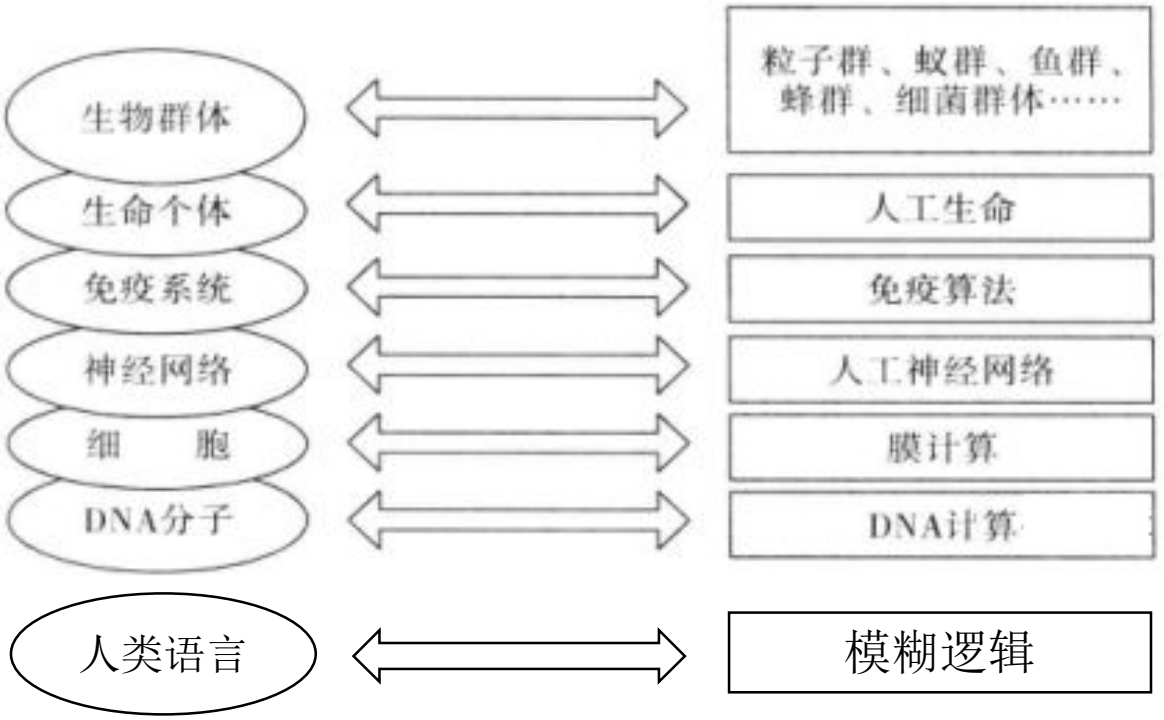
第10章 智能算法及其应用

Chapter 10 Intelligence Algorithms and Applications

背景

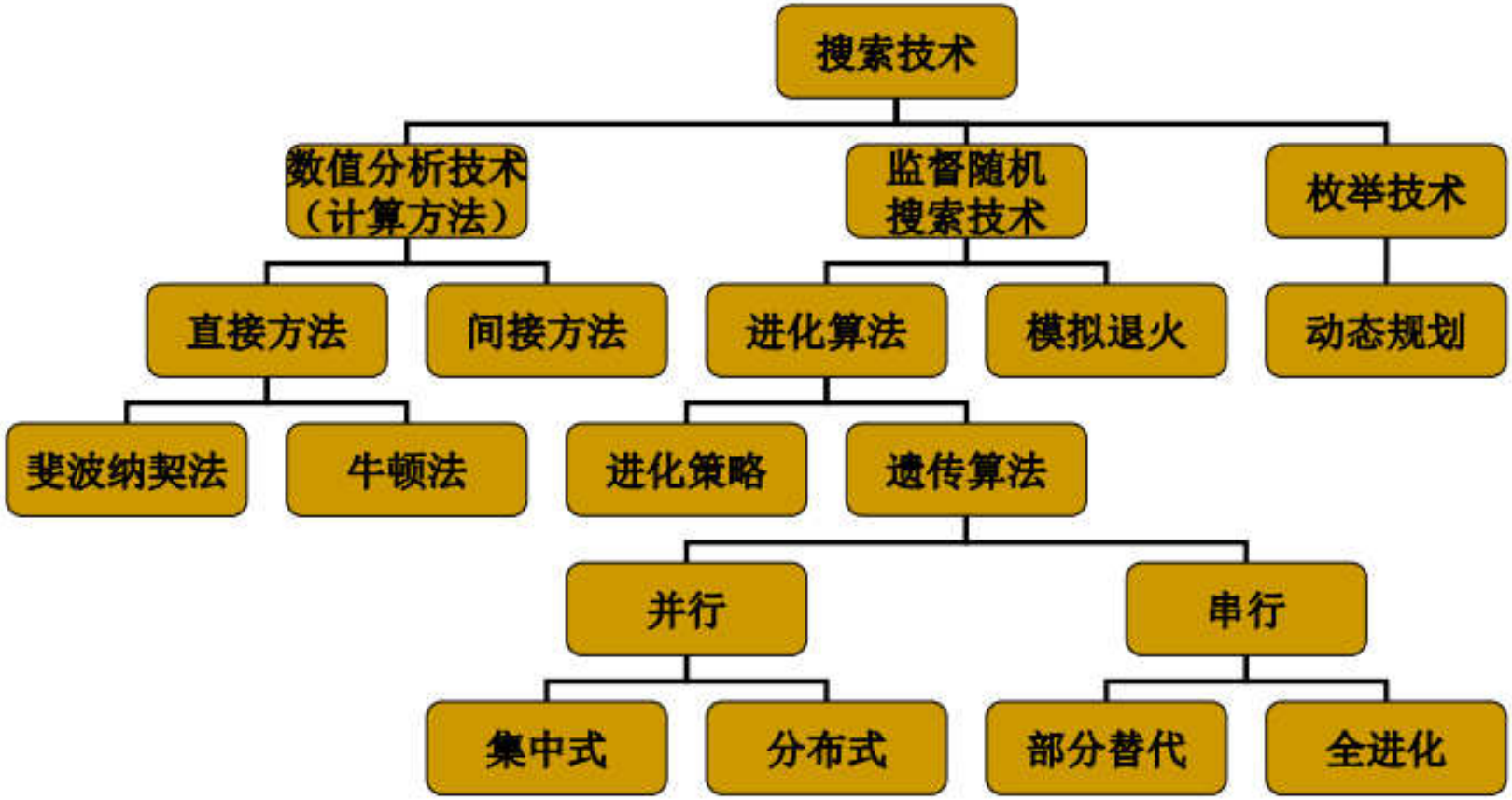
随着优化理论的发展，一些新的智能算法得到了迅速发展和广泛应用，成为解决传统控制问题的新方法，如遗传算法、 蚁群算法、 粒子群算法、差分进化算法等。这些优化算法都是通过模拟揭示自然现象和过程来实现，其优点和机制的独特，为具有非线性系统的控制问题提供了切实可行的解决方案。

背景


生物群体
生命个体
免疫系统
神经网络
细　胞
DNA分子
人类语言
粒子群、蚁群、鱼群、蜂群、细菌群体……
人工生命
免疫算法
人工神经网络
膜计算
DNA计算
模糊逻辑

仿生智能计算

背景


搜索技术
数值分析技术（计算方法）
监督随机搜索技术
枚举技术
直接方法
间接方法
进化算法
模拟退火
动态规划
斐波纳契法
牛顿法
进化策略
遗传算法
并行
串行
集中式
分布式
部分替代
全进化

背景

大多数算法都是基于梯度的应用优化算法，举个简单例子来说，求抛物线的极值点时，我们一般先要对函数求导，得到梯度变化最小的点，这个点就是极值点。但是群智能依靠的是概率搜索算法，虽然概率搜索算法通常采用较多的评价函数，但是与梯度方法及传统的演化算法相比，优点明显：

- 算法思想简单容易实现，只需遵循几个简单的规则。
- 以非直接的信息交流方式确保系统的扩张性，也就是说，每个点的行为是自主的，只会根据局部的变化改变自己的行为，不是由中心控制的。
- 具有并行性和分布式特点，可以利用多处理器予以实现。
- 对问题定义的连续性无特许要求，可以处理离散域问题。
- 没有集中控制的约束，不会因个别的故障影响整个问题的求解，确保了系统具备更强的鲁棒性（也就是说抗干扰能力强）。

群智能算法仅仅涉及基本的数学操作，数据处理过程对CPU和内存的要求不高。这种方法只需要计算目标函数的函数值，不用计算梯度信息。群智能理论已经完成的研究和应用证明这种方法是有效解决大多数全局优化问题的新方法。

内容提要

10.1 遗传算法的基本原理

遗传算法简称GA（Genetic Algorithms）是由美国Michigan大学的Holland教授提出的模拟自然界遗传机制和生物进化论而成的一种并行随机搜索最优化方法。

遗传算法的概念最早是由Bagley J.D在1967年提出的；而开始遗传算法的理论和方法的系统性研究的是1975年，这一开创性工作是由Michigan大学的J.H.Holland所实行。当时，其主要目的是说明自然和人工系统的自适应过程。

10.1 遗传算法的基本原理

约翰•霍兰德（John Holland，1929.2.2—2015.8.9），何许人也？他是复杂理论（Complexity）和非线性科学的先驱，“遗传算法”（Genetic Algorithm）之父！

主要研究领域为复杂自适应系统（CAS）、认知过程的计算机模型等。1950年获得麻省理工学院学士学位。后获得密歇根大学博士，并长期任教于该校。现为心理学和电气工程与计算机科学教授。

10.1 遗传算法的基本原理

那么“复杂的适应性系统”是什么呢？其实，包括人脑、免疫系统、生态系统、细胞、胚胎、蚂蚁群。。。他们似乎有某种直观重要的共性。

第一、每一个这样的系统都是一个由许多平行发生作用的“作用者”（Agent）组成的网络。更进一步说，一个复杂的适应性系统的控制力是相当分散的；而一个系统所产生的连续一致性结果，是产生于Agent之间的互相竞争与合作。

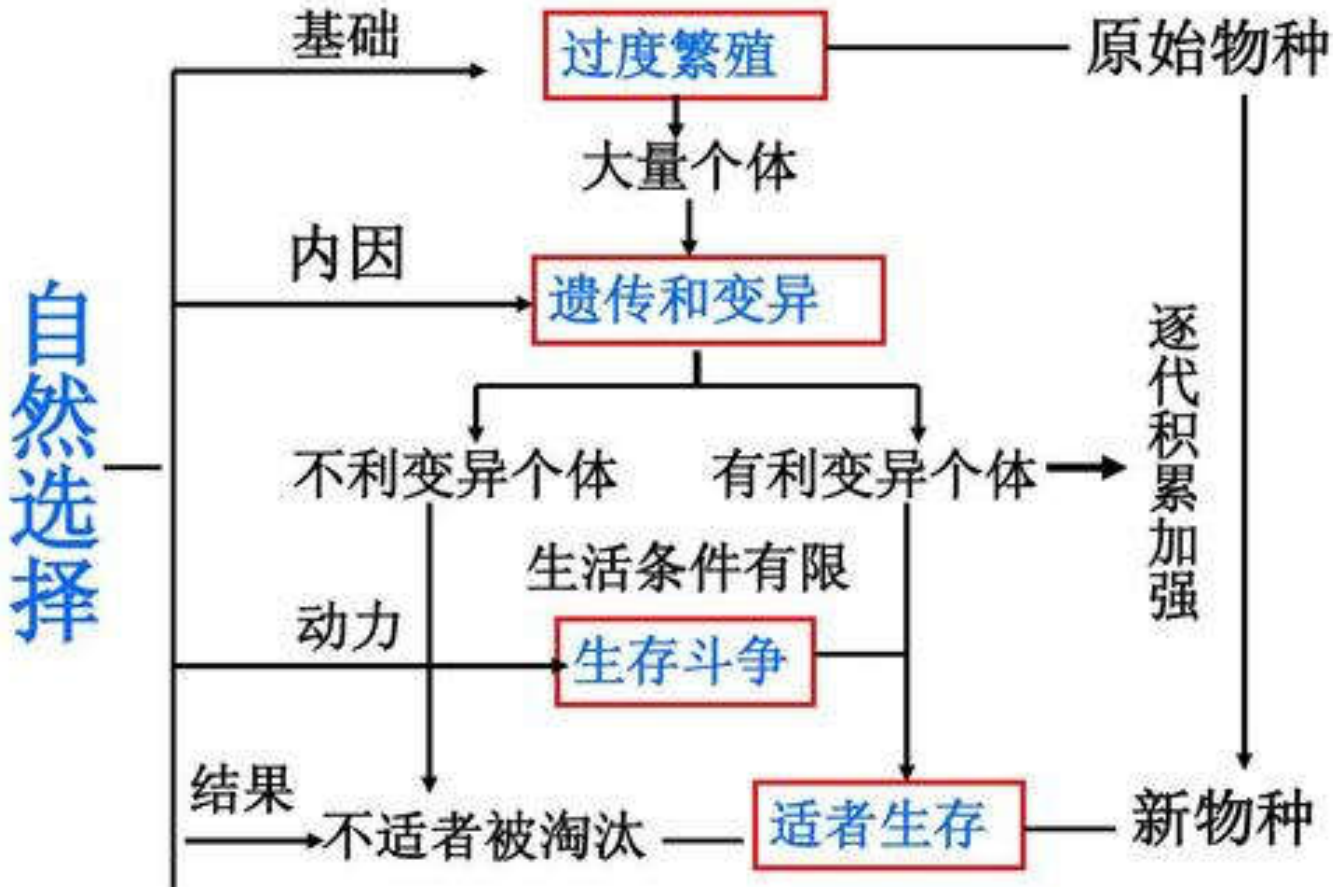
第二、一个复杂的适应性系统都具有多层次组织，每一个层次的Agent对更高层次的Agent来说，都起着建设砖块的作用。更重要的是：复杂的适应性系统能够吸取经验，从而经常改善和重新安排他们的建设砖块，这是最根本的适应机制之一。

10.1 遗传算法的基本原理

第三、所有复杂的适应性系统都会预期将来。就是内部的规则和假设，如：“在ABC情况下，可能要采取CDE行动”。可以把内心的假设模型想象成是行为的建设砖块。他们就像所有其他建设砖块一样，也能够随着系统不断吸取经验而被检验、被完善和被重新安排。

第四、复杂的适应性系统总是会有很多小生境（Niche），每一个这样的小生境都可以被一个能够使自己适应在其间发展的Agent所利用。每一个Agent填入一个小生境的同时又打开了更多的小生境。因此，复杂适应性系统的均衡就毫无意义：这种系统永远不可能达到均衡的状态，他总是处在不断展开，不断转变中。这里我们看到，复杂适应性系统的特点就是永恒的新奇性！

10.1 遗传算法的基本原理



10.1 遗传算法的基本原理

遗传算法的基本思想

- 以一个个体表示最优化问题的一个候选解；
- 以一个种群表示可行解的集合；
- 通过种群中个体之间的遗传操作（交叉和变异）产生新的候选解；
- 通过个体的适应度函数评价选择更优的可行解，构成新的种群。
- 反复进行上述遗传和选择过程，直至产生最优解。

10.1 遗传算法的基本原理

遗传算法是以达尔文的自然选择学说为基础发展起来的。自然选择学说包括以下三个方面：

（1）遗传：这是生物的普遍特征，亲代把生物信息交给子代，子代总是和亲代具有相同或相似的性状。生物有了这个特征，物种才能稳定存在。

（2）变异：亲代和子代之间以及子代的不同个体之间的差异，称为变异。变异是随机发生的，变异的选择和积累是生命多样性的根源。

（3）生存斗争和适者生存：具有适应性变异的个体被保留下来，不具有适应性变异的个体被淘汰，通过一代代的生存环境的选择作用，性状逐渐逐渐与祖先有所不同，演变为新的物种。

10.1 遗传算法的基本原理

遗传算法将“优胜劣汰，适者生存”的生物进化原理引入优化参数形成的编码串联群体中，按所选择的适应度函数并通过遗传中的选择/复制、交叉及变异对个体进行筛选，使适应度高的个体被保留下来，组成新的群体，新的群体既继承了上一代的信息，又优于上一代。这样周而复始，群体中个体适应度不断提高，直到满足一定的条件。遗传算法的算法简单，可并行处理，并能到全局最优解。

10.1 遗传算法的基本原理

遗传算法的基本操作为：

（1）复制（Reproduction Operator）

复制是从一个旧种群中选择生命力强的个体位串产生新种群的过程。具有高适应度的位串更有可能在下一代中产生一个或多个子孙。

复制操作可以通过随机方法来实现。首先产生0~1之间均匀分布的随机数，若某串的复制概率为40%，则当产生的随机数在0.40~1.0之间时，该串被复制，否则被淘汰。

10.1 遗传算法的基本原理

（2）交叉（Crossover Operator）

复制操作能从旧种群中选择出优秀者，但不能创造新的染色体。而交叉模拟了生物进化过程中的繁殖现象，通过两个染色体的交换组合，来产生新的优良品种。

交叉的过程为：在匹配池中任选两个染色体，随机选择一点或多点交换点位置；交换双亲染色体交换点右边的部分，即可得到两个新的染色体数字串。

10.1 遗传算法的基本原理

（3）变异(Mutation Operator)

变异运算用来模拟生物在自然的遗传环境中由于各种偶然因素引起的基因突变，它以很小的概率随机地改变遗传基因（表示染色体的符号串的某一位）的值。在染色体以二进制编码的系统中，它随机地将染色体的某一个基因由1变为0，或由0变为1。

若只有选择和交叉，而没有变异，则无法在初始基因组合以外的空间进行搜索，使进化过程在早期就陷入局部解而进入终止过程，从而影响解的质量。为了在尽可能大的空间中获得质量较高的优化解，必须采用变异操作。

10.2 遗传算法的特点

（1）遗传算法是对参数的编码进行操作，而非对参数本身，这就是使得我们在优化计算过程中可以借鉴生物学中染色体和基因等概念，模仿自然界中生物的遗传和进化等机理；

（2）遗传算法同时使用多个搜索点的搜索信息。传统的优化方法往往是从解空间的单个初始点开始最优解的迭代搜索过程，单个搜索点所提供的信息不多，搜索效率不高，有时甚至使搜索过程局限于局部最优解而停滞不前。

遗传算法从由很多个体组成的一个初始群体开始最优解的搜索过程，而不是从一个单一的个体开始搜索，这是遗传算法所特有的一种隐含并行性，因此遗传算法的搜索效率较高。

10.2 遗传算法的特点

（3）遗传算法直接以目标函数作为搜索信息。传统的优化算法不仅需要利用目标函数值，而且需要目标函数的导数值等辅助信息才能确定搜索方向。而遗传算法仅使用由目标函数值变换来的适应度函数值，就可以确定进一步的搜索方向和搜索范围，无需目标函数的导数值等其他一些辅助信息。

遗传算法可应用于目标函数无法求导数或导数不存在的函数的优化问题，以及组合优化问题等。

10.2 遗传算法的特点

（4）遗传算法使用概率搜索技术。遗传算法的选择、交叉、变异等运算都是以一种概率的方式来进行的，因而遗传算法的搜索过程具有很好的灵活性。随着进化过程的进行，遗传算法新的群体会更多地产生出许多新的优良的个体。

（5）遗传算法在解空间进行高效启发式搜索，而非盲目地穷举或完全随机搜索。

10.2 遗传算法的特点

（6）遗传算法对于待寻优的函数基本无限制，它既不要求函数连续，也不要求函数可微，既可以是数学解析式所表示的显函数，又可以是映射矩阵甚至是神经网络的隐函数，因而应用范围较广。

（7）遗传算法具有并行计算的特点，因而可通过大规模并行计算来提高计算速度，适合大规模复杂问题的优化。

10.3 遗传算法的发展及应用

（一）遗传算法的发展

遗传算法起源于对生物系统所进行的计算机模拟研究。早在20世纪40年代，就有学者开始研究如何利用计算机进行生物模拟的技术，他们从生物学的角度进行了生物的进化过程模拟、遗传过程模拟等研究工作。进入20世纪60年代，美国密执安大学的Holland教授及其学生们受到这种生物模拟技术的启发，创造出一种基于生物遗传和进化机制的适合于复杂系统优化计算的自适应概率优化技术—遗传算法。

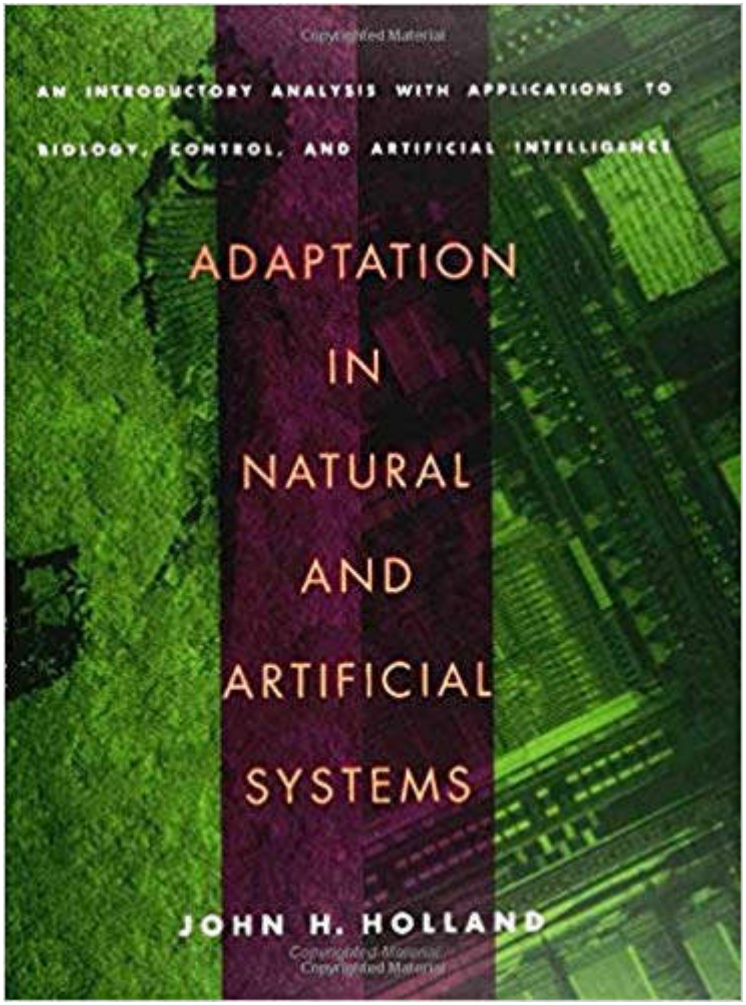
10.3 遗传算法的发展及应用

以下是在遗传算法发展进程中一些关键人物所做出的主要贡献：

（1）J.H.Holland

20世纪70年代初，Holland教授提出了遗传算法的基本定理—模式定理，从而奠定了遗传算法的理论基础。模式定理揭示了群体中优良个体（较好的模式）的样本数将以指数级规律增长，从理论上保证了遗传算法用于寻求最优可行解的优化过程。1975年，Holland出版了第一本系统论述遗传算法和人工自适应系统的专著《自然系统和人工系统的自适应性》。20世纪80年代，Holland教授实现了第一个基于遗传算法的机器学习系统—分类器系统，开创了基于遗传算法的机器学习的新概念。

10.3 遗传算法的发展及应用



10.3 遗传算法的发展及应用

（2）J.D.Bagley

1967年，Holland的学生Bagley在其博士论文中首次提出了“遗传算法”一词，并发表了遗传算法应用方面的第一篇论文。他发展了复制、交叉、变异、显性、倒位等遗传算子，在个体编码上使用了双倍体的编码方法。在遗传算法的不同阶段采用了不同的概率，从而创立了自适应遗传算法的概念。

10.3 遗传算法的发展及应用

（3）K.A.De Jong

1975年，De Jong博士在其博士论文中结合模式定理进行了大量纯数值函数优化计算实验，树立了遗传算法的工作框架。他推荐了在大多数优化问题中都较适用的遗传算法的参数，建立了著名的De Jong五函数测试平台，定义了评价遗传算法性能的在线指标和离线指标。

（4）D.J.Goldberg

1989年，Goldberg出版了专著《搜索、优化和机器学习中的遗传算法》，该书全面地论述了遗传算法的基本原理及其应用，奠定了现代遗传算法的科学基础。

10.3 遗传算法的发展及应用

（5）L. Davis

1991年，Davis编辑出版了《遗传算法手册》一书，为推广和普及遗传算法的应用起到了重要的指导作用。

（6）J.R. Koza

1992年，Koza将遗传算法应用于计算机程序的优化设计及自动生成，提出了遗传编程的概念，并成功地将遗传编程的方法应用于人工智能、机器学习和符号处理等方面。

10.3 遗传算法的发展及应用

（二）遗传算法的应用

（1）函数优化。

函数优化是遗传算法的经典应用领域，也是遗传算法进行性能评价的常用算例。尤其是对非线性、多模型、多目标的函数优化问题，采用其他优化方法较难求解，而遗传算法却可以得到较好的结果。

（2）组合优化。

随着问题的增大，组合优化问题的搜索空间也急剧扩大，采用传统的优化方法很难得到最优解。遗传算法是寻求这种满意解的最佳工具。例如，遗传算法已经在求解旅行商问题、背包问题、装箱问题、图形划分问题等方面得到成功的应用。

10.3 遗传算法的发展及应用

10.3 遗传算法的发展及应用

（3）生产调度问题

在很多情况下，采用建立数学模型的方法难以对生产调度问题进行精确求解。在现实生产中多采用一些经验进行调度。遗传算法是解决复杂调度问题的有效工具，在单件生产车间调度、流水线生产车间调度、生产规划、任务分配等方面遗传算法都得到了有效的应用。

10.3 遗传算法的发展及应用

（4）自动控制

在自动控制领域中有很多与优化相关的问题需要求解，遗传算法已经在其中得到了初步的应用。例如，利用遗传算法进行控制器参数的优化、基于遗传算法的模糊控制规则的学习、基于遗传算法的参数辨识、基于遗传算法的神经网络结构的优化和权值学习等。

10.3 遗传算法的发展及应用

10.3 遗传算法的发展及应用

（5）机器人

例如，遗传算法已经在移动机器人路径规划、关节机器人运动轨迹规划、机器人结构优化和行为协调等方面得到研究和应用。

（6）图像处理

遗传算法可用于图像处理过程中的扫描、特征提取、图像分割等的优化计算。目前遗传算法已经在模式识别、图像恢复、图像边缘特征提取等方面得到了应用。

10.3 遗传算法的发展及应用

（7）人工生命

人工生命是用计算机、机械等人工媒体模拟或构造出的具有生物系统特有行为的人造系统。人工生命与遗传算法有着密切的联系，基于遗传算法的进化模型是研究人工生命现象的重要基础理论。遗传算法为人工生命的研究提供了一个有效的工具。

（8）遗传编程

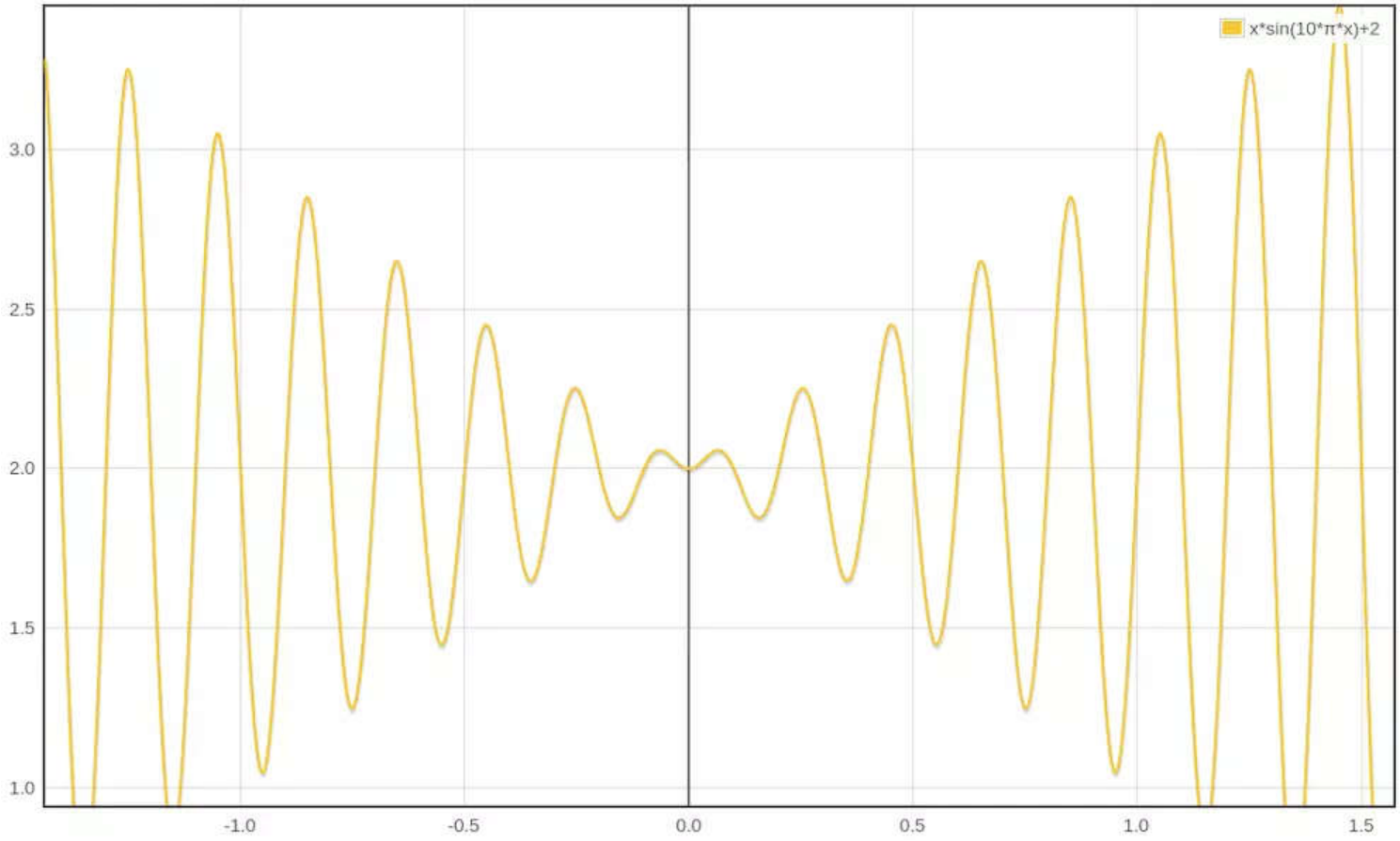
遗传算法已成功地应用于人工智能、机器学习等领域的编程。

10.3 遗传算法的发展及应用

（9）机器学习

基于遗传算法的机器学习在很多领域都得到了应用。例如，采用遗传算法实现模糊控制规则的优化，可以改进模糊系统的性能；遗传算法可用于神经网络连接权的调整和结构的优化；采用遗传算法设计的分类器系统可用于学习式多机器人路径规划。

10.4 遗传算法的设计



10.4 遗传算法的设计

10.4 遗传算法的设计

"袋鼠蹦跳"

既然我们把函数曲线理解成一个一个山峰和山谷组成的山脉。那么我们可以设想所得到的每一个解就是一只袋鼠，我们希望它们不断的向着更高处跳去，直到跳到最高的山峰。**所以求最大值的过程就转化成一个“袋鼠跳”的过程。**

- 爬山算法：一只袋鼠朝着比现在高的地方跳去。它找到了不远处的最高的山峰。但是这座山不一定是最高峰。这就是爬山算法，它不能保证局部最优值就是全局最优值。
- 模拟退火：袋鼠喝醉了，它随机地跳了很长时间。这期间，它可能走向高处，也可能踏入平地。但是，它渐渐清醒了并朝最高峰跳去。这就是模拟退火算法。
- 遗传算法：有很多袋鼠，它们降落到喜玛拉雅山脉的任意地方。这些袋鼠并不知道它们的任务是寻找珠穆朗玛峰。但每过几年，就在一些海拔高度较低的地方射杀一些袋鼠。于是，不断有袋鼠死于海拔较低的地方，而越是在海拔高的袋鼠越是能活得更久，也越有机会生儿育女。就这样经过许多年，这些袋鼠们竟然都不自觉地聚拢到了一个个的山峰上，可是在所有的袋鼠中，只有聚拢到珠穆朗玛峰的袋鼠被带回了美丽的澳洲。

10.4 遗传算法的设计

大体实现过程：

遗传算法中每一条染色体，对应着遗传算法的一个解决方案，一般我们用适应性函数（fitness function）来衡量这个解决方案的优劣。所以从一个基因组到其解的适应度形成一个映射。**遗传算法的实现过程实际上就像自然界的进化过程那样。**

下面我们用袋鼠跳中的步骤一一对应解释，以方便大家理解：

1.首先寻找一种对问题潜在解进行“数字化”编码的方案。（建立表现型和基因型的映射关系）

2.随机初始化一个种群（那么第一批袋鼠就被随意地分散在山脉上），种群里面的个体就是这些数字化的编码。

3.接下来，通过适当的解码过程之后（得到袋鼠的位置坐标）。

10.4 遗传算法的设计

4. 用适应性函数对每一个基因个体作一次适应度评估（袋鼠爬得越高当然就越好，所以适应度相应越高）。

5. 用选择函数按照某种规定择优选择（每隔一段时间，射杀一些所在海拔较低的袋鼠，以保证袋鼠总体数目持平。）。

6. 让个体基因变异（让袋鼠随机地跳一跳）。

7. 然后产生子代（希望存活下来的袋鼠是多产的，并在那里生儿育女）。

遗传算法并不保证你能获得问题的最优解，但是使用遗传算法的最大优点在于你不必去了解和操心如何去“找”最优解。（你不必去指导袋鼠向那边跳，跳多远。）而只要简单的“否定”一些表现不好的个体就行了。（把那些总是爱走下坡路的袋鼠射杀，这就是遗传算法的精粹！）

10.4 遗传算法的设计

遗传算法的一般步骤：

1. 随机产生种群。

2. 根据策略判断个体的适应度，是否符合优化准则，若符合，输出最佳个体及其最优解，结束。否则，进行下一步。

3. 依据适应度选择父母，适应度高的个体被选中的概率高，适应度低的个体被淘汰。

4. 用父母的染色体按照一定的方法进行交叉，生成子代。

5. 对子代染色体进行变异。

由交叉和变异产生新一代种群，返回步骤2，直到最优解产生。

10.4 遗传算法的设计

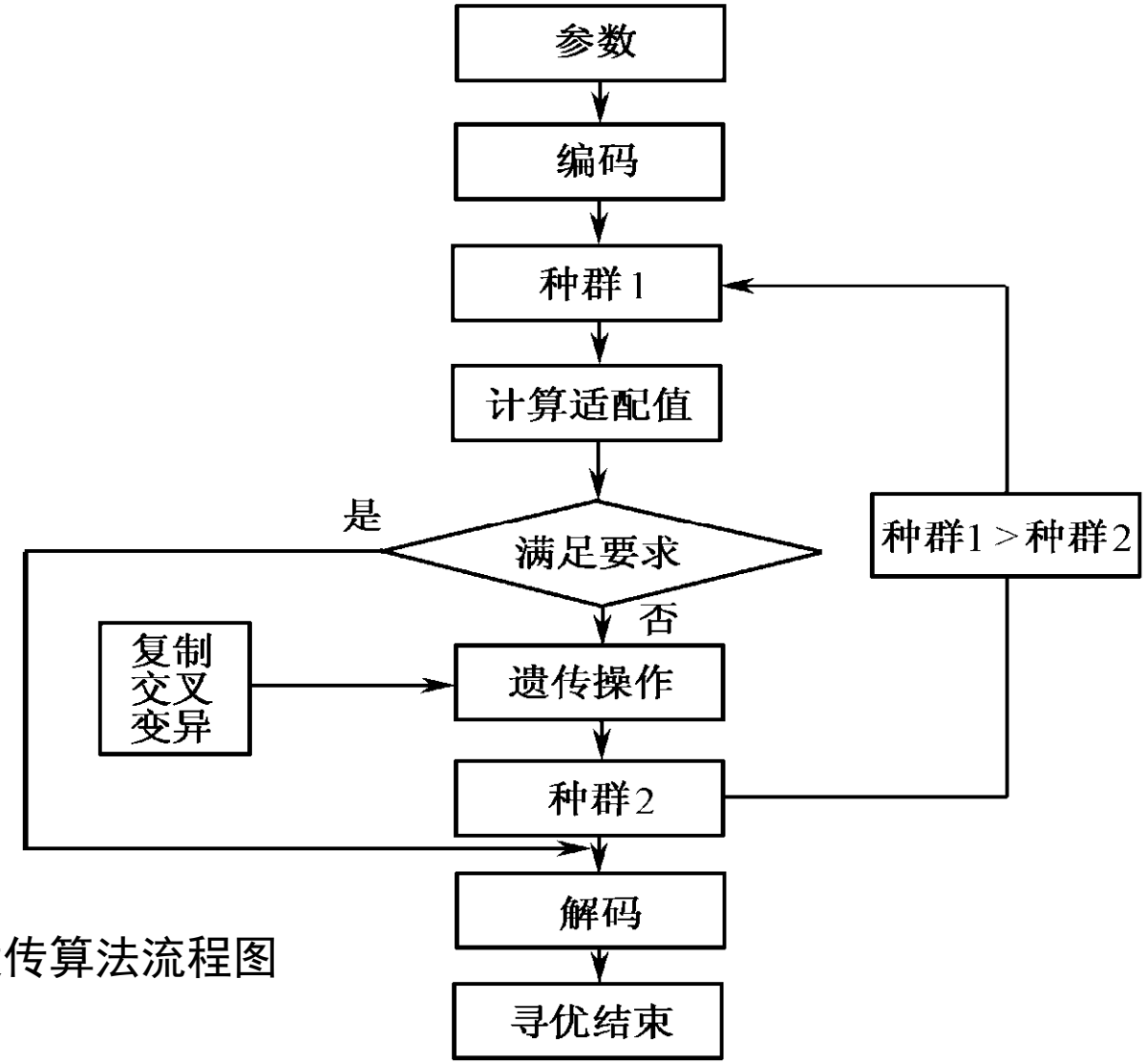


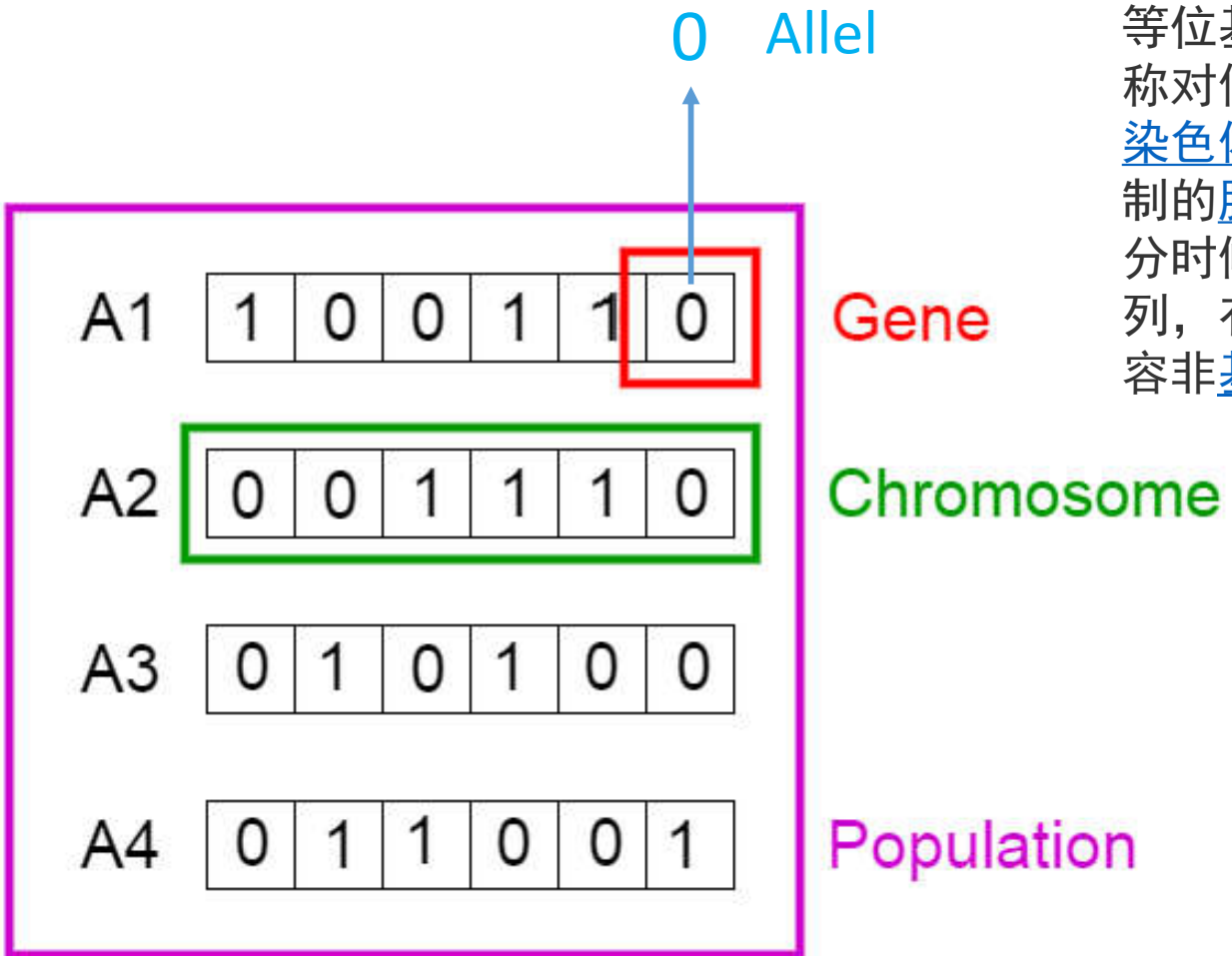
图10-1 遗传算法流程图

10.4 遗传算法的设计

相关生物学术语

- 基因型(genotype)：性状染色体的内部表现；
- 表现型(phenotype)：染色体决定的性状的外部表现，或者说，根据基因型形成的个体的外部表现；
- 进化(evolution)：种群逐渐适应生存环境，品质不断得到改良。生物的进化是以种群的形式进行的。
- 适应度(fitness)：度量某个物种对于生存环境的适应程度。
- 选择(selection)：以一定的概率从种群中选择若干个个体。一般，选择过程是一种基于适应度的优胜劣汰的过程。
- 复制(reproduction)：细胞分裂时，遗传物质DNA通过复制而转移到新产生的细胞中，新细胞就继承了旧细胞的基因。
- 交叉(crossover)：两个染色体的某一相同位置处DNA被切断，前后两串分别交叉组合形成两个新的染色体。也称基因重组或杂交；
- 变异(mutation)：复制时可能（很小的概率）产生某些复制差错，变异产生新的染色体，表现出新的性状。
- 编码(coding)：DNA中遗传信息在一个长链上按一定的模式排列。遗传编码可看作从表现型到基因型的映射。
- 解码(decoding)：基因型到表现型的映射。
- 个体（individual）：指染色体带有特征的实体；
- 种群（population）：个体的集合，该集合内个体数称为种群

10.4 遗传算法的设计



等位基因（*allele*），又称对偶基因，是一些占据染色体的基因座的可以复制的脱氧核糖核酸。大部分时候是脱氧核糖核酸序列，有的时候也被用来形容非基因序列。

10.4 遗传算法的设计

（一）遗传算法的构成要素

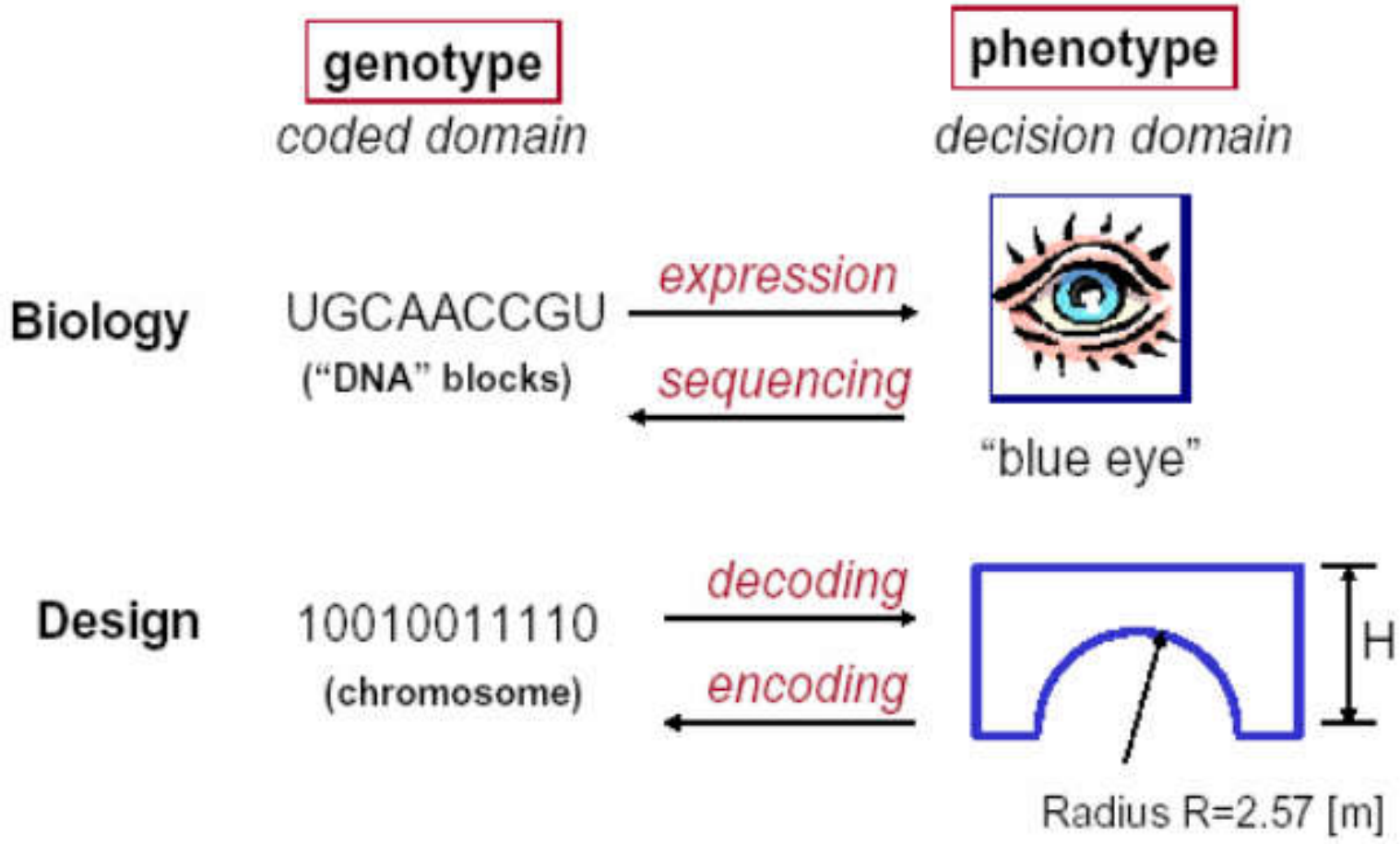
（1）染色体编码方法

基本遗传算法使用固定长度的二进制符号来表示群体中的个体，其等位基因是由二值符号集{0, 1}所组成。初始个体基因值可用均匀分布的随机值生成，如

$$x = 100111001000101101$$

就可表示一个个体，该个体的染色体长度是18。

10.4 遗传算法的设计


genotype
coded domain
phenotype
decision domain
Biology
UGCAACCGU
("DNA" blocks)
expression
sequencing
"blue eye"
Design
10010011110
(chromosome)
decoding
encoding
H
Radius R=2.57 [m]

10.4 遗传算法的设计

二进制编码法

就像人类的基因有AGCT 4种碱基序列一样。不过在这里我们只用了0和1两种碱基，然后将他们串成一条链形成染色体。一个位能表示出2种状态的信息量，因此足够长的二进制染色体便能表示所有的特征。这便是二进制编码。如下：

1110001010111

二进制编码的优点：

1. 编码、解码操作简单易行
2. 交叉、变异等遗传操作便于实现
3. 合最小字符集编码原则
4. 利用模式定理对算法进行理论分析。

二进制编码的缺点是：对于一些连续函数的优化问题，由于其随机性使得其局部搜索能力较差，如对于一些高精度的问题，当解迫近于最优解后，由于其变异后表现型变化很大，不连续，所以会远离最优解，达不到稳定。

10.4 遗传算法的设计

浮点编码法

二进制编码虽然简单直观，但明显地。但是存在着连续函数离散化时的映射误差。个体长度较短时，可能达不到精度要求，而个体编码长度较长时，虽然能提高精度，但增加了解码的难度，使遗传算法的搜索空间急剧扩大。

所谓浮点法，是指个体的每个基因值用某一范围内的一个浮点数来表示。在浮点数编码方法中，必须保证基因值在给定的区间限制范围内，遗传算法中所使用的交叉、变异等遗传算子也必须保证其运算结果所产生的新个体的基因值也在这个区间限制范围内。如下所示：

1. 2-3. 2-5. 3-7. 2-1. 4-9. 7

浮点数编码方法有下面几个优点：

1. 适用于在遗传算法中表示范围较大的数
2. 适用于精度要求较高的遗传算法
3. 便于较大空间的遗传搜索
4. 改善了遗传算法的计算复杂性，提高了运算交率
5. 便于遗传算法与经典优化方法的混合使用
6. 便于设计针对问题的专门知识的知识型遗传算子
7. 便于处理复杂的决策变量约束条件

10.4 遗传算法的设计

符号编码法

符号编码法是指个体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如{A,B,C…}。

符号编码的主要优点是：
1. 符合有意义积术块编码原则
2. 便于在遗传算法中利用所求解问题的专门知识
3. 便于遗传算法与相关近似算法之间的混合使用。

10.4 遗传算法的设计

为我们的袋鼠染色体编码

在上面介绍了一系列编码方式以后，那么，如何利用上面的编码来为我们的袋鼠染色体编码呢？首先我们要明确一点：编码无非就是建立从基因型到表现型的映射关系。这里的表现型可以理解为个体特征（比如身高、体重、毛色等等）。那么，在此问题下，我们关心的个体特征就是：袋鼠的位置坐标（因为我们要把海拔低的袋鼠给杀掉）。无论袋鼠长什么样，爱吃什么。我们关心的始终是袋鼠在哪里，并且只要知道了袋鼠的位置坐标（位置坐标就是相应的染色体编码，可以通过解码得出），我们就可以：

1. 在喜马拉雅山脉的地图上找到相应的位置坐标，算出海拔高度。（相当于通过自变量求得适应函数的值）然后判读该不该射杀该袋鼠。

2. 可以知道染色体交叉和变异后袋鼠新的位置坐标。

10.4 遗传算法的设计

（2）个体适应度评价：基本遗传算法与个体适应度成正比的概率来决定当前群体中每个个体遗传到下一代群体中的概率多少。为正确计算这个概率，要求所有个体的适应度必须为正数或零。因此，必须先确定由目标函数值J到个体适应度函数f之间的转换规则。

适应度函数也称评价函数，是根据目标函数确定的用于区分群体中个体好坏的标准。适应度函数总是非负的，而目标函数可能有正有负，故需要在目标函数与适应度函数之间进行变换。

评价个体适应度的一般过程为：

- ✓ 对个体编码串进行解码处理后，可得到个体的表现型。
- ✓ 由个体的表现型可计算出对应个体的目标函数值。
- ✓ 根据最优化问题的类型，由目标函数值按一定的转换规则求出个体的适应度。

10.4 遗传算法的设计

（3）遗传算子：基本遗传算法使用下述三种遗传算子。

① 选择/复制运算：比如，使用比例选择算子；

② 交叉运算：比如，使用单点交叉算子；

③ 变异运算：比如，使用基本位变异算子或均匀变异算子。

10.4 遗传算法的设计

遗传算法中的选择操作就是用来确定，如何从父代群体中按某种方法选取那些个体，以便遗传到下一代群体。选择操作用来确定重组或交叉个体，以及被选个体将产生多少个子代个体。我们希望海拔高的袋鼠存活下来，并尽可能繁衍更多的后代，也即，适应度高的袋鼠越能繁衍后代。但是，这只是概率意义上而已，某些适应度低的袋鼠也可能逃过我们的眼睛。

那么，怎么建立这种概率关系呢？

常用的选择算子：

- 轮盘赌选择（Roulette Wheel Selection）：是一种回放式随机采样方法。每个个体进入下一代的概率等于$\frac{\text{适应度值}}{\text{整个种群中个体适应度值之和}}$。选择误差较大。

- 随机竞争选择（Stochastic Tournament）：每次按轮盘赌选择一对个体，然后让这两个个体进行竞争，适应度高的被选中，如此反复，直到选满为止。

- 最佳保留选择：首先按轮盘赌选择方法执行遗传算法的选择操作，然后将当前群体中适应度最高的个体结构完整地复制到下一代群体中。

10.4 遗传算法的设计

- 无回放随机选择（也叫期望值选择Excepted Value Selection）。根据每个个体在下一代群体中的生存期望来进行随机选择运算:

（1） 计算群体中每个个体在下一代群体中的生存期望数目N。

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

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

- 确定式选择。按照一种确定的方式来进行选择操作：

（1） 计算群体中各个个体在下一代群体中的期望生存数目N。

（2） 用N的整数部分确定各个对应个体在下一代群体中的生存数目。

（3） 用N的小数部分对个体进行降序排列，顺序取前M个个体加入到下一代群体中。至此可完全确定出下一代群体中M个个体。

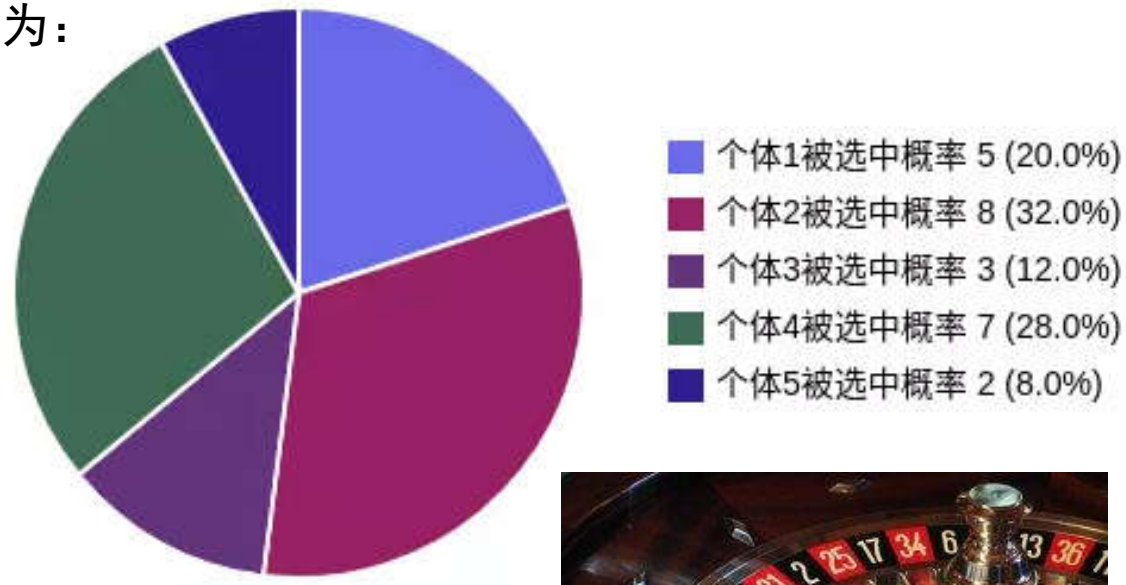
10.4 遗传算法的设计

- 无回放余数随机选择：可确保适应度比平均适应度大的一些个体能够被遗传到下一代群体中，因而选择误差比较小。
- 均匀排序：对群体中的所有个体按期适应度大小进行排序，基于这个排序来分配各个个体被选中的概率。
- 最佳保存策略：当前群体中适应度最高的个体不参与交叉运算和变异运算，而是用它来代替掉本代群体中经过交叉、变异等操作后所产生的适应度最低的个体。
- 随机联赛选择：每次选取几个个体中适应度最高的一个个体遗传到下一代群体中。
- 排挤选择：新生成的子代将代替或排挤相似的旧父代个体，提高群体的多样性。

10.4 遗传算法的设计

以轮盘赌选择为例：

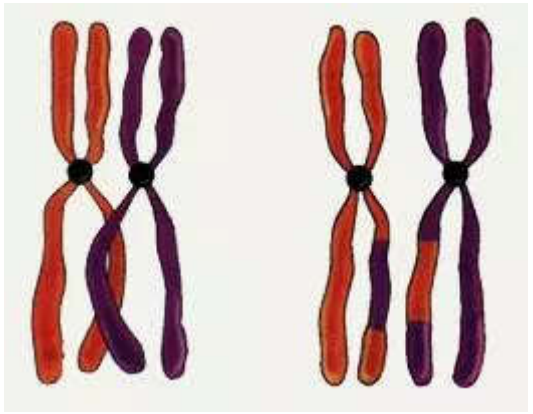
假如有５条染色体，他们的适应度分别为５、８、３、７、２。
那么总的适应度为：F = 5 + 8 + 3 + 7 + 2 = 25。
那么各个个体的被选中的概率为：
α1 = (5 / 25) * 100% = 20%
α2 = (8 / 25) * 100% = 32%
α3 = (3 / 25) * 100% = 12%
α4 = (7 / 25) * 100% = 28%
α5 = (2 / 25) * 100% = 8%
所以转盘如下：

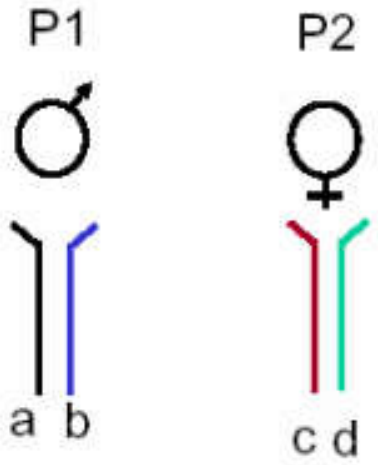


当指针在这个转盘上转动，停止下来时指向的个体就是天选之人啦。可以看出，适应性越高的个体被选中的概率就越大。

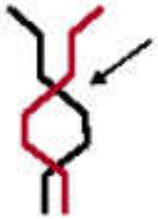
10.4 遗传算法的设计

Crossover in Biology





Child



This is where the word crossover comes from

ac

⇨ Crossover produces either of these results for each chromosome

ac OR ad OR bc OR bd

10.4 遗传算法的设计

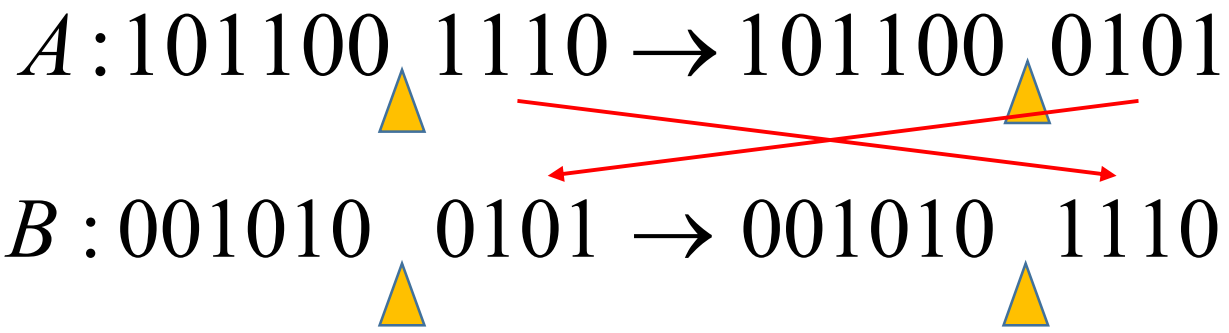
遗传算法的交叉操作，是指对两个相互配对的染色体按某种方式相互交换其部分基因，从而形成两个新的个体。

适用于二进制编码个体或浮点数编码个体的交叉算子：

- 单点交叉（One-point Crossover）：指在个体编码串中只随机设置一个交叉点，然后再该点相互交换两个配对个体的部分染色体。
- 两点交叉与多点交叉：
  - ✓ 两点交叉（Two-point Crossover）：在个体编码串中随机设置了两个交叉点，然后再进行部分基因交换。
  - ✓ 多点交叉（Multi-point Crossover）
- 均匀交叉（也称一致交叉，Uniform Crossover）：两个配对个体的每个基因座上的基因都以相同的交叉概率进行交换，从而形成两个新个体。
- 算术交叉（Arithmetic Crossover）：由两个个体的线性组合而产生出两个新的个体。该操作对象一般是由浮点数编码表示的个体。

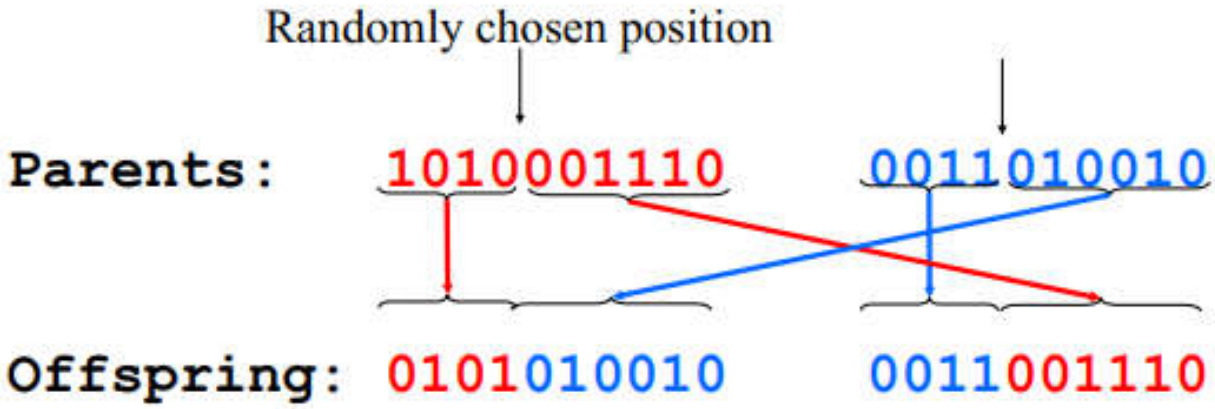
10.4 遗传算法的设计

交叉体现了自然界中信息交换的思想。交叉有一点交叉、多点交叉、还有一致交叉、顺序交叉和周期交叉。单点交叉是最基本的方法，应用较广。它是指染色体切断点有一处，例：



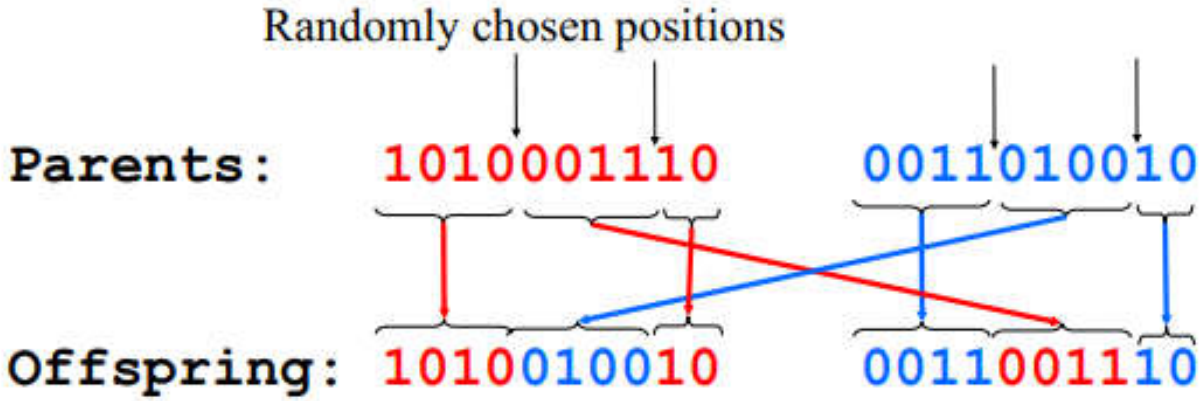
10.4 遗传算法的设计

- Randomly one position in the chromosomes is chosen
- Child 1 is head of chromosome of parent 1 with tail of chromosome of parent 2
- Child 2 is head of 2 with tail of 1



10.4 遗传算法的设计

- Randomly two positions in the chromosomes are chosen
- Avoids that genes at the head and genes at the tail of a chromosome are always split when recombined



10.4 遗传算法的设计

Single point crossover

Cross point

Two point crossover (Multi point crossover)

10.4 遗传算法的设计

变异--基因突变(Mutation)

遗传算法中的变异运算，是指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它等位基因来替换，从而形成新的个体。

例如下面这串二进制编码：
101101001011001

经过基因突变后，可能变成以下这串新的编码：
001101011011001

10.4 遗传算法的设计

以下变异算子适用于二进制编码和浮点数编码的个体：

- 基本位变异（Simple Mutation）：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变异运算。
- 均匀变异（Uniform Mutation）：分别用符合某一范围内均匀分布的随机数，以某一较小的概率来替换个体编码串中各个基因座上的原有基因值。（特别适用于在算法的初级运行阶段）
- 边界变异（Boundary Mutation）：随机的取基因座上的两个对应边界基因值之一去替代原有基因值。特别适用于最优点位于或接近于可行解的边界时的一类问题。
- 非均匀变异：对原有的基因值做一随机扰动，以扰动后的结果作为变异后的新基因值。对每个基因座都以相同的概率进行变异运算之后，相当于整个解向量在解空间中作了一次轻微的变动。
- 高斯近似变异：进行变异操作时用符号均值为 μ 的平均值，方差为 σ^2 的正态分布的一个随机数来替换原有的基因值。

10.4 遗传算法的设计

（4）基本遗传算法的运行参数

有下述4个运行参数需要提前设定：

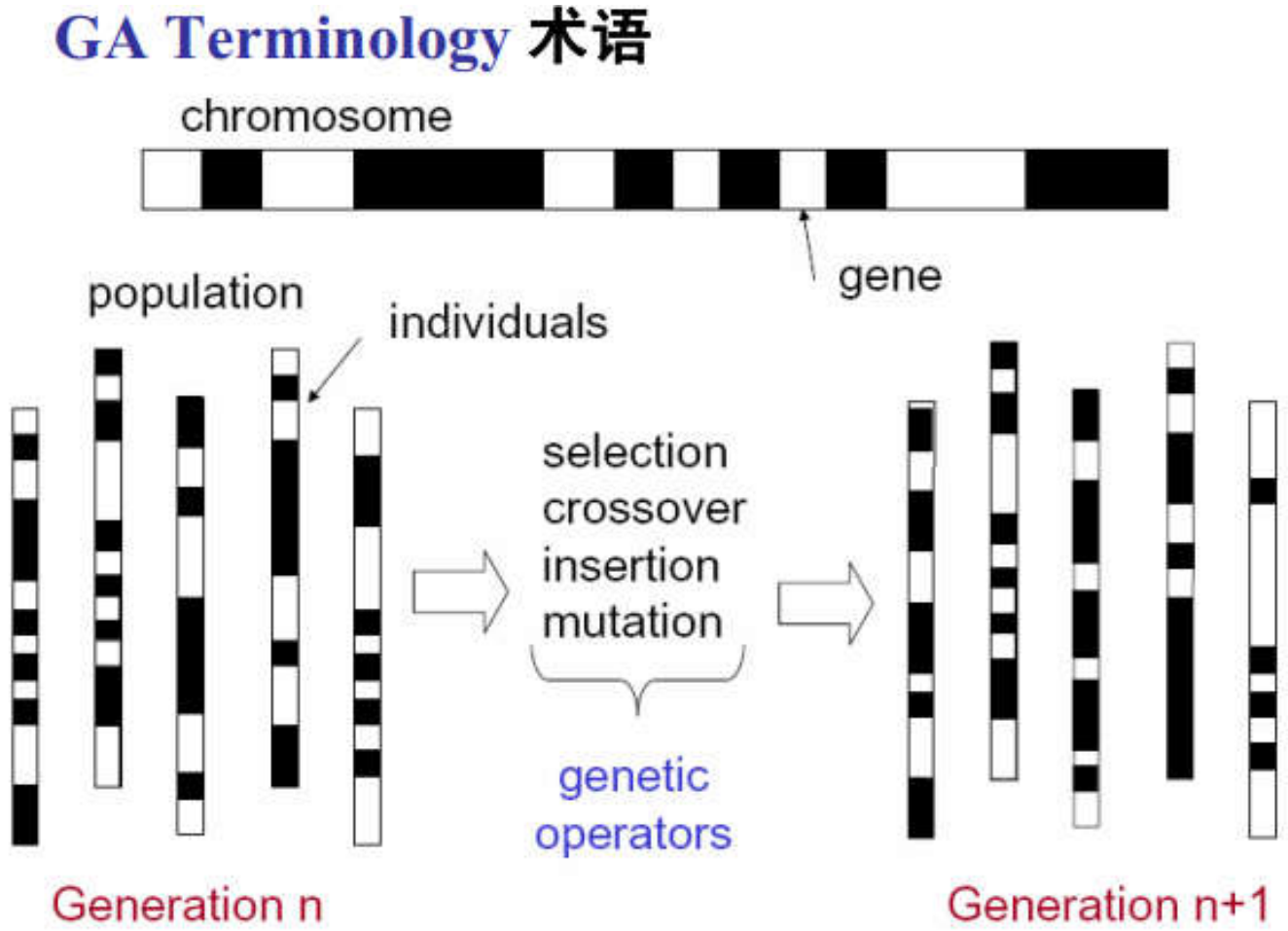
M：群体大小，即群体中所含个体的数量，一般取为20~100；

G：遗传算法的终止进化代数，一般取为100~500；

Pc：交叉概率，一般取为0.4~0.99；

Pm：变异概率，一般取为0.0001~0.1。

10.4 遗传算法的设计


GA Terminology 术语
chromosome
gene
population
individuals
selection
crossover
insertion
mutation
genetic
operators
Generation n
Generation n+1

10.4 遗传算法的设计

（二）遗传算法的应用步骤

对于一个需要进行优化的实际问题，一般可按下述步骤构造遗传算法：

第一步：确定决策变量及各种约束条件，即确定出个体的表现型X和问题的解空间；

第二步：建立优化模型，即确定出目标函数的类型及数学描述形式或量化方法；

10.4 遗传算法的设计

第三步：确定表示可行解的染色体编码方法，即确定出个体的基因型x及遗传算法的搜索空间；

第四步：确定个体适应度的量化评价方法，即确定出由目标函数值J(x)到个体适应度函数F(x)的转换规则；

第五步：设计遗传算子，即确定选择/复制运算、交叉运算、变异运算等遗传算子的具体操作方法；

第六步：确定遗传算法的有关运行参数，即M, G, Pc, Pm等参数；

第七步：确定解码方法，即确定出由个体表现型X到个体基因型x的对应关系或转换方法。

10.4 遗传算法的设计

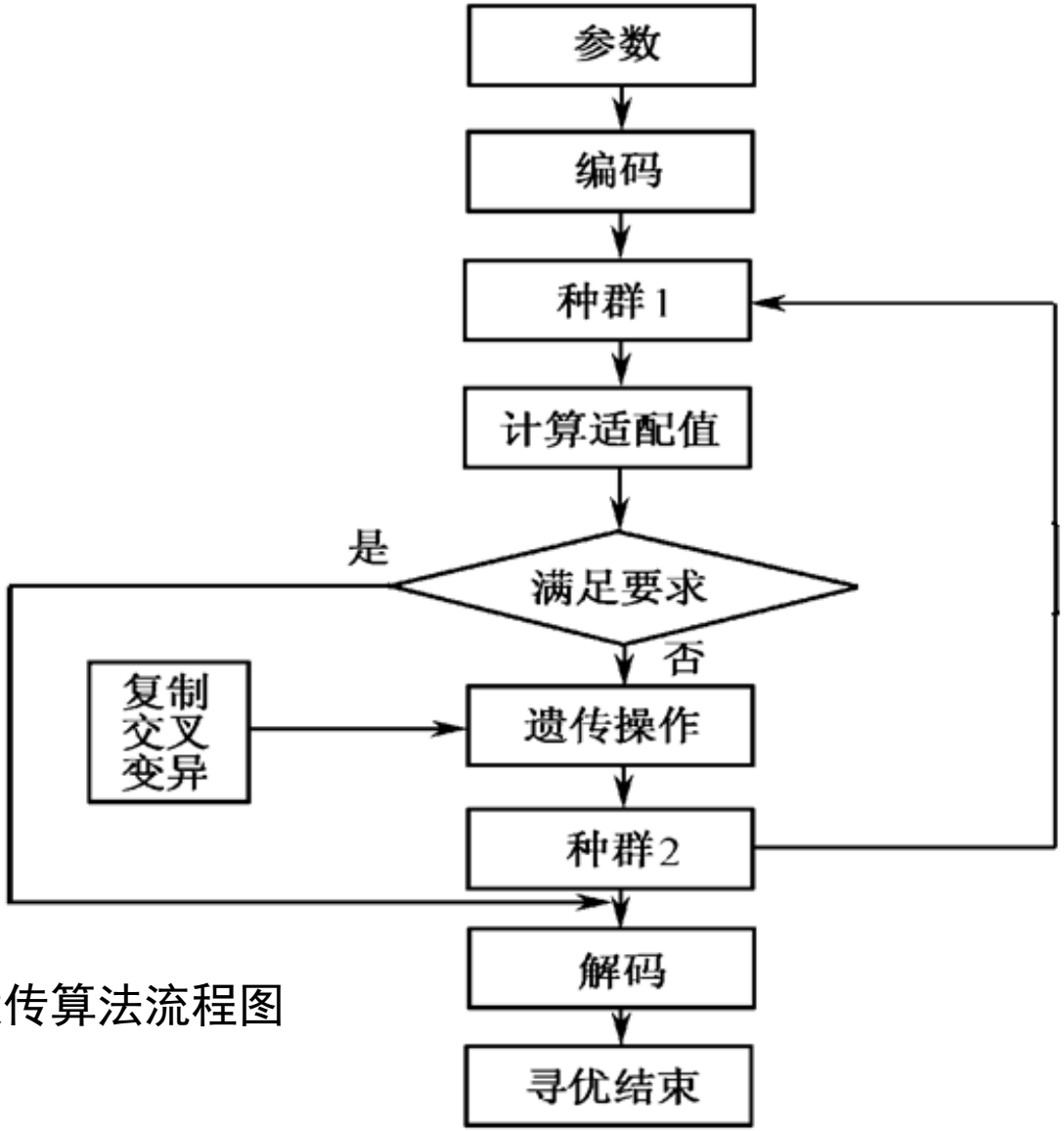


图10-1 遗传算法流程图

10.4 遗传算

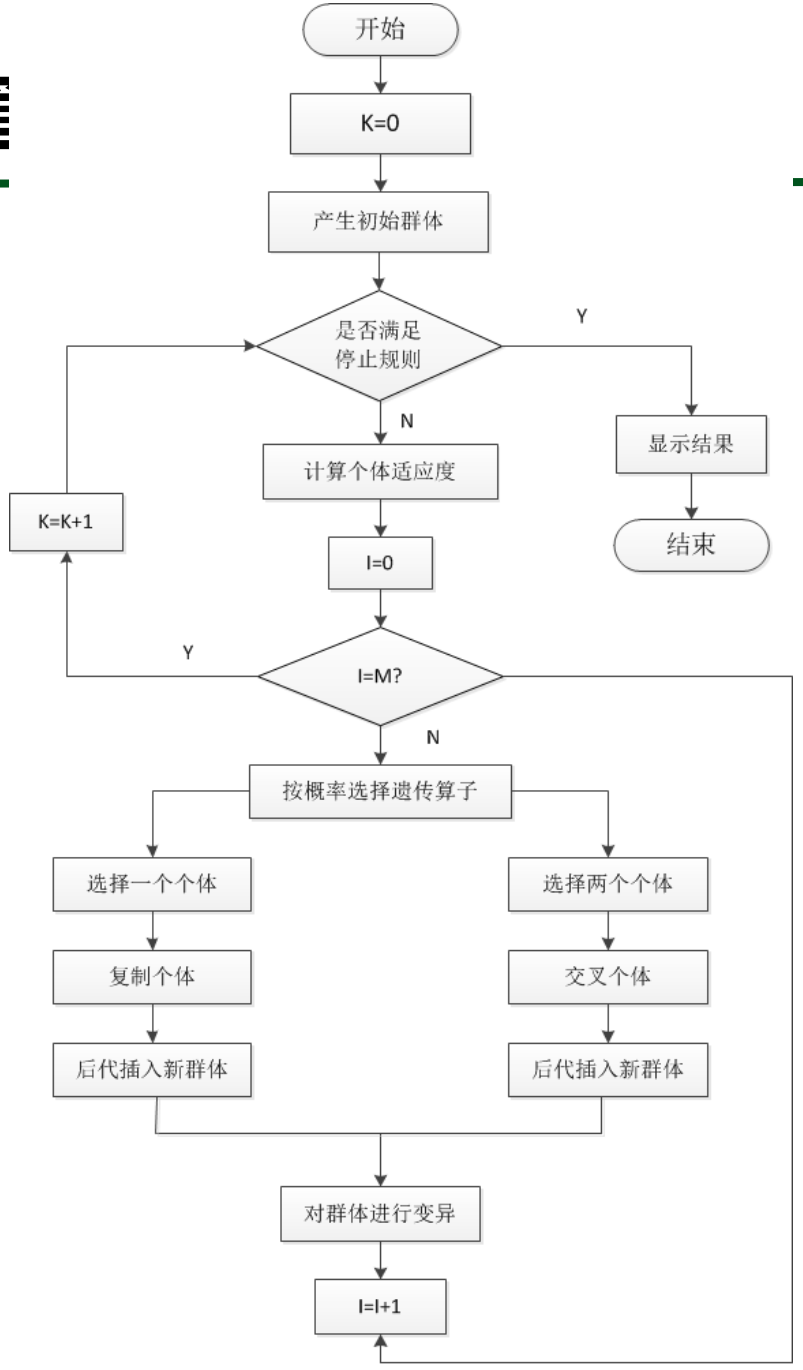


图 GA流程图

10.5 遗传算法求函数极大值

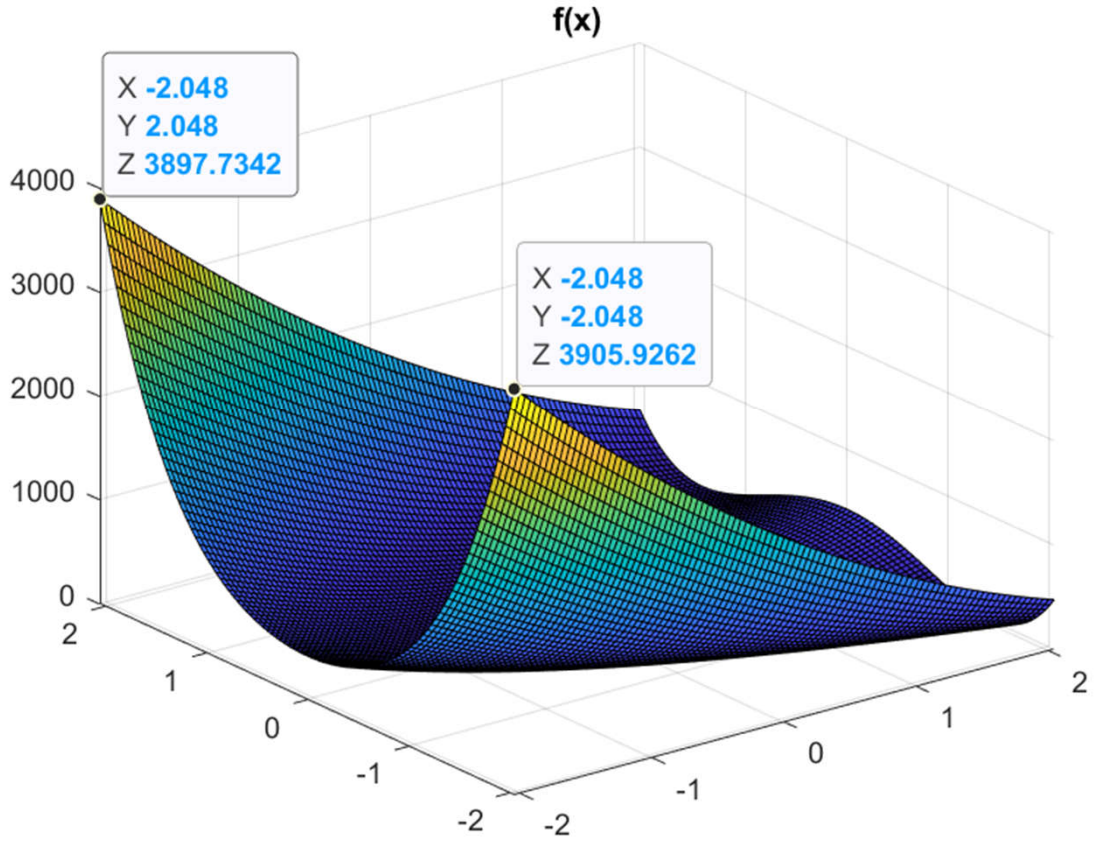
利用遗传算法求Rosenbrock函数的极大值

$$\begin{cases} f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \\ -2.048 \le x_i \le 2.048 \qquad\qquad (i = 1, 2) \end{cases}$$

该函数有两个局部极大点，分别是 $f(2.048, -2.048)=3897.7342$ 和 $f(-2.048, -2.048)=3905.9262$，其中后者为全局最大点。

10.4 遗传算法的设计

函数 $f(x_1, x_2)$ 的三维图如下图所示，可以发现该函数在指定的定义域上有两个接近的极点，即一个全局极大值和一个局部极大值。因此，采用寻优算法求极大值时，需要避免陷入局部最优解。



10.5 遗传算法求函数极大值

采用二进制编码遗传算法求函数极大值求解该问题遗传算法的构造过程：

（1）确定决策变量和约束条件；

（2）建立优化模型；

（3）确定编码方法：

10.5 遗传算法求函数极大值

用两个长度为10位的二进制编码串来分别表示两个决策变量x_1，x_2。10位二进制编码串可以表示从0到1023之间的1024个不同的数，故将x_1，x_2的定义域离散化为1023个均等的区域，包括两个端点在内共有1024个不同的离散点。

从离散点-2.048到离散点2.048，分别对应于从0000000000(0)到1111111111(1023)之间的二进制编码。

10.5 遗传算法求函数极大值

再将分别表示为x_1, x_2的两个10位二进制编码串连接在一起，组成一个20位长的二进制编码串，它就构成了这个函数优化问题的染色体编码方法。使用这种编码方法，解空间和遗传算法的搜索空间就具有一一对应的关系。

例如：

$$x : \underbrace{0000110111}_{x_1} \quad \underbrace{1101110001}_{x_2}$$

表示一个个体的基因型，其中前10位表示x_1，后10位表示x_2。

10.5 遗传算法求函数极大值

（4）确定解码方法：解码时需要将20位长的二进制编码串切断为两个10位长的二进制编码串，然后分别将它们转换为对应的十进制整数代码，分别记为y_1和y_2。

依据个体编码方法和对定义域的离散化方法可知，将代码y_i转换为变量x_i的解码公式为

$$x_i = 4.096 \times \frac{y_i}{1023} - 2.048 \qquad (i = 1,2) \qquad (10.1)$$

例如，对个体

$$x : 0000110111 \quad 1101110001$$

10.5 遗传算法求函数极大值

它由两个代码所组成

$$y_1 = 55, y_2 = 881$$

上述两个代码经过解码后，可得到两个实际的值

$$x_1 = -1.828, x_2 = 1.476$$

（5）确定个体评价方法：由于Rosenbrock函数的值域总是非负的，并且优化目标是求函数的最大值，故可将个体的适应度直接取为对应的目标函数值，即

$$F(x) = f(x_1, x_2) \qquad (10.2)$$

10.5 遗传算法求函数极大值

选个体适应度的倒数作为目标函数

$$J(x)=\frac{1}{F(x)} \tag{10.3}$$

（6）设计遗传算子：选择运算使用比例选择算子，交叉运算使用单点交叉算子，变异运算使用基本位变异算子。

（7）确定遗传算法的运行参数：群体大小M=80，终止进化代数G=100，交叉概率Pc=0.60，变异概率Pm=0.10。

上述七个步骤构成了用于求函数极大值的优化计算基本遗传算法。

10.5 遗传算法求函数极大值

采用上述方法进行仿真，经过100步迭代，最佳样本为

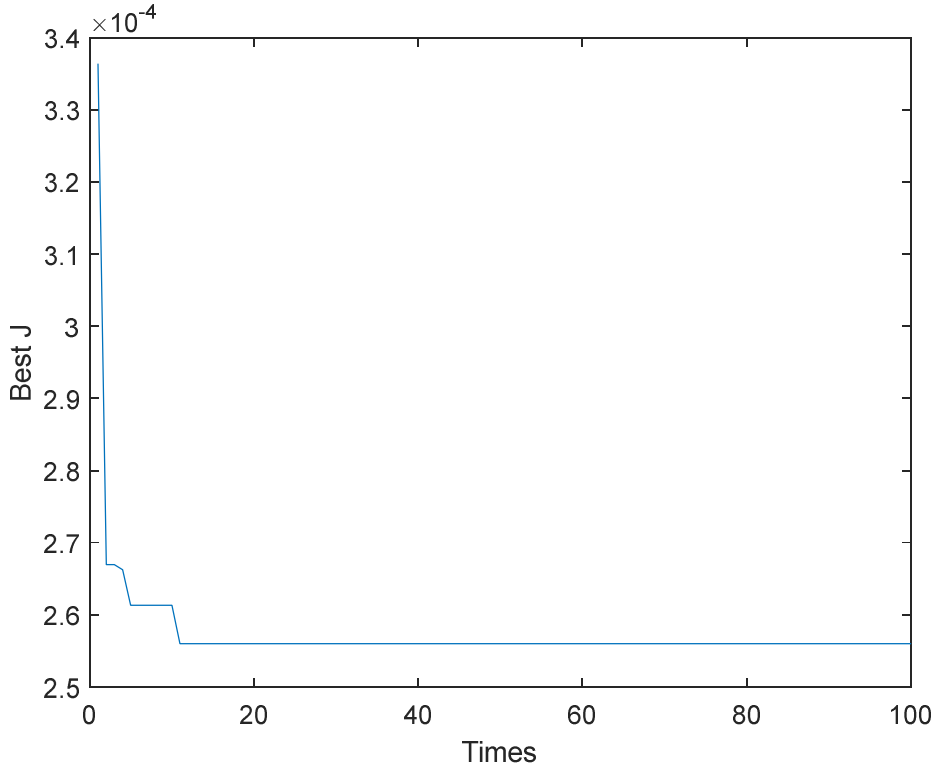
$$\text{Best}S = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0]$$

即当x1=x2=-2.0480时，Rosenbrock函数具有极大值，极大值为3905.9。

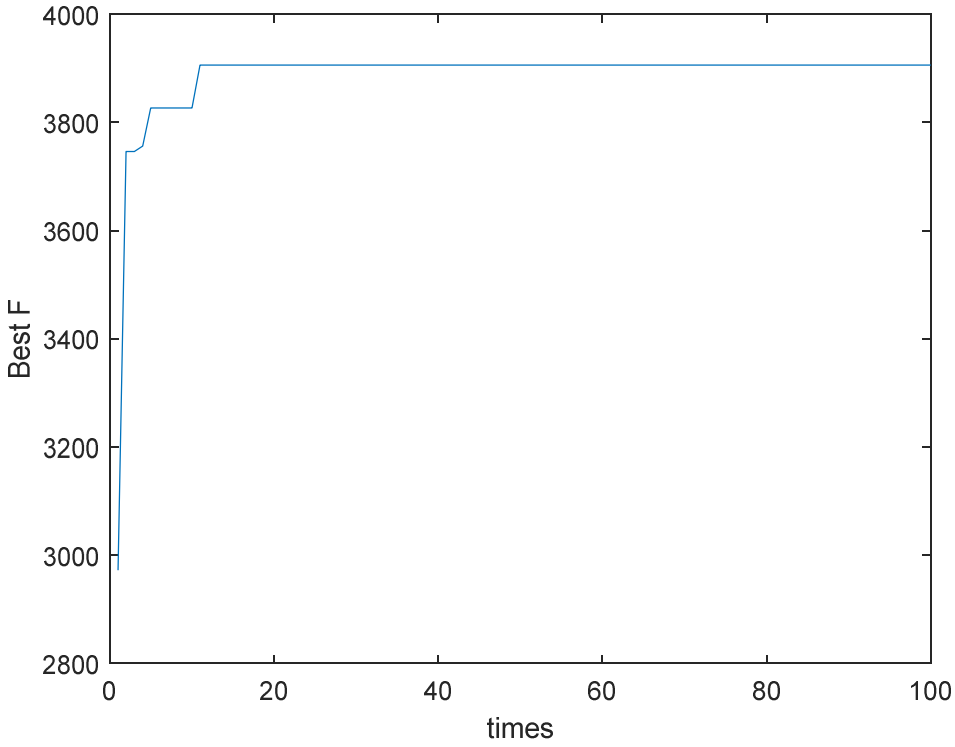
遗传算法的优化过程是目标函数J和适应度函数F的变化过程。

由仿真结果可知，随着进化过程的进行，群体中适应度较低的一些个体被逐渐淘汰掉，而适应度较高的一些个体会越来越多，并且它们都集中在所求问题的最优点附近，从而搜索到问题的最优解。

10.5 遗传算法求函数极大值



目标函数 J 的优化过程



适应度函数 F 的优化过程

10.6 基于遗传算法的TSP问题优化

在第8.4.2节已经对旅行商问题进行了描述。遗传算法由于其全局搜索的特点，在解决TSP问题中有明显的优势。

（一）TSP问题的编码

设 $\mathbf{D}=\{d_{ij}\}$ 是由城市i和城市j之间的距离组成的距离矩阵，旅行商问题就是求出一条通过所有城市且每个城市只通过一次的具有最短距离的回路。

10.6 基于遗传算法的TSP问题优化

在旅行商问题的各种求解方法中，描述旅行路线的方法主要有如下两种：

（1）巡回旅行路线经过的连接两个城市的路线的顺序排列；

（2）巡回旅行路线所经过的各个城市的顺序排列。

大多数求解旅行商问题的遗传算法是以后者为描述方法的，它们大多采用所遍历城市的顺序来表示各个个体的编码串，其等位基因为N个整数值或N个记号。

以城市的遍历次序作为遗传算法的编码，目标函数取路径长度。在群体初始化、交叉操作和变异操作中考虑TSP问题的合法性约束条件（即对所有的城市做到不重不漏）。

10.6 基于遗传算法的TSP问题优化

（一）TSP问题的遗传算法设计

采用遗传算法进行路径优化，分为以下几步：

第一步：参数编码和初始群体设定

一般来说遗传算法对解空间的编码大多采用二进制编码形式，但对于TSP一类排序问题，采用对访问城市序列进行排列组合的方法编码，即某个巡回路径的染色体个体是该巡回路径的城市序列。

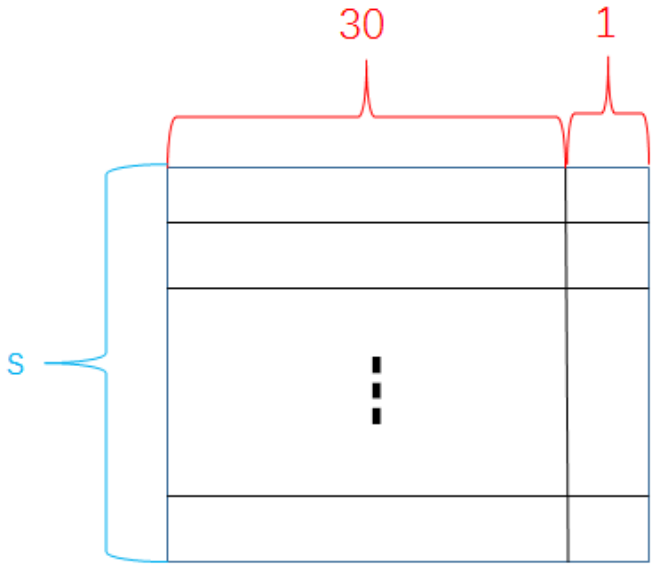
针对TSP问题，编码规则通常是取N进制编码，即每个基因仅从1到N的整数里面取一个值，每个个体的长度为N，N为城市总数。定义一个s行t列的pop矩阵来表示群体，t为城市个数N+1，即N+1，s为样本中个体数目。针对30个城市的TSP问题，t取值31，

10.6 基于遗传算法的TSP问题优化

即矩阵每一行的前30个元素表示经过的城市编号，
最后一个元素表示经过这些城市要走的距离。

参数编码和初始群体设定程序为：

- pop=zeros(s,t);
- for i=1:s
- pop(i,1:t-1)=randperm(t-1);
- end



10.6 基于遗传算法的TSP问题优化

第二步：计算路径长度的函数设计

在TSP的求解中，用距离的总和作为适应度函数，来衡量求解结果是否最优。将POP矩阵中每一行表示经过的距离的最后一个元素作为路径长度。

两个城市m和n间的距离为：

$$d_{mn}=\sqrt{\left(x_m-x_n\right)^2+\left(y_m-y_n\right)^2} \qquad (10.4)$$

用于计算路径长度的程序为chap10_1dis.m。

通过样本的路径长度可以得到目标函数和自适应度函数。根据t的定义，两两城市组合数共有t-2组，则目标函数为：

$$J(t)=\sum_{j=1}^{t-2}d(j) \qquad (10.5)$$

自适应度函数取目标函数的倒数，即：

$$f(t)=\frac{1}{J(t)} \qquad (10.6)$$

10.6 基于遗传算法的TSP问题优化

第三步：计算选择算子

选择就是从群体中选择优胜个体、淘汰劣质个体的操作，它是建立在群体中个体适应度评估基础上。仿真中采用最优保存方法，即将群体中适应度最大的c个个体直接替换适应度最小的c个个体。选择算子函数为chap10_4select.m。

第四步：计算交叉算子

交叉算子在遗传算法中起着核心的作用，它是指将个体进行两两配对，并以交叉概率pc将配对的父代个体加以替换重组而生成新个体的操作。如果当前随机值大于pc，则随机选择两个个体进行交叉。

10.6 基于遗传算法的TSP问题优化

有序交叉法实现的步骤是：

步骤 1 随机选取两个交叉点crosspoint(1)和crosspoint(2)；

步骤 2 两后代X_1'和X_2'先分别按对应位置复制双亲X_1和X_2匹配段中的两个子串A1和B1；

步骤 3 在对应位置交换X_1和X_2双亲匹配段A1和B1以外的城市，

如果交换后，后代X_1'中的某一城市a与X_1'子串中A1的城市重复，则在子传B1中找到与子串A1中城市a对应位置处的城市b,并用城市b取代城市a。

如果城市b与X_1'子串A1中的城市还重复，则在子串B1中找到与子串A1中b处对应位置处的城市c，并用城市c取代城市b，直到X_1'中的城市均不重复为止，对后代X_2'也采用同样方法，如图10-4所示：

10.6 基于遗传算法的TSP问题优化

X1:	9	8		4	5	6	7	1		3	2	0
						A1						
X2:	8	7		1	4	0	3	2		9	6	5
						B1						
						⇓						
X_1':	8	3		4	5	6	7	1		9	0	2
						A1						
X_2':	9	8		1	4	0	3	2		7	5	6

图10-4 有序交叉算子

10.6 基于遗传算法的TSP问题优化

以图10-4例，具体的有序交叉法实现步骤为：

（1）确定两个交叉点的位置分别为2和7；

（2）两后代X_1'和X_1'分别按对应位置复制双亲X_1和X_2匹配段中两个子串A_1和B_1，在对应位置交换X_1和X_2双亲匹配段A_1和B_1以外的城市，得到

X_1'：8 7 | 4 5 6 7 1 | 9 6 5。

（3）可以看到X_1'中的位置1处的元素8与A_1中没有重复，故位置1为8，位置2处的元素7与A_1中的第4个元素有重复，则用B_1中的第4个元素3来代替，且元素3与A_1中的元素没有重复，故X_1'中的位置2处的元素为城市3，此时X_1'：8 3 | 4 5 6 7 1 | 9 6 5。

10.6 基于遗传算法的TSP问题优化

X1: 9 8 | 4 5 6 7 1 | 3 2 0

X2: 8 7 | 1 4 0 3 2 | 9 6 5

（4） X_1'中第8个元素为9，此元素与A1中的元素没有重复，故X_1'中的位置8处为城市9， X_1'中的第9个元素为6，与A1中的第3个元素重复，则用B1中的第3个元素0取代，且0不与A1中的元素重复，则 X_1'中第9个元素为城市0，此时 X_1'：8 3 | 4 5 6 7 1 | 9 0 5。

（5） X_1'中第10个元素为5，此元素与A1中的位置2处的元素重复，则用B1中的位置2处的元素4取代，元素4仍与A1中的位置1处的元素重复，则用B1中位置1处的元素1取代，元素1仍与A1中的位置5处的元素重复，则用B1中位置5处的元素2取代，此时不与A1中的元素重复，故 X_1'中的第10个元素为城市2。

重复检查完成后，得到后代 X_1'，即

X_1'：8 3 | 4 5 6 7 1 | 9 0 2。至此 X_1' 交叉完毕。同理可得后代 X_2'，即X_2'：9 8 | 1 4 0 3 2 | 7 5 6。

10.6 基于遗传算法的TSP问题优化

从图10-4可知，有序交叉算子能够有效地继承双亲的部分基因成分，达到了进化过程中的遗传功能，使该算法并不是盲目搜索，而是趋向于使群体具有更多的优良基因，最后实现寻优的目的。交叉算子函数为

chap10_2corss.m

10.6 基于遗传算法的TSP问题优化

第五步：计算变异算子

变异操作是以变异概率P_m对群体中个体串某些基因位上的基因值作变动，若变异后子代的适应度值更加优异，则保留子代染色体，否则，仍保留父代染色体。

这里采用倒置变异法：假设当前个体X为(1 3 7 4 8 0 5 9 6 2)，如果当前随机概率值大于 P_m ，则随机选择来自同一个体的两个点mutatepoint(1)和mutatepoint(2)，然后倒置该两个点的中间部分，产生新的个体。

- 【例】假设随机选择个体X的两个点“7”和“9”，则倒置该两个点的中间部分，即将“4805”变为“5084”，产生新的个体X为(1 3 7 5 0 8 4 9 6 2)。变异算子函数为chap10_2mutate.m。

10.6 基于遗传算法的TSP问题优化

（三）仿真实例

分别以8个城市和30个城市的路径优化为例，其城市路径坐标保存在当前路径的文件cities8.txt和cities30.txt中。

8个城市优化时，遗传算法参数设定为：群体中个体数目S=30，交叉概率 P_c=0.10，变异概率P_m=0.80。通过改变进化代数为k，观察不同进化代数下路径的优化情况，经过50次进化，城市组合路径达到最小。最短路程为2.8937，如图10-5所示。仿真过程表明，在100次仿真实验中，有98次以上可收敛到最优解。

10.6 基于遗传算法的TSP问题优化

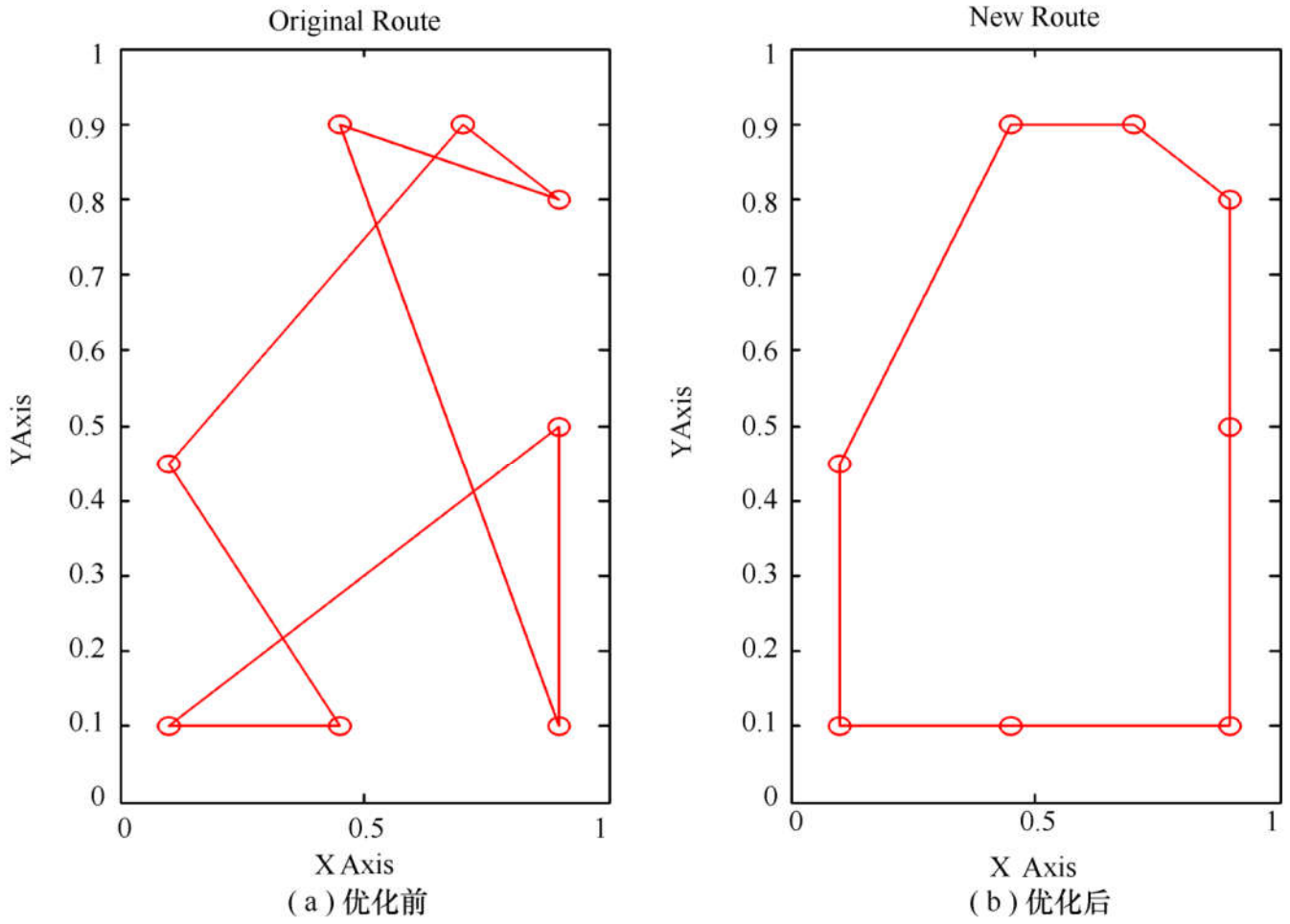


图10-5　8城市进化次数为50时的优化效果，距离L=2.8937(M=1)

10.6 基于遗传算法的TSP问题优化

30个城市优化时，遗传算法参数设定为：群体中个体数目s=1500，交叉概率P_c=0.10，变异概率P_m=0.80。取c=25，经过300次进化，城市组合路径达到最小。最短路程为424.8693，如图10-6所示。

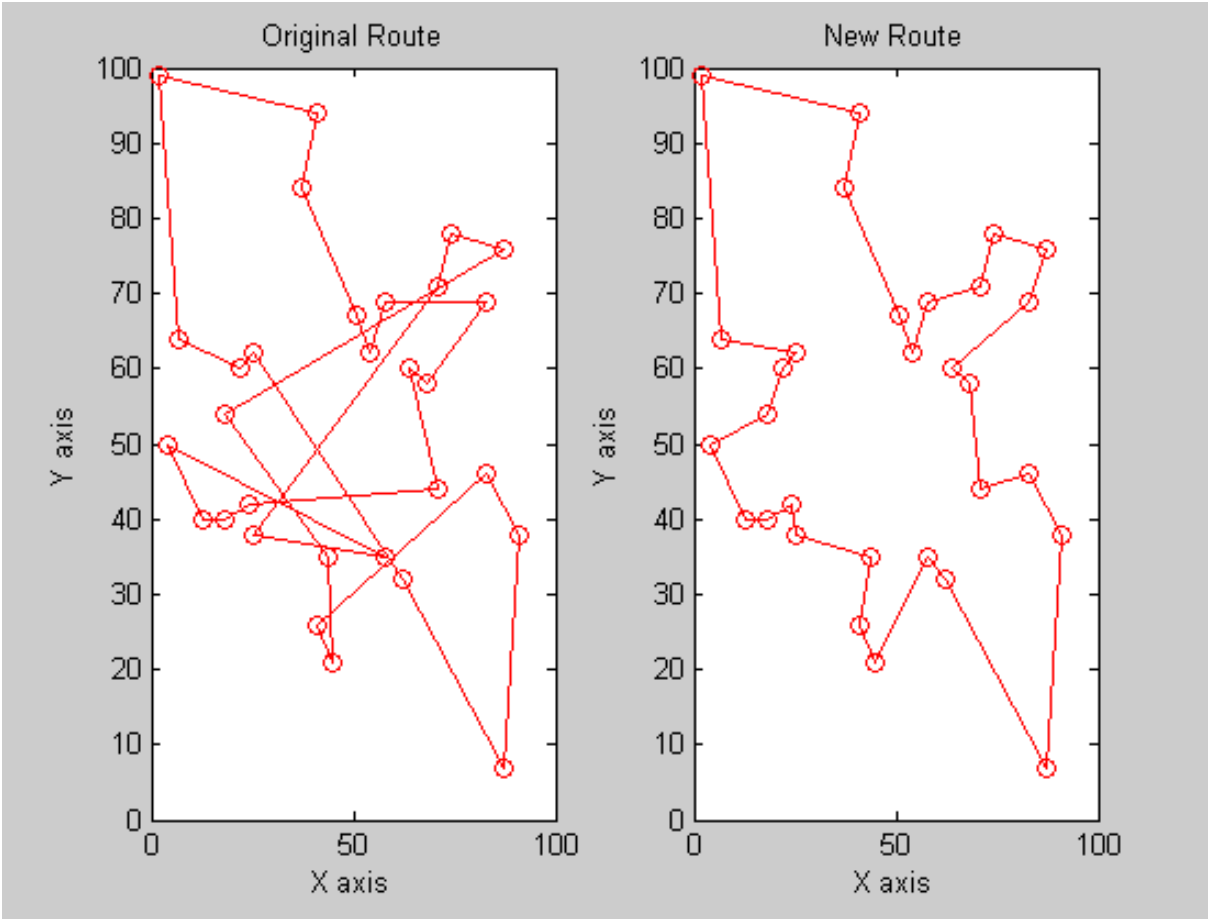


图10-6 30城市进化次数为300时的优化效果，距离L=424.8693（M=2）

10.7 粒子群优化算法

粒子群算法，也称粒子群优化算法(Particle Swarm Optimization)，缩写为PSO。粒子群优化算法也是一种进化计算技术，1995年由Eberhart博士和Kennedy博士提出，是通过模拟鸟群捕食行为设计的一种群智能算法。区域内有大大小小不同的食物源，鸟群的任务是找到最大的食物源（全局最优解），鸟群的任务是找到这个食物源。鸟群在整个搜寻的过程中，通过相互传递各自位置的信息，让其他的鸟知道食物源的位置最终，整个鸟群都能聚集在食物源周围，即我们所说的找到了最优解，问题收敛。

10.7 粒子群优化算法

PSO算法属于进化算法的一种，和遗传算法相似，

- ✓ 它也是从随机解出发，
- ✓ 通过迭代寻找最优解，
- ✓ 也是通过适应度来评价解的品质，

但它比遗传算法规则更为简单，

- ✓ 没有遗传算法的“交叉”和“变异”操作，
- ✓ 通过追随当前搜索到的最优值来寻找全局最优。

这种算法以其实现容易、精度高、收敛快等优点引起了学术界的重视，并且在解决实际问题中展示了其优越性。目前已广泛应用于函数优化、系统辨识、模糊控制等应用领域。

10.7 粒子群优化算法

（一）标准粒子群算法

PSO算法模拟鸟群的捕食行为。设想这样一个场景：一群鸟随机搜索食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远。那么找到食物的最优策略就是搜寻目前离食物最近的鸟的周围区域。

10.7 粒子群优化算法

PSO算法从这种模型中得到启示并用于解决优化问题，算法中，每个优化问题的解都是搜索空间中的一只鸟，称之为“粒子”。所有的粒子都有一个由被优化的函数决定的适应度值，适应度值越大越好。每个粒子还有一个速度决定他们飞行的方向和距离，粒子们追随当前的最优粒子在解空间中搜索。

PSO算法首先初始化为一群随机粒子(随机解)，然后通过迭代找到最优解。在每一次迭代中，粒子通过跟踪两个“极值”来更新自己的位置：

- ✓ 第一个极值是粒子本身所找到的最优解，这个解叫做个体极值。
- ✓ 另一个极值是整个种群目前找到的最优解，这个极值称为全局极值。另外也可以不用整个种群而只是用其中一部分作为粒子的邻居，称为局部极值，那么在所有邻居中的极值就是全局极值。

10.7 粒子群优化算法

（二）粒子群算法的参数设置

应用PSO算法解决优化问题的过程中有两个重要的步骤: **问题解的编码和适应度函数**。

(1) 编码：PSO的一个优势就是采用实数编码，例如对于问题 $f(x)=x_1^2+x_2^2+x_3^2$ 求最大值, 粒子可以直接编码为(x_1,x_2,x_3)，而适应度函数就是$f(x)$。

(2) PSO中需要调节的参数如下：

a) 粒子数：一般取20-40，对于比较难的问题, 粒子数可以取到100 或 200；

b) 最大速度V_{max}：决定粒子在一个循环中最大的移动距离，通常小于粒子的范围宽度。较大的V_{max}可以保证粒子种群的全局搜索能力，较小的V_{max}则使粒子种群的局部搜索能力加强；

10.7 粒子群优化算法

c)学习因子：c_1和c_2 通常可设定为2.0。c_1为局部学习因子， c_2为全局学习因子，一般取c_2大一些；

d)惯性权重：一个大的惯性权值有利于展开全局寻优，而一个小的惯性权值有利于局部寻优。当粒子的最大速度V_{max}很小时，使用接近于1的惯性权重。当V_{max}不是很小时，使用权重 w=0.80 较好。

还可使用时变权重。如果在迭代过程中采用线性递减惯性权值，则粒子群算法在开始时具有良好的全局搜索性能，能够迅速定位到接近全局最优点的区域，而在后期具有良好的局部搜索性能，能够精确的得到全局最优解。经验表明，惯性权重采用从0.90线性递减到0.10的策略，会获得比较好的算法性能；

e)中止条件：最大循环数或最小误差要求。

10.7 粒子群优化算法

（三）粒子群算法的基本流程

（1）初始化：设定参数运动范围，设定学习因子c_1, c_2，最大进化代数G，当前的进化代数kg。在一个 D 维参数的搜索解空间中，粒子组成的种群规模大小为Size，每个粒子代表解空间的一个候选解，其中第 i $(1 \le i \le \text{Size})$个粒子在整个解空间的位置表示为$X_i$，速度表示为$V_i$。第$i$个粒子从初始到当前迭代次数搜索产生的最优解为个体极值P_i，整个种群目前的最优解为BestS。随机产生Size个粒子，随机产生初始种群的位置矩阵和速度矩阵。

```
%初始化粒子群算法参数
min=-2.048;max=2.048;% 粒子位置范围
Vmax=1;Vmin=-1;% 粒子运动速度范围
c1=1.3;c2=1.7;   % 学习因子[0,4]

wmin=0.10;wmax=0.90;% 惯性权重
G=100;              % 最大迭代次数
Size=50;            % 初始化群体个体数目
```

10.7 粒子群优化算法

（2）个体评价(适应度评价)：将各个粒子初始位置作为个体极值，计算群体中各个粒子的初始适应值$f(X_i)$，并求出种群最优位置。

（3）更新粒子的速度和位置，产生新种群，并对粒子的速度和位置进行越界检查，为避免算法陷入局部最优解，加入一个局部自适应变异算子进行调整。

$$V_i^{kg+1} = w(t) \times V_i^{kg} + c_1 r_1 \left(p_i^{kg} - X_i^{kg} \right) + c_2 r_2 \left(\mathrm{BestS}_i^{kg} - X_i^{kg} \right) \quad (10.7)$$

$$X_i^{kg+1} = X_i^{kg} + V_i^{kg+1} \quad (10.8)$$

其中kg=1,2,…,G, i=1,2,…,Size，r_1和r_2为[0, 1]范围内的随机数，c_1为局部学习因子，c_2为全局学习因子，一般取c_2大些。

```
for i=1:G                    %采用时变权重
    w(i)=wmax-((wmax-wmin)/G)*i;
end
```

10.7 粒子群优化算法

对粒子的速度和位置进行越界检查

```
v(i,:)=w(kg)*v(i,:)+c1*rand*(Xl(i,:)-
X(i,:))+c2*rand*(BestS-X(i,:));%加权，
实现速度的更新
for j=1:CodeL   %检查速度是否越界
   if v(i,j)<Vmin
     v(i,j)=Vmin;
   elseif  v(i,j)>Vmax
     v(i,j)=Vmax;
   end
end
```

$$V_i = \begin{cases} V_{max}, & V_i > V_{max} \\ -V_{max}, & V_i < -V_{max} \\ V_i, & \text{otherwise.} \end{cases}$$

```
X(i,:)=X(i,:)+v(i,:); %实现位置的更新
    for j=1:CodeL%检查位置是否越界
       if X(i,j)<MinX(j)
          X(i,j)=MinX(j);
       elseif X(i,j)>MaxX(j)
          X(i,j)=MaxX(j);
       end
end
```

为避免算法陷入局部最优解，加入一个局部自适应变异算子进行调整

```
%自适应变异,避免陷入局部最优
    if rand>0.8
        k=ceil(4*rand);   %ceil为向上取整
        X(i,k)=5*rand;
    end
```

10.7 粒子群优化算法

（4）比较粒子的当前适应值 $f(X_i)$ 和自身历史最优值p_i，如果 $f(X_i)$ 优于p_i，则置p_i为当前值$f(X_i)$，并更新粒子位置。

（5）比较粒子当前适应值$f(X_i)$ 与种群最优值BestS。如果$f(X_i)$ 优于BestS，则置BestS为当前值$f(X_i)$，更新种群全局最优值。

（6）检查结束条件，若满足，则结束寻优；否则$kg=kg+1$，转至(3)。结束条件为寻优达到最大进化代数，或评价值小于给定精度。

$$V_i^{kg+1} = w(t) \times V_i^{kg} + c_1 r_1 \left(p_i^{kg} - X_i^{kg} \right) + c_2 r_2 \left(\mathrm{BestS}_i^{kg} - X_i^{kg} \right)$$

式(10.7)由三部分组成：

- 惯性或动量部分，反应粒子的运动习惯。
- 认知部分，粒子有向自身历史最佳位置逼近的优势。
- 社会部分，粒子有向群体或领域历史最佳位置逼近的趋势。

10.7 粒子群优化算法

$$V_i^{kg+1} = w(t) \times V_i^{kg} + c_1 r_1 \left(p_i^{kg} - X_i^{kg} \right) + c_2 r_2 \left(\mathrm{BestS}_i^{kg} - X_i^{kg} \right)$$

- V_{max}决定在当前位置与最好位置之间的区域的分辨率（或精度）。如果V_{max}太高，微粒可能会飞过好解，如果V_{max}太小，微粒不能进行足够的探索，导致陷入局部优值。该限制有三个目的：防止计算溢出；实现人工学习和态度转变；决定问题空间搜索的粒度。

- 惯性权重w使微粒保持运动的惯性，使其有扩展搜索空间的趋势，有能力探索新的区域。

- 加速常数c1和c2代表将每个微粒推向pbest和gbest位置的统计加速项的权重。低的值允许微粒在被拉回来之前可以在目标区域外徘徊，而高的值导致微粒突然的冲向或者越过目标区域。

- 如果没有后两部分，即$c_1 = c_2 = 0$，微粒将一直以当前的速度飞行，直到到达边界。由于它只能搜索有限的区域，将很难找到好的解。

10.7 粒子群优化算法

$$V_i^{kg+1} = w(t) \times V_i^{kg} + c_1 r_1 \left(p_i^{kg} - X_i^{kg} \right) + c_2 r_2 \left(\text{BestS}_i^{kg} - X_i^{kg} \right)$$

如果没有第一部分，即w = 0，则速度只取决于微粒当前的位置和它们历史最好位置pbest和gbest，速度本身没有记忆性。

- ✓ 假设一个微粒位于全局最好位置，它将保持静止。而其它微粒则飞向它本身最好位置pbest和全局最好位置gbest的加权中心。
- ✓ 在这种条件下，微粒群将统计地收缩到当前的全局最好位置，更象一个局部算法。

在加上第一部分后，微粒有扩展搜索空间的趋势，即第一部分有全局搜索的能力。这也使得w的作用为针对不同的搜索问题，调整算法全局和局部搜索能力的平衡。

如果没有第二部分，即$c_1 = 0$，则微粒没有认知能力，也就是“只有社会（social-only）”的模型。在微粒的相互作用下，有能力到达新的搜索空间。它的收敛速度比标准版本更快，但是对复杂问题，比标准版本更容易陷入局部优值点。

如果没有第三部分，即$c_2 = 0$，则微粒之间没有社会信息共享，也就是“只有认知（cognition-only）”的模型。因为个体间没有交互，一个规模为m的群体等价于m个单个微粒的运行。因而得到解的几率非常小。

10.7 粒子群优化算法

$f(X_i)$

$$V_i^{kg+1} = w(t) \times V_i^{kg} + c_1 r_1 \left(p_i^{kg} - X_i^{kg} \right) + c_2 r_2 \left(\mathrm{BestS}_i^{kg} - X_i^{kg} \right)$$

$$X_i^{kg+1} = X_i^{kg} + V_i^{kg+1}$$

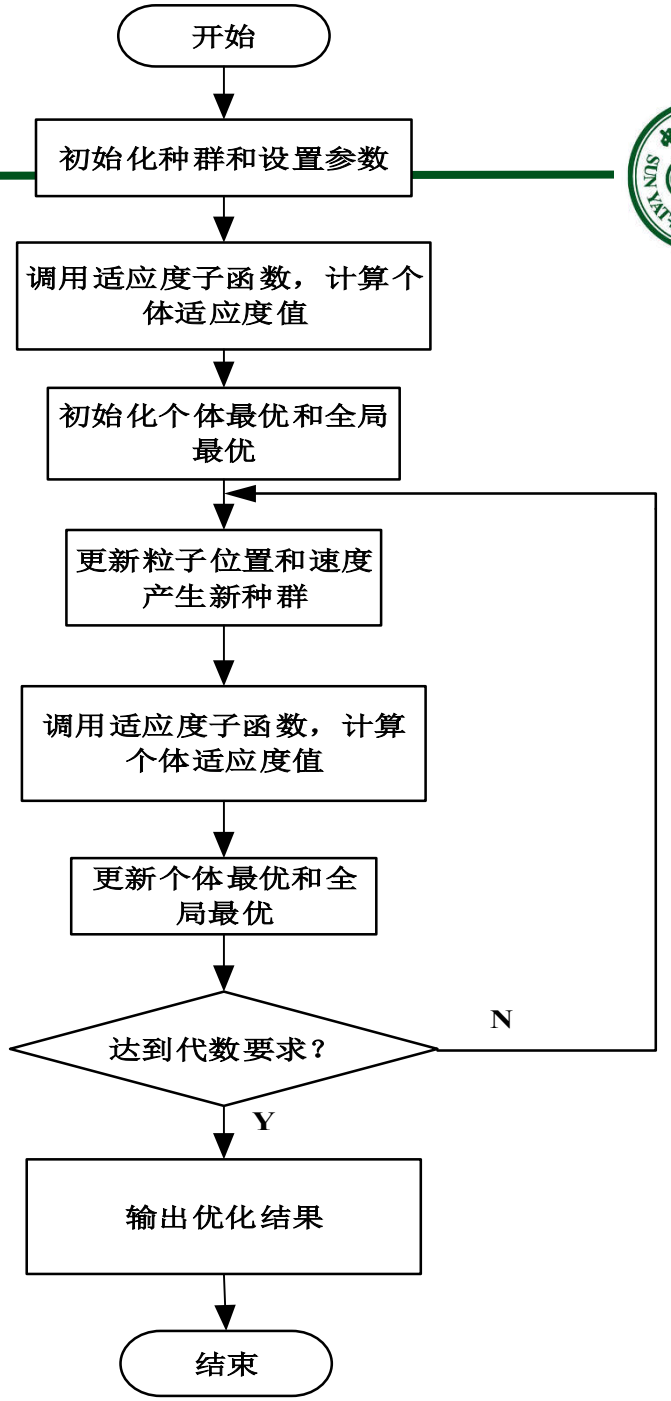


图10-7　PSO的算法流程图

10.8 粒子群算法的函数优化

（一）基于粒子群算法的函数优化

利用粒子群算法求Rosenbrock函数的极大值

$$\begin{cases} f(x_1, x_2) = 100(x_1^{\,2} - x_2)^2 + (1 - x_1)^2 \\ -2.048 \le x_i \le 2.048 \qquad (i = 1, 2) \end{cases} \tag{10.18}$$

该函数有两个局部极大点，分别是 $f(2.048, -2.048) = 3897.7342$ 和 $f(-2.048, -2.048) = 3905.9262$ ，其中后者为全局最大点。

全局粒子群算法中，粒子 i 的邻域随着迭代次数的增加而逐渐增加，开始第一次迭代，它的邻域粒子的个数为0，随着迭代次数邻域线性变大，最后邻域扩展到整个粒子群。全局粒子群算法收敛速度快，但容易陷入局部最优。而局部粒子群算法收敛速度慢，但可有效避免局部最优。

10.8 粒子群算法的函数优化

全局粒子群算法中，每个粒子的速度的更新是根据粒子自己历史最优值P_i和粒子群体全局最优值 P_g（BestS）。为了避免陷入局部极小，可采用局部粒子群算法，每个粒子速度更新根据粒子自己历史最优值P_i和粒子邻域内粒子的最优值P_{local}。

局部粒子群算法中，按如下两式更新粒子的速度和位置：

$$V_i^{kg+1} = w(t) \times V_i^{kg} + c_1 r_1 \left(p_i^{kg} - X_i^{kg} \right) + c_2 r_2 \left(p_{i\text{local}}^{kg} - X_i^{kg} \right) \tag{10.9}$$

$$X_i^{kg+1} = X_i^{kg} + V_i^{kg+1} \tag{10.10}$$

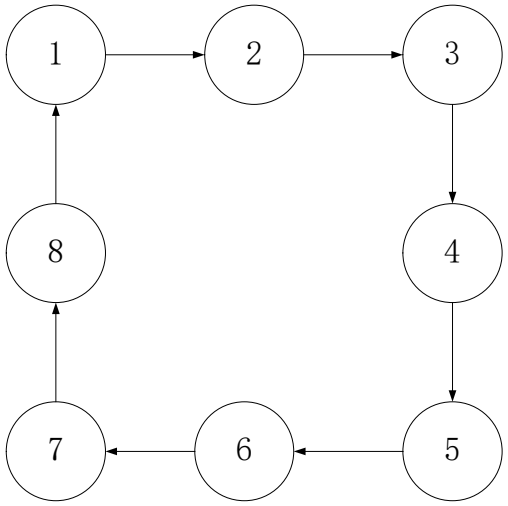
式中，$p_{i\text{local}}^{kg}$为局部寻优的粒子。

同样，对粒子的速度和位置要进行越界检查，为避免算法陷入局部最优解，加入一个局部自适应变异算子进行调整。

10.8 粒子群算法的函数优化

根据取邻域的方式的不同，局部粒子群算法有很多不同的实现方法。本节采用最简单的环形邻域法，如下图所示。

以8个粒子为例说明局部粒子群算法，在每次进行速度和位置更新时，粒子1追踪1、2、8三个粒子中的最优个体，粒子2追踪1、2、3三个粒子中的最优个体，依次类推。仿真中，求解某个粒子邻域中的最优个体是由函数chap8_3lbest.m来完成。




```
function f =evaluate_localbest(x1,x2,x3)
%求解粒子环形邻域中的局部最优个体
K0=[x1;x2;x3];
K1=[chap10_3func(x1),chap10_3func(x2),cha
p10_3func(x3)];
[maxvalue index]=max(K1);
plocalbest=K0(index,:);
f=plocalbest;
```

10.8 粒子群算法的函数优化

采用实数编码求函数极大值，用2个实数分别表示两个决策变量x_1和x_2，分别将x_1和x_2 的定义域离散化为从离散点-2.048到离散点2.048 的*Size*个实数。个体的适应度直接取为对应的目标函数值，越大越好。即取适应度函数为$F(x)=f(x_1, x_2)$ 。

在粒子群算法仿真中，取粒子群个数为 *Size* =50，最大迭代次数G=100，粒子运动最大速度为V_{max}=1.0，即速度范围为[-1,1]。学习因子取c_1=1.4, c_2=1.7，采用线性递减的惯性权重，惯性权重采用从0.90线性递减到0.10的策略。

10.8 粒子群算法的函数优化

根据M的不同可采用不同的粒子群算法。取M=2，采用局部粒子群算法。按式(10.9)和式(10.10)更新粒子的速度和位置，产生新种群。经过100步迭代，最佳样本为BestS=[-2.048,2.048]，即当x_1=-2.048, x_2=-2.048 时， 函数具有极大值，极大值为3905.9。

适应度函数F的变化过程如图10-9所示，由仿真可见，随着迭代过程的进行，粒子群通过追踪自身极值和局部极值，不断更新自身的速度和位置，从而找到全局最优解。通过采用局部粒子群算法，增强了算法的局部搜索能力，有效地避免了陷入局部最优解，仿真结果表明正确率在95%以上。

10.8 粒子群算法的函数优化

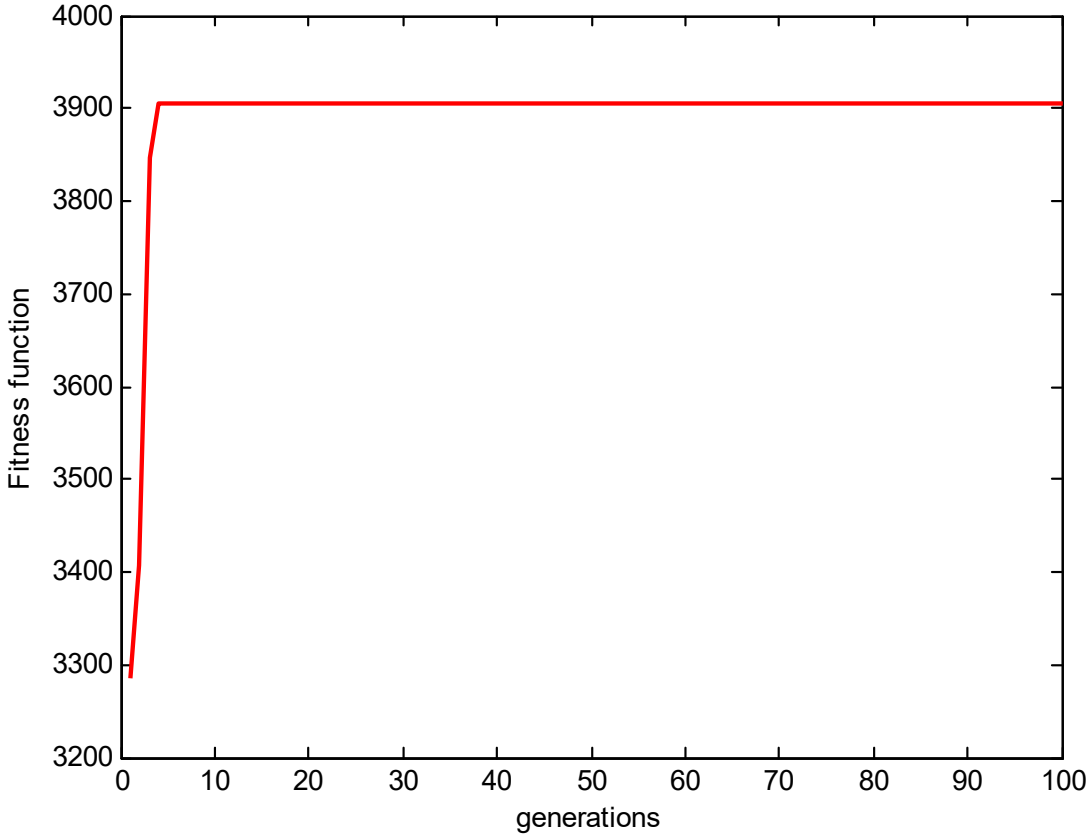


图10-9 适应度函数的优化过程

10.8 粒子群算法的函数优化

（二）基于粒子群算法的参数辨识

利用粒子群算法辨识非线性静态模型参数

$$y=\begin{cases}0\\ k_1\left(x-g\,\mathrm{sgn}\left(x\right)\right)\\ k_2\left(x-h\,\mathrm{sgn}\left(x\right)\right)+k_1\left(h-g\right)\mathrm{sgn}\left(x\right)\end{cases} \tag{10.11}$$

辨识参数集为 $\hat{\boldsymbol{\theta}}=\begin{bmatrix}\hat{g} & \hat{h} & \hat{k}_1 & \hat{k}_2\end{bmatrix}$，真实参数为

$$\boldsymbol{\theta}=\begin{bmatrix}g & h & k_1 & k_2\end{bmatrix}=\begin{bmatrix}1 & 2 & 1 & 0.5\end{bmatrix}$$

采用实数编码，辨识误差指标取

$$J=\sum_{i=1}^{N}\frac{1}{2}\left(y_i-\hat{y}_i\right)^{\mathrm{T}}\left(y_i-\hat{y}_i\right) \tag{10.12}$$

其中N为测试数据的数量，y_i为模型第i个测试样本的输出。

10.8 粒子群算法的函数优化

首先运行模型测试程序chap10_4.m，对象的输入样本区间为[-4,4]，步长为0.10，由式（10.11）计算样本输出值，共有81对输入输出样本对。

在粒子群算法仿真程序chap10_5.m中，将待辨识的参数向量记为X，取粒子群个数为Size=80，最大迭代次数G=500，采用实数编码，四个参数的搜索范围均为[0,5]，粒子运动最大速度为V_{max}=1.0，即速度范围为[-1,1]。学习因子取c_1=1.3，c_2=1.7，采用线性递减的惯性权重，惯性权重采用从0.90线性递减到0.10的策略。目标函数的倒数作为粒子群的适应度函数。将辨识误差指标直接作为粒子的目标函数，越小越好。

10.8 粒子群算法的函数优化

按式(10.7)和式(10.8)更新粒子的速度和位置，产生新种群，辨识误差函数 的优化过程如图10-10所示。

辨识结果为

$$\hat{\boldsymbol{\theta}}=\begin{bmatrix}\hat{g} & \hat{h} & \hat{k}_1 & \hat{k}_2\end{bmatrix}$$

$$=\begin{bmatrix}0.999999930217796 & 2.000000160922045 & 0.999999322205419 & 0.500000197043791\end{bmatrix}$$

最终的辨识误差指标为

$$J = 3.6166\times10^{-12}$$

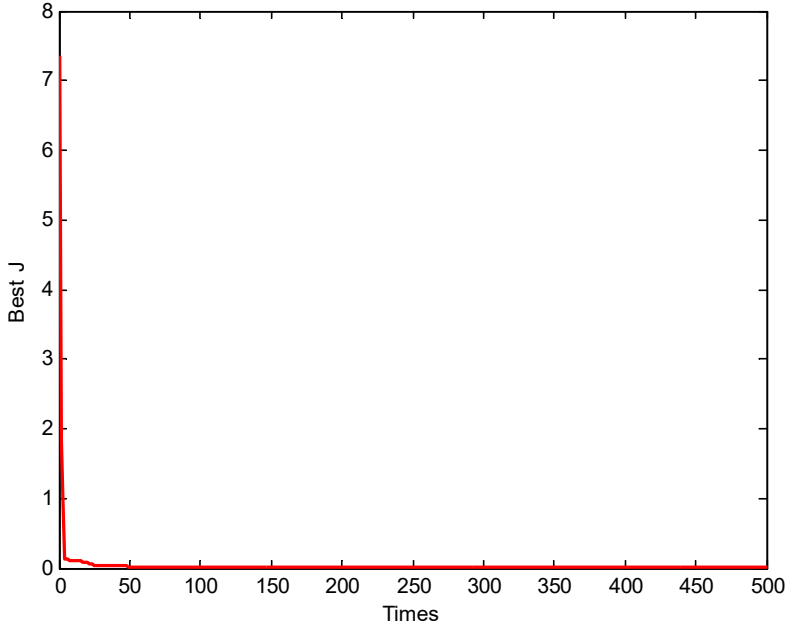


图10-10 辨识误差函数 的优化过程

10.9 差分进化算法

差分进化（Differential Evolution，DE）算法是模拟自然界生物种群以“优胜劣汰、适者生存”为原则的进化发展规律而形成的一种随机启发式搜索算法，是一种新兴的进化计算技术。它于1995年由Rainer Storn和Kenneth Price提出。由于其简单易用、稳健性好以及强大的全局搜索能力，使得差分进化算法已在多个领域取得成功。

10.9 差分进化算法

差分进化算法保留了基于种群的全局搜索策略，采用实数编码、基于差分的简单变异操作和一对一的竞争生存策略，降低了遗传操作的复杂性。同时，差分进化算法特有的记忆能力使其可以动态跟踪当前的搜索情况，以调整其搜索策略，具有较强的全局收敛能力和鲁棒性，且不需要借助问题的特征信息，适于求解一些利用常规的数学规划方法所无法求解的复杂环境中的优化问题，采用差分进化算法可实现复杂系统的参数辨识。

实验结果表明，差分进化算法的性能优于其它进化算法，该算法已成为一种求解非线性、不可微、多极值和高维的复杂函数的一种有效和鲁棒的方法。

10.9 差分进化算法

（一）标准差分进化算法

差分进化算法是基于群体智能理论的优化算法，通过群体内个体间的合作与竞争产生的群体智能指导优化搜索。它保留了基于种群的全局搜索策略，采用实数编码、基于差分的简单变异操作和一对一的竞争生存策略，降低了遗传操作的复杂性，同时它特有的记忆能力使其可以动态跟踪当前的搜索情况已调整其搜索策略。具有较强的全局收敛能力和鲁棒性。

差分进化算法的主要优点可以总结为以下三点：

待定参数少；不易陷入局部最优；收敛速度快。

10.9 差分进化算法

差分进化算法根据父代个体间的差分矢量进行变异、交叉和选择操作，其基本思想是从某一随机产生的初始群体开始，通过把种群中任意两个个体的向量差加权后按一定的规则与第三个个体求和来产生新个体，然后将新个体与当代种群中某个预先决定的个体相比较，如果新个体的适应度值优于与之相比较的个体的适应度值，则在下一代中就用新个体取代旧个体，否则旧个体仍保存下来，通过不断地迭代运算，保留优良个体，淘汰劣质个体，引导搜索过程向最优解逼近。

在优化设计中，差分进化算法与传统的优化方法相比，具有以下主要特点：

10.9 差分进化算法

（1）差分进化算法从一个群体即多个点而不是从一个点开始搜索，这是它能以较大的概率找到整体最优解的主要原因；

（2）差分进化算法的进化准则是基于适应性信息的，无须借助其它辅助性信息（如要求函数可导或连续），大大地扩展了其应用范围；

（3）差分进化算法具有内在的并行性，这使得它非常适用于大规模并行分布处理，减小时间成本开销；

（4）差分进化算法采用概率转移规则，不需要确定性的规则。

10.9 差分进化算法

（二）差分进化算法的基本流程

差分进化算法是基于实数编码的进化算法，整体结构上与其它进化算法类似，由变异、交叉和选择三个基本操作构成。标准差分进化算法主要包括以下4个步骤：

（1）生成初始群体

在 n维空间里随机产生满足约束条件的M个个体，实施措施如下：

$$x_{ij}(0)=\mathrm{rand}_{ij}(0,1)\left(x_{ij}^{\mathrm{U}}-x_{ij}^{\mathrm{L}}\right)+x_{ij}^{\mathrm{L}} \qquad (10.13)$$

其中x_{ij}^{U} 和x_{ij}^{L}分别是第j 个染色体的上界和下界，$\mathrm{rand}_{ij}(0,1)$ 是[0,1]之间的随机小数。

10.9 差分进化算法

（2）变异操作

从群体中随机选择3个个体$x_{\mathrm{p1},}\ x_{\mathrm{p2}}$和$x_{\mathrm{p3}}$，且$i\neq p_1\neq p_2\neq p_3$，则基本的变异操作为

$$h_{ij}\left(t+1\right)=x_{\mathrm{p1j}}\left(t\right)+F\left(x_{\mathrm{p2j}}\left(t\right)-x_{\mathrm{p3j}}\left(t\right)\right) \tag{10.14}$$

如果无局部优化问题，变异操作可写为

$$h_{ij}\left(t+1\right)=x_{\mathrm{bj}}\left(t\right)+F\left(x_{\mathrm{p2j}}\left(t\right)-x_{\mathrm{p3j}}\left(t\right)\right) \tag{10.15}$$

其中 $x_{p2j}(t)$- $x_{p3j}(t)$ 为差异化向量，此差分操作是差分进化算法的关键，F为缩放因子，p_1, p_2, p_3为随机整数，表示个体在种群中的序号，$x_{bj}(t)$为当前代中种群中最好的个体。由于式（10.15）借鉴了当前种群中最好的个体信息，可加快收敛速度。

10.9 差分进化算法

（3）交叉操作

交叉操作是为了增加群体的多样性，具体操作如下：

$$v_{ij}(t+1)=\begin{cases} h_{ij}(t+1), \text{rand}\, l_{ij} \le \text{CR} \\ x_{ij}(t), \text{rand}\, l_{ij} > \text{CR} \end{cases} \tag{10.16}$$

其中 $\text{rand}\, l_{ij}$ 为[0,1]之间的随机小数，CR为交叉概率，CRϵ[0,1] 。

10.9 差分进化算法

（4）选择操作

为了确定 $x_i(t)$ 是否成为下一代的成员，试验向量 $v_i(t+1)$ 和目标向量 $x_i(t)$ 对评价函数进行比较：

$$x_i(t+1)=\begin{cases} v_i(t+1), f(v_{i1}(t+1),\cdots,v_{in}(t+1)) < f(x_{i1}(t),\cdots,x_{in}(t)) \\ x_{ij}(t), f(v_{i1}(t+1),\cdots,v_{in}(t+1)) \geq f(x_{i1}(t),\cdots,x_{in}(t)) \end{cases} \tag{10.17}$$

反复执行步骤（2）至步骤（4）操作，直至达到最大的进化代数。

10.9 差分进化算法

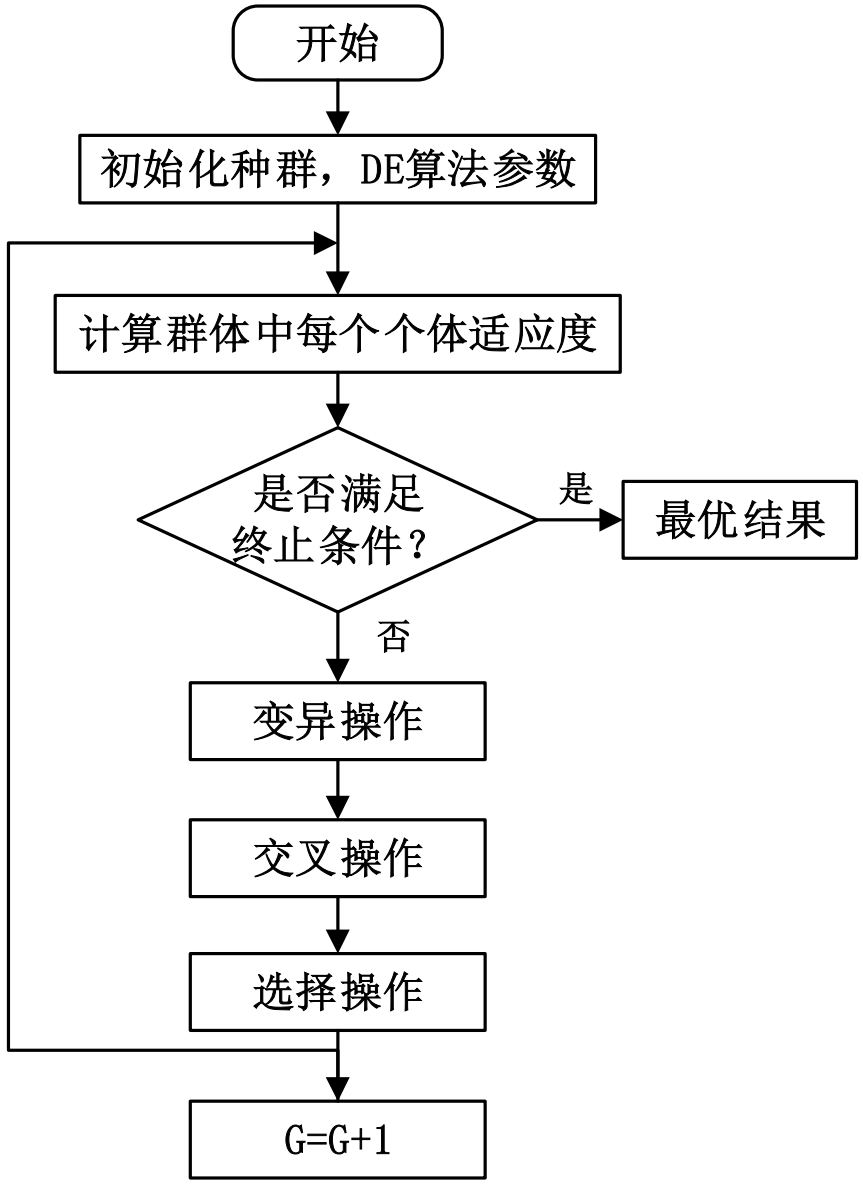


图10-11 差分进化基本运算流程

10.9 差分进化算法

（三）差分进化算法的参数设置

对于进化算法而言，为了取得理想的结果，需要对差分进化算法的各参数进行合理的设置。针对不同的优化问题，参数的设置往往也是不同的。另外，为了使差分进化算法的收敛速度得到提高，学者们针对差分进化算法的核心部分—变异向量的构造形式提出了多种的扩展模式，以适应更广泛的优化问题。

差分进化算法的运行参数主要有：缩放因子F，交叉因子CR，群体规模M和最大进化代数G。

10.9 差分进化算法

（1）变异因子F

变异因子F是控制种群多样性和收敛性的重要参数。一般在[0,2]之间取值。变异因子F值较小时，群体的差异度减小，进化过程不一跳出局部极值导致种群过早收敛。变异因子F值较大时，虽然容易跳出局部极值，但是收敛速度会减慢。一般可选在F=0.3~0.6 。

10.9 差分进化算法

（2）交叉因子CR

交叉因子CR 可控制个体参数的各维对交叉的参与程度，以及全局与局部搜索能力的平衡，一般在[0,1]之间。交叉因子CR越小，种群多样性减小，容易受骗，过早收敛。CR越大，收敛速度越大。但CR过大可能导致收敛变慢，因为扰动大于了群体差异度。根据文献一般应选在[0.6,0.9] 之间。

CR越大，F越小，种群收敛逐渐加速，但随着交叉因子CR 的增大，收敛对变异因子F的敏感度逐渐提高。

10.9 差分进化算法

（3）群体规模M

群体所含个体数量M 一般介于5D 与10D 之间(D为问题空间的维度)，但不能少于4，否则无法进行变异操作，M越大，种群多样性越强，获得最优解概率越大，但是计算时间更长，一般取20-50。

（4）最大迭代代数G

最大迭代代数G 一般作为进化过程的终止条件。迭代次数越大，最优解更精确，但同时计算的时间会更长，需要根据具体问题设定。

以上四个参数对差分进化算法的求解结果和求解效率都有很大的影响，因此，要合理设定这些参数才能获得较好的效果。

10.10 差分进化算法的函数优化与参数辨识

（一）基于差分进化算法的函数优化

利用差分进化算法求 Rosenbrock 函数的极大值

$$\begin{cases} f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \\ -2.048 \le x_i \le 2.048 \qquad (i = 1, 2) \end{cases} \tag{10.18}$$

该函数有两个局部极大点，分别是

f(2.048,-2.048)=3897.7342 和

f(-2.048,-2.048)=3905.9262，其中后者为全局最大点。

10.10 差分进化算法的函数优化与参数辨识

采用实数编码求函数极大值，用2个实数分别表示两个决策变量x_1, x_2，分别将x_1, x_2的定义域离散化为从离散点-2.048到离散点2.048 的Size个实数。个体的适应度直接取为对应的目标函数值，越大越好。即取适应度函数为$F(x) = f(x_1, x_2)$。

在差分进化算法仿真中，取F=1.2，CR=0.90，样本个数为Size=50，最大迭代次数G=30。按式(10.13)至式(10.17)设计差分进化算法，经过30步迭代，最佳样本为BestS=[-2048 -2.048]，即当x_1=-2.048，x_2=-2.048 时，函数具有极大值为3905.9。

10.10 差分进化算法的函数优化与参数辨识

适应度函数 F 的变化过程如图10-12所示，通过适当增大 F 值及增加样本数量，有效地避免了陷入局部最优解，仿真结果表明正确率接近100%。

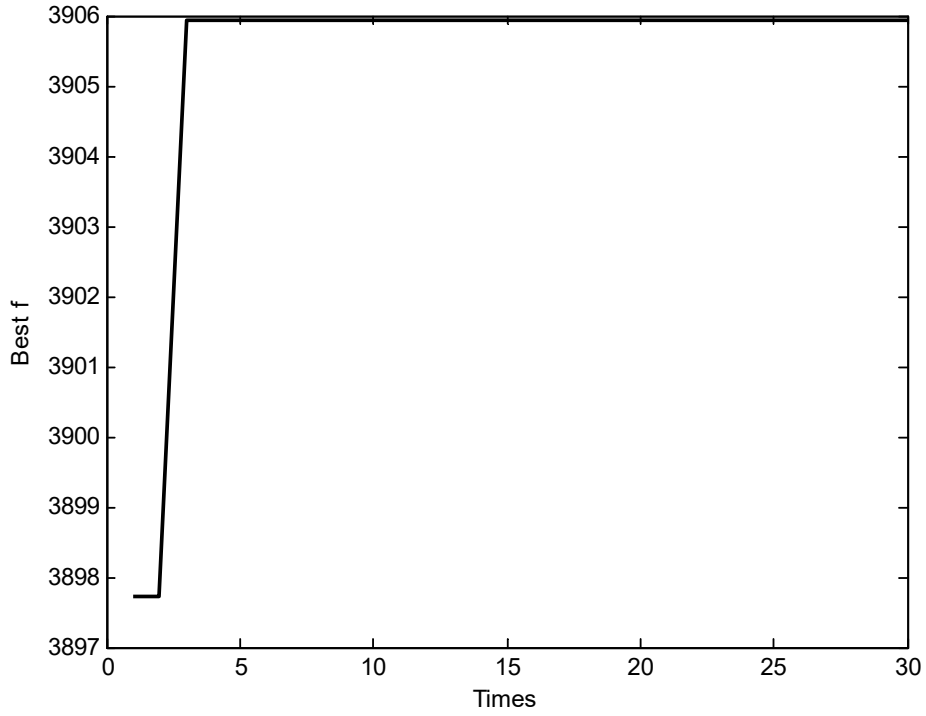


图10-12 适应度函数F的优化过程

10.10 差分进化算法的函数优化与参数辨识

（二）基于差分进化算法的参数辨识

利用差分进化算法辨识非线性静态模型参数：

$$y=\begin{cases}0\\ k_1\left(x-g\,\mathrm{sgn}\left(x\right)\right)\\ k_2\left(x-h\,\mathrm{sgn}\left(x\right)\right)+k_1\left(h-g\right)\mathrm{sgn}\left(x\right)\end{cases} \tag{10.19}$$

辨识参数集为 $\hat{\theta}=\begin{bmatrix}\hat{g} & \hat{h} & \hat{k}_1 & \hat{k}_2\end{bmatrix}$，真实参数为

$$\theta=\begin{bmatrix}g & h & k_1 & k_2\end{bmatrix}=\begin{bmatrix}1 & 2 & 1 & 0.5\end{bmatrix}$$

10.10 差分进化算法的函数优化与参数辨识

采用实数编码，辨识误差指标取:

$$J = \sum_{i=1}^{N} \frac{1}{2} \left(y_i - \hat{y}_i \right)^{\mathrm{T}} \left(y_i - \hat{y}_i \right) \qquad (10.20)$$

其中N为测试数据的数量，y_i为模型第i 个测试样本的输出。

首先运行模型测试程序chap10_7.m，对象的输入样本区间为[-4,4] 之间，步长为0.10，由式（10.19）计算样本输出值，共有81对输入输出样本对。

10.10 差分进化算法的函数优化与参数辨识

将待辨识的参数向量记为X，取样本个数为Size=200，最大迭代次数G=200，采用实数编码，四个参数的搜索范围均为[0,5] 。

在差分进化算法仿真中，取F=0.70，CR=0.60。按照式(10.13)至式(10.17)设计差分进化算法，经过200步迭代，辨识误差函数J的优化过程如图10-13所示。辨识结果为 $\hat{X}=[1 \quad 2 \quad 1 \quad 0.5]$，最终的辨识误差指标为

$$J = 9.0680 \times 10^{-23}$$

10.10 差分进化算法的函数优化与参数辨识

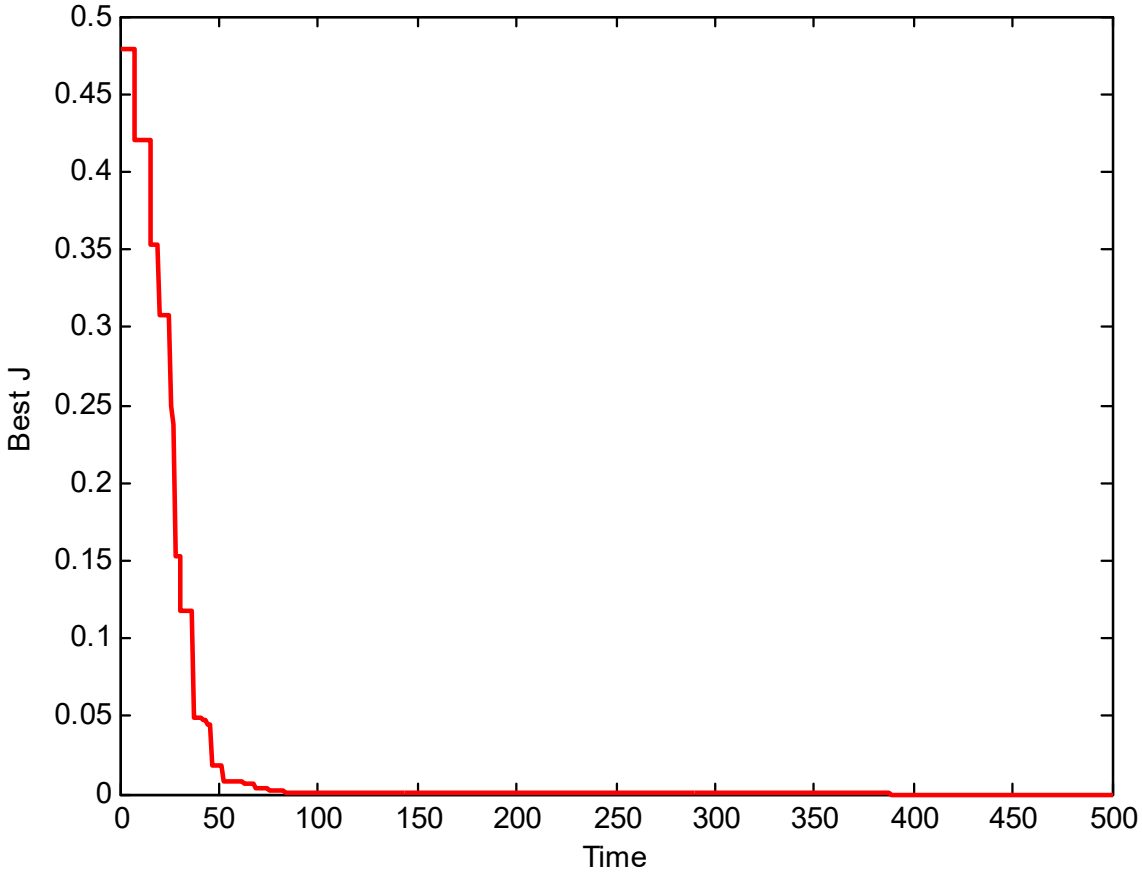


图10-13 辨识误差函数J 的优化过程

思考与练习

第十章的10-3、10-4、10-5

7月29日前交！

命名规则：姓名_学号_第十章作业

ic_sysu@163.com