3.2 遗传算法

3.2.1 遗传算法的概念

遗传算法是1975年由美国John.Holland 教授提出的模拟自然遗传和生物进化理论发展而成的解决最优化问题的计算模型。遗传算法是以达尔文的自然选择学说为基础发展起来的，其理论思想是模拟自然界生物种群进化过程中“优胜劣汰，适者生存”的选择规律，在数学上借助一个适应度函数，实现对优化问题寻找最优解的过程。由于该算法采用随机选择搜索，对求解空间无特殊要求，无需求导，具有运算简单、收敛速度快等优点，尤其适用于处理传统搜索方法难以解决的复杂和非线性问题。遗传算法直接以目标函数作为搜索信息，在一定范围内同时搜索多个点，比传统优化算法寻优效率高，适合处理较复杂的优化问题。如组合优化，生产调度，复杂布局，自动控制，神经网络，图像处理等。

利用遗传算法解最优化问题，首先应对可行域中的点进行编码,然后在可行域中随机挑选一些编码组成作为进化起点的第一代编码组，并计算每个解的目标函数值，也就是编码的适应度。接着就像自然界中一样，利用选择机制从编码组中随机挑选编码作为繁殖过程前的编码样本。选择机制应保证适应度较高的解能够保留较多的样本；而适应度较低的解则保留较少的样本，甚至被淘汰。在接下去的繁殖过程中，遗传算法提供了交叉和变异两种算子对挑选后的样本进行交换。交叉算子交换随机挑选的两个编码的某些位，变异算子则直接对一个编码中的随机挑选的某一位进行反转。这样通过选择和繁殖就产生了下一代编码组。重复上述选择和繁殖过程,直到结束条件得到满足为止。进化过程最后一代中的最优解就是用遗传算法解最优化问题所得到的最终结果。研究试验表明，遗传算法可以用非常快的速度找到与最优解近似程度达到90%的解。

3.2.2 遗传算法的原理

遗传算法是从代表问题可能潜在的解集的一个种群开始的，而一个种群则由经过基因编码的一定数目的个体组成。每个个体实际上是染色体带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码。初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度大小选择个体，并借助于自然遗传学的遗传算子进行组合交叉和变异，产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码，可以作为问题近似最优解。

与传统搜索算法从一个具体值出发不同，遗传算法从一组随机产生的数据开始搜索所求优化问题的最优解。由遗传问题产生的实际背景，开始搜索的一组数据称为初始解，并由它形成初始种群。同时，群体中的每个个体是问题的一个解，称为染色体.这些染色体在遗传算法中，通过选择、交叉和变异运算来模拟自然界物种进化过程中的基因复制、基因重组和基因突变，从而形成新一代群体，即后代.遗传算法利用适应度函数值来衡量每一代群体中每个个体的“优劣”程度，并由适应度值的大小从父代和子代中选择一定数量的个体，作为下一代群体。在理想模式下，如此不断进化，经过若干代之后，种群中的染色体保留了越来越适应环境的优良基因组合，在算法中的体现即是收敛于最终的最优解或次优解。

3.2.3 遗传算法的特点

因此，遗传算法具有以下七个特点：

1. 遗传算法是对问题参数的编码组进行编码，而不是直接对参数本身，因此可以使遗传算法可以直接对结构对象进行操作。
2. 遗传算法同时处理群体中的多个个体，即对搜索空间中的多个解进行评估，减少了陷入局部最优解的风险，同时算法本身易于实现并行化。
3. 遗传算法是从问题解的串级开始搜索,而不是从单个解开始。这是遗传算法与传统优化算法的极大区别。传统优化算法是从单个初始值迭代求最优解的；容易误入局部最优解。遗传算法从串集开始搜索，覆盖面大，利于全局择优。
4. 遗传算法使用目标函数值适应度这一信息进行搜索,而不需要其他信息。也就是说其对搜索空间没有任何特殊要求,只利用适应性信息,不需要导数等其它辅助信息, 这一特点使得遗传算法的应用范围大大扩展。
5. 遗传算法采用概率的变迁规则来指导他的搜索方向,而不是确定性规则。
6. 遗传算法具有自组织、自适应和自学习性。遗传算法利用进化过程获得的信息自行组织搜索时，适应度大的个体具有较高的生存概率，并获得更适应环境的基因结构。
7. 遗传算法本身也可以采用动态自适应技术，在进化过程中自动调整算法控制参数和编码精度。

3.2.3 遗传算法的构成

利用遗传算法解决实际优化问题的关键是，如何通过对算法实现过程进行合理的设计，来体现自然选择的规律，完成实际问题寻找最优解的任务。其主要步骤：

1. 根据实际问题选择合适的编码方式；
2. 根据目标函数所求解的要求，产生初始数据，即初始种群；
3. 评价种群中个体的优劣，即计算适应度函数值；
4. 进行遗传操作：选择复制、交叉、变异；
5. 对新种群重复步骤 3、4，直至进化完成，产生最优或次优解。

遗传算法一般不直接处理待优化问题的参数,而是将它们转换成由基因按一定结构组成的染色体或个体,即转化为对参数编码的处理。编码过程是遗传算法使用时的首要步骤，为实际问题选择一个合适的编码方案是后续步骤能顺利进行的关键，编码方式的确定主要从三方面考虑：

1. 编码形式直观易懂，最好能反映要解决的实际问题的特定知识；
2. 方便后续遗传进化部分的操作实现；

3、解码过程容易实现。

目前，常用的编码分为三种：二进制编码、浮点数编码和符号编码。编码方式在很大程度上决定了群体的遗传进化实现方式，并直接影响运算的效率。

3.2.3.1 二进制编码

二进制编码方式是最常用的一种编码方式。它以二值符号集{0,1}为编码字符集,所构成的个体基因型,也称染色体,是一个二进制符号串。如10110010是一个染色体长为8的个体。

二进制编码有五个优点：

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

同时，也存在着两大缺点：

1. 二进制编码存在着连续函数离散化时的映射误差。个体编码的长度较短时,可能达不到精度的要求，而个体编码的长度较大时,虽然能提高编码精度,但却会使遗传算法的搜索空间急剧扩大；
2. 它不能直接反映出所求问题的本身结构特征,这样也就不便于开发针对问题的专门知识的遗传运算算子,很难满足积木块编码原则。

3.2.3.2 浮点数编码

对于一些多维、高精度要求的连续函数优化问题,二进制编码可量测性不强,不直观,且不是因为字符串太长而导致遗传算法训练的解空间过大,寻优时间很长,就是因为字符串太短,使权值表达精度不够。这时可使用浮点数编码方式。浮点数编码中,个体的每个基因值用某一范围内的一个浮点数来表示,个体的编码长度等于其决策变量的个数。

在浮点数编码方式中,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使其运算结果所产生新个体的基因值在这个区间限制范围内。另外,当用多个字节来表示一个基因值时,单点交叉位置必须选在两个基因的分界处,而不能选在基因的中间字节分隔处。

浮点数编码的搜索能力一般稍逊于二进制编码，但浮点数编码在变异操作过程中，保持种群多样性的性能要比二进制编码优异。所以浮点数编码适合于较大空间、精度要求较高的场合。

3.2.3.3 十进制编码

十进制编码是符号编码中的一种,它将待处理的参数数值(实数)逐位数字地转化为一定长度的数字字符并形成字符串。其遗传操作可以完全类似于遗传算法对二进制字符串的处理。例如,对前例,当每个参数取值范围均包含在区间。

同浮点数编码方法一样,十进制编码也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、变异等遗传算子必须使其运算结果所产生新个体的基因值也在这个区间限制范围内。为方便算子设计起见,可以对各参数进行预处理,将各参数的取值范围映射为[O,999]。