第5章 智能优化算法

第4章介绍了搜索的概念，通过搜索方法的应用，从理论上讲能够找到所有优化问题的全局最优解。实际上并非如此，如某些问题的规模很大，状态空间极为复杂，即使采用目前最快的巨型计算机，也要花费很长时间（数天、数年甚至更长时间）才能得到结果，这在现实生活中是无法接受的。于是，考虑这样一类优化方法，能在可接受的时间内给出较优化的用户满意解（次优解）而非全局最优解。本章介绍的智能优化算法就是这样的方法。本章首先介绍邻域的概念、基于局部搜索算法的思路，在此基础上介绍几种常用的智能优化方法。

5.1 智能优化算法概述

5.1.1 优化问题的复杂度

在电子、通信、计算机、自动化、机器人、经济学和管理学等众多学科中不断出现了许多复杂的组合优化问题。

例 5.1 0-1背包问题。

给定 n 种物品和1个背包，物品 i 的重量是 W_i、价值为 V_i，背包的容量为 C。应如何选择装入背包的物品，使得装入背包中物品的总价值最大？

在该问题中，任一物品只有放入背包或不放入背包两种选择，对应可用1或0分别表示，所以称为0-1背包问题，其本质上属于组合优化问题，即通过对数学方法的研究去寻找离散事件的最优编排、分组、次序或筛选等。在此类问题中往往需要在庞大的搜索空间中寻找最优解或者准最优解。若采用传统优化方法（如搜索方法）需要遍历整个搜索空间，问题规模较大时会出现“组合爆炸”现象，算法难以在有效时间内完成搜索。鉴于实际工程问题的复杂性、非线性、约束性以及建模困难等特点，寻求高效的智能优化算法已成为热门研究内容之一。

优化问题按照其求解的复杂程度可以分为以下四类。

（1）P类问题：所有可以在多项式时间内求解的判定问题构成P类问题。

（2）非确定性多项式（Nondeterministic Polynomial，NP）类问题：复杂问题不能确定是否在多项式时间内找到答案，但可以在多项式时间内验证答案是否正确。显然，P类问题是NP类问题的子集。

长期以来，研究人员一直试图搞清楚NP类问题是否等于P类问题，即那些能在多项式时间内验证得出正确解的问题，是否都是具有多项式时间求解算法的问题呢？如果解决了这个问题，所有的NP问题都可以通过计算机解决，因为它们都存在多项式时间求解算法。

（3）NP-完全（NP-Complete，NPC）问题：为了论证“NP类问题是否等于P类问题”，研究人员想了很多办法，其中之一是问题约化。问题约化：如果用问题B的算法可以解决问题A，那么问题A可以简化成问题B。

如果存在这样一个NP问题，所有的NP问题都可以约化成它，则该问题称为NP完全问题。换句话说，只要找到某个NP完全问题的多项式时间求解算法，那么所有的NP问题都解决。

(4) NP-难(NP-Hard)问题：NP-完全问题需要满足它是一个 NP 问题，以及所有的 NP 问题都可以约化到它。

NP-难问题是满足 NP-完全问题定义的第二条但不一定满足第一条，即所有的 NP 问题都能约化到它，但它本身不一定是 NP 问题。显然，NP-难问题比 NP-完全问题的范围大，因为 NP-难问题没有限定属于 NP 类问题，即 NP-完全问题是 NP-难问题的子集。

目前尚未找到任何一个 NP-完全问题或 NP-难问题的多项式时间求解算法，只能用穷举法逐个检验，计算时间随问题的复杂程度通常呈指数增长。

在实际中，针对 NP-难问题，为了在有效时间内给出求解结果，通常是降低对 NP-难问题解的最优化要求，即不一定寻找最优解，而是寻找接近最优解的次优解(用户满意解)。本章要介绍的智能优化算法采用了这样的求解思路，是 NP-难问题最有效的求解方法之一。

5.1.2 典型智能优化算法

针对典型的 NP-难问题，传统优化算法搜索空间巨大，甚至可能搜索不到最优解。受到人类智能、生物群体社会性或自然现象规律的启发，研究人员提出很多智能优化算法来解决复杂优化问题，此类方法都是通过模拟或揭示某些自然界的现象和过程或生物群体的智能行为而得到发展。此类方法具有简单、通用、便于并行处理等特点，并可寻找到全局最优解或是接近全局最优的结果，在优化领域称它们为智能优化算法。

智能优化算法在解决大空间、非线性、全局寻优、组合优化等复杂优化问题方面所具有的独特优势，得到了国内外学者的广泛关注，目前已发展出许多有效的智能优化算法，比较典型的有依据固体物质退火过程和组合优化问题之间相似性提出的模拟退火算法、模仿自然界生物进化机制的遗传算法、模仿蚁群觅食路径选择的蚁群优化算法以及鱼群或鸟群运动行为的粒子群算法等。

本章将针对上述智能优化算法分别进行介绍。在介绍具体智能优化算法之前，首先引入邻域的概念，并介绍基于邻域的局部搜索算法。

5.1.3 邻域的概念

邻域是智能优化算法中一个非常重要的概念，也是众多智能优化算法完成求解的基础。对于函数优化问题，邻域可定义为在距离空间中以一点为中心的一个超球体。对于组合优化问题，邻域可定义为 $N: x \in D \rightarrow N(x) \in 2^D$，且 $x \in N(x)$，称为一个邻域映射。其中，2^D 表示 D 的所有子集组成的集合，$N(x)$ 称为 x 的邻域。如果 $y \in N(x)$，则称 y 为 x 的一个邻居。

邻域的构造依赖问题决策变量的表示，邻域的结构在现代智能化优化算法中起重要作用。下面通过例子说明邻域的构造方式和应用方法。

例 5.2 旅行商问题的邻域。

旅行商问题(Traveling Salesman Problem，TSP)是典型的 NPC 问题，其可描述为假设有一个旅行商人要访问 N 个城市，他必须选择所要走的路径，路径的限制是每个城市

只能访问一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径中的最小值。

TSP 解的一种表示方法为 $D=\{x=(i_1,i_2,\cdots,i_n)\mid i_1,i_2,\cdots,i_n$ 是城市序号的排列$\}$。可定义它的邻域映射为 2-opt，即 x 中的两个元素进行对换，即 $N(x)$中共包含 x 的 $C_n^2=n(n-1)/2$ 个邻居和 x 本身。

以 4 个城市的 TSP 为例，当 $x=(1,2,3,4)$时，表示旅行商从 1 号城市出发，以此经过 2、3、4 号城市，最终回到 1 号城市。此时，$C_4^2=6$，则 x 的邻域 $N(x)=\{(2,1,3,4),(3,2,1,4),(4,2,3,1),(1,3,2,4),(1,4,3,2),(1,2,4,3),(1,2,3,4)\}$。

类似可定义 k-opt($k\geqslant 2$)，即对 k 个元素按一定规则互换。k 的取值不同，邻域的结构也将完全不同。

5.1.4 局部搜索算法

基于邻域的概念可设计局部搜索算法求解 TSP。方法如下。

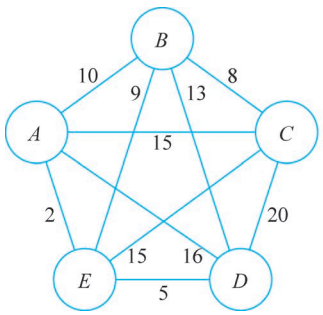
Step1：选定一个初始可行解 x_0，记录当前最优解 $x_{best}:=x_0$，设 $T=N(x_{best})$。

Step2：当 $T\backslash\{x_{best}\}=\varnothing$，或满足其他停止运算准则时，输出计算结果，停止运算；否则，从 T 中选一集合 S，得到 S 中的最好解 x_{now}；若 $f(x_{now})<f(x_{best})$，则 $x_{best}:=x_{now}$，$T=N(x_{best})$；重复 Step2。

其中，$f(\cdot)$表示旅行商问题中的评价准则，即旅行距离。

例 5.3 5 城市旅行商问题的局部搜索算法求解示例。

5 城市 TSP 中，各城市示意图和邻接矩阵如图 5.1 所示。



$$\boldsymbol{D}=(d_{ij})=\begin{bmatrix}0&10&15&6&2\\10&0&8&13&9\\15&8&0&20&15\\6&13&20&0&5\\2&9&15&5&0\end{bmatrix}$$

图 5.1 5 城市旅行商问题的示意图和邻接矩阵

假设，选定初始解为 $x_{best}=(ABCDE)$，$f(x_{best})=45$，定义邻域映射为对换两个城市位置的 2-opt，选定城市 A 为起点。

考虑两种利用邻域进行搜索的方法：

方法一：全邻域搜索。

第一步，计算 $N(x_{best})=\{(ABCDE),(ACBDE),(ADCBE),(AECDB),(ABDCE),(ABEDC),(ABCED)\}$。

对应目标函数为 $f(x)=\{45,43,45,60,60,59,44\}$。

则 $x_{best}:=x_{now}=(ACBDE)$。

第二步，计算 $N(x_{best})=\{(ACBDE),(ABCDE),(ADBCE),(AEBDC),(ACDBE),(ACEDB),(ACBED)\}$。

对应目标函数为 $f(x)=\{43,45,44,59,59,58,43\}$。

则 $x_{best}=(ACBDE)$ 没有变化，算法退出。

方法二：一步随机搜索。

第一步，从 $N(x_{best})$ 中随机选择一个解，如 $x_{now}=(ACBDE)$，其对应的目标函数为 $f(x)=43<45$，则 $x_{best}:=x_{now}=(ACBDE)$。

第二步，继续从 $N(x_{best})$ 中随机选择一个解，如 $x_{now}=(ADBCE)$，其对应的目标函数为 $f(x)=44>43$，则 $x_{best}=(ACBDE)$，不变。

上述步骤可继续进行，直到算法退出。

一步搜索算法又称为随机爬山算法。随机爬山算法通过在邻域中随机选择解的方式探索更优化解，但其只能接受比当前找到的最好解 x_{best} 更好的解，否则，保持 $x_{now}=x_{best}$ 不变。显然，这个特性容易使局部搜索算法陷入局部最优解，如图 5.2 所示。

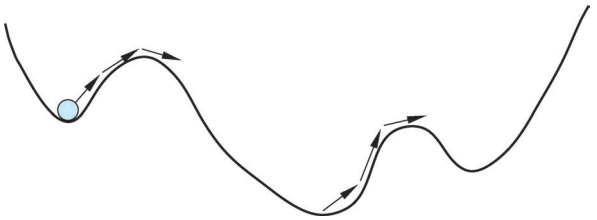


图 5.2　局部最优解与全局最优解示意图

由此可见，局部搜索算法有如下特点：

(1) 简单易行，但无法保证全局最优性；

(2) 局部搜索主要依赖起点的选取和邻域的结构；

(3) 为了得到好的解，可以比较不同的邻域结构和不同的初始点；

(4) 如果初始点的选择足够多，总可以计算出全局最优解。

显然，局部搜索算法的“智能性”还不够强，但其对于邻域的构造和利用是现代智能优化算法的思想基础。下面介绍几种典型的智能优化算法。

5.2　模拟退火算法

5.2.1　模拟退火算法的原理

模拟退火算法思想最早由 Metropolis 等于 1953 年提出，1983 年 Kirkpatrick 等将其应用于组合优化，用于解决 NP-Hard 问题。与局部搜索算法相比，该方法克服初值依赖性，而且改善优化过程，避免陷入局部极值。

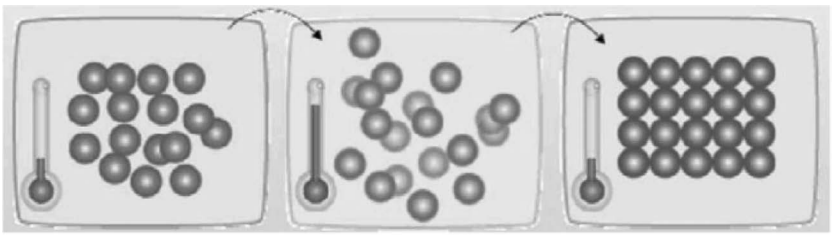
顾名思义，该算法是对现实生活中物理退火过程的模拟。在物理退火过程中，首先把固体加热到足够高的温度，使分子呈随机排列状态，然后逐步降温使之冷却，最后分子以低能状态排列，达到某种稳定固体状态。这一过程可概括为以下三个核心流程。

(1) 加温过程：增强粒子的热运动，消除系统原先可能存在的非均匀态。

(2) 等温过程：对于与环境换热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，系统达到平衡态。

(3) 冷却过程：使粒子热运动减弱并渐趋有序，系统能量逐渐下降，得到低能的晶体结构。

图 5.3(a)到图 5.3(b)是加温过程，图 5.3(b)到图 5.3(c)是冷却过程，而等温过程发生在图 5.3(b)中。



(a) (b) (c)

图 5.3 物理退火三个过程

模拟退火算法主要对其中的等温过程进行模拟。在该过程中，系统状态总体趋势是自由能由高到低变化，但也有随机的分子从低向高变化。对该过程进行理论描述，物理学的研究表明，在温度 T 时，分子停留在状态 r 的概率服从玻耳兹曼分布，即

$$P\{\bar{E}=E(r)\}=\frac{1}{Z(T)}\exp\left(-\frac{E(r)}{k_{\mathrm{B}}T}\right) \tag{5-1}$$

式中：$\bar{E}$ 为分子能量的一个随机变量；$E(r)$为状态 r 的能量；k_{B} 为玻耳兹曼常数，$k_{\mathrm{B}}>0$；$Z(T)$为概率标准化因子(常数)，且有

$$Z(T)=\sum_{s\in\mathcal{D}}\exp\left(-\frac{E(s)}{k_{\mathrm{B}}T}\right) \tag{5-2}$$

式中：$\mathcal{D}$为状态空间。

假设同一个温度 T，有两个能量状态 E_1 和 E_2，且 $E_1<E_2$，则

$$\begin{aligned}P\{\bar{E}=E_1\}-P\{\bar{E}=E_2\}&=\frac{1}{Z(T)}\exp\left(-\frac{E_1}{k_{\mathrm{B}}T}\right)-\frac{1}{Z(T)}\exp\left(-\frac{E_2}{k_{\mathrm{B}}T}\right)\\&=\frac{1}{Z(T)}\exp\left(-\frac{E_1}{k_{\mathrm{B}}T}\right)\left[1-\exp\left(-\frac{E_2-E_1}{k_{\mathrm{B}}T}\right)\right]\end{aligned} \tag{5-3}$$

由于

$$1-\exp\left(-\frac{E_2-E_1}{k_{\mathrm{B}}T}\right)>0$$

可知

$$P\{\bar{E}=E_1\}-P\{\bar{E}=E_2\}>0$$

可见，在同一温度下分子停留在低能量状态的概率大于停留在高能量状态的概率。由式(5-3)可进一步推断出，随着温度的降低，分子停留在最低能量状态的概率加大。值

得注意的是，这里存在一定随机性，即分子停留在低能量状态的概率会更大一些，并不代表分子一定只向能量低的状态变化。

为了模拟上述固体在恒定温度下达到热平衡的过程，可以用蒙特卡洛方法（计算机随机模拟方法）进行模拟；该方法虽然思路简单，但必须大量采样才能得到比较精确的结果，计算量很大。

为使模拟固体等温过程更加快速可行，Metropolis 于 1953 年提出以概率接受新状态的准则，也称为 Metropolis 准则。其核心思想为对于温度 T，由当前状态 i 转移到新状态 j，若 $E_j \leqslant E_i$，则直接接受状态 j 为当前状态；若概率 $P=\exp\left(-\dfrac{E_j-E_i}{k_B T}\right)$ 大于 $[0,1)$ 区间内的随机数，则接受状态 j 为当前状态，否则维持状态 i 不变。

Metropolis 准则表明：若新状态的能量变小，则必定接受；若新状态能量变大，则以一定概率接受。在相同温度下，状态之间的能量差值越大，由低能量态向高能量态转换成功的概率就越小。若温度很高，则系统以较大概率接受一切状态。若温度接近 0，则系统不接受任何更高能量的状态。可见，温度越高，接受从低能量状态到高能量状态转换的概率就越大。在高温下可接受与当前状态能量差较大的新状态，在低温下只接受与当前状态能量差较小的新状态。

将组合优化问题与物理退火过程相对比，各种状态或参数的对应相似性如表 5.1 所示。

表 5.1　组合优化与物理退火相似性

组合优化问题	物 理 退 火	组合优化问题	物 理 退 火
解	粒子状态	Metropolis 抽样过程	等温过程
最优解	能量最低状态	控制参数的下降	冷却
设定初温	熔解过程	目标函数	能量

因此，Metropolis 抽样准则为计算机模拟物理退火过程提供了可能。由表 5.1 列出的关联关系，可以通过计算机模拟物理退火过程求解复杂的优化问题，这就是模拟退火算法的核心思想。模拟退火算法流程如图 5.4 所示。

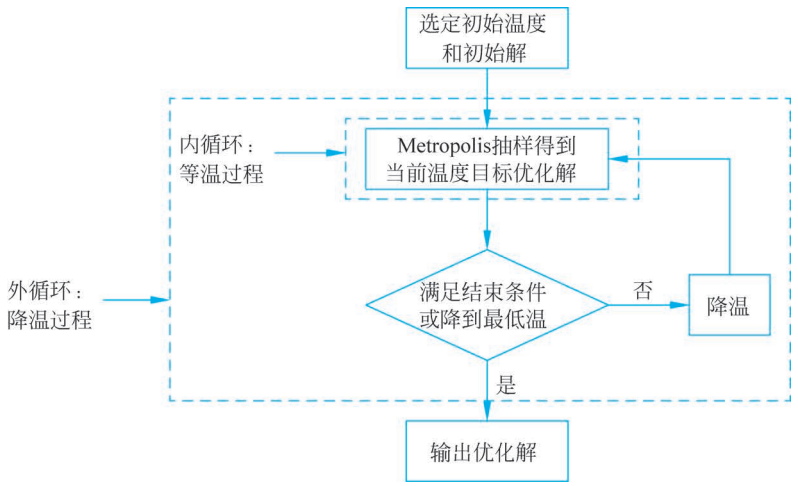


图 5.4　模拟退火算法流程

5.2.2 模拟退火算法的描述

算法 5.1 模拟退火算法。

给定初温 $t=t_0$，随机产生初始状态 $s=s_0$，令 $k=0$；

Repeat //外循环

  Repeat //内循环

    产生新状态 $s_j=\text{Genete}(s)$；

    if $\min\left\{1,\exp\left[-(C(s_j)-C(s))/t_k\right]\right\}>=\text{random}[0,1]$

      $s=s_j$；

  Until 抽样稳定准则满足；

  退温 $t_{k+1}=\text{update}(t_k)$，并令 $k=k+1$；

Until 算法终止准则满足；

输出算法搜索结果。

由算法 5.1 的描述可知，模拟退火算法的关键要素可以概括为“三函数两准则一初温”。

1. 三函数

1）状态产生函数

$$s_j=\text{Genete}(s)$$

原则：设计状态产生函数（邻域函数）的出发点应该是尽可能保证产生的候选解遍布全部的解空间。通常，状态产生函数由两部分组成，即产生候选解的方式和候选解产生的概率分布。

方法：在当前状态下的邻域结构内以一定的概率方式（均匀分布、正态分布、指数分布等）产生。

2）状态接受函数

$$\min\left\{1,\exp\left[-(C(s_j)-C(s))/t_k\right]\right\}>=\text{random}[0,1],\ s=s_j$$

式中：$C(s_j)$为当前解 s_j 的目标函数值。

状态接受函数一般以概率的方式给出，不同的状态接受函数的差别主要是接受概率的形式不同。设计状态接受概率，应该遵循以下原则。

（1）在固定温度下，接受使目标函数下降的候选解的概率大于使目标函数上升的候选解的概率；

（2）随温度的下降，接受目标函数上升的解的概率要逐渐减少；

（3）当温度趋于零时，只能接受目标函数下降的解。

方法：状态接受函数的引入是模拟退火算法实现全局搜索的关键因素。一般而言，状态接受函数只要满足上述三个原则即可，其具体形式对算法影响不大，通常用 $\min\left\{1,\exp\left[-(C(s_j)-C(s))/t_k\right]\right\}$ 作为模拟退火算法的状态接受函数。

3）退温函数

$$t_{k+1}=\text{update}(t_k)$$

即温度的下降方式，用于在外循环中修改温度值。降温方式对算法性能影响很大。降温过快可能丢失最优值点，降温过慢可能导致算法收敛速度极大降低。为保证全局收敛性，一般要求温度下降趋于 0。

常用的退温函数如下。

（1）比例退火方式：$t_{k+1}=\alpha \cdot t_k$，其中，$0<\alpha<1$。α 越接近 1，温度下降越慢。α 的大小可以不断变化。如果 α 大小不变，则比例退火方式的降温过程在高温区速度较快，低温区速度逐渐减慢。

（2）衰减退火方式：$t_{k+1}=t_0(K-k)/K$。其中 K 为温度下降的总次数（预先设定的常数）。

2. 两准则

1）内循环终止准则

内循环终止准则分为以下两种算法。

（1）非时齐算法：每个温度下只产生一个或少量候选解。

（2）时齐算法：常采用 Metropolis 抽样稳定准则，需要等温过程中晶体分子的状态转移达到稳定状态后再进行降温过程。确定在等温状态下系统稳定有如下三种方法。

① 检验目标函数的均值是否稳定（如均值变化小于某一个阈值）。

② 连续若干步的目标值变化较小（如目标值变化小于某一个阈值）。

③ 按一定步数抽样（抽样步数通常较多）。

2）外循环终止准则

外循环终止，代表算法计算结束。常用的外循环终止准则如下。

（1）设置终止温度的阈值（通常要求这个阈值应接近于 0）。

（2）设置外循环迭代次数（迭代达到一定步数，算法自动退出）。

（3）算法搜索到的最优值连续若干步保持不变。

3. 初温

原则上通过理论分析可以得到初温的解析式，但解决实际问题时难以得到精确的初温参数。实际应用中往往需要将初温设置得充分大，实验表明，初温越大，获得高质量解的概率越大，但也会花费较多的计算时间。常用的初温设置方法如下。

（1）均匀抽样一组状态，以各状态目标值的方差为初温。

（2）随机产生一组状态，确定两两状态间的最大目标差值，根据差值，利用一定的函数确定初温。

（3）其他一些经验公式。

5.2.3 模拟退火算法的应用

模拟退火算法的应用很广泛，可以有效求解 NP-Hard 问题。本节将介绍如何用模拟退火算法求解实际问题。

例 5.4 如图 5.5 所示，假设某小区内有 30 个安保巡逻打卡点，其在平面图上对应的坐标如下：

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

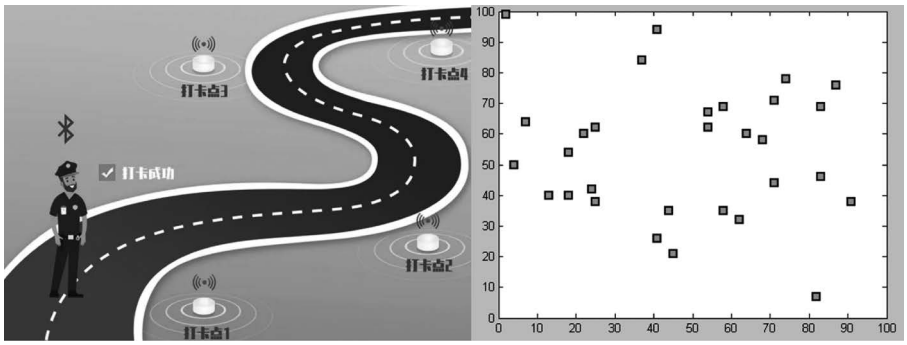


图 5.5 巡逻打卡问题

安保人员需要从第一个打卡点出发巡逻所有打卡点，最终返回第一个打卡点。如何设计保安的巡逻路线，使得巡逻路线总长度最小？

该问题可以抽象为 TSP，其已经被证明是一典型的 NP-难问题。目前尚未发现多项式时间求解方法。尝试用模拟退火算法求解。

1. 问题编码

采用与例 5.3 相同的编码方式，即采用访问打卡点的序号进行问题编码。后续的各种处理都基于该问题编码方式进行。

2. 设计状态产生函数

状态产生函数基于邻域的概念进行设计，设计了三种邻域结构操作，以生成多样化的新状态。具体如下所示。

(1) 互换操作，即随机选定两个打卡点的序号，直接将其交换，如图 5.6 所示。

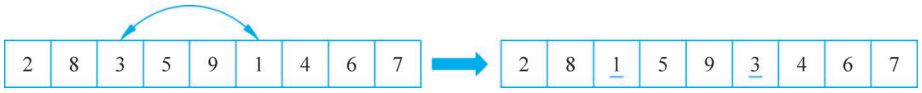


图 5.6 状态产生函数互换操作示意图

(2) 逆序操作，即随机选定连续若干打卡点，并将访问顺序逆转，如图 5.7 所示。

图 5.7 状态产生函数逆序操作示意图

(3) 插入操作，随机选择某个城市序号插入某随机位置，如图 5.8 所示。

3. 设置初始温度

随机地选择一些打卡点，并计算这些点之间的距离，然后用其中的最大值减去最小

图 5.8　状态产生函数插入操作示意图

值，再除以经验值 log0.9，作为初始温度。

4. 其他参数设定

截止温度 $t_f=0.01$；

退温函数采用比例退火方式，退温系数 $\alpha=0.9$；

内循环次数 $L=200\times \text{PointNum}$，其中 PointNum 为打卡点的数量。

采用上述设置后，使用模拟退火算法进行计算，运行结果如图 5.9 所示。

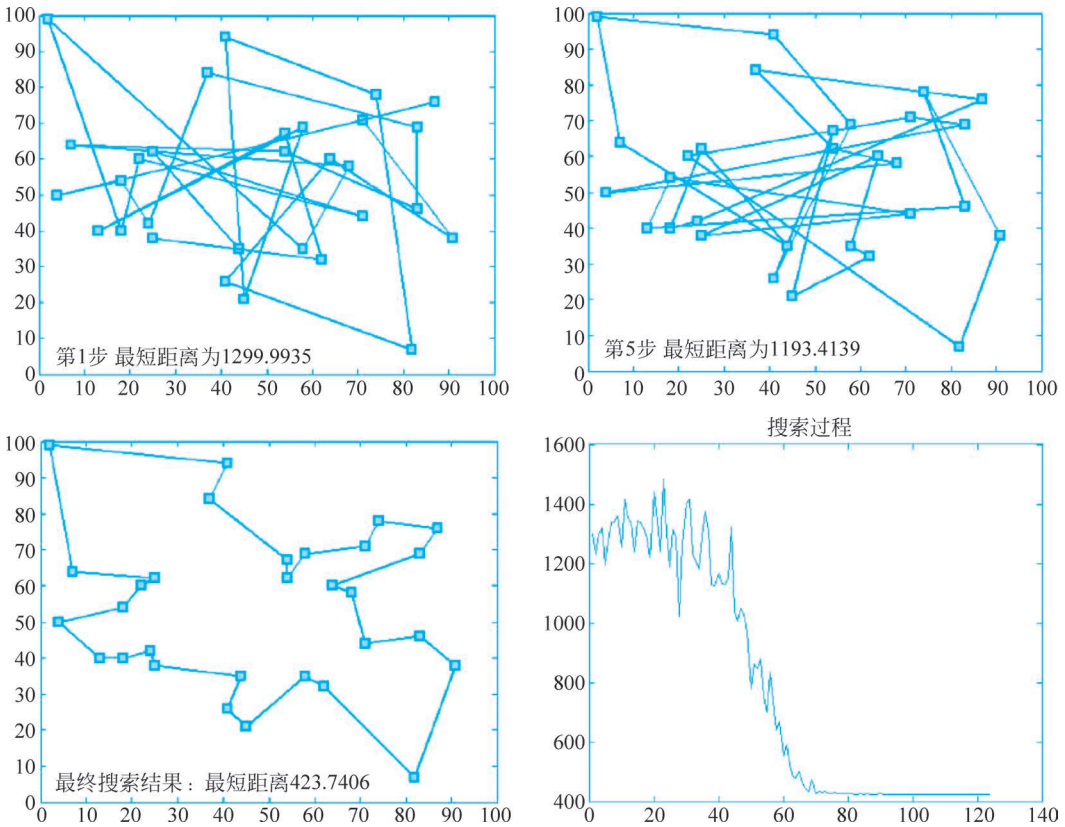


图 5.9　模拟退火算法求解巡逻打卡路径问题的计算结果

5.2.4　模拟退火算法的改进

模拟退火算法具有质量高、初值鲁棒性强、简单、通用、易实现等优点，其不足主要是较高的初始温度、较慢的降温速率、较低的终止温度，以及各温度下足够多次的抽样，因此优化过程较长，耗时较多。针对效率问题，常用的改进方法如下。

(1) 设计合适的状态产生函数。针对问题的特点，设计更加合理高效的状态产生函

数，在更有可能包含最优解的区域产生解。

（2）避免状态的迂回搜索。采用一定的记忆方式，使算法记住已经产生过的解，避免对局部空间反复搜索。

（3）采用并行搜索结构。利用并行计算方式，将搜索空间划分为多个子空间，进行并行搜索，然后汇总处理各子空间的搜索结果。

（4）避免陷入局部极小，改进对温度的控制方式。主要通过引入随机状态，使搜索具有跳出局部空间的能力。

（5）选择合适的初始状态。比如可引入先验信息或其他约束条件，通过控制搜索的起始位置减少搜索空间。

（6）设计合适的算法终止准则。除了以找到目标状态为终止条件外，也可合理设定内循环、外循环的次数，从而控制总的搜索计算量。

（7）增加升温或重升温过程，避免陷入局部极小。从模拟退火算法的原理可知，升温能够使系统处于不稳定状态，从而有利于跳出局部极值。

（8）结合其他智能优化算法或搜索机制的算法。比如将模拟退火算法的结果作为其他搜索算法的初始解。

（9）上述各方法的综合。

5.3 遗传算法

5.3.1 遗传算法的原理

1. 遗传算法的产生及发展历程

在达尔文(Darwin)的进化论和孟德尔(Mendel)的遗传变异理论的基础上，产生了一种在基因和种群层次上模拟自然界生物进化过程与机制的问题求解技术，称为演化计算，遗传算法是其最初形成的一种具有普遍影响的模拟进化优化算法。

20 世纪 50 年代，一些生物学家开始研究使用数字计算机模拟生物的自然遗传与自然进化过程。1963 年，德国柏林技术大学的雷肯伯格(I. Rechenberg)和施韦费尔(H. P. Schwefel)，在做风洞实验时产生了进化策略的初步思想。60 年代，福格尔(L. J. Fogel)在设计有限态自动机时提出进化规划的思想。1966 年福格尔等出版了《基于模拟进化的人工智能》，系统阐述了进化规划的思想。60 年代中期，美国密西根大学的霍兰德(J. H. Holland)教授提出借鉴生物自然遗传的基本原理用于自然和人工系统的自适应行为研究和串编码技术。1967 年，巴格利(J. D. Bagley)首次提出“遗传算法”一词。1975 年，Holland 出版了著名的《自然与人工系统中的适应》，标志遗传算法诞生。70 年代初，Holland 提出了“模式定理”，一般认为是“遗传算法的基本定理”，从而奠定了遗传算法研究的理论基础。1985 年，在美国召开了第一届遗传算法国际会议，并且成立了国际遗传算法学会(International Society of Genetic Algorithms，ISGA)。至今，以遗传算法为代表的演化计算仍然是国际上的研究热点。

2. 遗传算法的思想来源

遗传算法思想源自生物进化理论和遗传学，在计算机处理中借鉴了以下基本知识。

(1) 达尔文自然选择学说。

① 遗传：子代和父代具有相同或相似的性状，保证物种的稳定性。

② 变异：子代与父代，子代不同个体之间总有差异，是生命多样性的根源。

③ 自然选择：自然界中，适应性强的变异个体被保留，适应性弱的变异个体被淘汰，表现出生存斗争和适者生存的现象，称为自然选择。注意，自然选择是长期的、缓慢的、连续的过程。

(2) 遗传学基本概念与术语。

染色体：遗传物质的载体。

脱氧核糖核酸(DNA)：大分子有机聚合物，双螺旋结构。

遗传因子：DNA 或 RNA 长链结构中占有一定位置的基本遗传单位。

基因型：遗传因子组合的模型。

表现型：由染色体决定性状的外部表现。

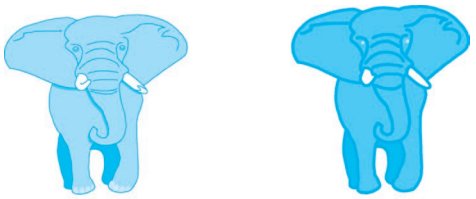


图 5.10　表现型基因对大象外貌的影响

基因型和表现型的例子如图 5.10 所示。假设大象的基因型表示为一串数字，白象的基因型为“1 1 1 1 1 1 1”，黑象的基因型为“1 1 1 0 1 1 1”。因为其基因型不同，个体的表现型不同，表现出白象与黑象两种特性。

基因座：遗传基因在染色体中所占据的位置，同一基因座可能有的全部基因称为等位基因。

个体：染色体带有特征的实体。

种群：个体的集合，该集合内个体数量称为种群的大小。

进化：生物在其延续生存的过程中，逐渐适应其生存环境，使得其品质不断得到改良，这种生命现象称为进化或演化。

适应度：度量某个物种对于生存环境的适应程度。对生存环境适应程度较高的物种将获得更多的繁殖机会，而对生存环境适应程度较低的物种，其繁殖机会就会相对较少，甚至逐渐灭绝。

选择：决定以一定的概率从种群中选择若干个体的操作。

复制：细胞在分裂时，遗传物质 DNA 通过复制而转移到新产生的细胞中，新的细胞就继承了旧细胞的基因。

交叉：在两个染色体的某一相同位置处 DNA 被切断，其前后两串分别交叉组合形成两个新的染色体。又称基因重组，俗称“杂交”。

变异：在细胞进行复制时可能以很小的概率产生某些复制差错，使 DNA 发生某种变异，产生出新的染色体，这些新的染色体表现出新的性状。

编码：表现型到基因型的映射。

解码：从基因型到表现型的映射。

1930—1947 年，达尔文进化论与遗传学走向融合，多布然斯基(Th. Dobzhansky)于

1937 年发表的《遗传学与物种起源》成为融合进化论与遗传学的代表作。

生物物种作为复杂系统，具有奇妙的自适应、自组织和自优化能力，这是生物在进化过程中体现的一种智能，也是人工系统梦寐以求的功能。

我们考虑这样一个场景。一片山地包含若干山峰，一群兔子立志找到最高峰。它们采用的方法是，首先随机选择一个地点，然后兔子之间交配生下子代兔子，子代兔子的位置为其父代的两个个体的中间。接下来，兔子可随机往某个方向随机移动一段距离。我们规定，位置海拔更高的兔子对环境的适应能力更强，其有更多的觅食和繁殖机会，被天敌吃掉的概率也更低。于是兔子在繁衍若干代之后，大量的兔子会集中在海拔更高的位置，直到找到最高峰。遗传算法正是用计算机模拟这个过程找到问题优化解。

5.3.2 遗传算法的实现

遗传算法是模拟自然界生物进化过程与机制的问题求解技术，基本遗传算法的流程（图 5.11）如下：

Step1：选择合适的编码形式，对待考查的个体进行编码；

Step2：初始化种群，设置算法运行参数；

Step3：评估种群中个体的适应度；

Step4：以适应度为依据，从种群中选择两个个体作为待交叉的父代个体；

Step5：将两个父代个体的染色体进行交叉，产生两个子代个体；

Step6：对子代的染色体进行变异；

Step7：转 Step4，直到子代种群产生；

Step8：如果满足算法结束条件，则算法退出，输出解；

Step9：转 Step3。

按照图 5.11 中基本遗传算法的流程，算法具有编码、选择、交叉、变异等关键操作。

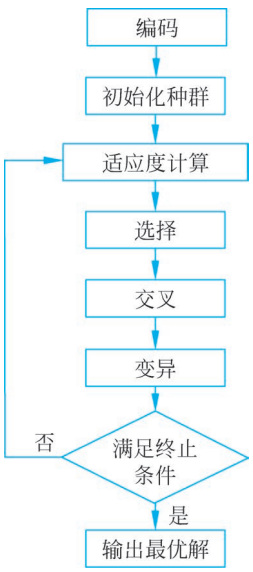


图 5.11 基本遗传算法流程

1. 编码

由设计空间向编码空间的映射，对应表现型到基因型的映射。编码的选择是影响算法性能和效率的重要因素。常用的一种编码方式是二进制编码，如数据对“(1,2)”的二进制编码可表示为“00010010”，其中编码的前半部分“0001”是十进制数字“1”的二进制编码，后半部分“0010”是十进制数字“2”的二进制编码。与编码对应的是解码，即由编码空间向设计空间的映射，对应基因型到表现型的映射。其可看作编码操作的逆映射。

在遗传算法操作中，种群中个体都以编码方式存在，后续的遗传操作都作用在编码后的个体上。

1）编码原则。

健壮的编码通常符合如下原则。

(1) 完备性：问题空间的所有解都能表示为所设计的基因型。

(2) 健全性：任何一个基因型都对应于一个可能解。

(3) 非冗余性：问题空间和表达空间一一对应。

2) 多种编码方式。

编码的方式多种多样，但通常有如下编码方式。

(1) 二进制编码：用一串“0”或“1”组成的二进制数进行编码，其编码、解码操作简单易行，交叉、变异等遗传操作便于实现。因为其变异后可能导致表现型变化很大，不连续，所以可能会远离最优解，达不到稳定。二进制编码也容易出现海明悬崖问题。

(2) 浮点数编码：又称为实数编码，其基于多个浮点数进行问题编码。

(3) 格雷码编码：两个相邻的数用格雷码表示，其对应的码位只有一个不同，可以提高算法的局部搜索能力。这是格雷码相比二进制码而言所具备的优势。

(4) 符号编码：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，城市编号的排列可构成一个表示旅行路线的个体。

(5) 复数编码：与实数编码相对，其用复数对个体进行编码。

2. 适应度函数

个体的适应度是个体在种群生存的优势程度度量，用于区分个体的“好与坏”，其具体数值使用适应度函数进行量化计算，也称为评价函数。适应度函数的设计原则如下：

(1) 单值，连续，非负，最大化；

(2) 合理，一致性；

(3) 计算量小；

(4) 通用性强。

适应度函数的选取直接影响遗传算法的收敛速度以及能否找到最优解。适应度函数设计不当有可能出现欺骗问题。比如，在进化初期个别超常个体可能会控制选择过程，在进化末期个体适应度差异太小，导致算法陷入局部极值。

一般而言，适应度函数是由目标函数变换而成的，对目标函数值域的某种映射变换称为适应度的尺度变换。常用的三种适应度函数的构造方式如下：

1) 直接转换法

目标函数为最大化问题：

$$\mathrm{Fit}(f(x)) = f(x) \tag{5-4}$$

目标函数为最小化问题：

$$\mathrm{Fit}(f(x)) = -f(x) \tag{5-5}$$

2) 截断式界限构造法

目标函数为最大化问题：

$$\mathrm{Fit}(f(x)) = \begin{cases} f(x) - c_{\min}, & f(x) > c_{\min} \\ 0, & \text{其他} \end{cases} \tag{5-6}$$

式中：$c_{\min}$ 为 $f(x)$的最小估计值。

目标函数为最小化问题：

$$\mathrm{Fit}(f(x))=\begin{cases}c_{\max}-f(x), & f(x)<c_{\max}\\ 0, & \text{其他}\end{cases} \tag{5-7}$$

式中：$c_{\max}$ 为 $f(x)$的最大估计值。

截断式界限构造法的作用是将部分目标函数评价不好的个体的适应度赋值为 0，即让其不可能通过选择、交叉、变异等遗传操作进入子代种群，也就是直接丢弃这部分适应度不高的个体。

3）比例式界限构造法

目标函数为最大化问题：

$$\mathrm{Fit}(f(x))=\frac{1}{1+c-f(x)},\quad c\geqslant 0, c-f(x)\geqslant 0 \tag{5-8}$$

式中：c 为目标函数 $f(x)$的保守估计值。

目标函数为最小化问题：

$$\mathrm{Fit}(f(x))=\frac{1}{1+c+f(x)},\quad c\geqslant 0, c+f(x)\geqslant 0 \tag{5-9}$$

式中：c 为目标函数 $f(x)$的保守估计值。

比例式界限构造法能够有效防止种群中的超常个体控制进化过程的问题。

3. 选择

遗传算法中的选择操作是用来确定如何从父代种群中按某种方法选取适应度较高的个体，以便将优良基因片段遗传到下一代种群。选择操作用来确定参与交叉操作的个体，以及被选个体将产生多少个子代个体。

1）个体选择概率计算

（1）按比例的适应度分配方法：假设种群中包含 n 个个体，某个个体 i 的适应度为 f_i，则其被选中的概率为

$$P_i=\frac{f_i}{\sum_{i=1}^{n}f_i} \tag{5-10}$$

由式(5-10)可知，在按比例的适应度分配方法中，当前个体 i 被选中的概率是其适应度与种群中所有个体适应度之和的比。

注意，在按比例的适应度分配方法中，个体适应度大小与其被选中的概率之间存在明确的比例关系，关联度较强，种群中个体适应度越高，则其基因被保留在种群中的概率也越大。其缺点是，如果种群中存在超强个体（适应度比其他个体高很多的个体），则易导致子代个体绝大部分来自该超强个体，使种群多样性丧失，也就是生物学中的“近亲繁殖”。这样容易导致算法早熟收敛。为避免此情况，也可以使用其他适应度分配方法，如下面要介绍的基于排序的策略。

（2）基于排序的适应度分配方法：在基于排序的适应度分配中，首先对种群中所有

个体按适应度进行排序，然后按排名的高低直接为每个个体赋予选择概率。常用的方法有线性排序和非线性排序。

线性排序：

$$P_i = \frac{1}{\mu}\left[\eta_{\max} - (\eta_{\max} - \eta_{\min}) \cdot \frac{i-1}{\mu - 1}\right] \tag{5-11}$$

式中：$1 \leqslant \eta_{\max} \leqslant 2$，$\eta_{\min} = 2 - \eta_{\max}$，是预先设定的超参数；$i$ 为排序后的个体序号；μ 为种群中个体的总数。

非线性排序：

$$P_i = c(1-c)^{i-1} \tag{5-12}$$

式中：i 为排序后的个体序号；c 为排序第一的个体的选择概率，是预先设定的超参数。

在基于排序的适应度分配方法中，个体选择概率与适应度的数值大小没有直接关系，其仅和个体适应度排序情况相关。因此，该方法能够较好地限制超常个体的选择概率，从而保证了种群的多样性。但正因为个体选择概率与个体适应度数值大小的联系不够直接，也容易导致种群中会丢失某些优质个体的基因，使算法收敛过程变得缓慢。

2）选择算子

（1）轮盘赌选择：假想有一个转盘如图 5.12 所示，个体的适应度大小代表了扇形区域的大小。每一次选择过程就是给转盘一个推力将转盘旋转起来，当转盘自然停下时，固定指针指向哪个扇形区域，该扇形区域代表的个体被选中。显然，个体适应度越大，其扇形面积越大，指针落到其扇形区域内的可能性也越大。因此，在轮盘赌算子中，个体被选中的概率与其适应度大小成正比。

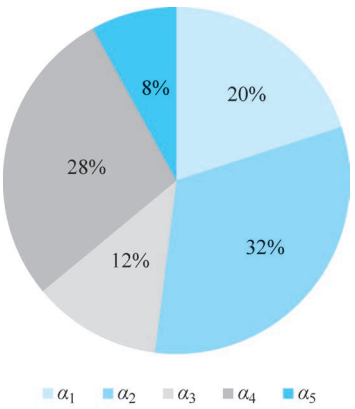


图 5.12　轮盘赌算子示例

在具体实现中，可以通过计算机采用如下方式模拟轮盘赌算子的过程：第一步，计算每个个体的累积概率。累积概率的计算方式如表 5.2 所示。第一个个体的累积概率等于其选择概率，从第二个个体开始，其累积概率等于其选择概率加上前一个个体的选择概率。第二步，生成一个在[0,1]区间内的随机数，若该随机数小于或等于某个个体的累积概率，但大于其前一个个体的累积概率，则表示该个体被选择算子所选中。第三步，重复第二步，直到选择的个体数量达到要求。

表 5.2　累积概率计算示例

个体编号	1	2	3	4	5	6	7	8	9	10
适应度	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2
选择概率	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02
累积概率	0.18	0.34	0.49	0.62	0.73	0.82	0.89	0.95	0.98	1.00

(2) 随机遍历抽样：随机遍历抽样算子的第一步和第二步操作与轮盘赌选择算子相同，也是先计算各个个体的选择概率和累积概率，并随机生成一个在 [0,1] 区间内的随机数 p_0，若该随机数小于或等于某个个体的累积概率，但大于其前一个个体的累积概率，则表示该个体被选择算子所选中。随机遍历抽样算子与轮盘赌选择算子的区别在第三步。在第三步中，随机遍历抽样算子随机生成一个在 [0,1] 区间内的随机数 b 作为偏移量，则计算 $p_i=\{p_0+i\cdot b\}$，其中，$i=1,2,3,\cdots$，算子 $\{\cdot\}$ 表示取小数部分。然后，根据 p_i 落在累积概率的区间确定后续被选中的个体，直到选择的个体数量达到要求。如表 5.2 所示的例子中，其随机遍历抽样结果如图 5.13 所示。

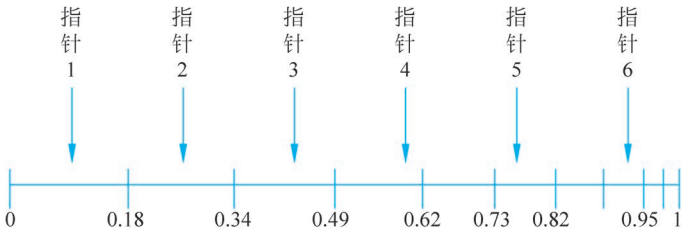


图 5.13　随机遍历抽样算法

在图 5.14 中，选中的个体序号分别为 1、2、3、4、6、8。

(3) 锦标赛选择：从种群中随机采样 s 个个体(注意采样是有放回的)，然后选择最优的个体进入下一代，只有个体的适应度值优于其他 $s-1$ 个竞争者时才能赢得锦标赛。其中，$2\leqslant s<M$，M 是种群中个体的总数量。显然，s 越大，选择强度越强，更优秀的个体容易被选出；但 s 过大容易破坏种群的多样性。值得注意的是，在锦标赛选择算子中，最差的个体肯定不会幸存，而最优的个体在其参与的所有锦标赛中都能获胜。

(4) 截断选择：根据适应度值对种群中的个体按照从优到劣的顺序进行排序，只有前 n 个最好的个体被选择进入下一代。截断选择是一种非常基础的选择算法，它的优势是能够快速地在大量种群中选择个体。截断选择是在动植物育种方面的标准方法，按照表现型价值排序，只有最好的动植物才会被选择繁殖。

4. 交叉

交叉算子是指把两个父代个体的部分结构加以替换、重组而生成新个体的操作。其目的是在编码空间进行有效搜索，同时降低对有效模式的破坏概率。习惯上对实数编码的操作称为重组，对二进制编码的操作称为交叉。交叉操作是遗传算法区别于其他进化算法的重要特征，在遗传算法的运算中起核心作用。

1) 实值重组

(1) 离散重组：子个体的每个基因可以按等概率随机地挑选父个体中的基因。离散重组的操作如图 5.14 所示。

父个体1　12　25　5
父个体2　123　4　34

子个体1　123　4　5
子个体2　12　4　34

图 5.14　离散重组操作

(2) 中间重组：中间重组操作中，子代个体的基因由父代个体基因通过计算得到，计算公式为

$$\text{子个体}=\text{父个体 1}+\alpha(\text{父个体 2}-\text{父个体 1}) \tag{5-13}$$

式中：α 为比例因子，由 $[-d,1+d]$ 上均匀分布的随机数产

生。当 $d=0$ 时，为中间重组，一般取 $d=0.25$。子代编码的每个基因均产生一个 α。如图 5.15 所示，为计算子个体 1 的第一个编码，首先随机生成 $\alpha=0.5$，然后根据父个体 1 和父个体 2 对应位置上的编码，采用式(5-13)可计算出子个体 1 的第一个编码为 67.5。

(3) 线性重组：线性重组和中间重组的操作方式类似，其区别是中间重组中针对子代编码的每个基因均产生一个 α，在线性重组中某子代个体的所有基因均用同一个 α，如图 5.16 所示。

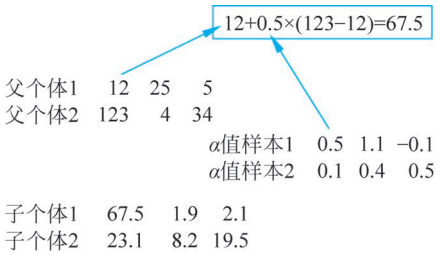


图 5.15 中间重组操作示意图

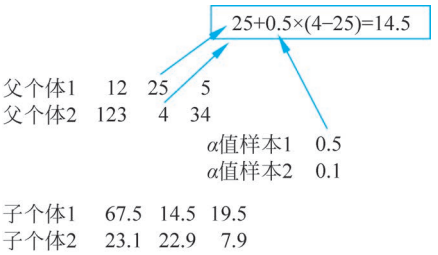


图 5.16 线性重组操作

2) 二进制交叉

(1) 单点交叉：首先在个体编码串中只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染色体，如图 5.17 所示。

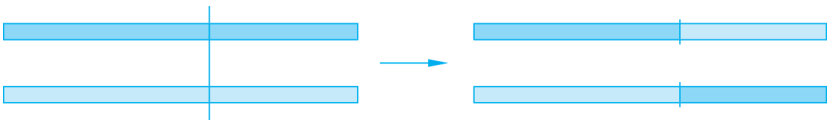


图 5.17 单点交叉操作示意图

(2) 多点交叉：与单点交叉不同，多点交叉首先会随机生成两个以上的交叉点，然后在多个交叉点相互交换两个配对个体的部分染色体，如图 5.18 所示。

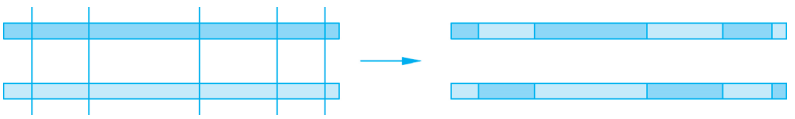


图 5.18 多点交叉操作示意图(5 个交叉点)

5. 变异

变异运算是指按一定概率将个体染色体编码串中的某些基因座上的基因值用该基因座上的其他等位基因来替换，从而形成新的个体的过程。因此，变异是按照一定的概率随机改变某个个体遗传信息的过程，变异使遗传算法具有局部的随机搜索能力，并使遗传算法可维持群体多样性，防止出现未成熟收敛现象。此外，研究发现，当遗传算法通过交叉算子已接近最优解邻域时，利用变异算子的这种局部随机搜索能力可以加速向最优解收敛。

变异操作的基本步骤：第一步，对种群中每个个体以事先设定的变异概率 p_m 判断是否进行变异；第二步，对需要变异的个体随机选择基因位进行变异。

针对不同的编码方式，通常采用不同的变异方法，下面将详细介绍。

1）面向二进制编码的变异方法

面向二进制编码的变异方法是改变染色体的某一个位点上基因，使这个基因变成它的等位基因，并且通常会引起一定的表现型变化。二进制编码的遗传操作过程和生物学中的过程非常类似，基因串上的“0”或“1”有一定概率变成与之相反的“1”或“0”。

常用的二进制编码变异操作是对个体编码串中以变异概率、随机指定的某一位或某几位基因座上的值做反转运算（“0”变为“1”或“1”变为“0”），称为随机反转变异算子，如图 5.19 所示。

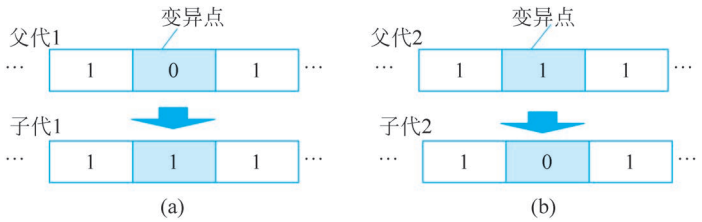


图 5.19　二进制编码变异操作

2）面向浮点型编码的变异方法

面向浮点型编码的变异方法一般是对原来的浮点数增加或者减少一个小随机数。比如，原来的浮点数串为 1.2，3.4，5.1，6.0，4.5，变异后得到浮点数串为 1.2，**3.5**，5.1，**5.9**，4.5。这个小随机数也有大小之分，称为“步长”。一般而言，步长越大，开始时进化的速度会比较快，但在进化过程末期比较难收敛到精确的点上；小步长与之相反。

值得注意的是，变异只能在少数基因位上进行，且变异的幅度通常不宜太大。变异过于剧烈，很难保证种群的遗传稳定性，从而丢失进化过程中积累得到的优良基因片段，导致遗传算法退化为随机搜索。

5.3.3　遗传算法的应用

遗传算法提供了一种求解复杂系统问题的通用框架，它不依赖问题的具体领域，且有很强的鲁棒性，其广泛应用于许多领域，典型的应用包括函数优化问题求解和组合优化问题求解两方面。

1. 函数优化

函数优化是遗传算法应用的经典领域，同时也是对基于演化计算的各种算法进行性能评价的常用算例，学者们构造出了各种各样复杂形式的测试函数，包括连续函数和离散函数、凸函数和非凸函数、低维函数和高维函数、单峰函数和多峰函数等。对于非线性、单峰、多目标等较难的函数优化问题，遗传算法能够获得较好的结果。

例 5.5　求一元函数 $f(x)=x\sin(10\pi x)+2.0(x\in[-1,2])$ 的最大值。

解：$f(x)$ 的函数图像如图 5.20 所示。这是一个复杂的多峰函数，在整个定义域内会出现多个极值。当 x 在 1.8～1.9 时，函数取到最大值。采用数学方法对该问题求解难

度较大，需要求解三角方程且在多个极值点上逐个判断。这里通过遗传算法求解。

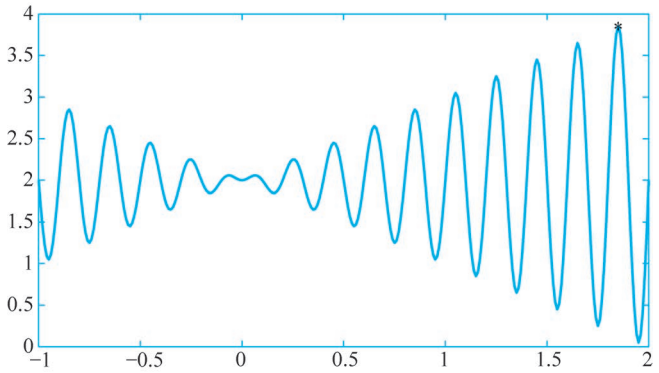


图 5.20　一元函数 $f(x)=x\sin(10\pi x)+2.0(x\in[-1,2])$图像

使用遗传算法求解过程如下。

Step1：编码。经过分析发现，本问题的表现型是自变量 $x(x\in[-1,2])$，将 x 的二进制形式作为其基因型，采用二进制编码。

注意，编码的长度取决于求解精度要求。若求解精度到 6 位小数，区间长度为 $2-(-1)=3$，即需将区间分为 $3/0.000001=3\times10^6$ 等份。因为 $2097152=2^{21}<3000000<2^{22}=4194304$，所以编码的二进制串长应为 22 位。

Step2：生成初始种群。首先需要设定种群的规模，即每一代的种群包含多少个个体。种群规模是遗传算法的超参数，事前必须设定，如可设定为 30、50 等。种群的生成方式可采用完全随机的方式生成个体，即生成多个长度为 22 的二进制串。

Step3：计算适应度。采用 $f(x)$作为适应度函数。注意，需要将个体的二进制编码形式映射为$[-1,2]$之间的实数，然后代入 $f(x)$计算其适应值。采用的方法如下。

(1) 将一个二进制串$(b_{21}b_{20}\cdots b_0)$转换为十进制数：

$$(b_{21}b_{20}\cdots b_0)_2=\left(\sum_{i=0}^{21}b_i2^i\right)_{10}=x' \tag{5-14}$$

(2) 将 x'对应为区间$[-1,2]$内的实数：

$$x=-1.0+x'\frac{2-(-1)}{2^{22}-1} \tag{5-15}$$

Step4：设计选择算子，选用轮盘赌选择算子。

Step5：设计交叉算子，选用二进制单点交叉算子。

Step6：设计变异算子，选用二进制随机反转变异算子。

算法超参数设置：种群大小 50，交叉概率 0.75，变异概率 0.05，最大代数 200(算法退出条件)。

遗传算法求解函数最大值如图 5.21 所示，求解函数最大值问题最优解变化过程如表 5.3 所示。

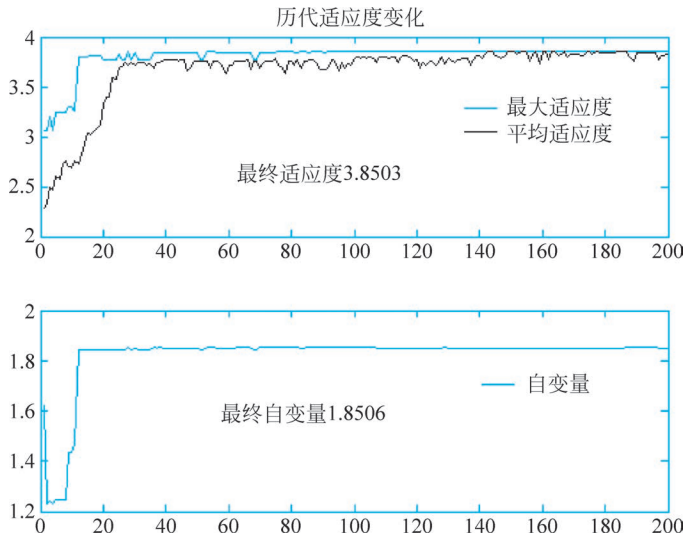


图 5.21　遗传算法求解函数最大值

表 5.3　求解函数最大值问题最优解变化过程

迭　代　数	自　变　量	适　应　度
1	1.4495	3.4494
9	1.8395	3.7412
17	1.8512	3.8499
30	1.8505	3.8503
50	1.8506	3.8503
80	1.8506	3.8503
120	1.8506	3.8503
200	1.8506	3.8503

最终得到最佳个体：

$S_{max}=<11110011001110111111100>$

$X_{max}=1.8506$

$f(X_{max})=3.8503$

值得注意的是，使用遗传算法求解函数优化问题不需要函数可导，甚至不要求函数连续。这是基于求导的数学方法所不具备的优势。

2. 组合优化

在有限个可行解的集合中找出最优解的一类优化问题称为组合优化问题。实践证明，遗传算法对于组合优化中的 NP-Hard 问题非常有效，在 0-1 背包问题、旅行商问题、装箱问题及图形划分等问题上已经成功应用了遗传算法。典型的组合优化问题可分为无约束组合优化问题和带约束组合优化问题。下面将分别介绍使用遗传算法对其求解的方法。

1）无约束组合优化问题求解

典型的无约束组合优化问题是旅行商问题，其只有最小化（或最大化）优化目标的要求，而对解本身没有约束。

本节使用遗传算法求解例 5.4 的 30 个巡逻点的巡逻打卡问题。前面已经分析，30 个巡逻点的巡逻打卡问题是典型的旅行商问题。

Step1：编码。直接采用解的表示形式，30 位（30 个打卡点）长，每位代表所经过的打卡点序号（无重复）。

Step2：生成初始种群。依然采用随机方式，生成整个种群。生成每个个体时，访问打卡点的顺序随机确定。

Step3：设计适应度函数。由于旅行商问题的优化目标是最小化路径距离，可以选择路径距离的倒数作为个体的适应度。

Step4：设计选择算子。采用轮盘赌方法。

Step5：交叉算子。交叉算子设计为随机有序交换，具体操作：随机选取两个交叉点；两个父个体交换中间部分；替换交换后重复的打卡点序号。

Step6：设计变异算子。随机选择同一个个体的两个点进行交换。

算法超参数设置：种群规模 100；交叉概率 0.8；变异概率 0.2；终止代数 2000。

采用上述设置后，使用遗传算法进行计算，运行结果如图 5.22 所示。

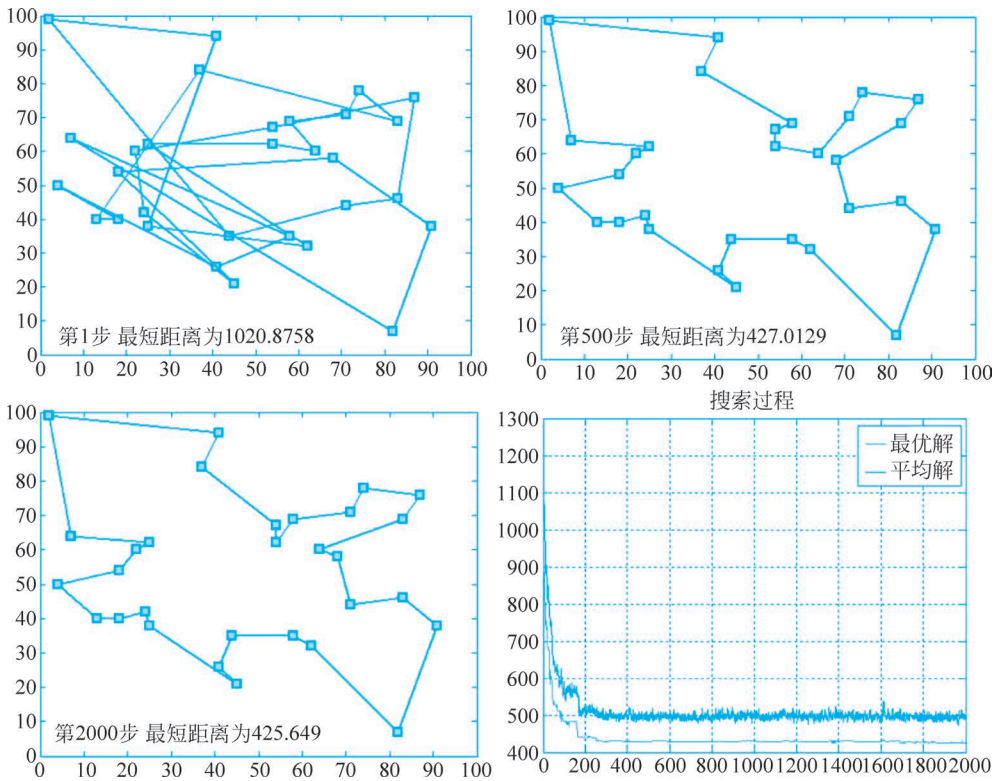


图 5.22　遗传算法求解巡逻打卡路径问题

这里通过遗传算法找到最优解为 425.649，比例 5.4 中用模拟退火算法找到的最优解 423.7406 差一些。需要说明的是，智能优化算法不一定能找到最优解，其目标是寻找足够好的次优解或用户满意解，且两次运行同一遗传算法所得到的结果也不一定相同。

2）带约束的组合优化问题求解

带约束的组合优化问题又称为约束最优化问题，其一般形式如下：

$$\begin{cases}\text{minimize } f(x)\\ \text{s.t. } g_i(x)\leqslant 0(i=1,2,\cdots,p)\\ \quad\quad h_j(x)=0(i=1,2,\cdots,q)\end{cases}\tag{5-16}$$

式中：$f(x)$是优化目标，不等式 $g_i(x)$和等式 $h_j(x)$均为约束。

从式(5-16)可知，约束最优化问题不仅要求目标函数达到最小，而且解 x 需要满足不等式 $g_i(x)$和等式 $h_j(x)$的约束。

直接按无约束问题求解的方式来处理约束最优化问题是行不通的，主要有两个原因：一是随机生成的初始种群中可能有大量不可行解；二是交叉、变异等遗传算子作用于可行解后也可能产生不可行解。

使用遗传算法解决约束最优化问题的关键是对约束的有效处理，主要包括以下两种方法。

(1) 罚函数法。

罚函数法的核心思想是将有约束问题转化为无约束问题。其构造惩罚项，并包含到适应度评价中，通常有加法和乘法两种形式。

罚函数法的加法形式：

$$\text{fitness}(x)=f(x)+rP(x),\quad \begin{cases}P(x)=0, & x\in X\\ P(x)>0, & x\notin X\end{cases},\quad r<0\tag{5-17}$$

式中：x 为遗传算法中个体；X 为可行解域；fitness(x)为个体 x 的适应度评价函数；$P(x)$为罚函数；r 为惩罚系数。

在式(5-17)中，$rP(x)$就是惩罚项。如果个体没有违反任何约束($x\in X$)，则惩罚项为 0；否则，惩罚项的存在会导致个体 x 的适应度降低。

罚函数法的乘法形式：

$$\text{fitness}(x)=f(x)\cdot P(x),\quad \begin{cases}P(x)=1, & x\in X\\ P(x)<1, & x\notin X\end{cases}\tag{5-18}$$

式中：x 为遗传算法中个体；X 为可行解域；fitness(x)为个体 x 的适应度评价函数；$P(x)$为罚函数。当个体 x 违反约束时，$P(x)<1$，也会导致个体 x 的适应度降低。

对简单约束问题采用定量惩罚，如只要违反约束，就适应度降低相同的数量；对复杂约束问题可采用变量惩罚。变量惩罚的罚函数由可变惩罚率和违反约束的惩罚量组成。

可变惩罚率是指个体违反约束程度不同或算法进化迭代次数的不同。调整罚函数的惩罚系数 r 有如下的处理方式。

① 基于解违反约束程度：随着违反约束程度变得严重而增加惩罚压力，称为静态惩罚。

② 基于算法的进化迭代次数：随着进化过程的进展而增加惩罚压力，称为动态惩罚。

违反约束的惩罚量是指在惩罚系数不变的情况下，合理设计罚函数 $P(x)$，使个体违反约束的程度越大，则 $P(x)$ 越大。

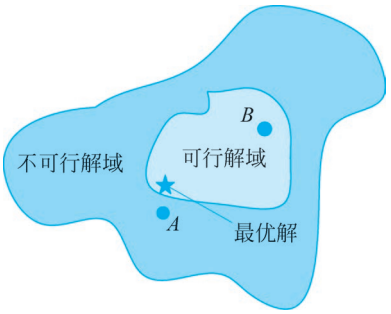


图 5.23　可行解域与不可行解域示意图及最优解分布位置

问题的解空间可以划分为可行解域和不可行解域，如图 5.23 所示。罚函数法的优势在于能够在可行解域和不可行解域搜索最优解。大量研究表明，最优解总是出现在可行解域与不可行解域的边界上。在遗传算法的进化过程中，一个不可行解（图 5.23 中 A 点）包含的染色体片段信息可能比可行解（图 5.23 中 B 点）中的染色体片段信息更接近最优解。

注意，使用罚函数法需要谨慎地在过轻或过重惩罚之间找到平衡。惩罚力度过大可能造成搜索只停留在可行解域内，且容易早熟收敛；惩罚力度过小可能造成种群中存在大量毫无意义的不可行解，搜索过程在无意义区域浪费大量时间，也难以找到最优解。

（2）解修正法。

解修正方法的思想是将不可行解转化为可行解，需要按照相关领域知识设计解修正算法，并且解修正算法的效果和效率对问题求解效果影响较大。在约束修正算法设计合理、修正算法运行代价较小的情况下，解修正方法相较于罚函数方法通常能取得更好的结果。

例如，使用遗传算法求解例 5.1 中的 0-1 背包问题。采用罚函数法，可设计个体的适应度函数等于装入物品价值减去超重部分物品价值乘以惩罚系数；采用解修正法，可按照性价比（或重量、价值、随机等）原则去除超重的背包内物品，直至背包不再超重。

5.3.4　遗传算法的改进

遗传算法的改进方法一直是研究的热点，目前已有大量成果。这里介绍精英解保持、自适应遗传算法和基于小生境技术的遗传算法三种常用的改进方法。

1. 精英解保持

遗传算法中的基因并不一定真实地反映待求解问题的本质，因此各个基因之间未必就相互独立。如果只是简单地进行杂交，很可能破坏了较好的基因片段，这样就没有达到通过进化过程累积优良基因的目的。精英保留策略可以避免杂交操作破坏最优个体。此外，轮盘赌等选择算子也可能以较小概率丢失种群中最优解。为了防止当前群体的最优个体在下一代发生丢失，导致遗传算法不能收敛到全局最优解，De Jong 提出了“精英选择”策略，也称为“精英保留”策略。该策略的思想是把群体在进化过程中迄今出现的最好个体（称为精英个体 Elitist）不进行配对交叉而直接复制到下一代中。

精英个体是种群进化到当前为止遗传算法搜索到的适应度最高的个体，它具有最好

的基因结构和优良特性。采用精英保留的优点是遗传算法在进化过程中出现的最优个体不会被选择、交叉和变异操作丢失和破坏。精英保留策略对改进标准遗传算法的全局收敛能力产生了重大作用，学者们已经从理论上证明了具有精英保留的标准遗传算法依概率1收敛到全局最优解。

2. 自适应遗传算法

交叉概率 P_c 和变异概率 P_m 的选择是影响遗传算法行为和性能的关键，直接关系到算法的收敛性。P_c 越大，新个体产生的速度就越快，但过大会使优秀个体的结构很快被破坏；P_c 过小，搜索过程缓慢，以至停滞不前。P_m 过大，变成纯粹的随机搜索；P_m 过小，不易产生新个体结构。

显然，当种群中各个体适应度趋于一致或趋于局部最优时，应使 P_c 和 P_m 增加；而当种群中个体适应度比较分散时，应使 P_c 和 P_m 减少。同时，适应度较高的个体，应设定较低的 P_c 和 P_m；适应度较低的个体，应设定较高的 P_c 和 P_m。

因此，可以按照一定的规则在遗传算法运行过程中动态地调整 P_c 和 P_m，使得算法具有自适应能力。

3. 基于小生境技术的遗传算法

小生境(niche)是生物学中的概念，表示特定环境中的一种组织功能；在自然界中，往往特征、性状相似的物种相聚在一起，并在同类中繁衍后代。在标准遗传算法中，容易“近亲繁殖”，即两个有类似基因的个体交叉产生的子代个体与双亲极为接近，破坏了种群中的潜在多样性。于是，学者提出了小生境遗传算法(Niche Generic Algorithm，NGA)，将种群中个体划分为若干类，这些类内个体的基因相似度相对较高，类间个体的基因相似度相对较低。选择操作时，从每类选出优秀个体进行交叉、变异操作，或组成新的种群。

基于小生境技术的遗传算法能够保持解的多样性，提高全局搜索能力，比较适合复杂多峰函数的优化。

5.4 其他典型智能优化算法简介

除了前面介绍的模拟退火算法和遗传算法之外，还有很多其他的智能优化算法。在自然界中各种生物群体显现出来的智能近几十年来得到了广泛关注，学者通过对简单生物体的群体行为进行模拟提出了群智能算法。其中，模拟蚁群觅食过程的蚁群优化算法(Ant Colony Optimization，ACO)和模拟鸟群运动方式的粒子群算法(Particle Swarm Optimization，PSO)是两种主要的群智能算法，本节将分别进行介绍。

5.4.1 蚁群优化算法

蚁群优化算法是一种源于大自然生物世界的新的仿生进化算法，是由意大利学者多里戈(M. Dorigo)和马聂佐(V. Maniezzo)等于20世纪90年代初期通过模拟自然界中蚂蚁集体寻径行为而提出的一种基于种群的启发式随机搜索算法。蚂蚁有能力在没有任

何提示的情形下找到从巢穴到食物源的最短路径，并且能随环境的变化，适应性地搜索新的路径，产生新的选择。目前，蚁群优化算法已被广泛应用于求解优化问题。

1. 算法原理

蚁群优化算法又称为人工蚁群优化算法，是模拟真实蚂蚁群行为而设计的求解优化问题的一种仿生算法。

通过对蚁群觅食行为的观察发现了如下现象。

(1) 蚁群觅食是群体行为，单只蚂蚁的觅食路径(从巢穴到达食物源的路径)并不一定是最优的，但通过群体协作总可找到最优路径。

(2) 蚁群觅食开始选择路径是随机的，但随着蚂蚁搬运食物来回于巢穴与食物源之间，路径的选择会越来越趋于有组织，当然也有少数蚂蚁有随机行走的现象。

进一步研究发现，蚂蚁会在其经过的路径上释放“信息素”，蚁群内的蚂蚁对“信息素”具有感知能力，它们会倾向于沿着“信息素”浓度较高路径行走，而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，这样经过一段时间后，整个蚁群就会沿着最短路径到达食物源。

受到真实蚁群觅食的启发，学者们设想了人工蚁群系统。人工蚁群系统具有如下特点。

(1) 每一只人工蚂蚁(简称蚂蚁)在行动路径上均释放信息素。

(2) 当“蚂蚁”碰到还没走过的路口，就随机挑选一条路走。“信息素”随时间挥发，对于某一条路径而言，“信息素”浓度与路径长度成反比。

(3) 其他觅食的蚂蚁根据不同路径上的“信息素”的浓度来选择路径，浓度越大，越有可能沿着其方向行进。

(4) 当更多的蚂蚁选择某条路径时，这条路径因为聚集了越来越多的“信息素”而变得更有吸引力，进而吸引更多的蚂蚁走这条路。

(5) 这种作用形成一种正反馈机制，使最优觅食路径被越来越多的蚂蚁选择，最优路径上的“信息素”浓度越来越大。

(6) 最终蚁群找到最优寻食路径。

在蚁群觅食过程中有以下两个重要的机制相互作用推进了这种最优行为。

(1) 蚂蚁在行进路径上不断累积信息素，使得不同路径变得不一样。

(2) 路径上信息素的不断累积改变了蚂蚁选择路径的概率。

从而形成一种蚂蚁行为不断改变所选路径，路径不断对蚂蚁行为产生影响的双向作用，直到绝大多数蚂蚁选择了最优路径。

2. 算法描述

这里以例 5.2 中的旅行商问题为例介绍基本蚁群优化算法及其流程。基本蚁群优化算法可以表述如下：在算法的初始时刻，将 m 只蚂蚁随机地放到 n 座城市，同时为每只蚂蚁保存一个禁忌表 tabu_k ($k=1,2,\cdots,m$)，用来记录蚂蚁 k 已经走过的城市。初始化禁忌表 tabu_k 时，将其第一个元素设置为蚂蚁 k 当前所在的城市。设 $\tau_{ij}(t)$ 表示在 t 时刻，城市 i 到城市 j 的“信息素”浓度。初始化 $\tau_{ij}(t)=c$ (c 是较小的常数)，表示刚开始

时，各路径上的信息素量相等，且接近于0。

算法运行过程中，每只蚂蚁根据路上残留的“信息素”量和启发式信息（两城市间的距离）独立地选择下一座城市，在时刻 t，蚂蚁 k 从城市 i 转移到城市 j 的概率可表示为

$$p_{ij}^{(k)}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum\limits_{s\in J_k(i)}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}]^{\beta}}, & j\in J_k(i)\\ 0, & \text{其他}\end{cases} \tag{5-19}$$

式中：$J_k(i)=\{1,2,\cdots,n\}-\text{tabu}_k$，表示蚂蚁 k 下一步允许选择的城市集合。禁忌表 tabu_k 记录了蚂蚁 k 已经走过的城市。当所有 n 座城市都加入禁忌表 tabu_k 中时，蚂蚁 k 便完成了一次周游，此时蚂蚁 k 所走过的路径便是 TSP 的一个可行解。式(5-19)中的 η_{ij} 是一个启发式因子，表示蚂蚁从城市 i 转移到城市 j 的期望程度。在蚁群优化算法中，η_{ij} 通常取城市 i 与城市 j 之间距离的倒数。α、β 分别表示信息素和期望启发式因子的相对重要程度。当所有蚂蚁完成一次周游后，各路径上的信息素根据下式更新：

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij} \tag{5-20}$$

式中：ρ 为路径上“信息素”的蒸发系数（$0<\rho<1$），则 $1-\rho$ 为“信息素”的持久性系数；$\Delta\tau_{ij}$ 为本次选代中边 $i\to j$ 上信息素的增量，即

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{(k)} \tag{5-21}$$

式中：$\Delta\tau_{ij}^{(k)}$ 为第 k 只蚂蚁在本次选代中留在边 (i,j) 上的“信息素”量，如果蚂蚁 k 没有经过边 (i,j)，则 $\Delta\tau_{ij}^{(k)}$ 的值为零。$\Delta\tau_{ij}^{(k)}$ 可表示为

$$\Delta\tau_{ij}^{(k)}=\begin{cases}\dfrac{Q}{L_k}, & \text{蚂蚁 } k \text{ 在本次周游中经过边}(i,j)\\ 0, & \text{其他}\end{cases} \tag{5-22}$$

式中：Q 为正常数；L_k 为第 k 只蚂蚁在本次周游中所走过路径的长度。

实际上，M. Dorigo 一共提出了三种群算法的模型，式(5-22)称为 ant-cycle 模型，另外两个模型分别称为 ant-quantity 模型和 ant-density 模型，其差别主要在于 $\Delta\tau_{ij}$ 的表达方式。

在 ant-quantity 模型中，$\Delta\tau_{ij}$ 表示为

$$\Delta\tau_{ij}^{(k)}=\begin{cases}\dfrac{Q}{d_{ij}}, & \text{蚂蚁 } k \text{ 在本次周游中经过边}(i,j)\\ 0, & \text{其他}\end{cases} \tag{5-23}$$

式中：Q 为正常数；d_{ij} 为城市 i 到城市 j 的距离。

在 ant-density 模型中，$\Delta\tau_{ij}$ 表示为

$$\Delta\tau_{ij}^{(k)}=\begin{cases}Q, & \text{蚂蚁 } k \text{ 在本次周游中经过边}(i,j)\\ 0, & \text{其他}\end{cases} \tag{5-24}$$

式中：Q 为正常数。

实际上，蚁群优化算法是正反馈原理和启发式算法相结合的一种算法。在选择路径

时，蚂蚁不仅利用了路径上的“信息素”，而且用到了城市间距离的倒数作为启发式信息。实验结果表明，通常情况下，ant-cycle 模型比 ant-quantity 和 ant-density 模型有更好的性能。这是因为 ant-cycle 模型利用全局信息更新路径上的信息素量，而 ant-quantity 和 ant-density 模型使用的是局部信息。

求解 TSP 的基本蚁群优化算法的步骤如下。

Step1：参数初始化。令时间 $t=0$ 和循环次数 $N_c=0$，设置最大循环次数 G，将 m 个蚂蚁随机置于 n 个元素（城市）上，令有向图上每条边(i,j)的初始化“信息素”量 $\tau_{ij}(t)=c$，其中 c 为常数，且初始时刻 $\Delta\tau_{ij}=0$；

Step2：设置蚂蚁的索引号 $k=1$；

Step3：设置当前蚂蚁 k 已访问的城市数量 $\text{num}_{\text{city}}=1$；

Step4：蚂蚁个体 k 根据式(5-19)计算的概率选择城市 j 并前进，$j\in J_k(i)$；

Step5：将城市 j 加入禁忌表 tabu_k 中，$\text{num}_{\text{city}}=\text{num}_{\text{city}}+1$；

Step6：如果 $\text{num}_{\text{city}}<n$，即蚂蚁 k 尚未完成周游，则跳转到 Step4；

Step7：记录本次周游路线；

Step8：根据式(5-20)和式(5-21)更新每条路径上的“信息素”量；

Step9：蚂蚁索引 $k=k+1$；

Step10：若 $k\leqslant m$（尚有蚂蚁没有完成本轮的周游），则跳转到 Step3；

Step11：循环次数 $N_c=N_c+1$；

Step12：若满足结束条件$(N_c\geqslant G)$，则循环结束并输出程序优化结果；否则，清空禁忌表，并跳转到 Step2。

上述算法中，G 是循环次数的上限。

5.4.2 粒子群算法

生物学家 Reynolds 在 1987 年提出了一个非常有影响力的鸟群聚集模型，在他的仿真模型中每个鸟类个体都遵循以下三条规则：第一，避免与邻域个体相冲撞；第二，匹配邻域个体的速度；第三，飞向鸟群中心，且整个群体飞向目标。1995 年，美国社会心理学家肯尼迪(J. Kennedy)和电气工程师埃伯哈特(R. Eberhart)提出了粒子群算法，该算法的提出是受对鸟类群体行为进行建模与仿真的研究结果的启发。粒子群算法一经提出，由于该算法简单，容易实现，立刻引起了演化计算领域学者的广泛关注，形成了研究热点。粒子群中的每个个体在一维或二维网格空间中与相邻个体相互作用，从而表现出自组织、自优化的特点。

1. 算法原理

鸟类在捕食过程中，鸟群成员可以通过个体之间的信息交流与共享获得其他成员的发现与飞行经历。在食物源零星分布并且不可预测的条件下，这种协作机制所带来的优势远远大于对食物的竞争所引起的劣势。

将优化问题解空间（也称为搜索空间）的解（数学上可用一个 n 维矢量表示）表示成搜索空间的一只“鸟”，称为粒子(particle)，粒子无质量、无体积。每个粒子都有一个被某

个优化函数确立的适应值(代表目标函数值)、一个坐标位置(代表解)和一个速度,速度决定粒子的移动方向和距离。粒子群算法就是让粒子根据当前周围粒子的运动情况,在解空间中搜索最优解。

粒子群算法的信息共享机制可以解释为一种共生合作的行为,即每个粒子都在不停地进行搜索,并且其搜索行为在不同程度上受到群体中其他个体的影响。同时,这些粒子还具备对所经历最佳位置的记忆能力,即其搜索行为在受其他个体影响的同时还受到自身经验的引导。粒子群算法首先随机初始化所有粒子的位置和速度,其中粒子的位置用于表征问题的可行解,然后通过种群间粒子个体的合作与竞争来求解优化问题。

2. 算法描述

假设在一个 n 维的目标搜索空间中,有 N 个粒子组成一个种群,其中第 i 个粒子的位置可表示为一个 n 维的矢量:

$$\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{in})^{\mathrm{T}},\quad i=1,2,\cdots,N \tag{5-25}$$

第 i 个粒子的速度也可以表示为一个 n 维的矢量:

$$\boldsymbol{V}_i=(v_{i1},v_{i2},\cdots,v_{in})^{\mathrm{T}},\quad i=1,2,\cdots,N \tag{5-26}$$

第 i 个粒子迄今为止搜索到的最优位置称为个体极值,记为

$$\boldsymbol{P}_i=(p_{i1},p_{i2},\cdots,p_{in})^{\mathrm{T}},\quad i=1,2,\cdots,N \tag{5-27}$$

当前种群迄今为止搜索到的最优位置称为种群极值,记为

$$\boldsymbol{P}_g=(p_{g1},p_{g2},\cdots,p_{gn})^{\mathrm{T}} \tag{5-28}$$

在找到这两个最优值时,粒子根据式(5-29)和式(5-30)来更新自己的速度和位置:

$$v_{ij}(t+1)=wv_{ij}(t)+c_1r_1\big(p_{ij}(t)-x_{ij}(t)\big)+c_2r_2\big(p_{gj}(t)-x_{ij}(t)\big) \tag{5-29}$$

$$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1) \tag{5-30}$$

式中:i 表示微粒序号;j 表示变量的第 j 个维度;t 表示迭代次数;c_1 和 c_2 为学习因子,也称加速常数,分别表示微粒向自身最好位置和全局最好位置飞行的调整步长;v_{ij} 为粒子的速度,$v_{ij}\in[-v_{\max},v_{\max}]$,$v_{\max}$ 为常数,由用户设定来限制粒子的速度。w 为权重,表示在多大程度上保留原来的速度。当 w 较大时,全局收敛能力较强,局部收敛能力较弱;反之,局部收敛能力较强,全局收敛能力较弱。$r_1\sim U(0,1)$,$r_2\sim U(0,1)$为两个服从均匀分布的独立随机变量,称为惯性因子,其值越大,表示搜索的范围越大。$p_{ij}(t)$是当前粒子的最优解位置的分量,$p_{gj}(t)$是粒子群中全局最优粒子的最优解位置的分量。式(5-29)右边由三部分组成:第一部分为“惯性”或“动量”部分,反映了粒子的运动“习惯”,代表粒子有维持自己先前速度的趋势;第二部分为“认知”部分,反映了粒子对自身历史经验的记忆或回忆,代表粒子有向自身历史最佳位置逼近的趋势;第三部分为“社会”部分,反映了粒子间协同合作与知识共享的群体历史经验,代表粒子有向群体或邻域历史最佳位置逼近的趋势。

标准粒子群算法的步骤如下。

Step1:初始化。确定粒子群规模,每个微粒的初始位置和初始速度。

Step2:计算所有微粒的适应度值。

Step3:更新微粒的当前最好位置。对于每个微粒,比较当前适应度值与所经历过的

最好位置 P_i，若当前位置更好，则更新微粒最好位置。

Step4：更新微粒的全局最好位置。对于每个微粒，比较其当前最好位置适应度值与当前全局最好位置适应度值，若某个微粒的当前最好值更好，则更新微粒全局最好位置。

Step5：根据式(5-29)和式(5-30)得到每个微粒下一个时间片段(或称为下一代微粒)的新的位置。

Step6：若不满足终止条件，则返回 Step2 继续上述过程。

3. 算法特点

粒子群算法本质是一种随机搜索算法，它是一种智能优化技术。该算法能以较大概率收敛于全局最优解。实践证明，它适合在动态、多目标优化环境中寻优，与传统优化算法相比，具有较快的计算速度和更好的全局搜索能力。其特点如下。

(1) 粒子群算法是基于群智能理论的优化算法，通过群体中粒子间的合作与竞争产生的群体智能指导优化搜索。与其他算法相比，粒子群算法是一种高效的并行搜索算法。

(2) 粒子群算法与遗传算法都是随机初始化种群，使用适应值来评价个体的优劣程度和进行一定的随机搜索。但粒子群算法根据自己的速度来决定搜索，没有遗传算法的交叉与变异。与进化算法相比，粒子群算法保留了基于种群的全局搜索策略，但是其采用的“速度-位移”模型操作简单，避免了复杂的遗传操作。

(3) 由于每个粒子在算法结束时仍保持其个体极值，即粒子群算法除了可以找到问题的最优解，还会得到若干较好的次优解，因此将粒子群算法用于调度和决策问题可以给出多种有意义的方案。

(4) 粒子群算法特有的记忆使其可以动态地跟踪当前搜索情况并调整其搜索策略。另外，粒子群算法对种群的大小不敏感，即使种群数目下降，性能下降也不是很大。

5.5 本章小结

本章介绍了智能优化算法的用途、主要智能优化算法的提出过程、算法思想、实现步骤、关键操作等知识，要点回顾如下。

- 优化问题按照其求解的复杂程度，可以分为 P 问题、NP 问题、NP-完全问题三类。
- 智能优化算法在解决大空间、非线性、全局寻优、组合优化等复杂优化问题方面具有的独特优势。
- 模拟退火算法的流程可以概括为“三函数两准则一初温”。
- 遗传算法的关键操作包括编码、选择、交叉、变异等。
- 蚁群优化算法核心思想是模拟蚁群的觅食过程，通过对信息素的利用，寻找最优路径。
- 粒子群算法模拟鸟群飞行和觅食过程，每个粒子既参考自身历史经验的记忆或回忆，也参考鸟群中最优粒子的历史信息，确定下一时刻自身的位置和速度，从而达到迭代寻优的目的。

习题

1. 简述优化问题的概念，按照复杂度优化问题可分为几类？

2. 列举几种典型的智能优化算法。

3. 当存在局部最优点干扰时，哪些手段可以有助于优化算法跳出局部最优，继续靠近全局最优？

4. 简述模拟退火算法中的“三函数两准则”。

5. 简述模拟退火算法流程。

6. 简述模拟退火算法的优缺点。

7. 遗传算法中包含哪些基本遗传算子？

8. 遗传算法中常用的编码方式有哪些？

9. 用遗传算法解函数 $f(x)=-3x^2-2x\sin x(x\in[2,8])$ 的极小值问题，应如何构造适应度函数？

10. 分别简单描述遗传算法与模拟退火算法是如何在优化过程中引入随机性的。

11. 简述遗传算法的优缺点。

12. 简述基本蚁群算法的算法思想和流程，如何用蚁群算法求解 0-1 背包问题，写出步骤。

13. 简述基本粒子群算法的流程，并说明其优点。

14. 画出思维导图，串联本章所讲的知识点。