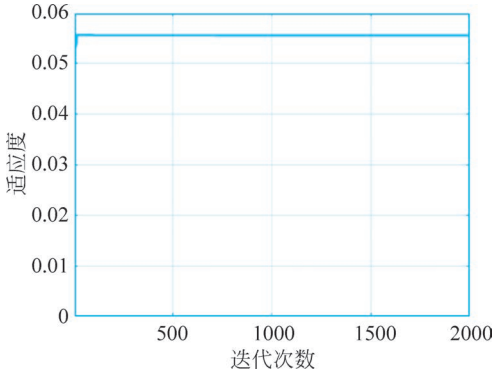


图 5.6 适应度曲线(一)

(2) 粒子数量为 200,迭代次数为 100,最短距离为 35.98,算法结果见图 5.7 和图 5.8。

从实验结果可以看出,PSO 算法同遗传算法一样,都是不稳定的,每次运行的结果都会不一致。粒子数量的选取对问题的求解有很大的影响。

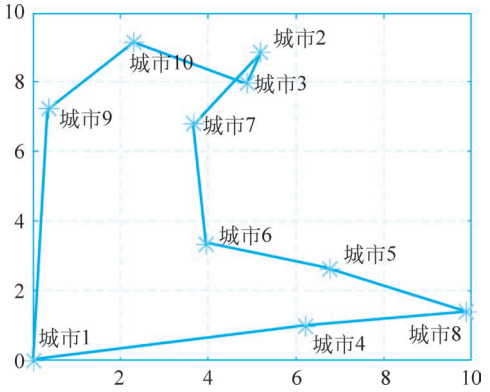


图 5.7　10 城市 TSP 问题最优路径(二)

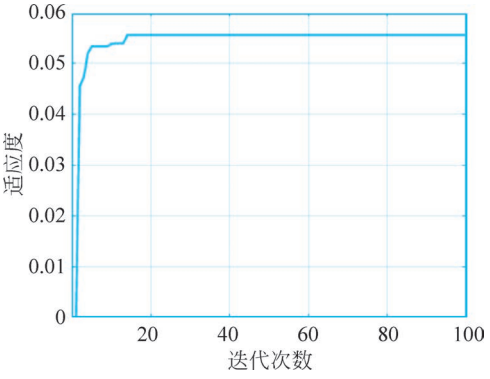


图 5.8　适应度曲线(二)

5.2.4　PSO 算法改进模型

PSO 算法提出至今虽然只有短短约二十年的历史,但研究人员对其进行了大量的研究,并给出了各种改进形式。本节介绍其中几种主流的改进方法。

1. 混合 PSO 算法模型

P. J. Angeline[13] 于 1998 年提出基于进化计算中的选择操作的改进型 PSO 算法模型,称为混合 PSO 算法。该混合 PSO 算法模型将每次迭代产生的新的粒子群根据适应度函数进行选择,用适应度较高的一半粒子的位置和速度向量取代适应度较低的一半粒子的相应向量,而保持后者个体极值不变。这样的 PSO 算法模型在提高收敛速度的同时保证了一定的全局搜索能力,在大多数的 Benchmark 函数的优化上取得较基础 PSO 算法模型更好的优化结果。

M. Lovbjerg、T. K. Rasmussen 和 T. Krink[14] 于 2000 年提出将进化算法中的交叉操作引入 PSO 算法的混合 PSO 算法模型。交叉机制首先以一定的交叉概率从所有粒子中选择待交叉的粒子,然后两两随机组合进行交叉操作产生后代粒子。交叉操作使后代粒子继承了双亲粒子的优点,在理论上加强了对粒子间区域的搜索能力。例如,两个双亲粒子均处于不同的局部最优区域,那么两者交叉产生的后代粒子往往能够摆脱局部最优,而获得改进的搜索结果。实验证明,与传统的 PSO 算法及传统的遗传算法比较,交叉型 PSO 算法搜索速度快,收敛精度高。目前,利用进化操作改进传

统 PSO 算法的探索仍在继续。

在 PSO 算法搜索的后期，将进化算法中的变异操作引入 PSO 算法形成了另一种混合 PSO 算法模型。因为在 PSO 算法搜索的后期，粒子群会向局部最优或全局最优收敛，此时，每个粒子的历史最优、所有粒子的历史最优解中每个粒子的当前位置都会趋向于同一点，而每个粒子的运动速度趋于零。在这种情况下，粒子群所趋向的那个点，即为粒子群算法的最终求解结果的极限值。如果此时得到的最优解不是理论最优解或者期望最优解，则粒子群陷入局部最优，算法将出现早熟收敛。为了提高解的质量，当粒子群收敛到一定程度时，就要进行变异。解的质量的优劣依赖于变异概率的大小，过大和过小的变异概率都不利于求得较高质量的解。当变异概率过大时，频繁的变异对解的质量的提高是不利的，前一次变异所带来的求得更优解的机会还没来得及被充分利用，又发生新的变异，新变异终止了正在进行的寻优过程，扰乱了粒子群收敛的趋势；当变异概率较小时，如此少的变异对解的质量的改善是有限的，无法达到变异的目的。因此，适当的变异概率可以获得较高的成功率与较快的收敛速度。

2. 考虑拓扑结构的 PSO 算法模型

标准 PSO 算法的网络结构是一个单群落的全连接型网络，标准 PSO 算法搜索高维复杂优化问题全局最优值的过程往往会出现因为某几维陷入局部最优而导致整个算法陷入局部最优。不同邻域拓扑结构的 PSO 算法对同一个优化问题效果有很大的差异，常见的拓扑结构有全连接形拓扑、环形拓扑、冯·诺依曼拓扑、星形拓扑和金字塔结构拓扑等，不同的邻域拓扑是对不同交流模式的模拟，选择合适的拓扑结构对解决优化问题有很大的影响[15]。常见拓扑结构中，全连接形拓扑信息传递的速度和算法收敛的速度较快，适合解决低维简单问题，但在求解高维问题时却很容易陷入局部最优。环形拓扑将所有的粒子首尾相连，相邻粒子可以保证更充分的信息交换，从而使信息在粒子中得到更好的分享，使算法充分搜索每个区域，不至于过早陷入局部最优，但也使得搜索速度变慢，收敛速度急剧下降。在冯·诺依曼拓扑中，粒子形成立体网状结构，加强了多个区域的搜索粒子之间的联系，使粒子可以更好地避开局部最优值。

3. 扩展 PSO 算法模型

PSO 算法的基础是粒子群中信息的社会共享，它通过粒子群中的个体最优位置和全局最优位置达到信息共享。然而，在 PSO 化算法的速度更新公式中，除了全局最优位置向其他粒子发布信息外，没有充分利用其他个体最优位置的信息，以至于粒子的多样性降低，易过早地陷入局部极值点。扩展 PSO 算法[16]利用粒子群中的所有个体最优位置的信息，使得其信息共享更加充分，提高了粒子的多样性。

除了这些改进的 PSO 算法外，还有带免疫性质的 PSO 算法[17]，它是将抗体多样性和免疫记忆特性引入 PSO 算法中，提高算法的全局搜索能力；混沌 PSO 算法[18]是利用混沌变量的随机性、遍历性及规律性，将最优粒子进行混沌寻优，再把混沌寻优的结果随机替换粒子群中的一个粒子，提高算法的收敛速度和精度。这些改进后的 PSO 算法各有千秋，它们的利弊及更多不同的改进方式还有待进一步的探索。

另外，差分进化[19](Differential Evolution，DE)也是一种基于种群的随机优化算法，在种群进化阶段通过对个体依次使用差分变异、交叉、选择等进化操作，使种群一代一代地进化，直到算法运行结束。与遗传算法相比，差分进化算法采用实数编码而

不是遗传算法的二进制编码，差分进化算法的核心操作是基于差分的简单变异，而遗传算法的核心操作是交叉操作，差分进化算法是父代自身的进化，而遗传算法是父代产生新子代来进化，与遗传算法的劣者概率淘汰不同的是，差分进化算法采用劣者绝对淘汰策略，总体上看，差分进化算法降低了遗传操作的复杂性。差分进化算法和粒子群算法都是智能优化算法，目的都是用来寻找最优适应度值，都是父代自身进化而不涉及子代，但相比 PSO 算法，差分进化算法依靠的是种群的不断变异来更新父代个体，而 PSO 算法是通过种群最优和粒子历史最优来引导粒子的搜索方向来更新父代个体，也就是说，两者的进化核心操作不同。

4. 粒子轨迹与收敛行为分析

本节首先提出了描述粒子轨迹的显式公式，从中对一个粒子的轨迹进行分析，进而给出保证粒子轨迹收敛的参数 c_1、c_2 和 w 的取值[20-23]。

本节中用到的术语“收敛”指的是粒子存在以下极限属性：

$$\lim_{k\to\infty} X(k) = p \tag{5.8}$$

其中，p 是搜索空间的任一位置；$X(k)$是粒子在第 k 代的位置。

为方便描述，在此重复标准 PSO 算法中惯性权重的速度和位置更新公式

$$v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \cdot \mathrm{rand}_1(p_{id} - x_{id}(k)) + c_2 \cdot \mathrm{rand}_2(p_{\mathrm{gbestd}} - x_{id}(k)) \tag{5.9}$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1), \quad d = 1,2,\cdots,D \tag{5.10}$$

式(5.9)给出的是速度更新公式的隐式形式，描述 PSO 算法在多维搜索空间中的多个粒子。为了简化描述，下面的分析限定在一维空间，因此省略下标 d。由于 PSO 算法中不同维度间没有相互影响，所以这样做也不失一般性。单独考察一个粒子的轨迹时，符号可以进一步简化，下标 i 也可以省略。这一简化假定在分析一个粒子的轨迹时其他粒子在空间上保持静止。一个粒子的轨迹将在离散时间步上被分析，因此用 x_k 表示 $x(k)$的值。

现在，把式(5.9)代入式(5.10)，可得到以下非齐次递归关系：

$$x_{k+1} = (1 + w - \phi_1 - \phi_2)x_k - wx_{k-1} + \phi_1 p + \phi_2 p_{\mathrm{gbest}} \tag{5.11}$$

其中，$\phi_1 = c_1\mathrm{rand}_1$，$\phi_2 = c_2\mathrm{rand}_2$、$\phi_1$、$\phi_2$ 和 w 为常数。ϕ_1 和 ϕ_2 分别是 $c_1\mathrm{rand}_1$ 和 $c_2\mathrm{rand}_2$ 的特例。

指定初始条件 $x(0) = x_0$ 以及 $x(1) = x_1$ 后，可用任何一种求解非齐次递归关系的方法得到式(5.11)的封闭形式。封闭形式的公式为

$$x_k = k_1 + k_2\alpha^k + k_3\beta^k \tag{5.12}$$

其中

$$k_1 = \frac{\phi_1 p + \phi_2 p_{\mathrm{gbest}}}{\phi_1 + \phi_2} \tag{5.13}$$

$$\gamma = \sqrt{(1 + w - \phi_1 - \phi_2)^2 - 4w} \tag{5.14}$$

$$\alpha = \frac{1 + w - \phi_1 - \phi_2 + \gamma}{2} \tag{5.15}$$

$$\beta = \frac{1 + w - \phi_1 - \phi_2 - \gamma}{2} \tag{5.16}$$

$$x_2 = (1 + w - \phi_1 - \phi_2)x_1 - wx_0 + \phi_1 p + \phi_2 p_{\text{gbest}} \tag{5.17}$$

$$k_2 = \frac{\beta(x_0 - x_1) - x_1 + x_2}{\gamma(\alpha - 1)} \tag{5.18}$$

$$k_3 = \frac{\alpha(x_1 - x_0) + x_1 - x_2}{\gamma(\beta - 1)} \tag{5.19}$$

注意以上公式假定 p 和 p_{gbest} 在第 k 代上保持不变。真正的 PSO 算法允许 p 和 p_{gbest} 的值改变。因此上述更新公式的封闭形式在发现更好的位置 x(及相应的 p 和 p_{gbest})前是有效的,发现更好的位置时重新计算 k_1、k_2 和 k_3 的值后又可以使用以上公式。发生这种情况的精确时间步视目标函数以及 p 和 p_{gbest} 的值而定。为了将结论推广,不如认定 p 和 p_{gbest} 为常数,由此推出 k_1、k_2 和 k_3 也是常数。虽然这种情况不会在实际的 PSO 算法中出现,但它有助于解释 PSO 算法的收敛特征。

粒子行为的一个重要方面是它的轨迹(由 x_k 确定)是收敛还是发散,即确定序列 $\{x_k\}_{k=0}^{+\infty}$ 在何种条件下可以收敛?

在 ϕ_1 和 ϕ_2 为常数值时,可以简单地对粒子轨迹的收敛性进行分析。但需要注意的是,真正的 PSO 算法中这些参数使用的是伪随机数,而不是常数。然而,正如后面将要说明的,系统的行为通常由这些值的上限确定。因此,通过使用 ϕ_1 和 ϕ_2 的最大可能取值,可以对最坏情况下系统的收敛行为进行分析。

假定 p、p_{gbest}、ϕ_1、ϕ_2、k_1、k_2 和 k_3 保持常值,则式(5.12)能用于计算一个粒子的轨迹。序列 $\{x_k\}_{k=0}^{+\infty}$ 的收敛性取决于由式(5.15)及式(5.16)计算出的 α 和 β 的大小。从式(5.14)中可以看到,如果 $(1+w-\phi_1-\phi_2)^2 < 4w$ 或者等价于以下成立条件:

$$(\phi_1 + \phi_2 - w - 2\sqrt{w} - 1)(\phi_1 + \phi_2 - w + 2\sqrt{w} - 1) < 0$$

γ 将是一个虚部非零的复数。

由复数 γ 得出的 α 和 β 也是虚部不为零的复数。α 和 β 的大小用向量的 L_2 范数衡量。对于一个复数 z,其 L_2 范数为

$$\| z \| = \sqrt{(\text{Re}(z))^2 + (\text{Im}(z))^2} \tag{5.20}$$

任意复数 z^k 可写为

$$\begin{aligned} z^k &= (\| z \| e^{i\theta})^k \\ &= \| z \|^k e^{i\theta k} \\ &= \| z \|^k (\cos(\theta k) + i\sin(\theta k)) \end{aligned} \tag{5.21}$$

其中 $\theta = \arg(z)$。只有当 $\| z \| < 1$ 时,极限 $\lim\limits_{k\to\infty} z^k = \lim\limits_{k\to\infty} \| z \|^k (\cos(\theta k) + i\sin(\theta k))$ 存在,极限值为 0。

现在考虑极限中 x_k 值,根据式(5.12)有

$$\lim_{k\to\infty} x_k = \lim_{k\to\infty} (k_1 + k_2\alpha^k + k_3\beta^k) \tag{5.22}$$

显然,式(5.22)间接描述了一个粒子的轨迹,只要 $\max(\| \alpha \|, \| \beta \|) > 1$,$\{x_k\}_{k=0}^{+\infty}$ 将发散,因为这种情况下极限不存在。相反,当 $\max(\| \alpha \|, \| \beta \|) < 1$ 时,$\{x_k\}_{k=0}^{+\infty}$ 就收敛,即有

$$\lim_{k\to\infty} x^k = \lim_{k\to\infty} (k_1 + k_2\alpha^k + k_3\beta^k) = k_1 \tag{5.23}$$

由于当 $\|\alpha\|<1$ 时 $\lim\limits_{k\to\infty}\alpha^k=0$，当 $\|\beta\|<1$ 时 $\lim\limits_{k\to\infty}\beta^k=0$。用 z 表示 α 或 β，如果 $\|z\|=1$，极限 $\lim\limits_{k\to\infty}z^k=\lim\limits_{k\to\infty}1^k(\cos(\theta k)+\mathrm{i}\sin(\theta k))$ 不存在，因此序列 $\{x_k\}_{k=0}^{+\infty}$ 发散。

注意，以上计算假定 ϕ_1 和 ϕ_2 值保持为常数，一般的 PSO 算法中并不如此。但 c_1 和 c_2 的值可认为是 ϕ_1 和 ϕ_2 的上限。以 ϕ_1 和 ϕ_2 的期望值考虑，可观察到系统的一般行为。假定 ϕ_1 和 ϕ_2 为均匀分布，即

$$E[\phi_1]=c_1\int_0^1\frac{x}{1-0}\mathrm{d}x=c_1\left.\frac{x}{2}\right|_0^1=\frac{c_1}{2}$$

$$E[\phi_2]=c_2\int_0^1\frac{x}{1-0}\mathrm{d}x=c_2\left.\frac{x}{2}\right|_0^1=\frac{c_2}{2}\tag{5.24}$$

假定选择的 ϕ_1、ϕ_2 和 w 使得 $\max(\|\alpha\|,\|\beta\|)<1$，即选择它们的值使得序列 $\{x_k\}_{k=0}^{+\infty}$ 收敛。从式(5.13)和式(5.23)可得

$$\lim_{k\to\infty}x_k=k_1=\frac{\phi_1 p+\phi_2 p_{\mathrm{gbest}}}{\phi_1+\phi_2}\tag{5.25}$$

将式(5.24)计算得到的 ϕ_1 和 ϕ_2 的期望值代入式(5.25)，可得

$$\begin{aligned}\lim_{k\to\infty}x_k&=\frac{\frac{c_1}{2}p+\frac{c_2}{2}p_{\mathrm{gbest}}}{\frac{c_1}{2}+\frac{c_2}{2}}\\&=\frac{c_1p+c_2p_{\mathrm{gbest}}}{c_1+c_2}\end{aligned}\tag{5.26}$$

粒子的轨迹收敛于 p 与 p_{gbest} 的加权平均值。

例如，如果 $c_1=c_2$，则

$$\lim_{k\to\infty}x_k=\frac{p+p_{\mathrm{gbest}}}{2}\tag{5.27}$$

从任意的 c_1 和 c_2 的取值中可以获得更一般的解

$$\begin{aligned}\lim_{k\to\infty}x_k&=\frac{c_1p+c_2p_{\mathrm{gbest}}}{c_1+c_2}\\&=\frac{c_1}{c_1+c_2}p+\frac{c_2}{c_1+c_2}p_{\mathrm{gbest}}\\&=\left(1-\frac{c_2}{c_1+c_2}\right)p+\frac{c_2}{c_1+c_2}p_{\mathrm{gbest}}\\&=(1-\alpha)p+\alpha p_{\mathrm{gbest}}\end{aligned}\tag{5.28}$$

其中 $\alpha=c_2/(c_1+c_2)$，因此 $\alpha\in[0,1]$。式(5.28)显示，粒子收敛于个体最优与全局最优的某个线性组合点。这一结果让人感到满意，因此它表明粒子将搜索位于它的个体最优与全局最优之间的具有更好解的区域。进一步分析可得到，当 c_1、c_2 和 w 满足以下关系时，能得到收敛的轨迹：

$$w>\frac{1}{2}(c_1+c_2)-1\tag{5.29}$$

由式(5.28)可知，如果粒子的轨迹收敛，则它朝自身的最优位置和全局最好粒子

的位置之间连线的某一个位置收敛。根据更新公式，粒子的个体最优位置将逐渐朝全局最优位置靠近，因此，粒子最终将收敛到全局最优粒子所在的位置。在这个点，粒子将停止运动，因此算法无法继续改善解的质量。但这和算法是否能真正发现函数 f 的最小值没有关系，事实上，甚至无法保证粒子最终收敛的位置是一个局部最小点。这一结论多少有点让人感到沮丧，因此有学者提出了保证收敛的 PSO 算法[24]，感兴趣的读者可以阅读相关文献，这里不做赘述。

5.2.5 PSO 算法应用

PSO 算法发展至今，其应用已非常广泛，主要有以下几个方面。

1. 神经网络训练

立方体机器人可等效为多变量、高阶、非线性、不稳定的多自由度空间动量轮倒立摆系统。为了解决立方体机器人系统非线性的特点和模型精度不高的问题，陈阳等[25]利用粒子群优化的模糊神经网络算法，通过对立方体机器人机体的输入/输出数据进行学习，建立立方体机器人的粒子群模糊神经网络模型，再引入粒子群实现模糊神经网络参数的优化调整，最后选用 PID 控制器进行稳定性控制。目前，采用 PSO 算法训练神经网络的方法正在更多的领域推广应用。

2. 电力系统

随着人们对风电的日益关注，新型风电场的容量在并网系统中所占比例不断增加，这对传统电力系统的经济调度问题提出了新的要求。特别是风电场输出功率的随机变化给系统的经济调度带来了更多的不确定性因素。陈海焱等[26]利用下降搜索思想对传统 PSO 算法进行了改进，以提高粒子的收敛速度和收敛精度，并将改进后的算法用于求解提出的动态经济调度问题，较好地解决了由于风电场输出功率难以准确预测而给传统机组经济调度带来的困难。

3. 机器人领域

李擎等[27]提出了一种基于保收敛 PSO 算法的移动机器人全局路径规划策略，为移动机器人在有限时间内找到一条避开障碍物的最短路径提供了一种解决方案。首先建立环境地图模型，将连接地图中起点和终点的路径编码成粒子，然后根据障碍物位置规划出粒子的可活动区域，在此区域内产生初始种群，使粒子在受限的区域内寻找最优路径。在搜索过程中，粒子群算法的加速系数和惯性权重均随迭代次数自适应调节，与其他文献所提的方法相比，该算法具有更快的搜索速度和更高的搜索质量。

事实上，PSO 算法的应用领域非常广泛，不胜枚举。它已经在多目标优化、自动目标检测、生物信号识别、决策调度、系统辨识以及游戏训练、分类、调度、信号处理、决策、机器人应用等方面都取得了一定的成果。在模糊控制器设计、车间作业调度、机器人实时路径规划、自动目标检测、语音识别、烧伤诊断、探测移动目标、时频分析和图像分割等方面也已经有成功应用的先例。

5.3 蚁群算法

作为群智能算法的另一代表性算法，蚁群算法相比粒子群算法有其不可替代的特点。蚁群算法是一种广泛应用于组合优化问题的启发式搜索算法。它的基本思想是

模拟自然界中蚂蚁的觅食行为，用信息素和正反馈机制更快地寻求最优解。通过本节的学习，读者可以深入了解模拟蚁群觅食行为解决实际问题的方法和过程，加深对群智能及应用群智能思想解决组合优化问题的理解，本节将从蚁群算法的生物学背景出发，介绍蚁群算法的基本模型、求解优化问题的实例、改进算法及应用。

5.3.1　蚁群算法背景

蚂蚁的食物源总是随机散布于蚁巢周围，按照人类的逻辑来说，蚂蚁找到食物的概率也应该具有一定的随机性，但实际上，经过一段时间后，蚂蚁总能找到一条从蚁巢到食物源的最短路径，这样的结果让人们好奇蚂蚁到底是如何做到的。其实单只蚂蚁的能力和智力非常简单，但它们通过相互协调、分工、合作来指挥完成筑巢、觅食、迁徙、清扫蚁穴等复杂行为。比如，蚂蚁在觅食过程中能够通过相互协作找到食物源和蚁巢之间的最短路径。自然界中的蚂蚁会分泌一种称为“信息素”的化学物质来与其他同伴通信。它们在觅食过程中，总存在信息素跟踪和信息素遗留两种行为，信息素跟踪行为使它们会沿着信息素浓度较高的路径行走，信息素遗留行为使每只路过的蚂蚁都会在路上留下信息素。这就形成一种类似正反馈的机制，经过一段时间后，整个蚁群就会沿着最短路径到达食物源了。

从上述信息中可以发现蚁群觅食的关键是信息素和正反馈机制。信息素是蚂蚁释放的一种易挥发物质，能够实现蚁群间的通信。蚂蚁在寻找食物时，会在其经过的路上释放信息素，而信息素浓度可以影响其他蚂蚁的路径选择，并且信息素浓度越高，对应的路径越短，蚂蚁选择该最短路径的概率越大。正反馈机制是指蚂蚁会以较大的概率选择信息素浓度较高的路径，并释放一定量的信息素，从而加强距离较短路径的信息素浓度，后来的蚂蚁大概率重复这样的过程，使较短的路径上累积的信息素浓度越来越高，选择该路径的蚂蚁个数也愈来愈多，这样的过程称为一个正反馈机制。

为了便于理解蚂蚁觅食的正反馈机制、信息素浓度越高的路径对应的路径长度越短的机理，图 5.9 给出了蚂蚁觅食过程。假设蚂蚁从蚁巢到食物的路线无障碍物时为直线，由于有障碍物阻挡，因此从蚁巢到食物之间有两条路径了，即 l_1 和 l_2，并且 l_2 比 l_1 长。初始状态时，两条路径上的信息素浓度相同。由于两条路径上的初始信息素浓度相同，一开始蚂蚁选择路径时，具有一定的随机性，蚁群中有些蚂蚁选择了 l_1 路径，有些蚂蚁选择了 l_2 路径，但是由于 l_1 路径的长度比较短，选择 l_1 路径的蚂蚁先行到达了食物附近，而此时选择 l_2 路径的蚂蚁还在路径上，未到达食物附近。

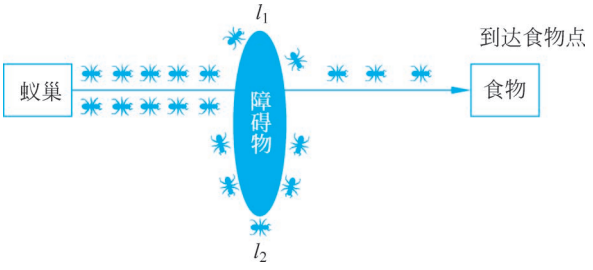


图 5.9　蚂蚁觅食第一阶段

在图 5.10 中，当选择 l_2 路径的蚂蚁到达食物点时，选择 l_1 路径的蚂蚁已经开始返回了，它在往返 l_1 路径的过程中，已经在 l_1 路径上留下了它的信息素，加强了路径

l_1 上的信息素。

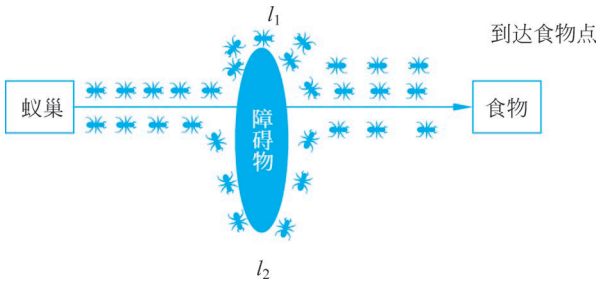


图 5.10 蚂蚁觅食第二阶段

在图 5.11 中，当选择 l_2 路径的蚂蚁返回时，l_1 路径上已经有新的蚂蚁已经开始进行下一次的搜索，此时 l_1 路径上的信息素已经大于 l_2 上的信息素，所以其他蚂蚁在进行选择时，就会更倾向于选择路径 l_1。最终，所有蚂蚁都会选择信息素浓度较大的 l_1 路径来搬运食物。

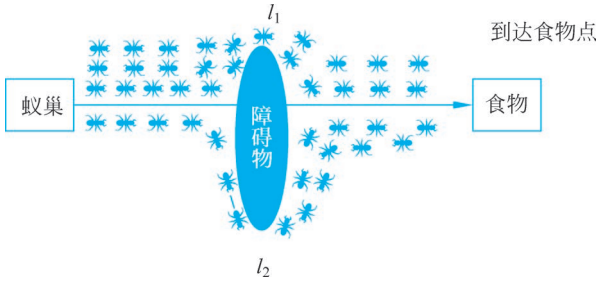


图 5.11 蚂蚁觅食第三阶段

下面从理论和实验验证两方面解释“信息素浓度越高的路径对应的路径长度越短”这一机理。理论上讲，假设有两条路可从蚁巢通向食物，开始时两条路上的蚂蚁数量差不多：当蚂蚁到达终点之后会立即返回，距离短的路上的蚂蚁往返一次时间短，重复频率快，在单位时间里往返蚂蚁的数目就多，留下的信息素也多，会吸引更多蚂蚁过来并留下更多信息素。而距离长的路正相反，因此越来越多的蚂蚁聚集到最短路径上来。从实验验证上讲，J. L. Deneubourg 及其同事在对阿根廷蚂蚁进行实验时，建造了一座有两个分支的桥，其中一个分支的长度是另一个分支的两倍，同时把蚁巢同食物源分隔开来。实验发现，蚂蚁通常在几分钟之内就选择了较短的那条分支。这个实验对蚁群算法的设计有至关重要的影响。

至此，自然界蚂蚁觅食行为的介绍已经结束，接下来介绍如何将蚁群觅食行为抽象为算法以解决优化问题的方法。

首先要了解蚂蚁的行为规则，具体如下。

（1）感知范围：蚂蚁能够察觉前方小范围区域内的状况，并判断出是否有食物或其他同类的信息素轨迹。

（2）环境信息：蚂蚁所在环境中有障碍物、其他蚂蚁、信息素，其中信息素包括食物信息素（找到食物的蚂蚁留下的）、蚁巢信息素（找到蚁巢的蚂蚁留下的），信息素以一定速率消失。

（3）觅食规则：蚂蚁在感知范围内寻找食物，如果感知到就会过去；否则朝信息

素多的地方走。每只蚂蚁会以小概率犯错误，并非都往信息素最多的方向移动。蚂蚁找蚁巢的规则类似，仅对蚁巢信息素有反应。

（4）移动规则：蚂蚁朝信息素最多的方向移动，当周围没有信息素指引时，会按照原来运动方向惯性移动。而且蚂蚁会记住最近走过的点，防止原地转圈。

（5）避障规则：当蚂蚁待移动的方向有障碍物时，将随机选择其他方向；当有信息素指引时，蚂蚁将按照觅食规则移动。

（6）散发信息素规则：在刚找到食物或者蚁巢时，蚂蚁散发的信息素最多；随着蚂蚁走远，散发的信息素将逐渐减少。正是这些简单的行为规则使蚁群具有智能行为，让蚁群具有多样性和正反馈[28]。

ACO 算法是计算机科学家 M. Dorigo[29] 及其同事于 1991 年提出的一种智能优化算法，是一种模拟蚂蚁群体智能行为的随机优化算法。其基本思想是试图模仿蚂蚁在觅食活动中找到最短路径的过程，通过它们释放信息素在所经过的路径上，就可以方便其他的蚂蚁利用和分享有用信息。同时，由于较短路径积累的信息素快、浓度值高，所有的蚂蚁向信息素浓度高的节点方向移动的概率比向浓度低的节点方向移动的概率大，这样经过一段时间后，利用整个群体的自组织，就能够发现最短的路径。

将 ACO 算法应用于解决优化问题的基本思路为：将蚂蚁觅食的觅食空间抽象为“问题的搜索空间”，将信息素抽象为一个“信息素浓度变量”，将蚁巢到食物的一条路径抽象为“一个有效解”，将蚁群搜索到的所有路径抽象为“搜索空间的一组有效解”，将蚁群找到的最短路径抽象为“问题的最优解”。初始时刻，蚂蚁随机选择路径，随着时间的推进，较短的路径上累积的信息素浓度逐渐增高，选择该路径的蚂蚁数也愈来愈多。最终，整个蚂蚁群会在正反馈的作用下集中到最佳的路径上，此时对应的便是待优化问题的最优解。

从上面的介绍可知，ACO 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，它具有较强的鲁棒性、优良的分布式计算机制、易于与其他方法相结合等优点。蚁群算法的正反馈机制会迅速地扩大初始解的差异，引导整个系统向最优解的方向进化，但是如果算法开始得到的较优解为次优解，那么正反馈会使次优解很快占据优势，使算法陷入局部最优，且难以跳出局部最优。

5.3.2　用于离散优化问题的蚁群算法模型

从真实蚂蚁觅食原理的多个角度出发，蚁群算法的基本模型才一步步塑造出抽象的人工蚂蚁[30]。人工蚂蚁的核心是路径构造和信息素更新。以旅行商问题为背景，人工蚂蚁的路径构建过程可描述如下。

1. 信息素

首先，每只真实蚂蚁在所经过的路径上都会留下信息素，人工蚂蚁如何表示这种留下来的这种物质呢？人工蚂蚁类似地用信息素增量表示该蚂蚁所经路径释放出的信息素（即给所经路径带来了增加量）：$\Delta\tau_{ij}^{k}$ 表示第 k 只蚂蚁留在路径(i,j)上的信息素增量。

接下来，应该考虑，每只蚂蚁留下的具体的信息素量到底是多少呢？必须采用数字信息来记录，才能让计算机看懂和识别。首先介绍一种性能较好的定义公式，称为蚁周（ant-cycle）模型：

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{第 } k \text{ 只蚂蚁经过路径}(i,j)\\ 0, & \text{其他}\end{cases} \tag{5.30}$$

其中，Q 是总信息量，表示蚂蚁在所走路径上释放信息素的恒定总量，通常设定为一常量，从出发点开始，每只蚂蚁身上都有相同的信息素总量；L_k 表示第 k 只蚂蚁走过的路径总长度。蚁周模型用比较简单的路径长度的倒数估量相应边的信息素增量，蚂蚁走的路径越长(表示解越差)，则释放到该路径中任何一条边的信息素越少，即信息素浓度和该边所在路径的优良度成正比，这样就会致使长路径上的所有边在路径选择中对蚂蚁的吸引力减小。

最后，知道了每只蚂蚁在所经路径上留下的信息素量，就可以迅速知道一条边上总的信息素增量为

$$\Delta\tau_{ij}=\sum_{k=1}^{N}\Delta\tau_{ij}^{k} \tag{5.31}$$

其中，N 是进行一次循环的蚂蚁数，每条边上的总信息素增量为 N 只蚂蚁的累加和。

2. 信息素更新公式

式(5.31)给出的信息素增量公式仅模仿了真实蚂蚁总是在所经路径上持续不断释放信息素的一方面，另一方面，真实环境中的信息素浓度随着时间的推移不停挥发，这一方面并没有考虑进来。为此，人工蚂蚁类似也需要使路径上积累的信息素随着时间逐渐蒸发，即在计算机经过一个时间单位之后执行一次挥发操作，不过这里的单位时间定义为所有蚂蚁完成一次循环后的离散时间点，虽与真实环境的连续挥发时间稍有不同，却便于计算机处理。

把信息素的挥发也考虑进去后，在每次迭代中，新的信息素释放之前，对原有的信息素进行一定量的挥发，对于每条边上的信息素有

$$\tau_{ij}(t)=(1-\rho)\cdot\tau_{ij}(t) \tag{5.32}$$

下一时刻的信息素总量为经挥发后残留的信息素总量加上此时增加的总信息素增量，即

$$\tau_{ij}=(1-\rho)\cdot\tau_{ij}+\Delta\tau_{ij} \tag{5.33}$$

其中，ρ 表示信息素挥发系数，$\rho\in[0,1]$，则$(1-\rho)$表示信息素残留因子，初始时刻 $\tau_{ij}=0$。

3. 原始状态转移概率公式

前面定义了蚂蚁在所走路径留下的信息素值大小，可是对于某只具体的蚂蚁该如何形成一条自己的路径呢？即蚂蚁是以什么方式在不同的节点间移动呢？

首先，由于真实蚂蚁从某一城市到下一城市的移动是以一定的概率选择移动方向的，选择的原则就是信息素浓度越大，选择该城市的概率越大。注意只是概率比较大，并不是一定会选择浓度较大的路径。类似地，人工蚂蚁移动到某一城市的概率为

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}}{\sum\limits_{x\in\text{allowed}_k}\tau_{ix}}, & j\in\text{allowed}_k\\ 0, & \text{其他}\end{cases} \tag{5.34}$$

其中，p_{ij}^{k} 表示从城市 i 决定选择下一城市 j 的概率大小，$\text{allowed}_k=\{N-\text{tabu}_k\}$表示蚂蚁 k 下一步允许访问的城市，tabu_k 表示当前蚂蚁已经访问过的城市，要禁止再次

访问，tabu_k 随着进化过程需做动态调整。

其次，上面只是给出了从某城市移动到其他城市的概率大小，但实际情况最终必须给出一个确定且唯一的选择，到底最终会选择哪个城市来加入该只蚂蚁的行走路径中呢？比较常用的是轮盘赌选择法(Roulette Wheel Selection，RWS)。通过该方法可以很好地与前面真实蚂蚁做出的选择方式相吻合。虽然它们每次具体会选择哪一个作为下一个城市的结果是随机的，每次可能不一样，但是要想有更多的机会被选中就得依靠其转移概率值的大小，即

$$j = \text{RWS}\{p_{ij}^k\} \tag{5.35}$$

4. 启发信息

式(5.34)和式(5.35)已经给出了蚂蚁如何产生一条路径以及对应路径该留下多少的信息素，整个抽象过程完美地模仿了真实蚁群觅食的行为特征，但是这种仿真系统存在一个缺陷：算法模型达到稳定状态需要耗费较长的时间。要想系统不再受该缺陷的影响还需要在决定蚂蚁行走方向的状态转移概率中引入一个快速搜索的过程，如引入启发信息。启发信息会依据所求问题解空间的整体特性，以一定权重在蚂蚁决策时给出快速的指引，这个指导信息的存在极大地改善了蚁群算法耗时的收敛速度：

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{5.36}$$

其中，d_{ij} 为城市 i 和城市 j 之间的距离。式(5.36)表示越短的路径对蚂蚁的启发越大，即蚂蚁更可能朝较短的路径行走，这与最终的优化目标完全一致，所以给了蚂蚁一个很好的引导作用。引入启发信息后的状态转移概率公式变为

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum\limits_{x \in \text{allowed}_{ix}} [\tau_{ix}]^\alpha \cdot [\eta_{ix}]^\beta}, & j \in \text{allowed}_k \\ 0, & \text{其他} \end{cases} \tag{5.37}$$

其中，α 为信息素因子；β 为启发信息因子/能见度。式(5.37)出现后，式(5.34)所示的原始状态转移概率公式就被淘汰了。α 和 β 是两个预先设置的参数，用来控制信息素浓度与启发式信息的重要程度。当 $\alpha=0$ 时，信息素这一项不起作用，算法演变成随机贪心算法，即距离城市 i 最近的城市被选中的概率最大。当 $\beta=0$ 时，启发式信息这一项不起作用，蚂蚁完全只根据信息素浓度确定路径，算法将快速收敛，这样构建出的路径往往与实际最优路径有较大的差异，算法的性能不够好。

5. 其他版本模型

通过按照前面的思路进行建模，到此已经形成了一个简单蚁群模型。其实，M. Dorigo 最初提出的基本蚁群算法模型有 3 种版本，除了蚁周模型，另外两种模型分别是蚁量(ant-quantity)模型和蚁密(ant-density)模型。三者之间的差别只在于信息素计算策略/方式的不同，即 $\Delta\tau_{ij}^k$ 求法的不同，其他公式都一样。

Ant-Quantity 模型的定义为

$$\Delta\tau_{ij}^k = \begin{cases} Q, & \text{第 } k \text{ 只蚂蚁从城市 } i \text{ 移动到城市 } j \\ 0, & \text{其他} \end{cases} \tag{5.38}$$

Ant-Density 模型的定义为

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{d_{ij}}, & \text{第 } k \text{ 只蚂蚁从城市 } i \text{ 移动到城市 } j\\ 0, & \text{其他}\end{cases} \tag{5.39}$$

通过研究者多次实验和理论分析发现，蚁周模型比蚁量模型和蚁密模型有更好的性能[31]。这是因为蚁周模型利用全局信息（蚂蚁走完所有城市后根据总的路径 L_k 长短）更新路径上的信息素量，而且每个蚂蚁所释放的信息素被表达为反映相应行程质量的函数。而蚁密模型使用局部信息，蚂蚁完成一步后根据当前所走一小段路径的长短 d_{ij} 更新信息素（蚂蚁在两个城市间每移动一次后即更新信息素），最差的蚁量模型则统一释放相同大小的信息素。

在以上知识的基础上，蚁群算法的原理如算法 5.3 所示，蚁群算法流程图如图 5.12 所示。

算法 5.3：蚁群算法

Step 1：初始化参数，在初始时刻，设各城市连接路径的信息素浓度具有相同的值，N 只蚂蚁放到 n 座城市；

Step 2：每只蚂蚁根据路径上的信息素和启发式信息，独立地访问下一座城市。根据式(5.8)确定从当前城市访问下一城市的概率；

Step 3：采用轮盘赌选择法的方式，选择下一座城市。更新每只蚂蚁的禁忌表，直至所有蚂蚁遍历所有城市 1 次；

Step 4：根据式(5.9)更新每条路径上的信息素；

Step 5：若满足结束条件，即达到最大循环次数，则循环结束并输出程序计算结果，否则清空禁忌表并跳转到 Step 2。

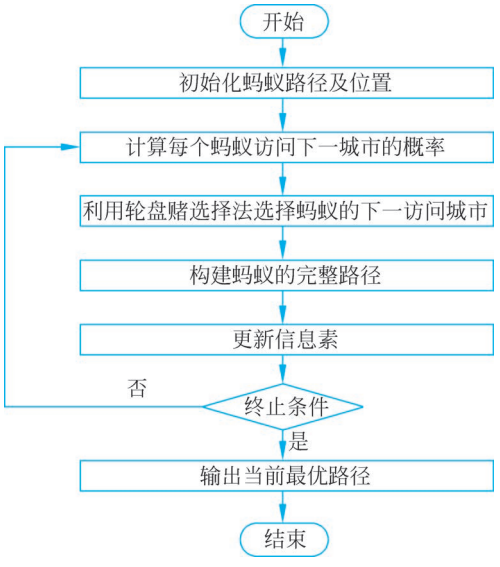


图 5.12 蚁群算法流程图

5.3.3 蚁群算法求解实例

以求解具有 10 城市的旅行商问题（即 10 点 TSP 问题）为例说明蚁群算法的具体求解过程，这 10 个城市的坐标见表 5.7，10 城市的位置分布见图 5.13。

表 5.7　10 点 TSP 问题

序号	1	2	3	4	5
X	0.0000	8.1500	9.0600	1.2700	9.1300
Y	0.0000	9.6500	1.5800	9.7100	9.5700
序号	**6**	**7**	**8**	**9**	**10**
X	6.3200	0.9800	2.7800	5.4700	9.5800
Y	4.5800	8.0000	1.4200	4.2200	9.1600

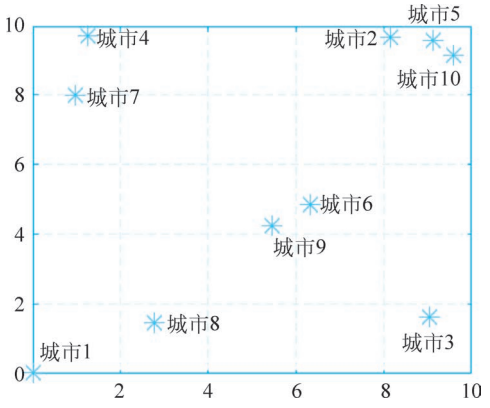


图 5.13　10 城市位置分布图

（1）蚁群规模设置为 $m=6$，迭代次数为 5000，控制信息素比例的参数设置为 $\alpha=1$，控制启发式信息比例的参数设置为 $\beta=2$，挥发系数为 $\rho=0.6$。最短距离为 39.42。结果见图 5.14 和图 5.15。

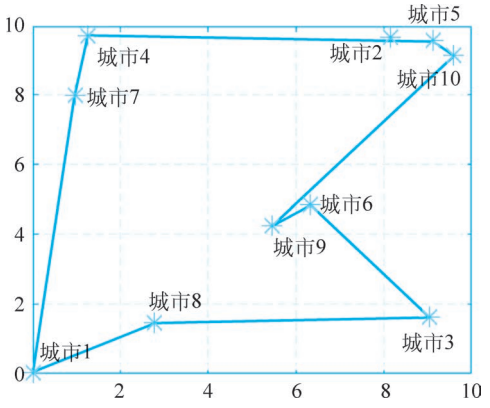


图 5.14　10 城市 TSP 问题最优路径（一）

（2）蚁群规模设置为 $m=10$，迭代次数为 5000，控制信息素比例的参数设置为 $\alpha=1$，控制启发式信息比例的参数设置为 $\beta=3$，挥发系数为 $\rho=0.6$。最短距离为 39.11。结果见图 5.16 和图 5.17。

在蚁群算法中，α 表示残留信息的挥发系数，$1-\alpha$ 表示残留信息的保留系数，α 直接影响算法的全局搜索能力及收敛速度。为了防止残留信息的无限积累同时为了使残留信息能够得到一定的保持，α 取值范围限制为[0,1]，如果 α 过大，会使先前访问过的城市再次被访问的机会增大，对全局搜索能力产生很大的影响，α 过小，虽然会使

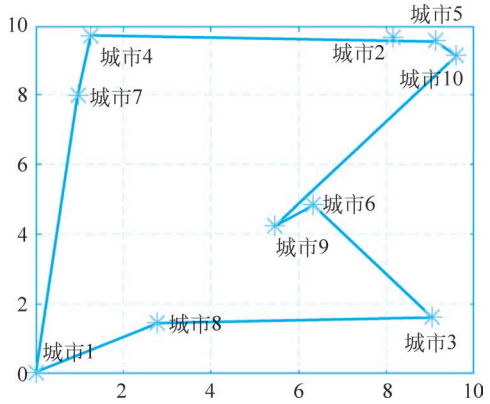


图 5.15 10 城市间信息素(一)

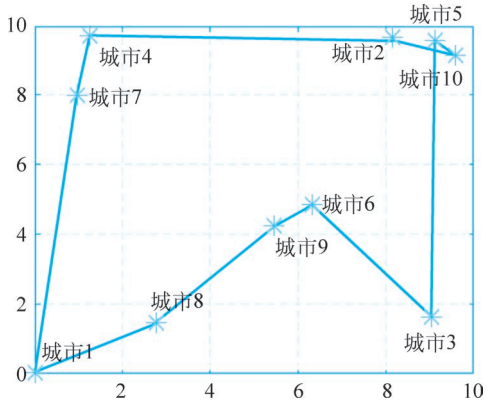


图 5.16 10 城市 TSP 问题最优路径(二)

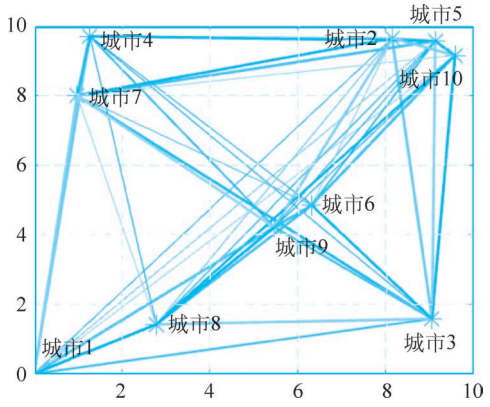


图 5.17 10 城市间信息素(二)

算法有较好的随机性以及全局搜索能力,但对算法的收敛速度会产生影响。β 表示启发信息受重视的程度,β 取值越大表示蚂蚁选择距离它近的城市的可能性越大。为了使启发式信息能够在蚁群算法中起到相应的作用,β 必须大于 0。如果 $\beta=0$,表示算法只使用了信息素,而没有利用启发信息带来的偏向性,这使得蚂蚁很快陷入停滞状态,达到局部最优,而不是全局最优,算法性能将大大降低。ρ 是信息挥发系数,挥发系数的大小反映了蚁群内蚂蚁相互之间的影响程度,ρ 使得刚刚被访问的路径上的信

息素减少，从而使得该路径被其他蚂蚁选择的概率减少，蚂蚁有更大的概率搜索其他的路径，因此算法不易陷入停滞状态。

5.3.4　蚁群算法改进模型

虽然蚁群算法思想在启发式方法范畴内已经形成为一个独立分支，但现阶段对蚁群算法的研究还只是停留在仿真阶段，远未像遗传算法、模拟退火等算法那样形成系统的方法，尚未能提出一个完善的理论分析，且由于蚁群中多个个体的运动是随机的，当群体规模较大时，要找出一条较好的路径需较长的搜索时间等。为此，很多学者提出了改进的模型以提高算法的性能和功效，下面介绍几种比较典型的改进模型。

1. 蚁群系统

蚁群系统(Ant Colony System，ACS)是 M. Dorigo 等[32]于 1996 年为改善原有蚁群算法性能而提出的。该系统的突出特点就是将蚁群算法和一种增强型学习算法(Q-learning)有机地结合在一块。所以，蚁群系统也简称 Ant-Q 算法。ACS 主要有三方面不同于蚁群算法。

1) 转移规则(State Transition Rule，STR)不同

蚁群系统在选择下一配送点时从原来单一的“随机性选择”中增加了一次“确定性选择”，两种方式结合使用，被称为伪随机选择规则(pseudorandom proportional rule)。从节点 i 选择节点 j 的方法重新定义为

$$j=\begin{cases}\mathrm{RWS}\{p_{ij}^{k}\}, & p>p_0\\ \mathrm{argmax}_{x\in \mathrm{allowed}_k}\{[\tau_{ix}]^{\alpha}\cdot[\eta_{ix}]^{\beta}\}, & p\leqslant p_0\end{cases} \tag{5.40}$$

其中，$x\in \mathrm{allowed}_k$ 表示第 k 只蚂蚁所有允许访问的节点；j 是第 k 只蚂蚁允许访问节点中的某一个；p 是一个随机数；p_0 是一个固定参数，称为调谐参数(tuning parameter)，决定了探索和利用的相对重要性。当 $p\leqslant p_0$ 时，蚂蚁将以 p_0 概率朝着由信息素和启发信息指示的方向移动，也就是蚂蚁利用先验知识进行移动。同时允许蚂蚁以$(1-p_0)$的概率进行有偏探索。有了 p_0 之后，就可以对探索的程度进行调整，确定是选择集中在较好解附近搜索还是探索其他新解。

当蚂蚁要选择下一个移动的城市时，算法先产生一个[0,1]范围内的随机数 p，然后判断该随机数与已知的固定参数 p_0 之间的大小关系，最后选择蚂蚁移动方向的其中一种方法：如果 $p\leqslant p_0$，蚂蚁确定性地选择信息素和启发信息乘积最大的节点；如果 $p>p_0$，则依然根据基本蚁群算法计算出每个节点被选择的概率，然后按轮盘赌选择法随机性地选择下一个节点。

2) 全局更新(global updating)规则不同

ACS 偏爱使用全局最优路径，即

$$\tau_{ij}=(1-\rho)\cdot\tau_{ij}+\rho\cdot\Delta\tau_{ij}^{*} \tag{5.41}$$

$$\Delta\tau_{ij}^{*}=\begin{cases}\dfrac{Q}{L_{\mathrm{gb}}}\\ 0\end{cases} \tag{5.42}$$

3) 新增局部更新(local updating)规则

蚂蚁在构造路径的每一步过程中，就立即用局部信息素更新规则修改目前已经访

问过路径上相应的信息素值

$$\tau_{ij}=(1-\xi)\cdot\tau_{ij}+\xi\cdot\tau' \tag{5.43}$$

其中，$\xi\in(0,1)$是信息素衰减系数(pheromone decay coefficient)，对于τ'有3种可供选择的计算方法[33]。

(1) 方法一：$\tau'=\gamma\cdot\max\limits_{z\in J_k(j)}\tau_{jz}$，类似于Q-learning公式，也叫作Ant-Q。

(2) 方法二：$\tau'=\tau_0$，τ_0是信息素初始值，也叫作ACS。在TSPLIB中不同的数据集上多次实验表明：$\tau_0=\dfrac{1}{nL_{nn}}$时结果较好，$n$是城市节点数，$L_{nn}$表示旅行商问题中按最近邻启发式方法生成的路径长度。

(3) 方法三：$\tau'=0$，实验证明方法三的性能是三者中最差的。

前两种方法的性能很相近，但是Ant-Q的计算量要比ACS的大很多，所以主要关注于ACS，并利用其进行局部更新。局部更新使蚂蚁已经访问过的路径对应的信息素减少，来逐渐降低它们的吸引力，使之后的蚂蚁有更多的机会去探索新路径而不是已经访问过的路径，所以局部更新的主要目的是通过减少集中在某些路径的信息素，使在每次迭代中后面的蚂蚁的搜索多样化。这样，后面的蚂蚁被鼓励选择其他的路径，就会产生不同的解，使不同的蚂蚁在一次迭代中产生相同解的可能性大大降低。

2. 最大最小蚁群系统

研究结果表明，蚁群系统ACS对基本蚁群算法的改进是行之有效的，它可以使得算法在搜索的过程中围绕最优解进行，保证了求解的质量，加速了算法的收敛速度。然而，由于ACS过于强调最优解的引导作用，导致算法容易陷入局部最优解，即出现搜索过程中的早熟现象。M. Dorigo等学者在基本ACS的基础上提出了称为Ant-Q System[34]的更一般的蚁群算法，仅让每一次循环中最短路径上的信息素更新，每次让信息素最大的路径以较大的概率被选中，以充分利用学习机制，强化最优信息的反馈。

为了克服Ant-Q中可能出现的算法停滞的现象，德国学者T. Stutzle和H. Hoos[35]提出了改进的最大最小蚁群系统(MAX-MIN Ant System，MMAS)，该算法允许各个路径上的信息素在一个限定的范围内变化。MMAS是到目前为止解决TSP最好的ACO类算法。

MMAS与AS主要有3个方面不同。

1) 将信息素浓度限定在$[\tau_{\min},\tau_{\max}]$区间内

无论采用迭代过程中的最优解还是全局搜索的最优解对信息素进行更新，都可能导致搜索的停滞。这是因为，在蚁群搜索过程中，采用这两种方式进行更新的路径，其信息素浓度明显高于其他路径上的信息素浓度，使得蚁群在搜索的过程中都会选择这样的路径，在正反馈机制的作用下，信息素浓度越高的路径会吸引更多的蚁群作为其下一步移动方向，而其他路径上的信息素浓度较低，选择的蚁群数量减少，从而使得浓度高的路径上浓度不断提高，同时在信息素挥发机制的影响下，其他路径上的信息素浓度降低。当差异达到一定阶段，停滞现象就发生了。为了避免这种停滞状态的发生，MMAS算法对信息素浓度进行了限定，规定了它的最大值和最小值。由于在信息素的更新过程中，对路径上的信息素浓度大小进行了限定，所以最优路径上的信息素浓度和最差路径上的信息素浓度差异就能够得到很好的控制，通过设置它们的大小，

减小最优路径的选择概率，保证最差路径的最小选择概率，很容易地解决了算法运行过程中各信息素浓度之间差异过大的问题。

MMAS 对路径上信息素浓度的最小值和最大值分别设置为 $\tau_{\min}$ 和 $\tau_{\max}$，使得所有路径上信息素浓度 τ_{ij} 都维持在一定范围之内，即 $\tau_{\min} \leqslant \tau_{ij} \leqslant \tau_{\max}$。在信息素更新的过程中，信息素浓度高于最大值，即 $\tau_{ij} > \tau_{\max}$，则将其大小设置成最大值，$\tau_{ij} = \tau_{\max}$。信息素浓度低于最小值，即 $\tau_{ij} \leqslant \tau_{\min}$，则设置 $\tau_{ij} = \tau_{\min}$。

2）只更新最优解上的信息素浓度

MMAS 综合了蚁群系统的优点，为加强最优解的引导作用，在每次算法迭代完成以后，只更新最优解路径上的信息素。其信息更新规则为

$$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{\text{best}} \tag{5.44}$$

$$\Delta\tau_{\text{best}} = \frac{1}{f(s_{\text{gb}})} \tag{5.45}$$

其中，$f(s_{\text{best}})$表示迭代过程中最优解(s_{ib})或全局最优解(s_{gb})的值。

3）信息素初始状态被设置为 $\tau_{\max}$

为使蚂蚁在算法的初始阶段能够更多地搜索新的解决方案，将信息素浓度大小初始化为 $\tau_{\max}$，而在蚂蚁系统中不存在这样的设置。

在 MMAS 中如何选择适当的信息素浓度限定值是很重要的。首先必须考虑算法的收敛性。若在每个选择点上，一个解元素上的信息素浓度为 $\tau_{\max}$，而其他所有可选择的解元素上信息素浓度均为 $\tau_{\min}$，则称 MMAS 是收敛的，算法始终选择信息素浓度最大的解元素所构造的解。

对于任意 τ_{ij}，有

$$\lim_{t\to\infty} \tau_{ij} = \tau_{ij} \leqslant \frac{1}{1-\rho} \times \frac{1}{f(s_{\text{opt}})} \tag{5.46}$$

在 MMAS 中，将最大信息素浓度 $\tau_{\max}$ 设置为渐进的最大值估计，是通过使用 $f(s_{\text{gb}})$代替 $f(s_{\text{opt}})$来实现的。每次找到一个新的最优解，$\tau_{\max}$ 都将被更新，使得 $\tau_{\max}$ 成为一个动态变化的值。

$\tau_{\min}$ 值的选取可以通过将算法的收敛性与最小信息素浓度的限制联系起来考虑。假设 P_{best} 在所有决策点上均为常数，蚂蚁需作 n 次“正确”的决策，则它将以概率 P_{dec}^{n} 来构造最优解

$$P_{\text{dec}}^{n} = P_{\text{best}}, \quad P_{\text{dec}} = \sqrt[n]{P_{\text{best}}} \tag{5.47}$$

$\tau_{\min}$ 的值可以通过以下两个公式计算：

$$P_{\text{dec}} = \frac{\tau_{\max}}{\tau_{\max} + (\text{avg}-1) \times \tau_{\min}} \tag{5.48}$$

$$\tau_{\min} = \frac{\tau_{\max}(1-P_{\text{dec}})}{(\text{avg}-1) \times P_{\text{dec}}} = \frac{\tau_{\max}(1-\sqrt[n]{P_{\text{best}}})}{(\text{avg}-1) \times \sqrt[n]{P_{\text{best}}}} \tag{5.49}$$

若 $P_{\text{best}} = 1$，则 $\tau_{\min} = 0$。如果 P_{best} 过小，可能会出现 $\tau_{\min} > \tau_{\max}$，这时需设置 $\tau_{\min} = \tau_{\max}$。

3. 自适应蚁群算法

蚁群算法是一种随机型的启发式搜索算法，在求解问题全局最优解的过程中主要

包含两个阶段：适应阶段和协作阶段。其中，适应阶段主要是根据当前获取的最优解信息不断调整搜索方向；协作阶段主要是指蚁群之间的信息交流。尽管蚁群算法是一种优秀的优化算法，在许多的应用领域都取得巨大的成功，但与其他进化算法一样，算法本身也存在一定的缺陷。自适应蚁群算法主要针对蚁群算法的不足，通过自适应地改变算法的信息挥发系数、自适应地调整蚁群路径选择策略、自适应地修正信息量大小、自适应地调节启发参数等方式，保证算法以较快的收敛速度找到全局最优解。

1）自适应调节信息素挥发系数

基本蚁群算法利用信息正反馈的机制，通过不断消减蚁群较少通过路径上的信息素，提高最优解包含路径上信息素浓度的方式，引导蚁群朝着最优解的方向进行。同时，由于信息素挥发系数 ρ 的存在，当信息素挥发系数较大时，一些蚁群较少通过的路径，其信息素浓度迅速减小，并不断向 0 靠近，最终使得这些路径以极小的概率被蚁群选中。然而，并非所有信息素浓度为 0 的路径都是非最优解路径，采用完全一致的信息素更新方式，就会降低算法获取全局最优解的能力。而采用较小的信息素挥发系数时，由于信息素挥发较少，各条路径上的信息素浓度差异较小，不利于引导蚁群朝着最优解方向进行，算法收敛时间较长。王颖等[36]提出了一种自适应改变信息素挥发系数的方法，设置信息素的初始挥发系数为 1，当算法在一定迭代次数出现停滞时，就自适应地改变挥发系数的大小。

2）自适应地调整蚁群路径选择策略和信息量大小

当采用蚁群算法求解大规模优化问题时，由于蚁群的数目远远小于路径条数，在蚁群搜索的过程中，有些路径有蚁群经过，而有些路径可能在整个搜索过程中都未被选择。因此，当进行信息素浓度更新时，这些未被选中路径上的信息素浓度就会被减小，而被选中路径上的信息素浓度被加大，即蚁群搜索过的路径上的信息素大于未选择路径上的信息素，在正反馈机制的作用下，蚁群的搜索过程朝着信息素浓度较高的方向进行，导致了蚁群在搜索的过程中，不断地朝着信息素浓度较高的几条路径上进行选择。这个过程中，因为规模较大时，蚁群运用概率选择策略选择下一次的移动方向，使得蚁群算法对问题求解不具有稳定性，每次搜索可能选择的路径不相同，最终搜索到的最优解变化就较大。针对这种不足，陈岐[37]提出了一种基于分布均匀度的自适应蚁群算法，根据优化过程中解的分布均匀度自适应地调节路径选择概率的确定策略和信息量更新策略。

3）自适应地调整信息启发参数

蚁群算法的引导作用主要是利用信息素浓度大小来实现的，当蚁群搜索进入一定阶段，各条路径上的信息素存在一定的差异，从而引导蚁群朝着信息素浓度较高的方向进行。由于基本蚁群算法在进化的初级阶段，各条路径上的信息素浓度相同，需要经过较多次迭代的搜索、多次的信息更新才能实现较优路径上的信息素浓度明显高于其他各条路径上的信息素浓度，因此，收敛时间较长。甘荣伟等[38]提出了一种具有路径平滑和信息动态更新的蚁群算法，根据当前蚁群搜索过程中求解的状态，通过路径平滑技术，对路径进行特定的平滑变换，保证蚁群在搜索过程中通过不断调整路径启发信息的大小（即间接地调节路径信息启发重要程度参数）的方式来实现蚁群搜索过程中的自适应性，减小搜索空间，加快算法收敛速度。

5.3.5　蚁群算法应用

蚁群算法提出至今已有十多年的时间，其理论正在形成一个较为严整的体系，有关基础也开始逐步奠定，而应用范围已几乎遍及各个领域，获得了极大的成功。它的主要应用领域有以下几部分[39]。

1. 通信问题

随着 Internet 上广泛的分布式多媒体应用对服务质量(QoS) 需求的增长，各种服务应用对网络所能提供的 QoS 提出了不同的要求。路由是实现 QoS 的关键之一，将蚁群算法用于解决受限路由问题，目前可以解决包括带宽、延时、包丢失率和最小花费等约束条件在内的 QoS 组播路由问题[40-42]，比现有的链路状态路由算法具有明显的优越性。但目前所研究的实例相对简单，没有对更复杂的 QoS 问题进行深入探讨。

2. 多目标分配问题

生产调度是一个复杂的动态管理优化问题。因为蚁群算法机制可以不断从过去的加工经历中学习，能自然地适应车间内外部环境的变化，从而实现动态调度，所以，它能适应动态的工件到达、不确定的加工时间以及机床故障等扰动，比静态确定性调度算法具有更好的应用前景[43-44]。

蚁群算法同样可求解混流装配线调度等动态任务多目标分配问题[45-46]，并对车间内外部环境变化具有良好的自适应性。但这些应用研究大多是针对小规模实例的仿真，用蚁群算法解决大规模生产调度和多目标分配问题是今后进一步的研究方向。

3. 电力系统优化问题

电力系统优化是一个复杂的系统工程，它包括无功优化、经济负荷分配、电网优化及机组最优投入等一系列问题，其中很多是高维、非凸、非线性的优化问题。侯云鹤等[47]将蚁群算法成功地用于解决经济负荷分配问题；王志刚等[48]则解决了该算法在配电网优化规划中的初步设计问题。

电力系统优化中的机组最优投入问题是寻求一个周期内各个负荷水平下机组的最优组合方式及开停机计划，使运行费用最小。利用状态、决策及路径概念，将机组最优投入问题设计成类似 TSP 问题的模式，从而可方便地利用蚁群算法进行求解。

4. 机器人路径规划问题

机器人路径规划就是在障碍有界空间内找到一条从出发点到目标位置的无碰撞且能满足一些特定要求的满意路径。近几年，许多学者开始用蚁群算法在这方面进行了一系列出色的研究工作。

为有效地解决机器人避障问题，并扩展其对具体问题的适应性，在蚁群算法中通过调整避障系数，可以得到不同的优化轨迹。澳大利亚学者 R. A. Russell[49] 设计了一种用于移动机器人的气味传感器导航机制，并深入分析了蚁群算法在该领域应用的鲁棒性。瑞士学者 J. B. K. Michael 等[50]将蚁群算法的程序编入微型机器人中，使众多微型机器人像蚂蚁一样协同工作，完成复杂任务。这项研究成果已被 *Nature* 报道。

此外，蚁群算法还在数据挖掘[51]、参数辨识[52]、图像处理[53]、图形着色[54]、分析化学[55]、岩石力学[56]以及生命科学[57]等领域的应用取得了很大进展。综合来看，蚁群算法与其他启发式算法相比，在求解性能上具有很强的鲁棒性，搜索能力较强且易与其他算法(如遗传算法、PSO 算法)结合并改善算法性能，因此蚁群算法应用前景十分广阔。

5.4 菌群算法

随着科学技术的发展，现实问题越来越复杂，学者们基于各种自然界存在的群智能现象，提出了许多群智能优化算法。5.2 节和 5.3 节中介绍的 PSO 算法和蚁群算法都是 20 世纪 90 年代早期提出的经典算法，经过二三十年的发展已经相对成熟和完善，本节将介绍一种新兴的群智能算法——菌群算法，菌群算法是受大肠杆菌的觅食行为启发而提出的。

5.4.1 菌群算法背景

菌群优化(Bacterial Foraging Optimization，BFO)算法也被称为 BFA(Bacterial Foraging Algorithm)，是由 K. M. Passino 教授[58]基于生物学家 H. C. Berg[59]对大肠杆菌群体的研究成果所提出的用来解决优化问题的群智能优化算法。

大肠杆菌是人体大肠内的一种常见细菌，它生活在人体大肠内的溶液环境中，溶液中有大肠杆菌生存所需要的各种营养物质。大肠杆菌群体在觅食时，个体会通过向自己的周围散播化学信息(一种称为引诱剂的化学物质)，来通知同伴在环境中的营养物质的分布情况，实现通信的目的[60]。引诱剂的释放范围是以细菌为中心的一个区域，蚂蚁的信息素浓度随时间增加而减小，相似地，引诱剂的浓度随着离开细菌的距离增大而减小。引诱剂有两种，分别发挥吸引与排斥的作用，其中吸引的作用范围大于排斥的作用范围。这两种化学信息的作用类似于物理中的引力与斥力作用，因此，也可以把细菌间的这种通信机制看成引力/斥力机制。具有吸引作用的引诱剂浓度越高则代表该位置上的营养物质多，因此细菌群体就朝着环境中引诱剂浓度高的方向上运动。在大肠杆菌群体中，引诱剂的浓度信息就是细菌群体之间的通信介质，大肠杆菌接收来自其他细菌的化学信息，进而决定自己的移动趋势，从而完成觅食行为。

5.4.2 菌群算法原理

细菌的觅食过程是一种趋利避害的行为，它通过自身周围的纤毛为其他细菌进行信息传递，通过自身周围的鞭毛的摆动进行移动。随着对环境的适应，细菌会进行分裂或是消亡，并且对所经过的环境状态还有一定的记忆性，因此可以在觅食的过程中避开有毒物质而沿着食物源的方向前进。菌群算法是对大肠杆菌在趋向性过程中的行为进行模拟，以此进行建模迭代来产生最优解，依靠概率性向最优结果逼近的一种仿生搜索算法。

菌群算法将每个细菌看作待求解问题的一个可行解，然后利用细菌的趋向(chemotaxis)、聚群(swarming)、繁殖(reproduction)和迁徙(elimination and dispersal) 4 个生物行为完成解的搜索过程。以求目标函数极小化问题为例，下面详细介绍这 4 个生物行为的实现过程[61]。

1. 趋向行为

在觅食过程中，细菌会首先执行翻转操作以产生一个随机移动方向，沿着该方向

前进一步。然后，细菌比较移动前后食物营养的丰富程度，如果移动后食物营养更加丰富，细菌会继续沿着该方向移动，直到食物营养变得匮乏或者已经在同一方向上移动了足够长的距离；否则细菌会通过翻转产生另一个随机方向后再进行如上的觅食行动。细菌在觅食过程中沿着相同方向连续前进的行为称为游动或者游泳。具体地，细菌沿着某个方向移动一步所发生的解的变化可表示为

$$X^i(j+1,k,l)=X^i(j,k,l)+C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\cdot\Delta(i)}} \tag{5.50}$$

其中，$X^i(j,k,l)$是第 i 个细菌在第 j 次趋向、第 k 次繁殖、第 l 次迁徙操作时的位置即第 i 个细菌所表示的解(有时会略去索引，用 X^i 表示第 i 个细菌的位置)；$\Delta(i)=(\delta_1(i),\delta_2(i),\cdots,\delta_n(i))$是一个 n 维随机向量，每个分量 $\delta_p(i)(p=1,2,\cdots,n)$是$(-1,1)$内一个均匀分布的随机数，$\Delta(i)/\sqrt{\Delta^{\mathrm{T}}(i)\cdot\Delta(i)}$是通过向量单位化得到的 n 维单位向量，它表示细菌通过翻转操作确定的觅食方向；$C(i)$是细菌 i 的移动步长。细菌在 $X^i(j,k,l)$处的目标函数值或者花费(cost) 表示为 $f(X^i(j,k,l))$，也可写为 $f(i,j,k,l)$。当 $f(X^i(j+1,k,l))<f(X^i(j,k,l))$时，细菌 i 将沿着相同方向继续向前运动，直到目标函数值不再降低或者已经在相同方向上移动了足够多的步数为止，最大的步数习惯用 N_s 表示。从本质上讲，趋向机制是菌群算法的核心操作，是个体解向最优解逼近的驱动力。从实现方式上说，它是翻转操作和游动操作复杂交织的运动过程，其中翻转操作控制寻优的方向，游动操作决定在某个方向上搜索优良解的程度。

2. 聚群行为

细菌个体在移动过程中通常会同时释放引诱信号和排斥信号。引诱信号吸引其他细菌个体不断靠近自己，排斥信号保证其他细菌个体不能无限制地靠近自己，而是保持在一定距离之外。细菌个体之间的上述相互作用表述为

$$\begin{aligned}f_{\mathrm{cc}}(X^i,P(j,k,l))&=\sum_{r=1}^{s}f_{\mathrm{cc}}^r(X^i,X^r)\\&=\sum_{r=1}^{s}\left[-d_{\mathrm{att}}\exp\left(-w_{\mathrm{att}}\sum_{p=1}^{n}(x_p^i-x_p^r)^2\right)\right]+\\&\quad\sum_{r=1}^{s}\left[-h_{\mathrm{rep}}\exp\left(-w_{\mathrm{rep}}\sum_{p=1}^{n}(x_p^i-x_p^r)^2\right)\right]\end{aligned} \tag{5.51}$$

其中，S 是菌群中细菌个体数目；$P(j,k,l)=\{X^r(j,k,l)\mid r=1,2,\cdots,S\}$是菌群中所有个体在第 j 次趋向、第 k 次繁殖、第 l 次迁徙操作时的位置；d_{att} 和 w_{att} 分别表示引诱信号的释放量和扩散率；h_{rep} 和 w_{rep} 分别表示排斥信号的释放量和扩散率；$f_{\mathrm{cc}}(X^i,P(j,k,l))$表示其他个体对细菌 i 的作用力。

聚群机制是菌群算法中一个可选的机制。当有聚群机制时，细菌个体 i 根据 $f(X^i(j,k,l))+f_{\mathrm{cc}}(X^i,P(j,k,l))$而不是单一的 $f(X^i(j,k,l))$进行寻优判断。从表面上看，聚群机制使每个个体与其他个体都发生了联系，貌似起到了信息沟通的作用。但实际上，聚群机制并没有给予个体具体的行动指导，仅仅在目标函数值上加

了一个附加值，因而所起到的信息交流作用非常有限。除此之外，聚群机制的计算量比较大，而且含有多个参数。因此许多菌群算法研究并不采用该机制。

3. 繁殖行为

随着营养的吸收，细菌会逐渐变长。在适当的条件下，营养充足的细菌个体会发生无性繁殖，即一个个体分裂为两个完全相同的子个体，而营养匮乏的细菌个体会消亡。在菌群算法中，N_c 次趋向操作被看作细菌生命的存活长度，即每经过 N_c 次趋向操作，菌群发生一次繁殖过程。通常设定 $S_r = S/2$ 个优良细菌个体进行无性繁殖。具体实现过程为：首先根据健康度对菌群进行降序排列，然后前一半比较健康的细菌中的每个个体分裂为两个相同的子个体(两个子个体有相同的解)，另一半不健康的细菌则全部被丢弃。在该过程中，菌群中每个个体的健康程度根据其 N_c 次趋向操作累积的函数值来评价，即

$$f_{\text{health}}^{i} = \sum_{j=1}^{N_c+1} f(i,j,k,l) \tag{5.52}$$

本质上，繁殖行为可以看作一个相对抽象的达尔文进化模型，类似于遗传算法的选择操作。它起到了优胜劣汰的作用，有助于提高算法的收敛速度。

4. 迁徙行为

菌群生活的环境发生剧烈变化后，有的个体可能会迁移到新环境中去寻找营养源。菌群算法发生迁徙行为的总数记为 N_{ed}。菌群每完成 N_{re} 次繁殖操作，发生一次迁徙过程。细菌个体按照执行迁徙的规则为

$$X = \begin{cases} X', & q < P_{\text{ed}} \\ X, & \text{其他} \end{cases} \tag{5.53}$$

其中，X' 是细菌通过重新初始化到达的新位置；q 是在(0,1)区间中采样的一个均匀分布的随机数；P_{ed} 是迁徙概率。

在以上知识的基础上，给出菌群算法[62]原理，如算法 5.4 所示，菌群算法的流程图如图 5.18 所示。

算法 5.4：菌群算法(BFO)

Step 1：初始化参数。初始化算法中的所有参数：菌群大小 S，迭代次数 G，趋向次数 N_c，繁殖次数 N_{re}，迁移次数 N_{ed}，趋向步长 $C(i)$，游动次数 N_s，迁徙概率 P_{ed} 以及细菌个体的初始位置。

Step 2：对细菌个体进行趋向操作。首先对细菌进行翻转操作，然后计算其适应度值，如果翻转后的适应度值得到改善，则按照翻转的方向继续对细菌进行游动操作，直至达到最大游动步数 N_s 或适应度值不再改善。

Step 3：对菌群进行繁殖操作。趋向操作完成后，计算每个细菌的适应度值，并且根据适应度值的大小进行排序，对适应度值较强的半数细菌进行繁殖操作，对适应度值较弱的细菌进行消亡操作。

Step 4：对细菌个体进行迁徙操作。菌群繁殖完成后，随机生成一个[0,1]的常数，并将其与迁徙概率进行比较，如果小于 P_{ed}，则将细菌随机分散在搜索空间中，以此使细菌个体有一定的机会重新开始优化。

Step 5：判断终止条件。如果满足终止条件，程序结束输出结果，否则，返回 Step 2。

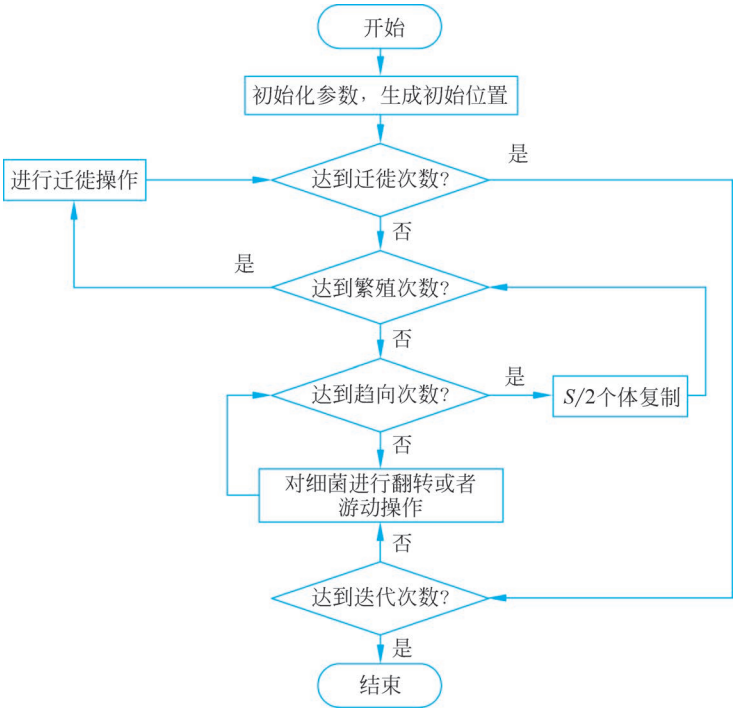


图 5.18　菌群算法流程图

5.4.3　菌群算法应用

作为一种新兴的群智能算法，菌群算法的理论体系尚未完整建立。菌群算法中参数较多，目前对于相关参数的选取没有成系统的方法。但菌群算法具有并行的寻优能力且对初值不敏感等特点，因此它在一些领域得到了有效应用。下面介绍目前几个比较主流的应用领域。

1. 多目标优化

大量工程和科学领域经常遇到相互冲突的多个目标均可能是最佳的优化解的问题，即最优解通常是一个解集，这一类问题称为多目标优化问题。传统方法求解多目标优化问题经常只能得到一个解而不是 Pareto 的最优解集，而且求解效率不够高。因此将细菌觅食优化算法应用于多目标优化问题，菌群中的全局最优位置和细菌个体的局部极优位置共同决定每个细菌个体的行为。在多目标优化问题中，如何选择下一步的最佳位置指导菌群个体进行运动成为解决该问题的关键点。乔英等[63]在多目标分布估计算法中融入细菌觅食行为，利用细菌觅食行为局部搜索能力强、搜索的多向性和全局性好等特点，能够在一次运行中获取多个 Pareto 最优解集。细菌觅食算法的引入使得问题求解在收敛性和分散性两方面表现良好，是一种有效的多同标进化算法，提高了算法的性能。

2. 路径规划

移动机器人路径规划需要机器人能够准确、及时地利用传感器获得的静态或动态的环境信息做出综合、合理的判断，并根据这些获取的信息做出正确的决策。相对于传统的动态路径规划算法，生物启发式算法都是基于生物在复杂的动态环境中寻找食物或迁徙的行为得到的启发，所以在处理路径规划问题上拥有优良的全局寻优能力。

梁晓丹等[64]将自然界中细菌的自适应觅食现象与移动机器人动态路径规划相类比，在菌群算法的基础上，融合了自然生物的局部区域搜索策略和自适应觅食策略，使其具有发现动态环境的营养梯度方向、继续发掘潜在解区域进而发现全局最优或局部最优解所在区域的能力。在该改进菌群算法驱动的搜索过程中，机器人可以顺利避开障碍，同时快速找到目标地点。

3. 神经网络训练

径向基函数(Radial Basis Function，RBF)神经网络是一种高效的前馈式神经网络，它具有其他前向网络所不具有的最佳逼近性能和全局最优特性，并且结构简单，训练速度快。同时，它也是一种可以广泛应用于模式识别、非线性函数逼近等领域的神经网络模型。缪凯[65]在细菌群体趋向性优化算法的基础上提出了微细菌群体趋向性算法，并借用微细菌群体趋向性算法的思想来确定 RBF 神经网络隐层神经元的控制参数，来全结构优化 RBF 神经网络，使 RBF 神经网络不但可以得到合适的结构，同时也可以得到合适的控制参数。结果表明，这种方法相对于传统的 K-means 和改进的 RBF 算法效率高，泛化能力好。

4. 图像处理

图像匹配作为图像处理领域中一个重要的课题，是指机器在识别物体过程中，将已知图像与未知图像的整体或部分进行空间上的对准，根据已知模式的图像在一幅陌生图像中寻找对应模式的子图像的过程。周美茹[66]提出将优化后的动态细菌觅食优化算法应用于图像匹配搜索，以最大互相关值为适应度函数，结合图像匹配这一具体应用问题设计种群，通过自适应趋向操作、动态概率繁殖操作、自适应迁徙操作，对细菌个体进行迭代寻优，最终找出图像中的最佳匹配点。实验结果表明，相比于传统图像匹配算法，基于动态细菌觅食优化算法的图像匹配有效地提高了匹配概率和精度，缩短了匹配时间。

5.5 其他群智能模型

大自然中存在着形形色色的以集群方式生存的生物，它们利用群体智能，经历了漫长的自然界的优胜劣汰而生存至今，它们表现出来的群体智能不断启发着学者们一次次去模拟和探索，为复杂优化问题的求解带来新的生机。继粒子群算法、蚁群算法和菌群算法之后，作为附加知识，本节将简要介绍人工鱼群算法和狼群算法的基本原理，扩展读者对群智能领域的了解。

5.5.1 人工鱼群算法基本原理

鱼群在我们的生活中较为常见，通过对鱼类生活习性的观察，可以总结出鱼群有 4 种典型行为：觅食行为、聚群行为、追尾行为和随机行为。觅食行为是生物的一种最基本的行为，也就是趋向食物的一种活动，一般可以认为这种行为是通过视觉或味觉感知水中的食物量或浓度来选择趋向的。聚群行为是鱼类较为常见的一种现象，大量或少量的鱼都能聚集成群，这是它们在进化过程中形成的一种生存方式，可以进行集体觅食和躲避敌害。追尾行为指当某一条鱼或几条鱼发现食物时，它们附近的鱼会尾随其后快速游过来，进而导致更远处的鱼也尾随过来。随机行为指鱼在水中悠闲地自由游动，基本上是随机的，其实它们也是为了更大范围地寻觅食物或同伴。鱼的这

些行为会在不同时刻相互转换，而这种转换通常是鱼通过对环境的感知来自主实现的，这些行为与鱼的觅食和生存都有着密切的关系。

人工鱼群算法是由李晓磊等[67]于 2002 年在鱼群寻找食物时表现的种种移动寻觅特点中得到启发而提出的一种群智能优化算法。在一片水域中，鱼生存数目最多的地方一般就是该水域中营养物质含量最高的地方。依据这一特点来模仿鱼群的觅食等行为，以期完成寻优目的，从而实现全局寻优[68]，这就是人工鱼群算法的基本思想。

鱼群的活动中，觅食行为、聚群行为、追尾行为和随机行为与寻优命题的解决有着密切的关系，如何构造并实现上述鱼群行为，从而在计算机设备上解决实际问题呢？

鱼群的觅食行为是循着食物多的方向游动的一种行为，在寻优算法中则是向较优方向前进的迭代方式。在聚群行为中，每条人工鱼遵守两个规则：一是尽量向临近伙伴的中心移动；二是避免过分拥挤。追尾行为是向临近的最活跃者追逐的行为，在寻优算法中可以理解为向附近的最优伙伴前进的过程。人工鱼群算法将每个备选解视为一条“人工鱼”，多条人工鱼共存，实现合作寻优（类似鱼群寻找食物）。在构建整体算法时，首先初始化为一群人工鱼（随机解），然后通过迭代搜寻最优解。在每次迭代过程中，人工鱼通过觅食、聚群及追尾等行为来更新自己，从而实现寻优。也就是说，算法的进行是人工鱼个体的自适应行为活动，即每条人工鱼根据周围的情况进行游动，人工鱼的每次游动就是算法的一次迭代。欲了解详细的算法构造实现过程，可参考文献[67]。

5.5.2　狼群算法基本原理

在自然界中，狼是处在食物链顶端的捕食者。狼群有着严密的等级制度，一个狼群一般有一个首领，称为头狼；若干探狼负责找寻好的食物；若干猛狼负责围捕猎物，狼群过着各司其职的生活。头狼是整个狼群中的关键，是最具智慧和最强壮的，是在“弱肉强食”式的残酷竞争中产生的首领。头狼负责整个狼群的决策，关乎着群体的兴衰。狼群在搜索猎物时并不会全体出动，而是先派出一些精锐的狼，称为探狼。探狼通过气味搜寻优质猎物。一旦探狼发现猎物，就会向头狼报告，头狼通过嚎叫呼唤周围的猛狼进行围攻。猎物分配上，狼群有着严格的等级制度，猎物并不是平均分配给群体中的每只狼，而是强者多分，弱者少分或不分。这样会使弱小的狼饥饿而亡，遭到自然界的淘汰，同时可以使狼群持续强盛[69]。

狼群研究（Wolf Pack Search，WPS）最早是由杨晨光等[70]提出的。2011 年，刘长安等[71]为解决优化问题提出了一种新狼群算法（Wolf Colony Algorithm，WCA）。吴虎胜等[72]于 2013 年提出了全新的狼群算法（Wolf Pack Algorithm，WPA），该算法更加详细地将狼群分为头狼、探狼与猛狼，并抽象出游走、召唤、奔袭与围攻等智能行为与“胜者为王”的头狼产生规则和“强者生存”的狼群更新机制。

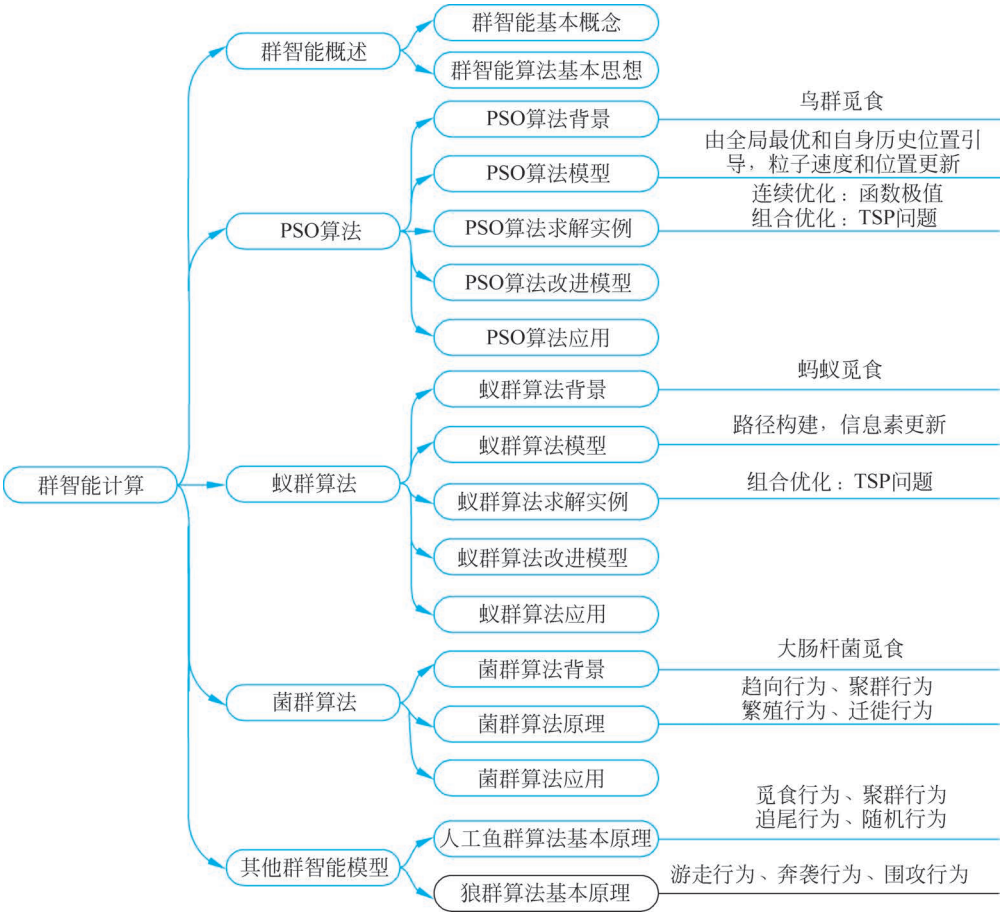
狼群算法旨在模拟狼群的捕猎行为来处理函数优化问题，将狼群分为三类：头狼、探狼和猛狼。算法的基本思想是：从待寻优空间中的某一初始猎物群开始，其中具有最佳适应度值的狼作为头狼，该操作称为头狼生成准则。然后，选取除头狼外最佳的 m 匹狼作为探狼，进行预定方向上的寻优搜索，采用新旧猎物规则保留较优质的猎物，一旦发现比当前头狼更优质的猎物，则具有该猎物的探狼成为头狼，此过程称为探狼游走行为。头狼发起嚎叫，通知周围猛狼迅速向头狼靠拢，探寻优质猎物，如果探寻到的优质猎物比头狼更优，则该狼代替头狼再次发起嚎叫，直到猛狼距离猎物一定距离时停止，此过程称为猛狼奔袭行为。当猛狼距离猎物达到预先设定的阈值时，转

变为围攻行为，对头狼附近的优质猎物进行寻优，此过程称为狼群围攻行为。将适应度值最差的 R 匹狼淘汰，同时在寻优空间内随机产生 R 匹狼进行补充，此过程称为“强者生存”的狼群更新机制。欲了解狼群算法的详细内容，请参考文献[72]。

本章小结

群智能算法是指受生物群体行为研究的启发，人类模拟自然界群居动物的觅食和繁殖等行为或者动物群体的捕猎策略等对问题进行求解的优化算法。

到目前为止，人们已经发明了大量群体计算模型，它们表现出了与单个个体完全不同的非凡计算能力，可以用于求解大量复杂的科学与工程问题，受到了广泛的关注和研究，并发展成为群智能优化计算的新型研究方向。本章从群智能算法的基本概念开始介绍，总结了群智能算法的特点，并简单介绍了群智能算法的基本思想。在此基础上，详细介绍了群智能算法中的主流算法——PSO 算法、蚁群算法。总体来说，群智能算法都是基于概率计算的随机搜索进化算法，在结构、研究内容、方法及步骤上有较大的相似性，其结果也都偏随机性。目前群智能算法在数学理论基础研究上还比较薄弱，且参数设置大都没有确切的理论依据，对具体问题和应用环境的依赖性比较大。通过本章的学习，读者可以对群智能算法有更深入的了解，并尝试将这些算法应用于其他领域，解决领域中面临的问题。



习题

1. 群智能算法的基本思想是什么？

2. 常见的群智能算法有哪些？

3. 简述群智能算法与进化算法的异同。

4. 请按自己的理解用自己的语言举例说明 PSO 算法的基本思想。

5. 简述鸟群觅食被抽象为 PSO 算法的对应关系。

6. 简述 PSO 算法的流程。

7. 计算 5.2.3 节中第三代和第四代更新后粒子的速度和位置。

8. 简述 PSO 算法中各参数的意义及选择方法。

9. 根据式(5.4)线性调整惯性权重，计算 5.2.3 节中标准 PSO 算法第二代、第三代和第四代更新后粒子的速度和位置。

10. 试自行编程实现 PSO 算法，具体求解一个优化问题。

11. PSO 算法中惯性权重如何进行自适应？试设计出满足自适应变化的惯性权重变换公式。

12. 举例说明蚁群算法的基本原理。

13. 简述蚁群觅食被抽象为蚁群算法的对应关系。

14. 简述蚁群算法的流程。

15. 计算 5.3.3 节蚁群算法实例中第一次信息素更新时每条路径上的信息素浓度。

16. 简述蚁群算法中各参数的意义，试改变 5.3.3 节蚁群算法实例中参数的值并计算结果。

17. 5.3.3 节蚁群算法求解 TSP 问题的结果是唯一的吗？结合 5.3.3 节蚁群算法实例分析蚁群算法的优缺点。

18. 试自行编程实现 ACO 算法，具体求解一个组合优化问题。

19. 试自行编程实现最大最小蚁群系统，求解第 16 题中的组合优化问题，比较结果并总结两种算法的异同。

20. 简述菌群算法的基本原理。

21. 简述菌群算法的流程。

22. 简述菌群算法与粒子群算法、蚁群算法的异同，并简述菌群算法的优缺点。

23. 试自行编程实现菌群算法，具体求解一个优化问题。

24. 简述人工鱼群算法和狼群算法的基本原理，并查阅相关文献，了解人工鱼群算法和狼群算法的流程。

25. 群智能算法有哪些共同的特点？目前研究存在的问题是什么？

参考文献

[1]　雷秀娟. 群智能优化算法及其应用[M]. 北京：科学出版社，2012.

[2]　王万良. 人工智能导论[M]. 北京：高等教育出版社，2017.

[3] Beni G, Wang J. Swarm intelligence[C]//Proceedings for the 7th Annual Meeting of the Robotics Society of Japan, 1989.
[4] 张青, 康立山, 李大农. 群智能算法及其应用[J]. 黄冈师范学院学报, 2008(06): 44-48.
[5] Langton C G. Artificial life: The proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems[M]. New Mexico: Longman Publishing, 1989.
[6] 霍兰. 隐秩序:适应性造就复杂性[M]. 周晓牧, 韩晖, 译. 上海: 上海科技教育出版社, 2001.
[7] Reynolds C W. Flocks, herds and schools: A distributed behavioral model[J]. Computer Graphies, 1987, 21(4): 25-34.
[8] Heppner F, Grenander U. A stochastic nonlinear model for coordinated bird flocks[C]// The Ubiquity of Chaos, 1990.
[9] Kennedy J, Eberhart R C. Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neural Networks, 1995.
[10] Shi Y H, Eherhart R C. Parameter selection in particle swarm optimization[J], Lecture Notes in Computer Science, 1998, 1447: 591-600.
[11] 王东风, 孟丽. 粒子群优化算法的性能分析和参数选择[J]. 自动化学报, 2016, 42(10): 1552-1561.
[12] 郭文忠, 陈国龙. 求解 TSP 问题的模糊自适应粒子群算法[J]. 计算机科学, 2006(06): 161-162+185.
[13] Angeline P J. Using selection to improve particle swarm optimization[C]//IEEE International Conference on Evolutionary Computation Proceedings, 1998.
[14] Lovbjerg M, Rasmussen T K, Krink T. Hybrid particle swarm optimizer with breeding and subpopulations[C]//Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, 2001.
[15] 石松, 陈云. 层次环形拓扑结构的动态粒子群算法[J]. 计算机工程与应用, 2013(08): 1-5.
[16] 高鹰. 一种自适应扩展粒子群优化算法[J]. 计算机工程与应用, 2006(15): 12-15.
[17] 胡春霞, 曾建潮, 王清华, 等. 一种免疫微粒群优化算法[J]. 计算机工程, 2007(19): 213-214.
[18] Liu B, Wang L, Jin Y H. Improved particle swarm optimization combined with chaos[J]. Chaos, Solitons and Fractals, 2005, 25(5): 1261-1271.
[19] Storn R, Price K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal Global Optimization, 1997, 11(4): 341-359.
[20] Clerc M. The swarm and the queen: towards a deterministic adaptive particle swarm optimization[C]//Proceedings of the IEEE Congress on Evolutionary Computation, 1999: 1951-1957.
[21] Trelea I C. The particle swarm optimization algorithm: convergence analysis and parameter selection[J]. Information Processing Letters, 2003, 85(6): 317-325.
[22] Zheng Y, Ma L, Zhang L, et al. On the convergence analysis and parameter selection in particle swarm optimization[C]//Proceedings of International Conference on Machine Learning and Cbernetics, 2003.
[23] 王芳. 粒子群算法的研究[D]. 重庆: 西南大学, 2006.
[24] Van Den Bergh F. An analysis of particle swarm optimizers[D]. Pretoria, Gauteng: University of Pretoria, 2007.
[25] 陈阳, 黄卫华, 何佳乐, 等. 基于粒子群模糊神经网络的立方体机器人建模[J]. 组合机床与自动化加工技术, 2022(07): 22-25, 29.
[26] 陈海焱, 陈金富, 段献忠. 含风电场电力系统经济调度的模糊建模及优化算法[J]. 电力系统自动化, 2006(02): 22-26.
[27] 李擎, 徐银梅, 张德政, 等. 基于粒子群算法的移动机器人全局路径规划策略[J]. 北京科技大

学学报,2010,32(03):397-402.
[28] 马良,朱刚,宁爱兵.蚁群优化算法[M].北京:科学出版社,2007.
[29] Colorni A,Dorigo M,Maniezzo V. Distributed optimization by ant colonies[C] //Proceedings of the first European conference on artificial life,1991.
[30] 李建军.基于蚁群算法的车辆路径规划问题的研究[D].西安:西安电子科技大学,2015.
[31] Dorigo M, Montes d O M A, Oliveira S, et al. Ant colony optimization[J]. Alphascript Publishing,2010,28(3):1155-1173.
[32] Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation,1997,1(1):53-56.
[33] Dorigo M,Birattari M,Stutzle T. Ant colony optimization: artificial ants as a computational intelligence technique[J]. IEEE Computational Intelligence Magazine,2006,1(4):28-39.
[34] Gambardella L M, Dorigo M. Ant Q: A reinforcement learning approach to the traveling salesman problem [C]//Proceedings of the 12th International Conference on Machine Learning,1995.
[35] Stutzle T, Hoos H. MAX-MIN Ant system and local search for the traveling salesman problem[C]//IEEE International Conference on Evolutionary Computation,1997.
[36] 王颖,谢剑英.一种自适应蚁群算法及其仿真研究[J].系统仿真学报,2002,14(1):31-33.
[37] 陈岐,沈洁,秦玲,等.基于分布均匀度的自适应蚁群算法[J].软件学报,2003,14(8):1379-1387.
[38] 甘荣伟,郭清顺,常会友,等.具有路径平滑和信息动态更新的蚁群算法[J].计算机科学,2010,37(01):233-235.
[39] 段海滨,王道波,朱家强,等.蚁群算法理论及应用研究的进展[J].控制与决策,2004(12):1321-1326,1340.
[40] LU G Y, LIU Z M. QoS Multicast Routing Based on Ant Algorithm in Internet[J]. The Journal of China Universities of Posts and Telecommunications,2000(04):12-17.
[41] Chu C H, Gu J, Hou X, et al. A heuristic ant algorithm for solving QoS multicast routing problem[C]//Proceedings of the 2002 Congress on Evolutionary Computation,2002.
[42] Ding J L,Chen Z Q, Yuan Z Z. Dynamic optimization routing method based on ant adaptive algorithm[J]. Control and Decision,2003,18(06):751-753,757.
[43] Shervin N. Agent-based approach to dynamic task allocation[C]// Proceedings of the 3th International Workshop on Ant Algorithms,2002.
[44] Li Y J, Wu T J. A nested ant colony algorithm for hybrid production scheduling[C]// Proceedings of the 2002 American Control Conference,2002.
[45] Li Y J, Wu T J. A nested hybrid ant colony algorithm for hybrid production scheduling problems[J]. Acta Auomatica Sinica,2003,29(1):95-101.
[46] Joaquin B, Jordi P. Ant algorithms for assembly line balancing[C]//Proceedings of Third International Work shop on Ant Algorithms,2002.
[47] Hou Y H,Wu Y W,Lu L J,et al. Generalized ant colony optimization for economic dispatch of power systems [C]//Proceedings of the International Conference on Power System Technology,2002(1).
[48] 王志刚,杨丽徙,陈根永.基于蚁群算法的配电网网架优化规划方法[J].电力系统及其自动化学报,2002(06):73-76.
[49] Russell R A. Ant trails-an example for robots to follow[C]//IEEE International Conference on Robotics and Automation,1999.
[50] Michael J B K,Jean-Bernard B,Laurent K. Ant-like task and recruitment in cooperative robots

[J]. Nature,2000,406(6799): 992-995.

[51] Tsai C F, Wu H C, Tsai C W. A new data clustering approach for data mining in large databases [C]//International Symposium on Parallel Architectures, Algorithms and Networks,2002.

[52] Abbaspour K C, Schulin R, Van Genuchten M T. Estimating unsaturated soil hydraulic parameters using ant colony optimization[J]. Advances inWater Resources, 2001, 24(8): 827-841.

[53] Salima Q, Mohamed B, Catherine G. Ant Colony System for Image Segmentation Using Markov Random Field [C]//Proceedings of the Third International Workshop on Ant Algorithms,2002.

[54] Costa D,Hertz A. Ants can colour graphs[J]. Journal of theOperational Research Society, 1997,48(3): 295-305.

[55] Ding Y P, Wu Q S, Su Q D. Ant colony algorithm and optimization of test conditions in analytical chemistry[J]. Chinese Journal of Chemistry,2003,21(6): 607-609.

[56] Wang C H,Xia X Y,Li G X. Ant algorithm in search of the critical slip surface in soil slops based stress fields[J]. Chinese Journal of Rock Mechanics and Engineering,2003,22(5): 813-819.

[57] Ando S,Iba H. Ant algorithm for construction of evolutionary tree[C]//Proceedings of the 2002 Congress on Evolutionary Computation,2002.

[58] Passino K M. Biomimicry of bacterial foraging for distributed optimization and control[J]. IEEE Control Systems Magazine,2002,22(3): 52-67.

[59] Berg H C. Motile behavior of bacteria[J]. Physics Today,2000,53(1): 24-29.

[60] 樊非之. 菌群算法的研究及改进[D]. 北京：华北电力大学,2010.

[61] 杨翠翠. 细菌觅食优化算法及其应用研究[D]. 北京：北京工业大学,2017.

[62] Das S, Biswas A, Dasgupta S, et al. Bacterial foraging optimization algorithm: theoretical foundations,analysis,and applications[M] Berlin,Heidelberg:Springer,2009,3: 23-55.

[63] 乔英,高岳林,江巧永. 基于细菌觅食行为的多目标分布估计算法[J]. 计算机应用研究,2011, 28(10): 3681-3683,3686.

[64] 梁晓丹,蔺娜,陈瀚宁. 基于细菌觅食行为的移动机器人动态路径规划[J]. 仪器仪表学报, 2016,37(06): 1316-1324.

[65] 缪凯. RBF 神经网络的研究与应用[D]. 青岛:青岛大学,2007.

[66] 周美茹. 细菌觅食优化算法研究及其在图像匹配中的应用[D]. 西安:西安电子科技大学,2014.

[67] 李晓磊,邵之江,钱积新. 一种基于动物自治体的寻优模式：鱼群算法[J]. 系统工程理论与实践,2002(11): 32-38.

[68] 王闯. 人工鱼群算法的分析及改进[D]. 大连：大连海事大学,2008.

[69] 李国亮. 狼群算法的研究与应用[D]. 南昌：东华理工大学,2016.

[70] Yang C G,Tu X Y,Chen J. Algorithm of marriage in honey bees optimization based on the wolfpack search[C]//Proceedings of the International Conference on Intelligent Pervasive Computing,2007.

[71] Liu C,Yan X,Liu C,et al. The wolf colony algorithm and its application[J]. Chinese Journal of Electronics,2011,20(2): 212-216.

[72] 吴虎胜,张凤鸣,吴庐山. 一种新的群体智能算法——狼群算法[J]. 系统工程与电子技术, 2013,35(11): 2430-2438.