

面向对象编码技术下的大个体问题的遗传算法

裴焱栋, 顾克江

(江苏油田勘探局, 江苏 扬州 225009)

[摘要] 针对遗传算法引入面向对象编码技术并对其进行一定的修改. 实际问题中常见大个体、超长个体, 传统遗传算法表现不足. 提出相应的解决方法, 引入较大的概率让所有属性均有几率发生改变. 给出具体概率的值. 随后采用测试函数, 对本文提出的方法进行了测试, 证明该方法有效.

[关键词] 面向对象技术, 大个体, 遗传算法

[中图分类号] TP3 [文献标志码] A [文章编号] 1001-4616(2015)01-0086-05

Gene Algorithm for Long Individual Problems in Object-Oriented Coding Method

Pei Yandong, Gu Kejiang

(Jiangsu Petroleum Exploration Bureau, Yangzhou 225009, China)

Abstract: This article introduces the method of object-oriented of encoding. And it improves this method and does some modification. In practical problems, it's common to see big individuals and very long individuals. Due to the process of the algorithm, the traditional GA gives a so-so performance. It comes up with a operation which matches this problem. It use a large probability to make all the properties change probably. At last, we use a function for testing. And the result proves that this method is useful.

Key words: method of object-oriented, big individuals, GA

遗传算法起源于上世纪 60 年代, 通俗易懂、解决复杂优化问题有很好的效果, 适用范围广, 大多数时候都能获得满意的解^[1]. 就像猴子爬山一样, 给予其合适的淘汰机制, 合适的奖励机制, 给其足够的空间, 它自己就能向着最终目标靠近^[2]. 它不需要对解空间做太多限制, 采用多点同时进行, 并行趋近的策略, 向下自行演化, 就能趋近最优解^[3]. 随着算法应用的增多, 解决复杂问题时会遇到更复杂的实际情况, 即遇到大个体问题时, 仍然需要一定的修改.

面向对象方法符合对世界客观现实的认识, 改变了思维方式, 打开了一扇通往未来的大门.

实际问题中常见复杂问题, 对应的解的数学模型往往非常复杂, 解的结构非常长. 这类问题使用遗传算法时个体会很大. 由于遗传算法自身的搜索特点, 在处理大个体时, 有时会陷入困境.

1 编码方案与进化方案

常用编码方式有两种, 一种是符号编码, 如二进制编码; 另一种方式则 1 个属性与 1 个参数对应的方法, 如实数编码(浮点型)、整数编码(整型), 这种方法把相应的遗传操作转变成适合实数运算的操作, 也保证了精度和速度, 便于阅读和分析. 两种方法都会把整个个体捏合成 1 个整串.

二进制编码方式更类似生物学里的 DNA 的组成方式. 生物学中 DNA 的核心部分就是由 4 种不同的物质排列组合而成的.

对于不同的问题而言, 使用遗传算法就会有相同的思路 and 结构: 种群、个体、适应度函数、进化操作等, 差异性主要表现在个体的组成方式、编码值、适应度函数的函数式等^[4]. 因而可以结合面向对象的思维方

式. 面向对象编码方案是用面向对象的方式去对个体进行编码, 区分各个属性的类型和取值范围. 若某属性的取值范围是连续的, 可以简单封装为 1 个常见的带有取值范围的数值类型的属性^[5]. 若某属性的取值范围是离散的, 可以封装为 1 个集合. 对于单一属性而言, 无论是连续的或者离散的, 都可以视为是线性的.

相应的, 种群处理为个体的集合. 种群密度这些属性一并保留并应用.

使用面向对象方法的好处在于模仿了实际问题的解的构成方式, 这也体现在程序设计中. 封装过的种群带入到进化过程中. 不需要额外编写代码.

例如对函数:

$$f(x_1, x_2) = x_1^2 + x_2^2, x_1, x_2 \in [-3, 3].$$

运用面向对象编码方式, 可以做如图 1 设计.

面向对象编码方案下, 个体的表现形式不再是 1 个整体, 而是对各个属性进行了封装和保护^[6]. 所以交叉变异等操作也需要额外的改动.

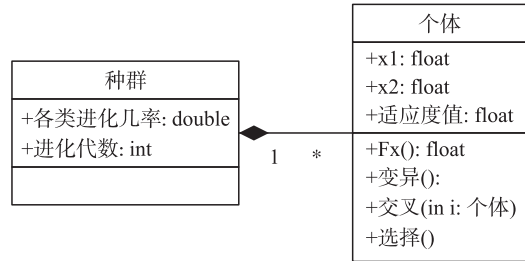


图 1 经典问题的面向对象编码类图

Fig. 1 Class diagram of object-oriented code in traditional problems

表 1 二进制编码下变异操作过程描述表

Table 1 Description of mutation in binary code

4 位变异前部分基因序列	4 位变异前部分基因序列的编码	4 位变异后部分基因序列的编码	4 位变异后部分基因序列的解码
211	11010011	11000011	195

表 2 二进制编码下单点交叉操作过程描述表

Table 2 Description of single point crossover in binary code

2、5 位交叉前部分基因序列	2、5 位交叉前部分基因序列的编码	2、5 位交叉后部分基因序列的编码	2、5 位交叉后部分基因序列的解码
211	11010011	10011011	155
172	10101110	11100110	230

对变异进行适当的修改. 原本遗传算法通过修改位的方式间接修改某一属性的值, 从外部来看, 这个修改几乎没有什么直接的规律. 因为属性取值是线性的. 设定变异操作如下: 当属性取值范围是连续的时候, 在原值 a 的两边随机选取 1 个新的值. 表示为 $a+k$, k 为偏移量. k 的取值范围与属性的值的区间长度有关. 自适应的 k 在算法中表现得更优秀. 当属性值的集合是离散的时候, k 与属性的值集合的元素个数有关.

变异示例如图, 因属性的取值区间长度为 20, 所以偏移量 k 的上限可以取 ± 3 . 对交叉进行适当的修改. 原本遗传算法通过两个个体的位交换间接修改两个个体中某属性的值. 从外部来看, 属性的值的改变也没有什么直接的规律. 面向对象方法中, 若属性直接交换, 则与原本的交叉含义有区别, 降低了搜索能力. 所以参考上述的变异, 设定交叉操作如下: 两个个体交换 1 个属性, 然后按照几率 p 在新值的周边范围进行 1 次微小的变异^[7].

交叉示例如图, 与交叉类似, 这里偏移量 k 的上限取 ± 3 .

表 3 面向对象编码下变异操作过程描述表

Table 3 Description of mutation in object-oriented code

个体属性值	变异后属性值
12(0, 20)	$12+k$ $-3 < k < 3$

表 4 面向对象编码下交叉操作过程描述表

Table 4 Description of cross in object-oriented code

个体属性值	变异后属性值	变异后 p 概率小变异属性值
A: 12(0, 20)	$a: 6(0, 20)$	$a: 6+k$ $-3 < k < 3$
B: 6(0, 20)	$b: 12(0, 20)$	$b: 12+j$ $-3 < j < 3$

2 大个体与遗传算法

个体是问题的解的直接映射. 个体的组成和解的组成有关联. 解的复杂程度影响了个体的复杂程度. 越是复杂的解, 其对应个体中包含的信息就越多.

考虑下列两个函数:

$$\min f(x) = \sum_{i=1}^3 x_i^2, \quad x_i \in [-3, 3],$$

$$\min F(x) = \sum_{i=1}^{100} x_i^2, \quad x \in [-1, 1].$$

前者的解由 3 部分组成,分别表示 x_1, x_2, x_3 的值. 后者的解由 100 个部分组成,分别表示 x_1 到 x_{100} 的值. 比较起来,后者的解的复杂程度是前者的很多倍.

一般遗传算法的处理方式是随机位交叉或变异. 解规模小、个体小的问题时往往一般只需要 1 位. 解规模大、个体复杂时,会增加变异、交叉长度来保证种群多样性和效率.

表 5 二进制编码下增加交叉长度描述表

Table 5 Description of increasing the length of crossover in binary code

交叉前部分基因序列	交叉前部分基因序列的编码	234 位交叉后部分基因序列的编码	234 位交叉后部分基因序列的解码
211	11010011	10100011	163
172	10101110	11011110	222

遗传算法通过制造个体内元素取值的改变,达到改变个体的函数值的目的,继而通过对个体的适应度值的计算和优胜劣汰的进化机制,达到搜索较优可行解的目的.

大个体在实际生活中较为常见,更符合实际问题的实际情况. 复杂的实际问题一般具有非常复杂的解形式. 如时间表问题,完整的解需要包含所有元素的排列结果. 当解变得复杂、个体变得非常庞大的时候,内部属性的每一次改变对个体的适应度函数值改变都不大. 运用遗传算法,会发现收敛速度较慢、局部收敛等状况. 这就是遗传算法在大规模或超大规模的多变量求解问题中性能较差的原因. 这时就需要对算法本身再进行修改.

返回头再考虑一下生物学中实际的情况. 个体中,染色体成双成对地出现、作用. 繁衍时,两个相同代的个体分别解开各自的成对的染色体,提供出其中 1 条,组合成新的染色体对. 这个过程中,父代个体的所有信息都得到了利用,参与了繁殖的过程.

面向对象编码下,我们没有把个体拉伸为线性排列的、类似 DNA 那样的结构. 结合增大交叉长度的做法,需要进一步设计交叉和变异这两个过程. 效仿增加交叉长度的方式,设定对个体的所有属性按照几率进行变异或交叉的操作.

交叉时,两个个体按照对位交叉的方式,从第一个属性开始,对所有属性按照几率 A 进行对位交叉. 交叉方式为前一节所述的方式,内部包含 1 个小变异操作的可能. 交叉内变异几率记为 P_{cm} . 因为个体内的每个属性都按照均等几率进行交叉操作,所以对两个个体进行多次交叉,得到的可能是不同的子代. 结合自适应的方式,交叉中小变异的偏移量 k 如下.

$$-\frac{R}{\log_2 10n} < k < \frac{R}{\log_2 10n}.$$

假设个体 $X(12, 4, 5, 11)$ 和个体 $Y(19, 7, 3, 5)$ 两个个体,分别包含 4 个属性,所有属性的取值范围为 $[0, 20]$ 的整数. X 和 Y 发生交叉. 对各个属性按几率 A 进行对位交叉. 这里取 $A=0.75$,最终 1、2、3 位发生交叉. 没有发生交叉内小变异时,得到子代个体 X' 和 Y' . 按几率 P_{cm} 发生小变异时得到新的子代个体 X' 和 Y' . 这里 $P_{cm}=0.8$,变异的 $k=3$. 整个过程示意如表 6.

表 6 交叉操作结果示意表

Table 6 Results of crossover

父代个体 X	父代个体 Y	1、2、3 交叉后 子代个体 X'	1、2、3 交叉后 子代个体 Y'	1、2 属性触发小变 异后子代个体 X'	1、2 属性触发小变 异后子代个体 Y'
12	19	19	12	18	14
4	7	7	4	6	3
5	3	3	5	5	3
11	5	1	5	1	5

变异时,对单一个体,从第一个属性开始,对所有属性按照几率 B 进行变异. 结合自适应的方式,变异的偏移量 k 如下.

$$-\frac{R}{\log_4 2n} < k < \frac{R}{\log_4 2n}.$$

取上文的例子个体 $X(12,4,5,11)$, 发生变异, 各属性按照几率 B 发生变异. 这里取 $B=0.5$. 变异的 $k=3$. 最终 12 位发生变异. 过程示意如表 7.

当 A, B 等于 1 时, 即对所有属性都进行了操作. 当 A, B 等于 0 时, 则不会对任何属性进行操作.

遗传算法的搜索方式是自发的、随机的. 大个体的 1 个特点是, 单一属性的取值发生变化时, 整个个体的适应度值变化量不一定明显. 为了放大差异, 必要的时候要选用更好的适应度函数. 另一方面可以增加变化次数, 进行更多的交叉或变异.

修改进化过程, 提出如下的步骤.

(1) 构造种群 $P1$, 种群密度设为 70.

(2) 对种群内个体按照适应度值进行降序排序.

(3) 最优的个体另存为 I .

(4) 建立空种群 $P2$.

(5) 遍历种群 $P1$, 对第 i 个个体按照几率 P_c 和第 $71-i$ 个个体进行双向交叉. 交叉过程如上节. 得到两个子代, 存入种群 $P2$. 这个, 过程完成后可以得到两倍于原种群密度, 140 个子代个体的新种群.

(6) 对种群 $P2$ 按照适应度值进行降序排序, 将前 70 个复制到原种群 $P1$ 中.

(7) 遍历种群 $P1$, 对每个个体按照几率 P_m 进行变异. 变异过程如上节所述.

(8) 对 $P1$ 按照适应度值进行降序排序, 删除最差的个体, 用 I 代替.

(9) 重复步骤 (2) ~ (5), 直到符合结束条件.

前文所述几率中, 个体内属性交叉的几率 $A=0.75$, 个体内属性变异几率 $B=0.5$, 交叉内变异几率 $P_{cm}=0.8$. 个体双向交叉几率 $P_c=0.8$, 个体变异几率 $P_m=0.1$.

第 3 步的意义在于模拟生物进化环境中两个父代个体的多个子代个体一起存在, 在自然大环境中参加优胜劣汰的残酷选择. 劣质的会立刻被淘汰, 优质的会获得生存和繁衍的权利^[8].

设定终止条件为迭代 200 次. 遗传算法的诸多例子中, 200 代时皆已收敛.

过程的流程图如图 2.

3 函数测试

采用一个测试函数对上述的方法进行测试:

$$\min F(x) = \sum_{i=1}^{100} x_i^2, \quad x \in [-1, 1].$$

该函数为一个经典的遗传算法测试用例函数的变种^[9]

$$\min f(x) = \sum_{i=1}^3 x_i^2, \quad x_i \in [-3, 3].$$

表 7 变异操作结果示意表

Table 7 Results of mutation

变异前属性 X	1, 2 位变异后属性 X'
12	10
4	7
5	5
11	11

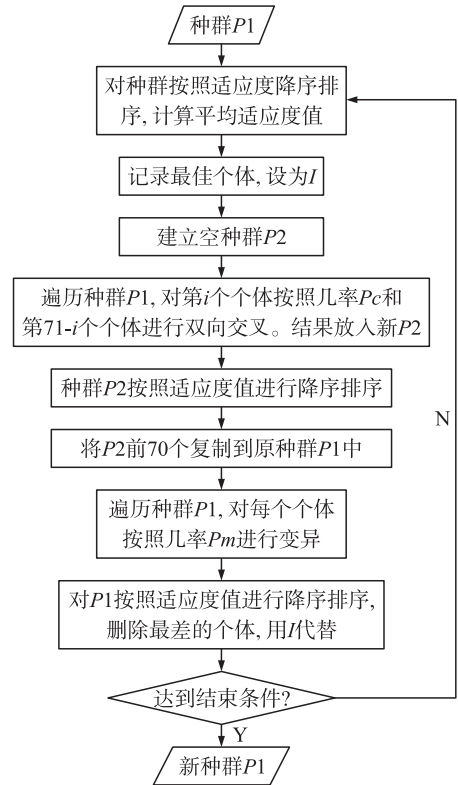


图 2 进化过程流程图

Fig. 2 Flow chart of the process of evolution

原函数 f 是一个 3 元函数, 在所有的 $x=0$ 时有函数值取最小值 0. 相应的, 根据简单的推断, 得出函数 F 在所有的 $x=0$ 时有最小值 0. 应用遗传算法解决该问题时, 构建的个体含有 100 个属性, 个体规模较大.

设计适应度函数:

$$F_{\text{fitness}} = 100 - \sum_{i=1}^{100} x_i^2, \quad x_i \in [-1, 1],$$

记录每一代的最优解的函数值. 横轴表示进化代数, 纵轴表示函数值, 可以得到图象如图 3 所示.

其中曲线为最优个体对应的函数值. 每进化 50 代显示 1 次最优和平均函数值. 可以看出整体在慢慢收敛, 证明方法可靠.

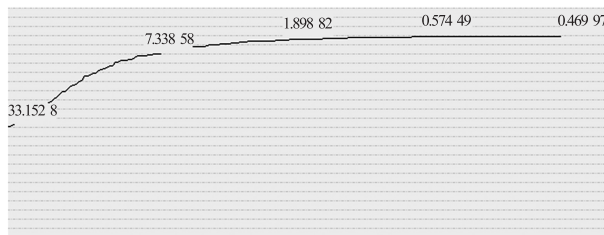


图 3 算法运算结果结果图

Fig. 3 Chart of the operation results

4 结论

本文介绍了面向对象编码技术在遗传算法中的应用, 分析了复杂问题的大个体情况, 提出了改进意见, 并进行了测试. 大个体问题在实际生活中较为常见, 需要更多地考虑. 虽然测试结果还不尽如人意, 但是这次大胆的尝试是可行的.

[参考文献]

- [1] 叶碧虾. 基于遗传和禁忌搜索算法的排课系统研究与实现[D]. 厦门: 厦门大学软件学院, 2009.
- [2] 孙建平, 梅晓勇, 肖政宏, 等. 关联规则在高校智能排课系统中的应用[J]. 计算机应用, 2002, 22(5): 37-39.
- [3] 王倩, 张锦华. 基于 GATS 算法的面向对象测试用例自动生成[J]. 郑州轻工业学院学报: 自然科学版, 2011, 26(6): 31-34.
- [4] 周海清, 陈正汉. 面向对象的深度搜索遗传算法及其工程应用(I)-算法与程序[J]. 岩石力学与工程学报, 2005, 11(24): 1996-2002.
- [5] 王小平, 曹立明. 遗传算法-理论、应用及软件实现[M]. 西安: 西安交通大学出版社, 2002.
- [6] 刘道华, 原思聪, 邬长安, 等. 面向对象的改进遗传算法优化研究[J]. 华中科技大学学报: 自然科学版, 2008, 36(7): 89-92.
- [7] 刘兴隆. 遗传算法中交叉操作研究及应用[J]. 东北电力学院学报, 2008(8): 34-37.
- [8] 陈文伟. 智能决策技术[M]. 北京: 电子工业出版社, 1998.
- [9] 关志华, 寇纪淞, 李敏强. 一种改进的遗传算法 Scatter GA[J]. 控制与决策, 2002, 17(5): 579-582.

[责任编辑: 顾晓天]