计算智能

计算智能(ComputationalIntelligence，CI)是信息科学、控制科学，生命科学及智能科学与技术等不同学科相互交叉的产物，它是指采用数值计算的方去去模拟和实现人类的智能、生物智能、其他社会和自然规律。本章首先介绍计算智能的基本概念，然后就计算智能主要研究领域如进化计算、群体智能、模糊计算、神经计算等内容分别展开介绍。

1. 进化计算

达尔文于 1859 年完成的科学巨著《物种起源》中，提出了自然选择学说，指出物种是在不断演变的，而且这种演变是一种由低级到高级、由简单到复杂的过程。1868 年，达尔文的第二部科学巨著《动物和植物在家养下的变异》问世，进一步阐述了他的进化论观点，提出了物种的变异和遗传、生物的生存斗争和自然选择的重要论点。生物种群的生存过程普遍遵循达尔文的物竞天择、适者生存的进化准则。种群中的个体根据对环境的适应能力而被大自然选择或淘汰。进化过程的结果反映在个体结构上，其染色体包含若干基因，相应的表现型和基因型的联系体现了个体的外部特性与内部机理间的逻辑关系。生物通过个体间的选择、交叉、变异来适应大自然环境。生物染色体用数学方式或计算机方式来体现就是一串数码，仍叫染色体，有时也叫个体；适应能力用对应一个染色体的数值来衡量；染色体的选择或淘汰问题是按求最大还是求最小问题来进行的。为求解优化问题，人们试图从自然界中寻找启迪。优化是自然界进化的核心，每个物种都在随着自然界的进化而不断优化自身结构，以适应自然的变化。20 世纪 60 年代以来，如何模仿生物来建立功能强大的算法，进而将它们运用于复杂的优化问题，越来越成为一个研究热点。对优化与自然界进化的深入观察和思考，促使进化算法诞生，其已发展成为一个重要的研究方向。

1.1 进化算法的原理

为了求解优化问题，研究人员试图从自然界中寻找答案。优化是自然界进化的核心，比如，每个物种都在随着自然界的进化而不断优化自身结构。

对进化算法的研究可追溯到 20 世纪 50 年代。当时，研究人员已经开始意识到达尔文的进化论可用于求解复杂问题。受达尔文进化论“物竞天择，适者生存”思想的启发，20 世纪 60 年代，美国密歇根大学的 J.霍兰德(J.Holland)提出了遗传算法。K.德容(K.De Jong)率先将遗传算法应用于函数优化。20 世纪 60 年代中期，L.J.福格尔(L.J. Fogel)等美国学者提出了

进化编程(evolutionary programming, EP)。几乎在同一时期，德国学者 I.雷兴贝格(I.Rechenberg)和 H.P.施韦费尔(H. P. Schwefel)开始了对进化策略(evolution strategy, ES)的研究。在随后的 15 年中，上述三种算法独立得到了发展。直到 20 世纪 90 年代，遗传算法、进化编程和进化策略才逐步走向统一，被统称为进化算法。很快，进化算法又增加了新的成员遗传编程(genetic programming, GP)，它由美国斯坦福大学的 J.R.科扎(J. R. Koza)于 20 世纪 90 年代早期创立。遗传算法、进化编程、进化策略和遗传编程是进化算法的四大经典范例，它们为研究人员求解优化问题提供了崭新的思路。

由于进化算法求解优化问题的巨大潜力，大批研究人员开始参与进化算法的研究，并取得了显著进展。目前，进化算法已经在许多领域得到了非常广泛的应用，并受到生物学、心理学、物理学等众多学科的关注。进化计算领域的第一本国际期刊 Evolutionary Computation 于 1993 年问世，由麻省理工学院出版社出版。美国电气和电子工程师协会于 1996 年创办了国际期刊 IEEE Transactions on Evolutionary Computation。进化计算在美国电气和电子工程师协会中起初隶属于神经网络协会。2004 年 2 月，美国电气和电子工程师协会神经网络协会正式更名为美国电气和电子工程师协会计算智能协会。此后，进化计算、神经网络、模糊计算作为 3 个主要分支共同隶属于美国电气和电子工程师协会计算智能协会。进化计算领域有许多高水平的国际学术会议，如美国电气和电子工程师协会每年举办的进化计算大会、国际计算机组织(Association for Computing Machinery, ACM)每年举办的遗传和进化计算大会等

事实上，随着对自然界的进一步认识和研究的不断深入，研究人员此后又提出了大量的进化算法范例。根据对文献的搜集发现，现有的进化算法范例已超过 30 种。例如，M.多里戈(M.Dorigo)于 1992 年提出了蚁群算法；R.G.罗伯特(R.G. Robert)于 1994 年提出了文化算法；R.埃伯哈特(R.Eberhart)和 J.肯尼迪(J.Kennedy)于 1995 年发明了粒子群优化算法；同年，R.斯通(R.Storn)和 K.普赖斯(K.Price)提出了差分进化算法。此外，人工免疫系统、量子进化算法、和声搜索算法、细菌觅食算法、人工鱼群算法、人口迁移算法、文化基因算法、人工蜂群算法，生物地理算法、组搜索算法、萤火虫算法、布谷鸟算法、蝙蝠算法、教学算法等进化算法范例也相继被提出。

上述进化算法范例的出现极大地促进了进化计算领域的发展，进化计算领域因此呈现出一派欣欣向荣的景象。

在解释进化算法的主要原理之前，先通过图 1 介绍基于梯度的优化方法的主要缺陷。假设图 1 中的优化问题为极小化问题。图 1(*a*)被称为单峰优化问题，这类优化问题仅包含一个局部最优解，因此全局最优解与局部最优解相同。图 1(*b*)被称为多峰优化问题，这类优化问

题同时包含多个局部最优解，通常全局最优解为其中的某个局部最优解。在求解优化问题时，基于梯度的优化方法首先确定一个初始点，接着基于梯度来计算下降方向和步长。利用下降方向和步长，可以产生一个新的点。基于梯度的优化方法通过反复执行上述过程来搜索全局最优解。对于单峰优化问题，基于梯度的优化方法非常有效。如图 1(*a*)所示，通过对初始点 x_a 不断施加下降方向和步长，最后可收敛于全局最优解 x^*。然而，对于图 1(*b*)中的多峰优化问题，执行同样的过程，则可能收敛于局部最优解^x，而非全局最优解 x^*。对于多峰优化问题，基于梯度的优化方法不能找到全局最优解的原因似乎非常直观：多峰优化问题包含多个局部最优解，然而基于梯度的优化方法往往从单点出发进行搜索，因此极易收敛于某个局部最优解

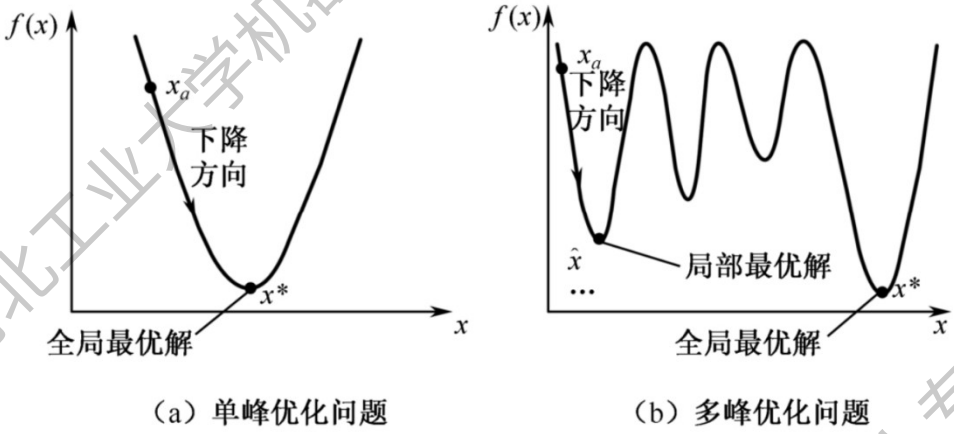


图 1 基于梯度的优先方法的主要缺陷

那么，对于复杂优化问题，能否通过多点出发来同步搜索全局最优解呢？这便是进化算法的主要特点。如图 2 所示，对于与图 1(b)中相同的多峰优化问题，进化算法采用多点出发(例如 x_a，x_b，x_c 和 x_d) 方式，沿着多个方向同时进行搜索。虽然每个点最后收敛于一个局部最优解，但是这些局部最优解中有可能包含全局最优解 x^*。显然，与基于梯度的优化方法相比，进化算法能够有更大的概率找到优化问题的全局最优解。

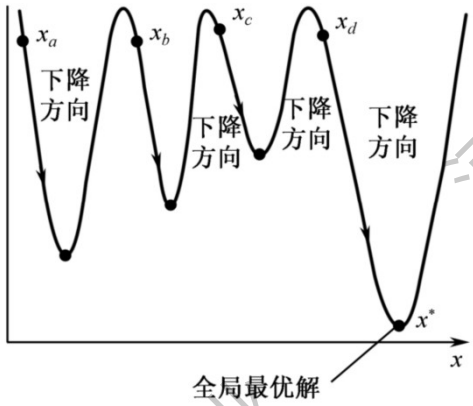


图 2 进化算法的主要特点

1.2 进化算法框架

如前所述，进化算法基于多点同时进行搜索。在进化算法中，这些点被称为个体(individual)，所有的个体构成了一个群体(population)。进化算法从选定的初始群体出发，通过不断迭代逐步改进当前群体，直至最后收敛于全局最优解或满意解。这种群体迭代进化的思想给优化问题的求解提供了一种全新思路。

一般来说，在求解优化问题时，进化算法的整体框架如图 4-8 所示。在进化算法的迭代过程中，首先应产生一个包含 N 个个体的群体，接着通过选择算子从群体中选择某些个体组成父代个体集，然后利用交叉算子和变异算子对父代个体集进行相关操作，产生子代个体集，最后将替换算子应用于旧的群体和子代个体集，得到下一代群体。其中，初始群体一般在搜索空间中随机产生，交叉算子和变异算子用于发现新的候选解，选择算子和替换算子则用于确定群体的进化方向。

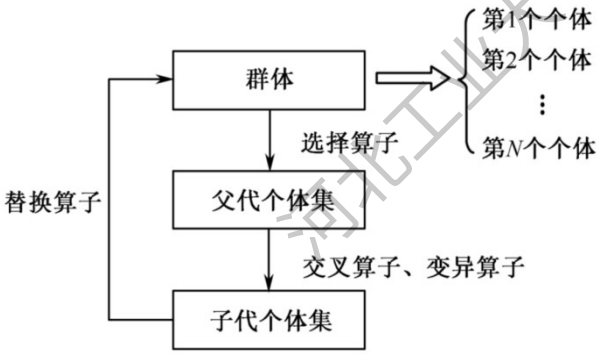


图 3 进化算法整体框架

1.3 遗传算法

遗传算法(GenelicAlgorithm，GA)是模拟达尔文的遗传选择和自然淘汰的生物进化过程的计算模型，它是由美国密歇根大学的 Holland 教授于 1975 年首先提出的。遗传算法作为一种新的全局优化搜索算法，具有简单通用、鲁棒性强、适于并行处理及应用范围广等显著特点。

1.3.1 遗传算法原理

1. 基本思想

遗传算法受生物进化论和遗传学说的启发而提出。它把问题的参数用基因代表，把问题的解用染色体代表(在计算机里用二进制码表示)，这就得到一个由具有不同染色体的个体组成的群体。群体在特定的环境里生存竞争、不断进化，最后收敛到一组最适应环境的个体——问题的最优解。

图 4 给出了基本遗传算法的过程，具体说明如下：

(1) 问题的解表示成染色体，在算法中也就是以二进制编码的串。在执行遗传算法之前，给出一群染色体(父个体) ，也就是假设的可行解。

(2) 把这些假设的可行解置于问题的环境中，并按适者生存的原则，从中选择出较适应环境的染色体进行复制，再通过交叉、变异过程产生更适应环境的新一代染色体(子个体)群。

(3) 经过一代一代地进化，最后收敛到的最适应环境的一个染色体上就是问题的最优解。

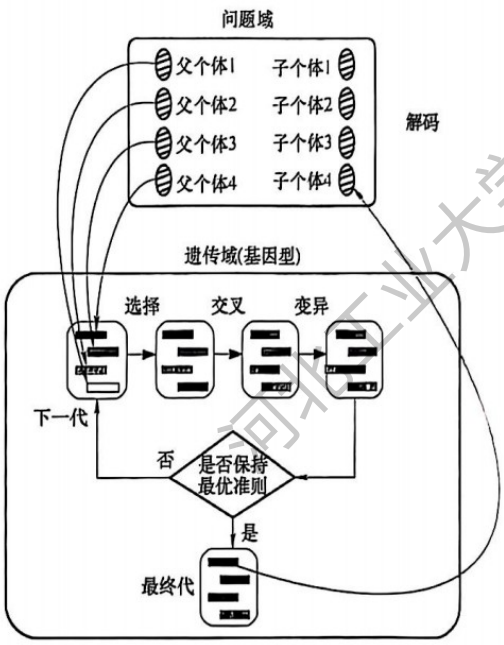


图 4 基本遗传算法的过程

2. 遗传算法中的术语

遗传算法计算优化的操作过程就如同生物学上生物遗传进化的过程，主要有三个基本操作(或称为算子)：选择(Selecbion)、交叉(Crossover)、变异(Mutation)。相关遗传学概念、遗传算法概念和数学概念三者之间的对应关系见下表。

表 1 遗传学概念、遗传算法概念和数学概念三者之间的对应关系

序号	遗传学概念	遗传算法概念	数学震念
1	个体	要处理的基本对象、结构	可行解
2	群体	个体的集合	被选定的一组可行解
3	染色体	个体的表现形式	可行解的编码
4	基因	染色体中的元素	编码中的元案
5	基因位	某一基因在染色体中的位置	元素在编码中的位置
6	适应值	个体对于环境的适应程度，或在环境压力下的生存能力	可行解所对应的适应函数值
7	种群	被选定的一组染色体或个体	根据入选概率定出的一组可行解
8	选择	从群体中选择优性个体，淘汰劣质个体	保留或复制适应值大的可行解，去掉小的可行解
9	交叉	一组染色体上对应基因段的交换	根据交叉原则产生的一组新解
10	交叉概串	染色体对应基因段交换的概率(可能性大小)	闭区间[0、1]上的一个值，一般为0.65~0.90
11	变异	染色体水平上基因变化	编码的某些元素被改变
12	变异概率	染色体上基因变化的概率(可能性大小)	开区间(0,1)内的一个值，一般为0.001~0.01
13	进化、适者生存	个体优胜劣汰，一代又一代地进化	目标函数取到最优可行解

(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。根据位串的适配值复制，也就是指具有高适配值的位串更有可能在下一代中产生一个或多个子孙。它模仿了自然现象，应用了达尔文的适者生存理论。复制操作可以通过随机方法来实现。若用计算机程序来实现，可考虑首先产生 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。此外，还可以通过计算方法实现，其中较典型的几种方法为适应度比例法、期望值法、排位次法等，适应度比例法较常用。选择运算是复制中的重要步骤。

(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。它的过程为：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。交换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。一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，例如：

A：101100 1110→101100 0101

B：001010 0101→001010 1110

(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。

3. 遗传算法的构成要素

遗传算法的核心构成要素包括：

(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}所组成的。初始个体的基因值可用均匀分布的随机值来生成，如 x=100111001000101101 就可表示一个个体，该个体的染色体长度是 n=18。

(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到个体适应度之间的转换规则。

(3) 遗传算子

基本遗传算法中的 3 种运算使用下述 3 种遗传算子：

1) 选择运算使用比例选择算子；

2) 交叉运算使用单点交叉算子；

3) 变异运算使用基本位变异算子或均匀变异算子。

(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。

P_c：交叉概率，一般取为 0.4～0.99。

P_m：变异概率，一般取为 0.0001～0.1。

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 1 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 2 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；

第 3 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 4 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 5 步：设计遗传算子，即确定选择运算、交叉运算、变异运算等遗传算子的具体操作方法；

第 6 步：确定遗传算法的有关运行参数，即 M, G, P_c, P_m 等参数；

第 7 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。

以上操作过程可以用**图 5** 来表示

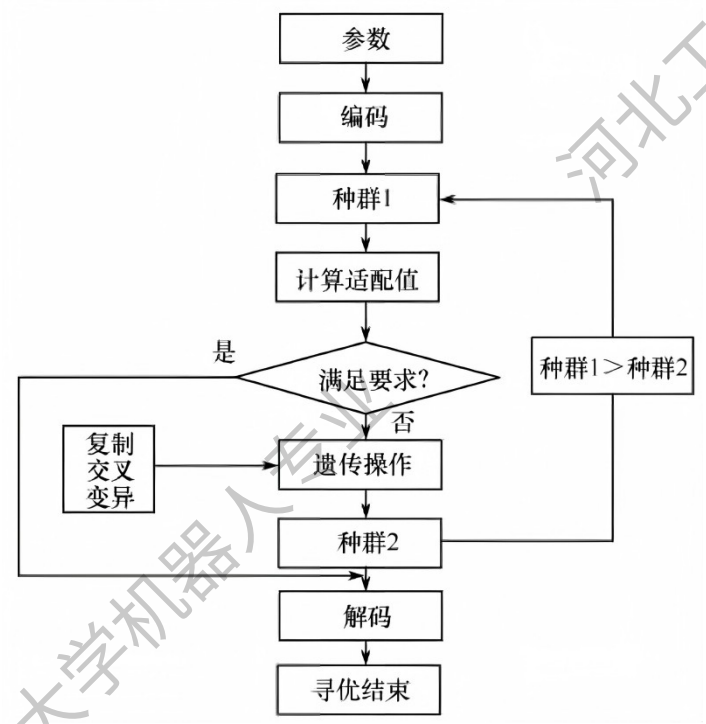


图 5 遗传算法流程图

4. 遗传算法的特点

遗传算法主要有以下几个特点。

(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一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。

(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，无须目标函数的导数值等其他一些辅助信息。因此，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。而且，直接利用目标函数值或个体适应度，也可将搜索范围集中到适应度较高的部分搜索空间中，从而提高搜索效率。

(4) 遗传算法使用概率搜索技术。许多传统的优化算法使用的是确定性搜索算法，一个搜索点到另一个搜索点的转移有确定的转移方法和转移关系，这种确定性的搜索方法有可能使得搜索无法达到最优点，因而限制了算法的使用范围。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。理论已经证明，遗传算法在一定条件下以概率 1 收敛于问题的最优解。

(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。

(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。

5. 遗传算法的应用领域

遗传算法的应用领域包括：

(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。

(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。

(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

(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、图像恢复和图像边缘特征提取等方面得到了应用

6. 遗传算法求函数极大值

利用遗传算法求 Rosenbrock 函数的极大值，Rosenbrock 函数的形式如下：

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

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。

函数 $f(x_1, x_2)$的三维图如**图 6** 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

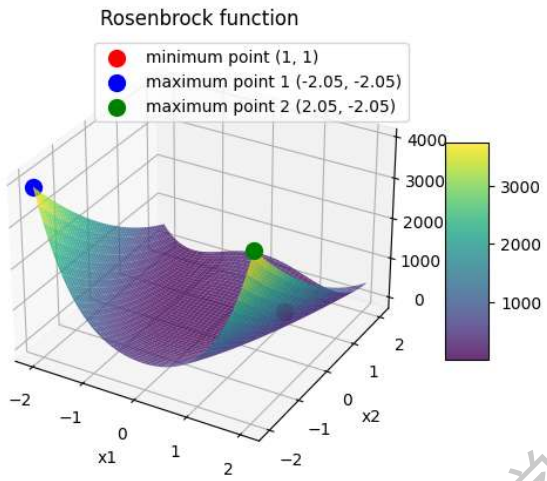


图 6 $f(x1, x2)$的三维图

采用二进制编码遗传算法求函数极大值，遗传算法的构造过程如下：

(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：用长度为 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，依次让它们分别对应于从 0000000000(0)到 1111111111(1023)之间的二进制编码。再将分别表示 x_1，x_2 的两个 10 位长的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：0000110111 1101110001 就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。

(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。由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：

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

例如，对个体 x：0000110111 1101110001，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

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

(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

$$F(x) = f(x_1, x_2) \tag{3}$$

选个体适应度的倒数作为目标函数

$$J(x) = \frac{1}{F(x)} \tag{4}$$

(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=500，终止进化代数 G=300，交叉概率 Pc=0.80，变异概率 Pm=0.10。

上述 7 个步骤构成了用于求 Rosenbrock 函数极大值优化计算的二进制编码遗传算法。经过 100 步迭代，最佳样本为 *BestS*=[00000000000000000000]，即当 x_1=-2.048，x_2=-2.048 时，*Rosenbrock* 函数具有极大值，极大值为 3905.9。

遗传算法的优化过程中目标函数 J 和适应度函数 F 的变化过程如**图 7** 和**图 8** 所示，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

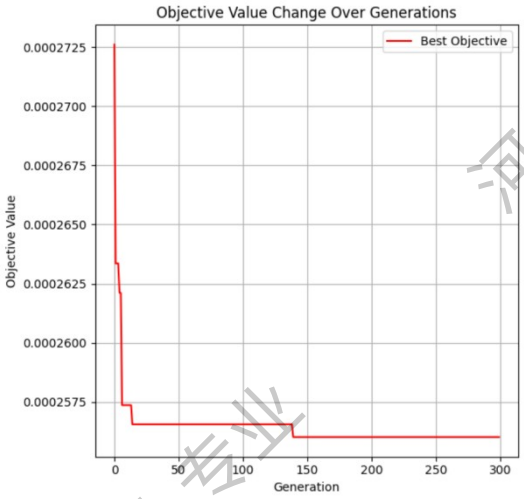


图 7 目标函数 *J* 的优化过程

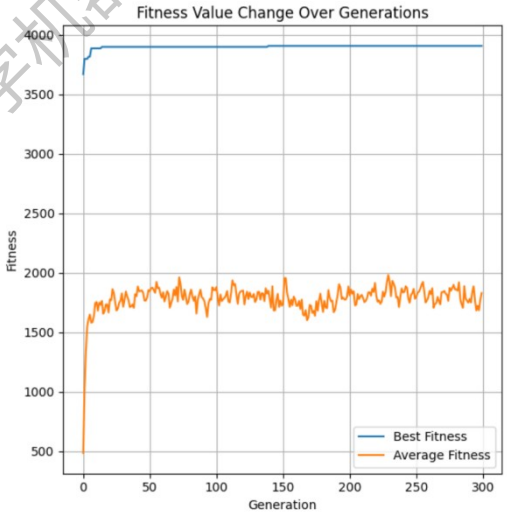


图 8 适应度 *F* 的优化过程

2. 群体智能

群体智能(SwarmIntelligence)受群居生物集体社会行为的启发，其特点是个体行为简单，但群体工作却能够突显出复杂(智能)的行为特征。目前，群体智能研究领域主要有蚁群算法(蚂蚁觅食的过程)、粒子群算法(鸟群捕食的过程)、人工蜂群算法(蜜蜂采蜜的过程)、萤火虫算法(萤火虫相互吸引，改变位置的过程)、细菌觅食优化算法(细菌觅食行为的一种模拟过程)。群体智能方法的应用领域已扩展到多目标优化、数据分类、模式识别、神经网络训练、信号处理、决策支持等多个方面。

2.1 粒子群优化算法

粒子群算法，也称为粒子群优化算法(Particle Swarm Optimization，PSO)。粒子群优化算法是一种进化计算技术，1995 年由 Eberhart 博士和 Kennedy 博士提出，该算法源于对鸟群捕食的行为研究，是近年来迅速发展的一种新的进化算法。

最早的 PSO 是模拟鸟群觅食行为而发展起来的一种基于群体协作的随机搜索算法，让一群鸟在空间里自由飞翔觅食，每只鸟都能记住它曾经飞过最高的位置，然后就随机地靠近那个位置，不同的鸟之间可以互相交流，它们都尽量靠近整个鸟群中曾经飞过的最高点，这样，经过一段时间就可以找到近似的最高点。

PSO 算法和遗传算法相似，也是从随机解出发，通过迭代寻找最优解；它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。

2.2 粒子群算法的基本原理

PSO 算法模拟鸟群的捕食行为。设想这样一个场景：一群鸟在随机搜索食物，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。

PSO 算法从这种模型中得到启示并用于解决优化问题。PSO 算法中，每个优化问题的解都是搜索空间中的一只鸟，称为“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它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，这个解称为个体极值。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，那么在所有邻居中的极值就是全局极值。

2.3 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

(1) 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。

(2) 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} 则保证粒子种群的局部搜索能力加强。

(3) 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。

(4) 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=0.8 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，能够精确地得到全局最优解。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。

(5) 中止条件：最大循环数或最小误差要求。

2.4 粒子群算法的流程

(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，kg 表示当前的进化代数。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 $i(1\leqslant i\leqslant Size)$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、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。

(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。

$$V_i^{kg+1}=w(t)\times V_i^{kg}+c_1r_1\left(P_i^{kg}-X_i^{kg}\right)+c_2r_2\left(Best\,\mathrm{S}_i^{kg}-X_i^{kg}\right) \tag{5}$$

$$X_i^{kg+1} = X_i^{kg} + V_i^{kg+1} \tag{6}$$

其中，kg=1，2，…，G，i=1，2，…，$Size$，r_1 和 r_2 为 0 到 1 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 大一些。

(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。

PSO 的算法流程图如图所示

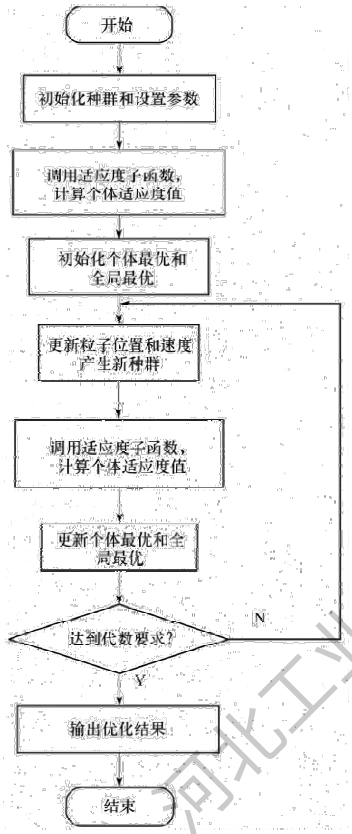


图 9 粒子群算法流程图

2.5 粒子群算法应用举例

利用粒子群算法求 Rosenbrock 函数的极大值

$$\begin{cases} f(x_1, x_2) = 100\left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2 \\ -2.048 \le x_i \le 2.048 \quad (i = 1, 2) \end{cases} \tag{7}$$

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。

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。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}。

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，如图所示。

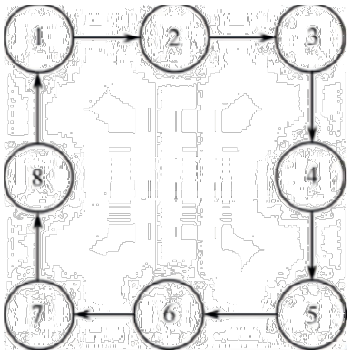


图 10 环形邻域

图中，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，依此类推。

局部粒子群算法中，按如下两式更新粒子的速度和位置：

$$V_i^{kg+1}=\omega(t)\times V_i^{kg}+c_1r_1\left(p_i^{kg}-X_i^{kg}\right)+c_2r_2\left(p_{ilocal}^{kg}-X_i^{kg}\right) \tag{8}$$

$$X_i^{kg+1}=X_i^{kg}+V_i^{kg+1} \tag{9}$$

其中，$p^{kg}{}_{i\,local}$ 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。

采用实数编码求函数极大值，用两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=1.3，*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。按式(8)和式(9)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，*Rosenbrock* 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。

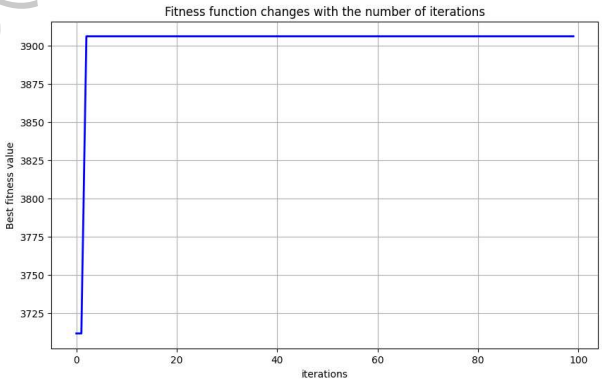


图 11 适应度函数 F 的优化过程

3. 模糊理论

1960 年，加州大学伯克利分校电子工程系的扎德教授(L.A.Zadeh)首次提出具有划时代意义的"模糊"概念。五年后，他通过创立模糊集合论，开创性地构建了模糊系统理论这门新兴学科。1973 年，扎德教授系统性地提出基于模糊语言的系统描述方法，为后续模糊控制技术的实际应用奠定了方法论基础。随着学术影响力的扩大，1984 年国际模糊系统协会(IFSA)在学界瞩目中正式成立，标志着该学科正式进入国际主流学术视野，完成了从理论探索到学术建制化进程的重要跨越。

3.1 特征函数和隶属函数

在数学上经常用到集合的概念，例如，集合 A 由 4 个离散值 x_1，x_2，x_3，x_4 组成，记 $A=\{x_1, x_2, x_3, x_4\}$。再如，集合 A 由 0 到 1 之间的连续实数值组成，记 $A=\{x, x\in R, 0\leqslant x\leqslant 1\}$。以上两个集合是完全不模糊的，对任意元素 x，只有两种可能：属于 A，不属于 A。这种特性可以用特征函数 $\mu_A(x)$来描述，即

$$\mu_A(x)=\begin{cases}1 & x\in A\\ 0 & x\notin A\end{cases} \tag{10}$$

为了表示模糊概念，需要引入模糊集合和隶属函数的概念：

$$\mu_A(x)=\begin{cases}1 & x\in A\\ (0,1) & x\text{ 属于}A\text{ 的程度}\\ 0 & x\notin A\end{cases} \tag{11}$$

其中 A 称为模糊集合，由 0，1 及 $\mu_A(x)$构成，表示元素 x 属于模糊集合 A 的程度，取值范围为[0，1]，称 $\mu_A(x)$为 x 属于模糊集合 A 的隶属度。

3.2 模糊算子

模糊集合的逻辑运算实质上就是隶属函数的运算过程。采用隶属函数的取大(MAX)-取小(MIN)进行模糊集合的并、交逻辑运算是目前最常用的方法。但还有其他公式，这些公式统称为“模糊算子”。

设有模糊集合 A、B 和 C，常用的模糊算子如下：

1. 交运算算子

设 $C=A\cap B$，有以下 3 种模糊算子。

(1) 模糊交算子

$$\mu_C(x)=\min\{\mu_A(x),\ \mu_B(x)\} \tag{12}$$

(2) 代数积算子

$$\mu_C(x)=\mu_A(x)\cdot\mu_B(x) \tag{13}$$

(3) 有界积算子

$$\mu_C(x)=\max\{0,\ \mu_A(x)+\mu_B(x)-1\} \tag{14}$$

2. 并运算算子

设 $C=A\cup B$，有 3 种模糊算子。

(1) 模糊并算子

$$\mu_C(x)=\max\{\mu_A(x),\ \mu_B(x)\} \tag{15}$$

(2) 概率或算子

$$\mu_C(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)\times\mu_B(x) \tag{16}$$

(3) 有界和算子

$$\mu_C(x)=\min\{1，\mu_A(x)+\mu_B(x)\} \tag{17}$$

3. 平衡算子

当隶属函数取大和取小运算时，不可避免地要丢失部分信息，采用一种平衡算子可起到补偿作用。设 $C=A\cup B$，则 $\mu_C(x)=[\mu_A(x)\cdot\mu_B(x)]^{1-\gamma}\cdot[1-(1-\mu_A(x))\cdot(1-\mu_B(x))]^{\gamma}$，$\gamma$取值为[0，1]。当 $\gamma=1$ 时，$\mu_C(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)\times\mu_B(x)$，相当于 $A\cup B$ 时的算子。平衡算子目前已经应用于德国 Inform 公司研制的著名模糊控制软件 Fuzzy-Tech 中

3.3 隶属函数

普通集合用特征函数来表示，模糊集合用隶属函数来描述。隶属函数很好地描述了事物的模糊性。隶属函数有以下两个特点。

① 隶属函数的值域为[0，1]，它将普通集合只能取 0 和 1 两个值，推广到[0，1]闭区间上连续取值。隶属函数的值 $\mu_A(x)$越接近于 1，表示元素 x 属于模糊集合 A 的程度越大。反之，$\mu_A(x)$越接近于 0，表示元素 x 属于模糊集合 A 的程度越小。

② 隶属函数完全刻画了模糊集合，隶属函数是模糊数学的基本概念，不同的隶属函数所描述的模糊集合也不同。

典型的隶属函数有 11 种，即双 S 形隶属函数、联合高斯型隶属函数、高斯型隶属函数、广义钟形隶属函数、Π形隶属函数、双 S 形乘积隶属函数、S 状隶属函数、S 形隶属函数、梯形隶属函数、三角形隶属函数、Z 形隶属函数。

在模糊控制中应用较多的隶属函数有以下 6 种隶属函数。

(1) 高斯型隶属函数

高斯型隶属函数由两个参数 σ 和 c 确定，即

$$f(x,\sigma,c)=e^{-\frac{(x-c)^2}{2\sigma^2}} \tag{18}$$

其中，参数 σ 通常为正，参数 c 用于确定曲线的中心。

(2) 广义钟形隶属函数

广义钟形隶属函数由三个参数 a，b，c 确定，即

$$f(x,a,b,c)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{19}$$

其中，参数 b 通常为正，参数 c 用于确定曲线的中心。

(3) S 形隶属函数

S 形函数 *sigmf*(x，[a，c])由参数 a 和 c 决定，即

$$f(x,a,c)=\frac{1}{1+e^{-a(x-c)}} \tag{20}$$

其中，参数 a 的正、负符号决定了 S 形隶属函数的开口朝左或朝右，用来表示“正大”或“负大”的概念。

(4) 梯形隶属函数

梯形曲线可由四个参数 a，b，c，d 确定，即

$$f(x,a,b,c,d)=\begin{cases}0 & x\le a\\ \dfrac{x-a}{b-a} & a\le x\le b\\ 1 & b\le x\le c\\ \dfrac{d-x}{d-c} & c\le x\le d\\ 0 & x\ge d\end{cases} \tag{21}$$

其中，参数 a 和 d 确定梯形的“脚”，而参数 b 和 c 确定梯形的“肩膀”。

(5) 三角形隶属函数

三角形曲线的形状由三个参数 a，b，c 确定，即：

$$f(x,a,b,c)=\begin{cases}0 & x\le a\\ \dfrac{x-a}{b-a} & a\le x\le b\\ \dfrac{c-x}{c-b} & b\le x\le c\\ 0 & x\ge c\end{cases} \tag{22}$$

其中，参数 a 和 c 确定三角形的“脚”，而参数 b 确定三角形的“峰”。

(6) Z 形隶属函数

Z 形隶属函数是基于样条函数而设计的，因其呈现 Z 形状而得名。参数 a 和 b 确定了曲线的形状。

针对上述描述的 6 种隶属函数进行仿真。$x \in [0, 10]$，M 为隶属函数的类型，其中 M=1 为高斯型隶属函数，M=2 为广义钟形隶属函数，M=3 为 S 形隶属函数，M=4 为梯形隶属函数，M=5 为三角形隶属函数，M=6 为 Z 形隶属函数。仿真结果如图 7-1 至图 7-6 所示。

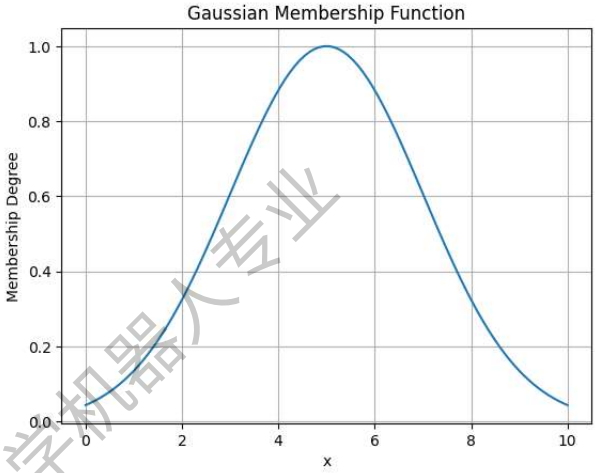


图 12 高斯型隶属函数(M=1)

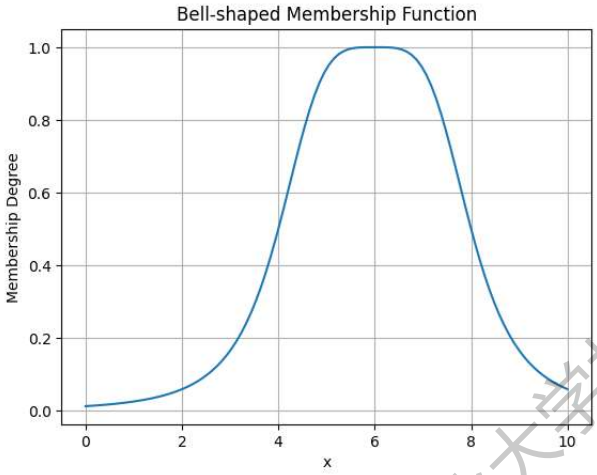


图 13 广义钟形隶属函数(M=2)

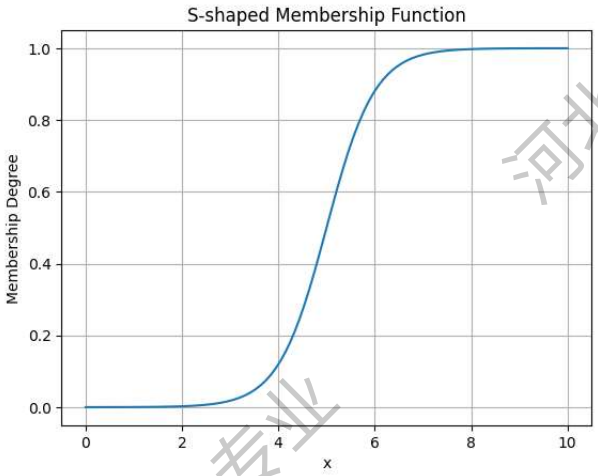


图 14 *S* 形隶属函数(*M*=3)

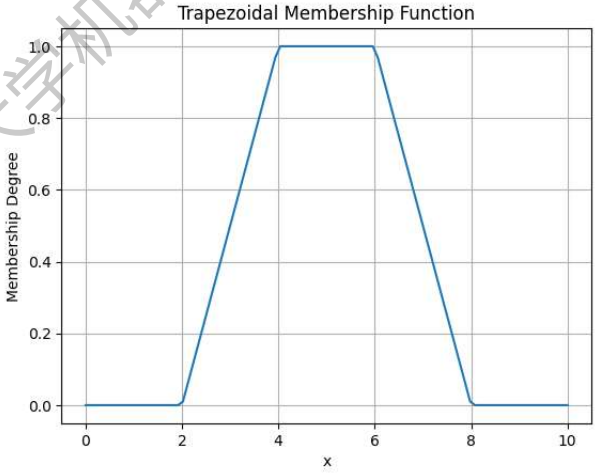


图 15 梯形隶属函数(*M*=4)

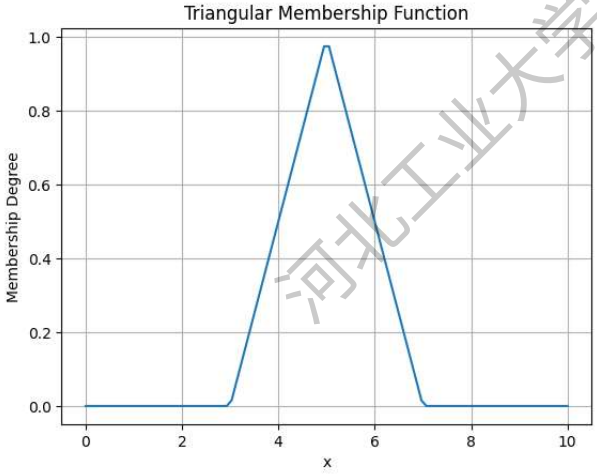


图 16 三角形隶属函数(*M*=5)

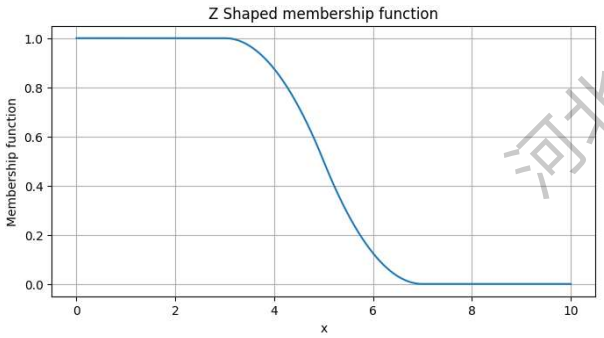


图 17 *Z* 形隶属函数(*M*=6)

实例：设计评价一个学生成绩的隶属函数，在[0，100]之内按 A、B、C、D、E 分为五个等级，即{不及格，及格，中，良，优}。分别采用五个高斯型隶属函数来表示，建立一个模糊系统。

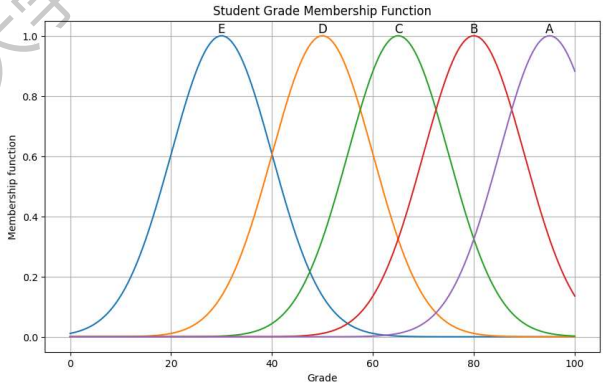


图 18 高斯型隶属函数曲线

```python
def gaussian_membership(x, mean, sigma):
    return np.exp(-((x - mean) ** 2) / (2 * sigma ** 2))

# 定义五个等级的隶属函数参数
grades = {
    'E': {'mean': 30, 'sigma': 10},  # 不及格
    'D': {'mean': 50, 'sigma': 10},  # 及格
    'C': {'mean': 65, 'sigma': 10},  # 中
    'B': {'mean': 80, 'sigma': 10},  # 良
    'A': {'mean': 95, 'sigma': 10}   # 优
}
```

3.4 模糊控制

1. 模糊控制的基本思想

模糊控制原理图如图 19 所示，模糊控制和传统控制的系统结构是完全一致的。点画线框内表明了模糊控制基于模糊化、模糊推理、解模糊等运算过程。最常见的模糊控制系统有 Mamdani 型模糊逻辑系统和高木-关野(Takagi-Sugeno)型模糊逻辑系统。

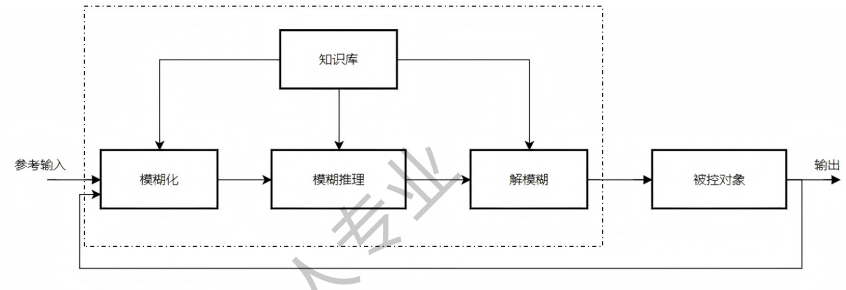
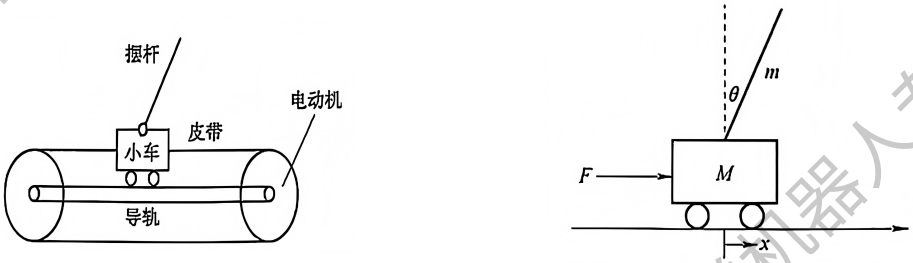


图 19 模糊控制原理图

现以图 5-2 所示的一级倒立摆系统来简单说明模糊控制器设计的一般方法。在忽略了空气阻力和各种摩擦之后，可将一级倒立摆系统抽象成小车和均质杆组成的系统，摆杆与小车之间为自由链接，小车在控制力的作用下沿滑轨在 x 方向运动，控制目的是使倒立摆能够尽可能稳定在铅直方向，同时小车的水平位置也能得到控制。



(a) 系统结构　　(b) 系统建模

图 20 以及倒立摆

(1）模糊化

以摆杆的倾角和速度作为输入变量，可以将摆杆倾角定义为：向左倾角大、中、小、垂直，向右倾角小、中、大几个模糊子集；摆杆速度定义为：向左非常快、快、慢、静止，向右慢、快、非常快等几个模糊子集。它们都可以用模糊语言变量 NB、NM、NS、ZE、PS、PM、PB 来表示。控制小车运动的输出也可类似定义。接着按照一定的隶属度函数确定每个模糊子集隶属度。这个确定隶属度的过程就是对变量进行模糊化的过程。倒立摆系统二维模糊控制器输入量为倾角θ和角速度$\Delta\theta$，输出量是施加在小车上的控制力 u。

对输人输出变量θ、$\Delta\theta$、u 各定义七个模糊集合：NB、NM、NS、ZE、PS、PM、PB；它们的基本论域分别为[-0.5，0.5]、[-1，1]和[-10，10］。各变量的模糊集合的隶属函数均是对称、均匀分布的三角形。例如，输人变量$\Delta\theta$的隶属函数如图 21 所示。

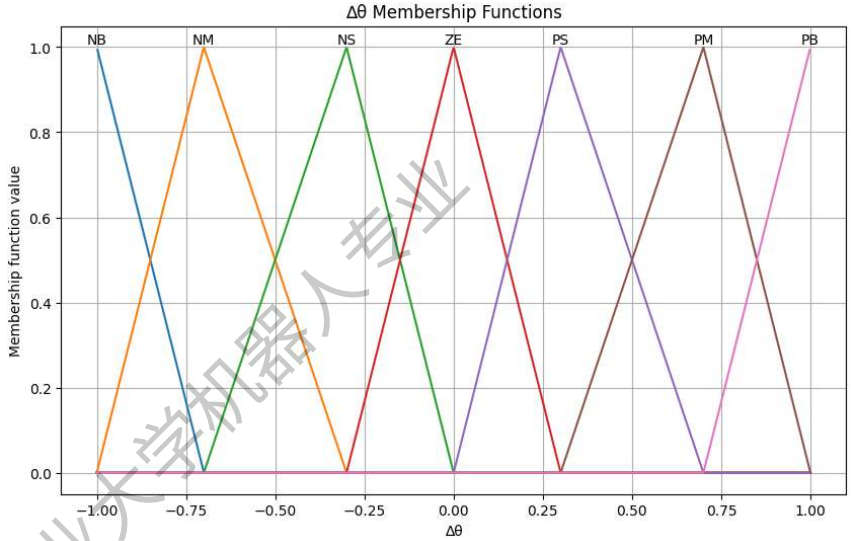


图 21 输入变量Δ0的隶属函数

（2）模糊推理

建立一系列的模糊规则，来描述各种输人所产生的作用。比如可以建立如下一些规则：如果摆杆向左倾斜大并倒得非常快，那么小车快速向左运动；如果摆杆向左倾斜大并倒得较快，那么小车中速向左运动；如果摆杆向左倾斜小并倒的慢，那么小车慢速向左运动。因为θ和$\Delta\theta$ 各有七个模糊集合，所以最多有 $7^2=49$ 条规则，根据经验可以减少规则的数量，只用 29 条即可，倒立摆的模糊控制规则表见表 2。

表 2 倒立摆的模糊控制规则表

$\Delta\theta$	θ						
	NB	NM	NS	0	PS	PM	PB
	u						
NB				NB			
NN			NB	NM	NS	0	
NS		NB	NM	NS	0	PS	PM
0	NB	NM	NS	0	PS	PM	PB
PS	NM	NS	0	PS	PM	PB	
P		0	PS	PM	PB		
PB				PB			

利用模糊规则编辑器，可以方便地编辑模糊规则。例如，表 3 中第一行中间的控制规则可用模糊条件语句描述：

If θ is O and $\Delta\theta$ is NB，then u is NB.

(3)解模糊。

模糊输出量被反模糊化为能够用于对物理装置进行控制的精确量，这个过程称为解模糊。这里，可以通过重心法解模糊得到施加在小车上的控制力 u 实现倒立摆的稳定控制。上述倒立摆模糊控制实例是一个典型的 Mamdani 型模糊逻辑系统。Takagi 和 Sugeno 于 1985 年提出了一种 T-S 模型，也称之为 Sugeno 模糊模型。T-S 模型和 Mamdani 模型的最主要的区别在于：推理规则后项结论中的输出变量的隶属度函数只能是关于输人的线性组合或是常值函数。可见，T-S 型系统解模糊过程与模糊推理过程是结合在一起的，因此其在形式上更加紧凑和易于计算，使得它也可以很方便地采用自适应的思想来创建系统模型。

2. 模糊控制的特点

对于一个熟练的操作人员，并非需要了解被控对象精确的数学模型，而是凭借其丰富的实践经验，采取适当的对策可巧妙地控制一个复杂过程。模糊控制器具有如下一些显著特点：

(1) 无须知道被控对象的精确数学模型。

(2) 易被人们接受。模控制的核心是模糊推理，它是人类通常智能活动的体现。

(3) 鲁棒性好。干扰和参数变化对控制效果的影响被大大减弱，尤其适合于非线性、时变及纯滞后系统的控制。

另一方面，模糊控制尚有许多问题有待解决：

(1) 在理论上还无法像经典控制理论那样证明运用模糊逻辑的控制系统的稳定性。

(2) 模逻辑控制规则是靠人的经验制定的，它本身并不具有学习功能。

(3) 模糊控制规则越多，控制运算的实时性越差。

3. 模糊控制的实质

从系统建模的角度而言，模糊系统建立在被人容易接受的“如果-则”表达方法之上，模糊系统将知识存储在规则集中。

(1) 从知识表示方式看：模糊系统可以表达人们的经验性知识，便于理解。

(2) 从知识的运用方式看：模糊系统具有并行处理的特点，模糊系统同时激活的规则不多，计算量也不大。

(3) 从知识的获取方式看：模糊系统的规则是靠专家提供或设计的，对于具有复杂系统的专家知识，往往很难由直觉和经验获取，规则形式也是很困难的。

(4) 从知识的修正方式看：模糊系统可以非常容易地对系统规则进行分析，可由删除和增加模糊规则来完成知识的修正。

从上述分析可见，当输入信号进入模糊系统时，所有的模糊规则将依据条件部分的适用度决定是否被激发，并且由被激发的规则决定系统的输出。其输入/输出(I/O)非线性映射关系体现在模糊规则上。这一点在接下来神经网络的学习中，可以体会到误差逆传播(BackPropagation，BP)神经网络亦是一种非线性映射，其输人/输出(I/O)非线性映射关系体现在神经网络的结构和参数上。BP 神经网络与模糊系统具有一定的等价性。