进化算法简介

XXX XXX XXX 刘汶鑫张毅张甲栋

March 21, 2018

浙江大学计算机科学与技术学院

目录

前言

定义

又叫演化计算，是模拟自然界中的生物的演化过程产生的一种群体导向的随机搜索技术和方法。

是一种通用的问题求解方法，具有自组织、自适应、自学习性和本质并行性等特点，不受搜索空间限制性条件的约束，也不需要其它辅助信息。

思想

进化算法是受生物进化过程中“优胜劣汰”的自然选择机制和遗传信息的传递规律的影响，通过程序迭代模拟这一过程，把要解决的问题看作环境，在一些可能的解组成的种群中，通过自然演化寻求最优解。

种类

- 遗传算法 (Genetic Algorithms)

种类

- 遗传算法 (Genetic Algorithms)
- 演化策略 (Evolution Strategy)

种类

- 遗传算法 (Genetic Algorithms)
- 演化策略 (Evolution Strategy)
- 演化规划 (Evolution Programming)

种类

- 遗传算法 (Genetic Algorithms)
- 演化策略 (Evolution Strategy)
- 演化规划 (Evolution Programming)
- 遗传程序设计 (Genetic Programming)

种类

- 遗传算法 (Genetic Algorithms)
- 演化策略 (Evolution Strategy)
- 演化规划 (Evolution Programming)
- 遗传程序设计 (Genetic Programming)
- 多种群协同进化 (multi-species cooperative)

种类

- 遗传算法 (Genetic Algorithms)
- 演化策略 (Evolution Strategy)
- 演化规划 (Evolution Programming)
- 遗传程序设计 (Genetic Programming)
- 多种群协同进化 (multi-species cooperative)
- 差分进化算法 (Differential Evolutionary)

演化规划 (EPA)

定义

定义

演化规划 (Evolutionary Programming Algorithm, EPA) 是由美国学者 Lawrence J. Fogel 于 1960 年提出的，它适用于解决目标函数或约束条件不可微的复杂非线性实值连续优化问题。它与遗传算法类似，但要优化的程序结构是固定的，而其数值参数则可以进化。

思想

EP 模拟生物种群层次上的进化，因此在进化过程中主要强调生物种群行为上的联系，即强调种群层次上的行为进化而建立父、子代间的行为链。

EP 算法最重要的一个操作是变异操作。通过变异，父代群体中的每一个个体产生一个子代个体，父代和子代中最好的那一半被选择生存下来。

Algorithm 1: Evolutionary Programming Algorithm

Input: 个体表现型 X, 群体规模 N, 迭代次数 G 等

Output: 子代

```
随机产生初始群体并计算适应值 (含 N 个个体)
while not done do
    //终止条件：达到规定的进化代数，或若干代内种
      群中最好个体的函数值不再发生变化，则终止进化
    for i = 1; i < N; i++ do
        对 X_i 进行变异得到 X'_i
        对 X_i 进行可行性检查
        计算 X_i 的适应值
    end
    从 2N 个个体中选择 N 个个体 //随机型 q-竞争法
end
return 子代;
```

q-竞争选择算法

演化规划算法的选择策略采用的是 q-竞争机制，这也是与进化策略算法最大的不同点，q-竞争说白了就是选择优质解的同时，以一定的随机概率接受较差的解。多数是优解，少数是比较差的解，共同组成一组父代，为下一次进化做准备。

q-竞争选择算法

思想

将 N 个父代进化的 N 个子代一起放在一起，从中随机选择不重复 q 个个体组成一个组，然后依次对 2N 个个体的每一个个体进行计分，将 2N 个依次每次一个与随机挑选出的群组的每一个成员进行比较，相比优的话，则会对应的个体的分数加 1，最后对分数进行排序，选择分数最高的 N 个个体！

缺点

大量的研究证明，由于过度选择、变异操作破坏有效模式、参数选取不当等原因，该算法存在一些亟待克服的**缺点**：

（1）容易出现早熟收敛现象

（2）进化后期，个体之间的竞争趋缓导致算法后期的搜索效率降低等。

改进

- 基于柯西变异的进化规划算法 1996

改进

- 基于柯西变异的进化规划算法 1996
- 基于 Lévy 概率分布的进化规划算法 (LEP) 2004

改进

- 基于柯西变异的进化规划算法 1996
- 基于 Lévy 概率分布的进化规划算法 (LEP) 2004
- 一种求解数值优化问题的快速进化规划算法 2004

改进

- 基于柯西变异的进化规划算法 1996
- 基于 Lévy 概率分布的进化规划算法 (LEP) 2004
- 一种求解数值优化问题的快速进化规划算法 2004
- ...

遗传程序设计

背景

自计算机出现以来，计算机科学的一个方向性目标就是让计算机自主进行程序设计，即只要告诉计算机要解决的问题，而不需要告诉它具体如何去做。遗传程序设计便是在该领域的一种尝试。

遗传程序设计最早由 John Holland 的遗传算法衍生而来，是一种由生物学引发的不依赖于领域的方法，它从问题需求的高层次描述中自动地产生计算机程序。

20 世纪 80 年代以来，演化计算成为国际上诸多领域研究的热点。作为演化计算最新的分支，遗传程序设计 (GP) 自 1990 年前后提出后便成为广大学者关注的方向。

研究现状

目前在国外，GP 在某些方面已经进入了应用性研究的阶段。在模拟与数字电路设计与分布式模糊控制方面，用 GP 自动设计的成果已经有了相当优良的性能。

国内对 GP 的研究起源于 1990 年前后，现在已开始成为诸多领域的研究热点。武汉大学软件工程国际重点实验室在动态系统的演化建模方面具有国际领先的水平。

定义

遗传程序设计是借鉴生物界的自然选择和遗传机制，在遗产算法的基础上发展起来的搜索算法，以树型结构作为个体结构的遗传算法称为遗传程序设计。

思想

在标准的遗传算法中，由定长字符串组成的群体借助于复制、交叉、变异等遗传操作不断进化找到问题的最优解或次优解。

遗传程序设计运用遗传算法的思想，常采用树的结构来表示计算机程序，从而解决问题。

对于许多问题，包括人工智能和机器学习上的问题都可看做是需要发现一个计算机程序，即对特定输入产生特定输出的程序，形式化为程序归纳，那么遗传程序设计提供了实现程序归纳的方法。

基本原理

遗传程序设计借助达尔文进化论中适者生存的理论，模拟自然界生物体的自然选择和进化的过程，从产生一个随机的计算机程序群体出发，以适应值度量为衡量计算机程序解决特定问题好坏的标准，基于适应值按概率方式从群体中选出计算机程序进行复制和杂交等操作，以适当的停止准则终止循环，迭代执行其中的每一个计算机程序，在可能的计算机程序空间中寻找适应性最好的程序，最终获得对特定输入产生所要求输出的计算机程序。

对象

与传统的数值优化算法不同，遗传程序设计的操作对象是规模和形状都能够动态变化的具有分层结构的计算机程序，从而不再局限于数值优化的范畴。

优点

遗传程序设计由已知的数据在适应值度量的推动下推导出其内在的未知规律即程序，避免了对求解过程的限制和先验性假设的要求。

框架

对程序设计问题，先产生一个不费事的有错误的解，然后再修改使它正确工作，这种做法一般要比坚持要求第一个解就完全没有缺陷的做法有效的多。遗传程序设计正是基于这样一种思想而发展起来的，它通过树的改变分层节点和结构链接关系来优化以判断是否满足终止准则，从而得到最优解。

流程图

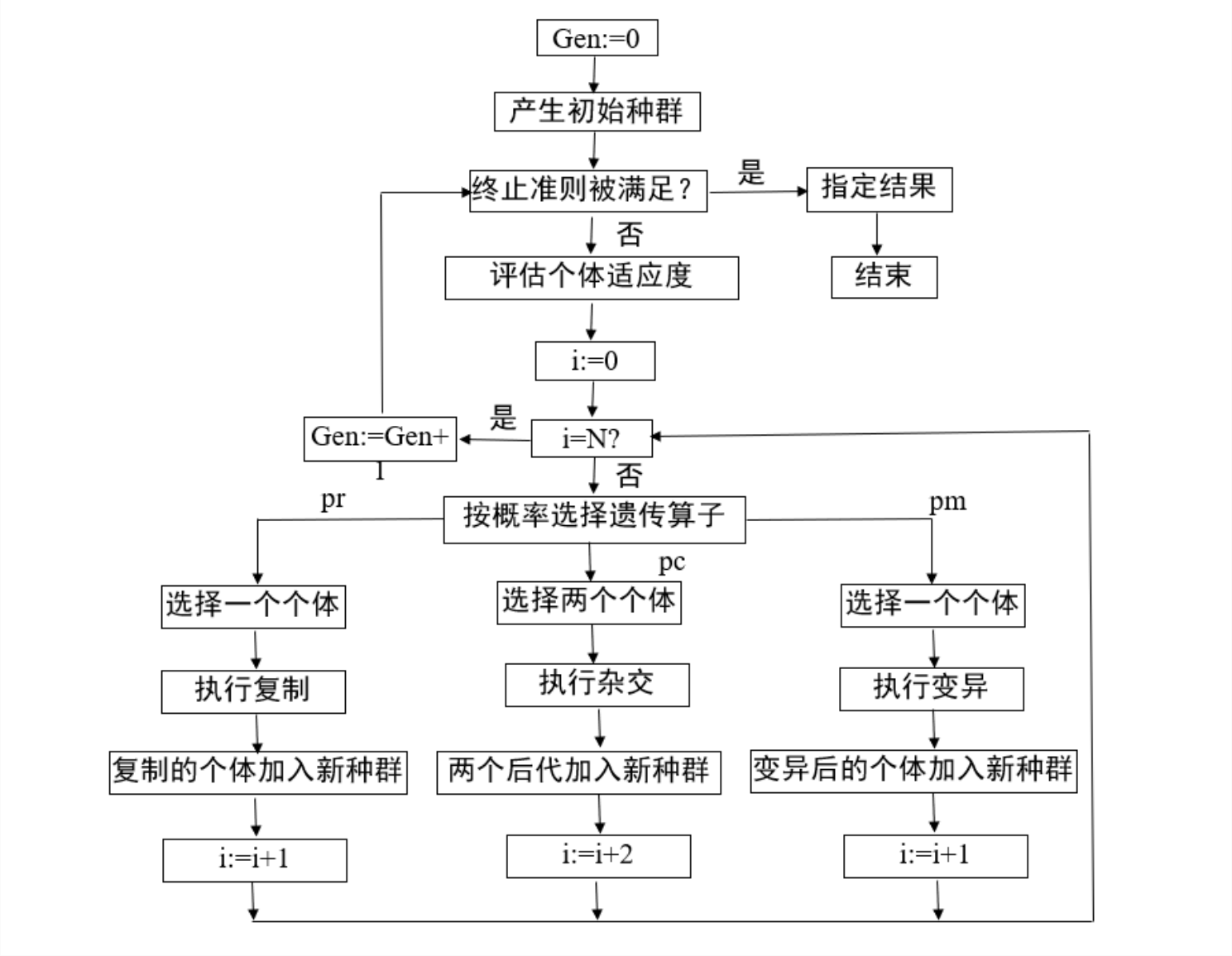


Figure 1: 遗传程序设计的流程图

算法

遗传程序设计的基本算法：

(1) 随机生成初始群体，其中个体是由表示问题的函数和终止符随机组合而成的计算机程序。

(2) 循环执行下列各步直到满足终止准则为止：

I) 运行群体中的每个计算机程序，并对其赋予适应度；

II) 运行下面两个主要操作产生新的计算机程序群体：

i) 把当前一代计算机程序复制成新一代计算机程序，被复制的个体依其适应度随机选定。

ii) 随机选定双亲个体部位进行交叉操作产生新个体，双亲个体也依适应度随机选定。

(3) 用结果标定法确定的程序被认为是运行的结果，它可能是一个正确(或近似) 的问题答案。

关键

遗传程序设计所需解决的关键问题：

(1) 程序的表示；

(2) 程序好坏的评价标准；

(3) 遗传算子的设计。

程序的表示

在遗传程序设计中，种群中的个体是计算机程序。为了程序表示的简单性和容易验证程序的句法，遗传程序设计用 LISP 语言来表示程序。

考虑一个简单的 LISP 程序，该程序简单地返回一个自然数 n 的平方：

>(defun square(n)(*nn))

SQUARE

下面是一个计算实数绝对值的 LISP 程序：

>(defun abs(x)(if(< x 0)(-x)(x)))

ABS

LISP 程序的主体由类似于 (*nn) 和 (if(< x 0)(-x)(x)) 的一些 S-表达式所组成。

一般来说一个表示 LISP 程序的 S-表达式通常由一些函数和端点组成。

程序的表示

构造 LISP 程序的 S-表达式是以前缀表达式形式表示的。给定一个 S-表达式，我们容易构造出它的语法树，而对该树进行先序遍历便可得到给定的 S-表达式。

例如 S-表达式 (+12(if(> x 10)56)) 所对应的语法树如下图所示：

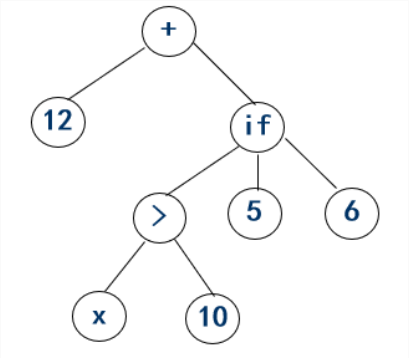


Figure 2: S-表达式的语法树

在语法树中，树的外部结点 (叶结点) 分别用变量 x 和常量 5，6，10，12 来标记，这些变量和常量通常称为端点，而内部结点分别用函数 +，>，IF 来标记。

实现

用遗传程序设计求解问题有以下五个基本步骤：

- 选择端点集；

实现

用遗传程序设计求解问题有以下五个基本步骤：

- 选择端点集；
- 选择函数集；

实现

用遗传程序设计求解问题有以下五个基本步骤：

- 选择端点集；
- 选择函数集；
- 确定适应函数；

实现

用遗传程序设计求解问题有以下五个基本步骤：

- 选择端点集；
- 选择函数集；
- 确定适应函数；
- 确定控制参数；

实现

用遗传程序设计求解问题有以下五个基本步骤：

- 选择端点集；
- 选择函数集；
- 确定适应函数；
- 确定控制参数；
- 确定指定结果的方法和终止程序运行的条件。

选择端点集和函数集

- 在遗传程序设计中，计算机程序用 LISP 程序的语法树来表示。每一棵语法树的内部结点由一些函数构成，而叶结点由一些变量和常数构成。

选择端点集和函数集

- 在遗传程序设计中，计算机程序用 LISP 程序的语法树来表示。每一棵语法树的内部结点由一些函数构成，而叶结点由一些变量和常数构成。
- 对于一个给定的问题，首先要确定由所有可能求解该问题的程序所组成的程序空间 S。由于 LISP 程序通常可以由一些简单的函数、变量和常数经过有限次运算和复合而生成。因此若要指定程序空间 S，我们只要指定可以构成中程序的函数集和端点集。

选择端点集和函数集

- 在遗传程序设计中，计算机程序用 LISP 程序的语法树来表示。每一棵语法树的内部结点由一些函数构成，而叶结点由一些变量和常数构成。
- 对于一个给定的问题，首先要确定由所有可能求解该问题的程序所组成的程序空间 S。由于 LISP 程序通常可以由一些简单的函数、变量和常数经过有限次运算和复合而生成。因此若要指定程序空间 S，我们只要指定可以构成中程序的函数集和端点集。
- 函数集可以包含：
  <1> 算数运算 +,-,*,/等；
  <2> 数学函数 sin,cos,exp,log 等；
  <3> 布尔运算 AND,OR,NOT 等；
  <4> 条件运算 IF-THEN-ELSE 等；
  <5> 循环运算 DO-UNTIL,FOR,WHILE 等。

选择端点集和函数集

- 在遗传程序设计中，计算机程序用 LISP 程序的语法树来表示。每一棵语法树的内部结点由一些函数构成，而叶结点由一些变量和常数构成。
- 对于一个给定的问题，首先要确定由所有可能求解该问题的程序所组成的程序空间 S。由于 LISP 程序通常可以由一些简单的函数、变量和常数经过有限次运算和复合而生成。因此若要指定程序空间 S，我们只要指定可以构成中程序的函数集和端点集。
- 函数集可以包含：
  <1> 算数运算 +,-,*,/等；
  <2> 数学函数 sin,cos,exp,log 等；
  <3> 布尔运算 AND,OR,NOT 等；
  <4> 条件运算 IF-THEN-ELSE 等；
  <5> 循环运算 DO-UNTIL,FOR,WHILE 等。
- 端点集通常由变量和常量组成。

初始化

- 假设问题的端点集和函数集分别为 T 和 F。

初始化

- 假设问题的端点集和函数集分别为 T 和 F。
- 初始种群中的每个 S-表达式，即每棵树，可按以下步骤产生：
  <1> 首先建立一个根结点，从函数集 F 中随机地选择一个函数作为根结点的标记；
  <2> 每当树中的一个结点被标记为函数集 F 中的一个函数 f 后，则为该结点建立 z(f) 个儿子结点，而每个儿子结点的标记从 C=FYT 中随机地选取；
  <3> 每当树中一个结点被标记为端点集 T 中的一个元素时，则该结点成为树的一个叶结点；
  <4> 如此反复进行，直到所有结点都被标记。

初始化

- 假定 F={+,-,*,/}，T={a,b,c,d}。下面给出了建立一棵树结构的过程：

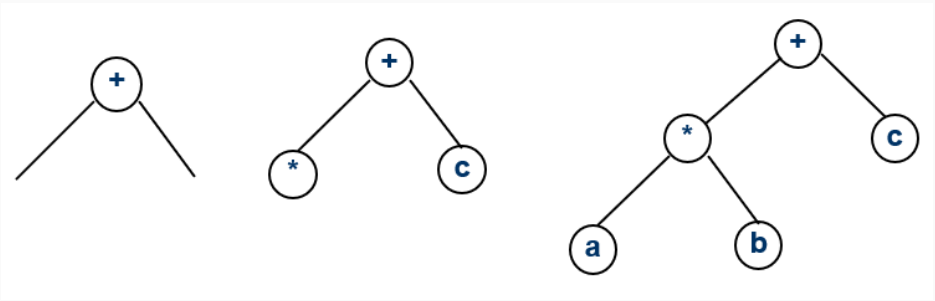


Figure 3: 语法树的构建

初始化

- 假定 F={+,-,*,/}，T={a,b,c,d}。下面给出了建立一棵树结构的过程：

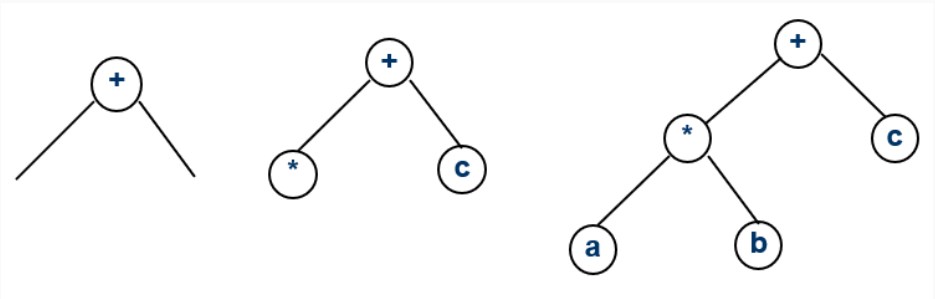


Figure 3: 语法树的构建

- 上述树生成过程可以有几种不同的实现方式：
<1> 完全法：该方法预先定义所要生成的树的最大深度 Di，当要生成的结点的深度小于最大深度时，则从函数集 F 中随机地选取一个元素作为该结点的标记，当要生成的结点的深度等于最大深度时，则从端点集 T 中随机地选取一个元素作为该结点的标记。

初始化

<2> 生长法：该方法预先定义所要生成的树的最大深度 Di，当要生成的结点的深度小于最大深度时，则从中随机地选取一个元素作为该结点的标记，当要生成的结点的深度等于最大深度时，则从端点集 T 中随机地选取一个元素作为该结点的标记。

<3> 混合法：该方法预先定义所要生成的树的最大深度 Di，对从 2 至最大深度的每一个深度，生成相同个数的树。对每一个深度，其中一半的树用完全法生成，而另一半用生成方法生成。

例如，假定树的最大深度定义为 6，种群中个体的个数为 100，则用混合方法将分别产生深度为 2,3,4,5,6 的树各 20 个，其中深度为的树有 10 个用完全法产生，有 10 个用生成方法产生。

适应函数

适应性函数是遗传算法和遗传程序设计得以实现的关键因素，是评价个体好坏的定量表述，决定着演化过程中群体选择复制及群体整体性的质量。

与其它演化算法相同，在遗传程序设计中，个体的适应性函数值是判断个体的质量的唯一办法。

个体适应性函数值的计算方法是问题相关的，有多种方式度量个体的适应性函数值，有些方法是明确的，有些方法是隐含的。

例如，若个体是下列程序：

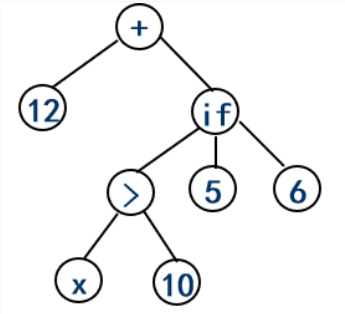


Figure 4: 程序示例

适应函数

评估该程序好坏的一种方法是：

(1) 建立一个样本集 $\{(x_i, y_i)|i = 1, 2, \Lambda, s\}$，其中 y_i 是当变量 x 取值为 x_i 时的期望输出；

(2) 当 $x = x_i(i = 1, 2, \Lambda, s)$ 时，运行该程序得到输出 z_i；

(3) 计算所得到的输出与期望输出之间的误差的绝对值之和 $e = \Sigma|z_i - y_i|$；

(4) 最后，用 e 作为该个体的适应性函数值。显然，适应性函数值越小，个体越好。

若个体是判定树，则判定树的分类准确率可以作为个体的适应性函数值。若个体是游戏策略，则游戏策略与其它策略对奕时取胜的次数可以作为个体的适应性函数值。

适应函数

个体适应性函数值有以下几种形式：

(1) 原始适应性函数值；

原始适应性函数值是以问题本身的自然术语叙述的适应性函数值度量。

(2) 标准适应性函数值；

标准适应性函数值对原始适应性函数值作简单变换，使得标准适应性函数值越小，个体越好。

(3) 调整适应性函数值；

调整适应性函数值 $\epsilon(0,1]$。调整适应性函数值越大，个体越好。

(4) 正规化适应性函数值。

正规化适应性函数值 $\epsilon[0,1]$，在基于适应性函数值比例的选择策略中可直接用作选择概率。

适应函数

设计适应性函数，一般有以下几个步骤：

(1) 原始适应性函数；

按照目标函数的计算方法直接计算原始适应性函数值。

(2) 标准化适应性函数；

将得出的原始适应性函数表述为标准适应性函数。

(3) 优化调整适应性函数；

调整最优个体与次优个体的微小差别。

(4) 正规化适应性函数。

正规化 (归一化) 已优化调整的适应性函数值。

父体选择策略

(1) 通常，遗传程序设计使用基于适应性函数值比例的选择策略。

(2) 若种群规模在 1000 以上，则经常使用一种贪婪过度选择策略。

(3) 该策略首先将种群中的个体按适应性函数值排序，然后将种群中的个体分为两部分。第一部分包含种群 $x\%$ 的最好个体，另一部分包含其它 $(1 - x\%)$ 的个体。当进行父体选择时，80% 的选择在第一部分中进行，20% 的选择在第二部分中进行。x 的取值以来于种群规模，通常通过实验确定。

种群规模	x
1000	32
2000	16
4000	8
8000	4

Figure 5: x 的取值

从上表可以看出，随着种群规模的增加，x 不断减小，选择压力逐步增大。

遗传算子

遗传程序设计中的遗传算子主要有复制、杂交和变异。变异算子的作用不及遗传算法中重要。

(1) 复制

复制算子首先按照某种基于适应性函数值比例的选择策略从种群中选择一个个体，然后将该个体不加改变地复制到下一代种群。

复制算子有一个参数 p_r，通常取 $p_r = 0.1$。

(2) 杂交

杂交算子分别从两个父体中随机地选择一个杂交点，然后交换父体中以杂交点为根结点的子树产生两个后代。

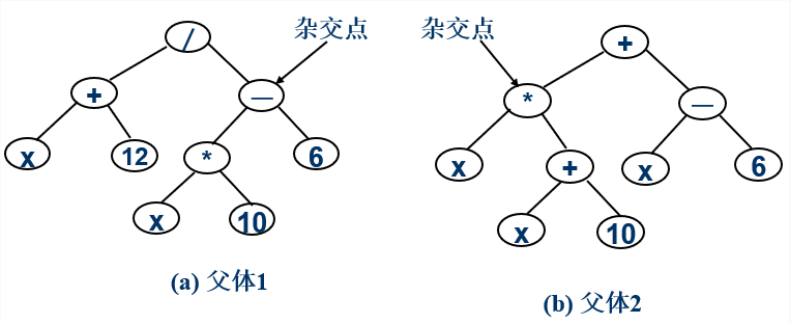


Figure 6: x 的取值

遗传算子

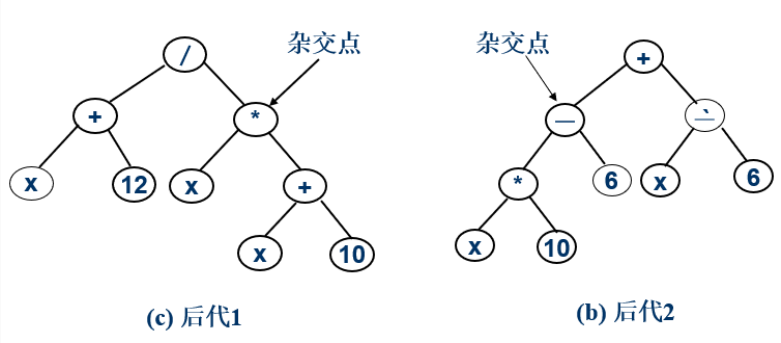


Figure 7: x 的取值

杂交算子有两个参数 p_c 和 p_{cn}，p_c 是按概率选择遗传算子时选择杂交算子的概率，而 p_{cn} 是在进行杂交时，在父体中选择内结点作为杂交点的概率。通常取 $p_c = 0.9$ 和 $p_{cn} = 0.9$。

在对父体进行杂交后，后代树的深度有可能加大。为了有效地利用计算机资源，防止产生巨型个体，遗传程序设计通常设置一个最大允许深度 D_c 进行控制。

若杂交后，有一个后代的深度超过了最大允许深度，则随机地选取一个父体代替该后代；若两个后代的深度都超过了最大允许深度，则用两个父体代替这两个后代。

遗传算子

(3) 变异

变异算子首先在父体中随机地选择一个结点，然后删除以该结点为根结点的子树，并在该结点处插入一个随机生成的子树。

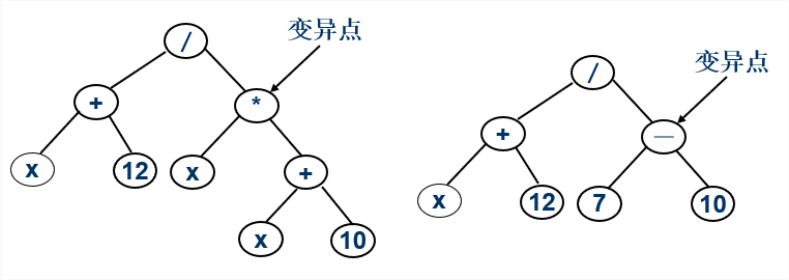


Figure 8: x 的取值

变异算子有两个参数 p_m 和 p_{mm}。p_m 是按概率选择遗传算子时选择变异算子的概率，而 p_{mm} 是在进行变异时，在父体中选择内结点的概率。

值得注意的是 J.R.Koza 建议 p_m，也就是说在遗传程序设计中不使用变异算子。近年来的遗传程序设计实践建议使用变异算子，但是以较低的概率。

应用

- 预测和分类：使用历史数据库来预测新事例。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。
- 人工生命：用计算机模拟生物的自然进化或发现规律。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。
- 人工生命：用计算机模拟生物的自然进化或发现规律。
- 神经网络设计“设计神经网络结构、发现学习规则和相关权值，以使神经网络完成指定的任务。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。
- 人工生命：用计算机模拟生物的自然进化或发现规律。
- 神经网络设计“设计神经网络结构、发现学习规则和相关权值，以使神经网络完成指定的任务。
- 图像和信号处理：图像识别、图像恢复、图像和声音的压缩等。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。
- 人工生命：用计算机模拟生物的自然进化或发现规律。
- 神经网络设计“设计神经网络结构、发现学习规则和相关权值，以使神经网络完成指定的任务。
- 图像和信号处理：图像识别、图像恢复、图像和声音的压缩等。
- 多 Agent 系统的自动设计：多个 Agent 协作共同处理实际问题。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。
- 人工生命：用计算机模拟生物的自然进化或发现规律。
- 神经网络设计“设计神经网络结构、发现学习规则和相关权值，以使神经网络完成指定的任务。
- 图像和信号处理：图像识别、图像恢复、图像和声音的压缩等。
- 多 Agent 系统的自动设计：多个 Agent 协作共同处理实际问题。
- 游戏策略：发现一个策略打败对手。

应用

- 预测和分类：使用历史数据库来预测新事例。
- 符号压缩：发现各变量间的隐含关系。
- 机器人：控制机器人行为，使其对环境做出反应。
- 人工生命：用计算机模拟生物的自然进化或发现规律。
- 神经网络设计“设计神经网络结构、发现学习规则和相关权值，以使神经网络完成指定的任务。
- 图像和信号处理：图像识别、图像恢复、图像和声音的压缩等。
- 多 Agent 系统的自动设计：多个 Agent 协作共同处理实际问题。
- 游戏策略：发现一个策略打败对手。
- 艺术：声音、三维图像的生成、计算机动画等。

应用实例 1

简单符号回归

问题：给定20个样本点 $(x_1, y_1), (x_2, y_2), \Lambda, (x_{20}, y_{20})$，其中 $x_i (i = 1,2,\Lambda, 20)$ 是区间 $[-1, 1]$ 中的随机数，而

$$y_i = x_i^4 + x_i^3 + x_i^2 + x_i$$

求出拟合这些样本点的函数。

应用实例 1

求解该问题的遗传程序设计如下：

(1) 端点集

端点集可取为 $T=\{x\}$。

(2) 函数集

函数集可仅由算术运算加法和乘法组成。一个更一般的选择是 $F=\{+,-,*,\%,\sin,\cos,\exp,\ln\}$。

(3) 适应值度量

个体f的原始适应值用下式计算：$err(f)=\sum_{i=1}^{20}\left|f(x_i)-y_i\right|$

应用实例 1

(4) 父体选择策略

使用轮盘赌选择。

(5) 种群初始化和算法终止准则

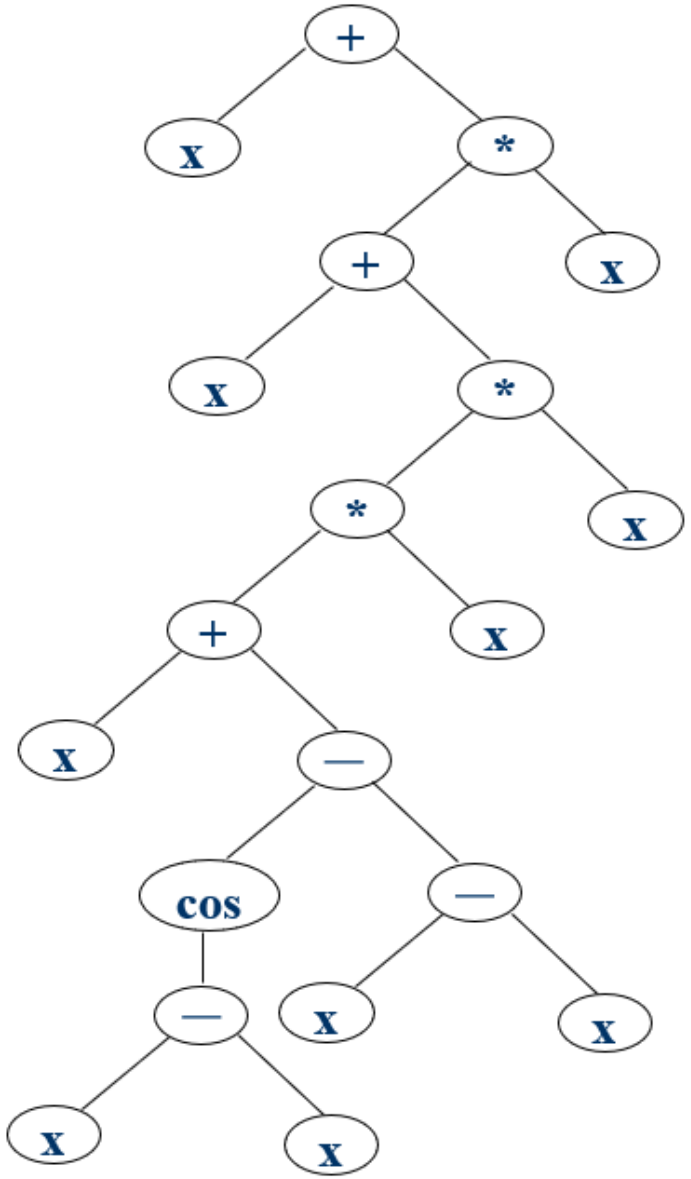
用混合法产生初始种群。而当$|f(x_i)-y_i|<0.01\,(i=1,2,\Lambda\ ,n)$或算法运行到预先指定的最大演化代数时，终止算法运行。

应用实例 1

(6) 参数设置

种群规模	500
最大演化代数	50
概率p_r	0.1
概率p_c	0.9
概率p_m	0
概率p_{cn}	0.9
最大深度D_i	6
最大深度D_c	17

应用实例 1



$$x^4 + x^3 + x^2 + x$$

应用实例 2

问题: 给定数学表达式 cos(2x)，我们希望发现与 cos(2x) 恒等的数学表达式。

将该问题视为一个符号回归问题。通过在某个区间内抽取一个随机样本，并计算给定函数在该样本的函数值，这样形成一组样本点，对所形成的样本点作符号回归。

应用实例 2

由于 $\cos 2x$ 是一个三角函数，所以在区间 $[0, 2\pi]$ 内随机地抽取20个值。利用这20个样本点 $(x_i, y_i)(i = 1,2,\Lambda\ ,20)$ 进行符号回归。

求解该问题的遗传程序设计如下：

(1) 端点集

$$T = \{x, 1.0\}$$

(2) 函数集

$$F = \{+, -, *, \%, \sin\}$$

应用实例 2

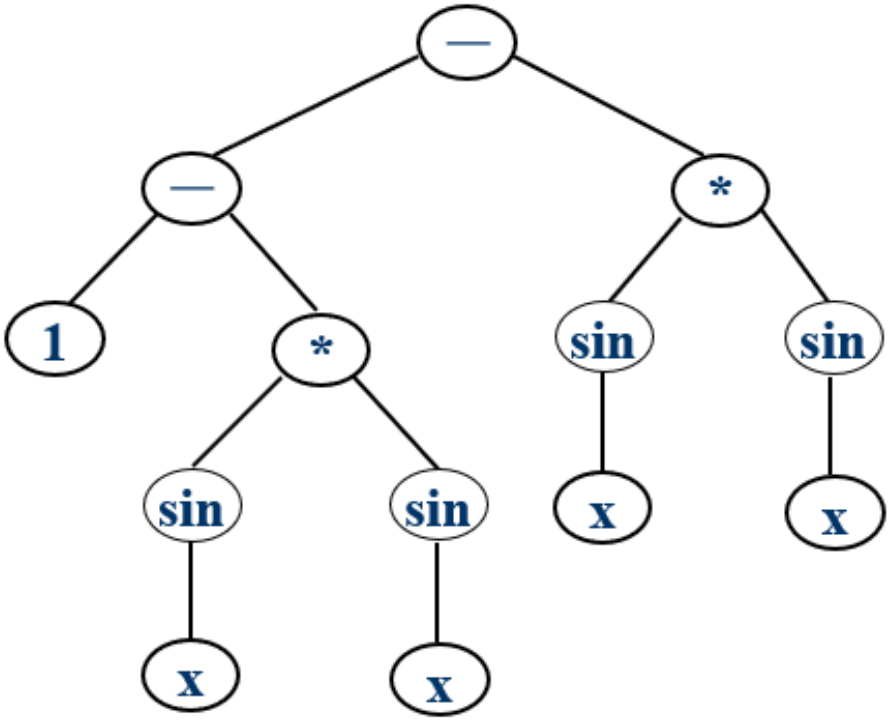
(3) 适应值度量

个体f的原始适应值用下式计算：

$$err(f)=\sum_{i=1}^{20}\left|f(x_i)-y_i\right|$$

父体选择策略、种群初始化、算法终止准则和参数设置均与上面相同.

应用实例 2

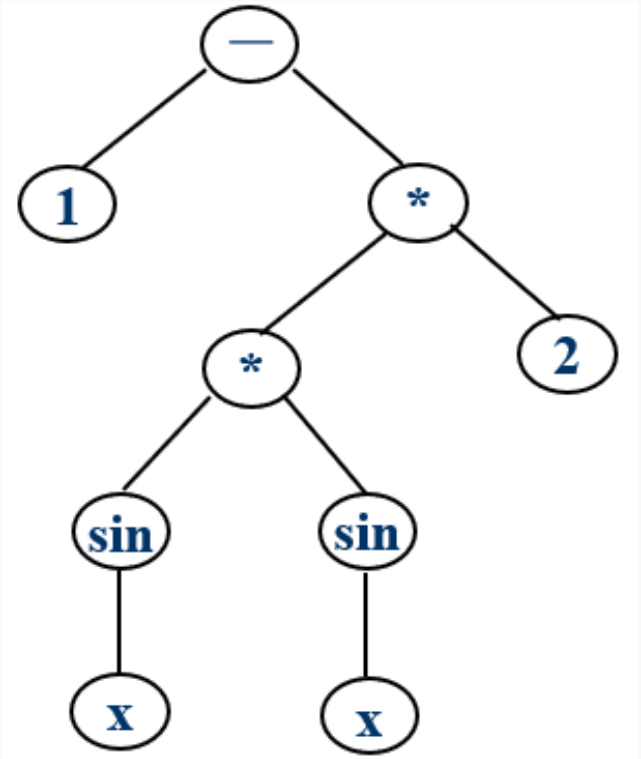


运行结果 1

$$1-2\sin^2 x$$

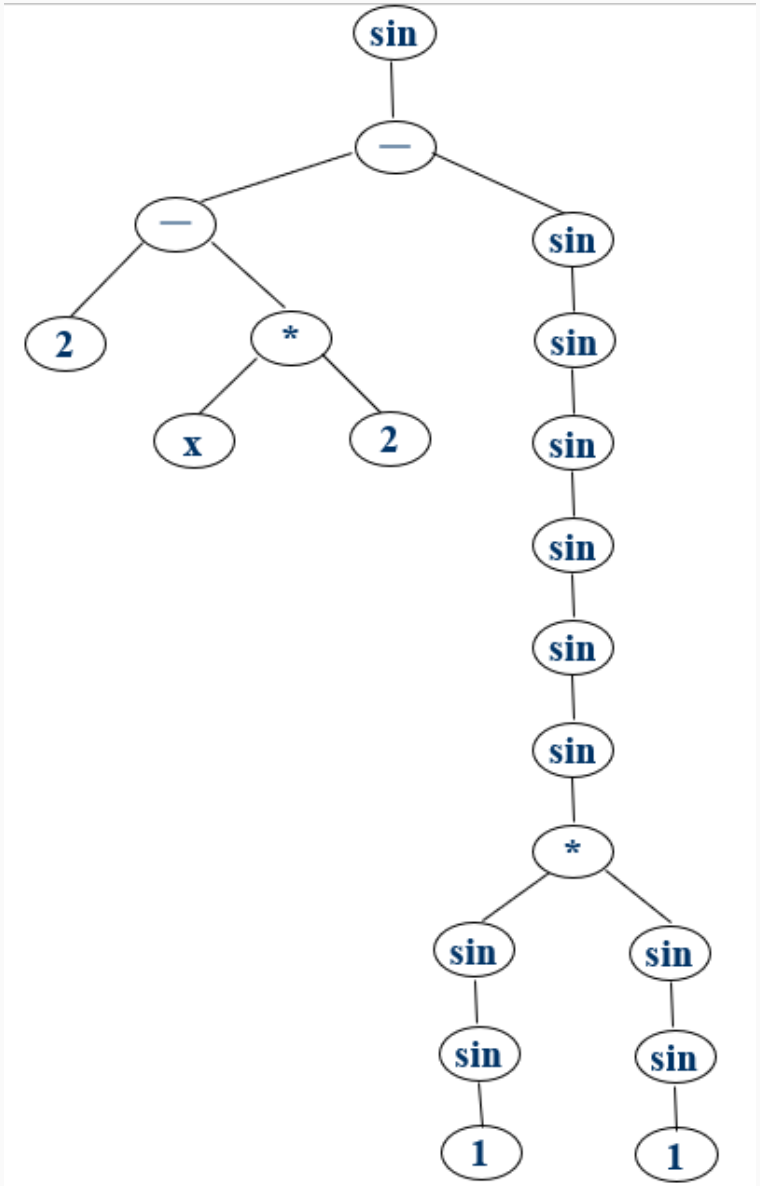
$$\cos 2x = 1-2\sin^2 x$$

应用实例 2



运行结果 2

应用实例 2



运行结果 3

应用实例 2

$$\sin(2-\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(1))*\sin(\sin(1)))))))-2x)$$

$$2-\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(1))*\sin(\sin(1)))))))$$

$$\approx \pi/2$$

$$\cos 2x = \sin(\pi/2-2x)$$

改进

遗传程序设计作为一种优化算法具有不需要先验性假设而可以进行全局搜索和进化最终得到最优解的特点，近年来人们在研究使用这种算法的过程中，也针对实际应用对其作了一些相应的改进。

遗传程序设计主要处理树状具有分层结构的程序，在实际编程中多采用 LISP 语言实现。LISP 语言由于缺乏统一的标准和方便的对外接口，作为一种解释语言运行速度较慢，可以采用几种具有不同数据类型的语言如 PROL0G 和 C 语言来加速程序进化的进程。

传统的遗传程序设计限制少、搜索速度慢，可以用定向的非周期图 (DAG) 取代树和森林存储程序群体，针对子树进行改进，以节省程序的运行时间和空间。

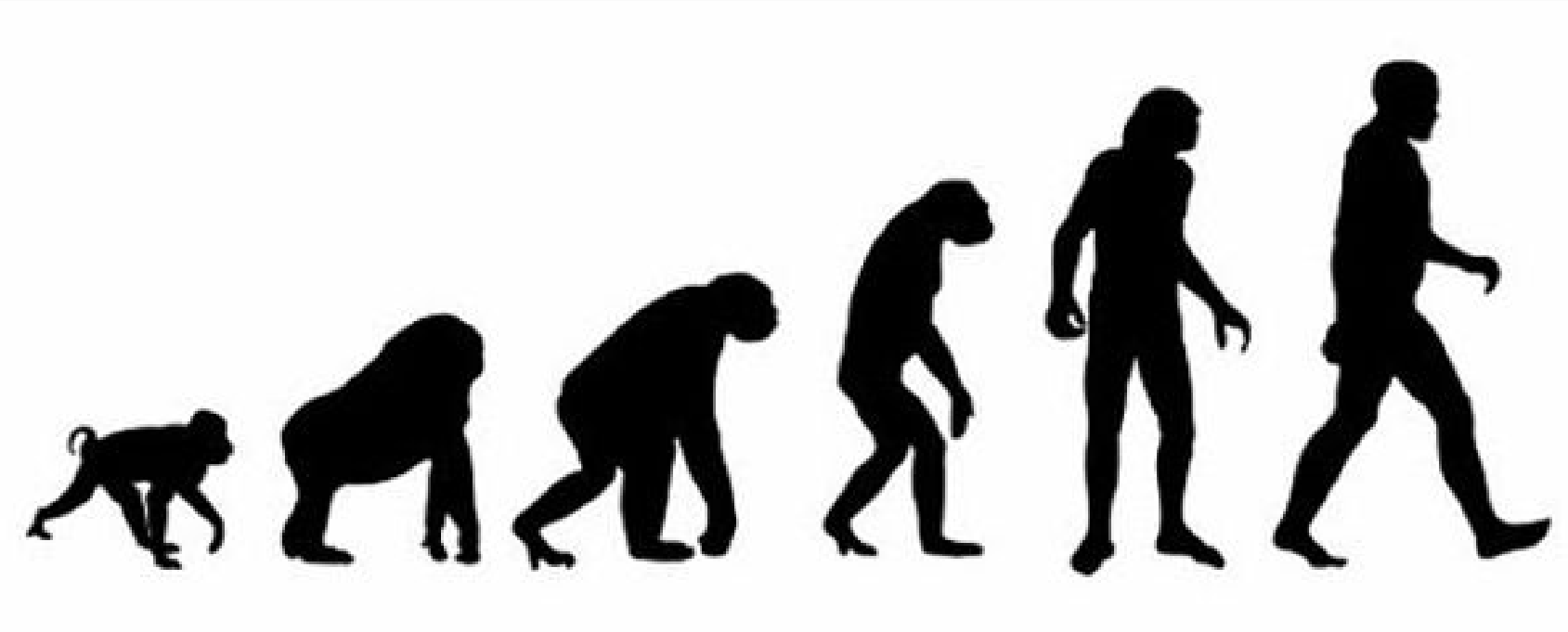
在算法方面，遗传程序设计可以与包括遗传算法、神经网络和模拟退火等的其他优化方法相结合，以得到更好的应用。

差分进化

定义

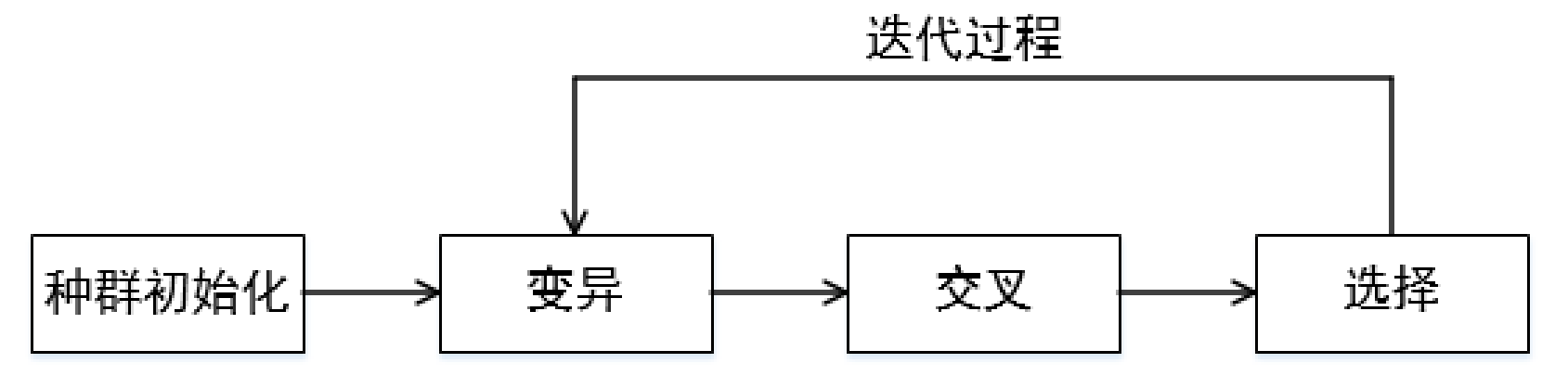
是一种基于群体智能的全局优化方法，其主要通过种群内个体之间的协同合作和相互竞争来产生群体智能，进一步指导进化过程的全局搜索。

算法思想



通过种群之间的**个体差异**和**优胜劣汰**的竞争策略产生新的个体，最终使种群接近或达到全局最优解。

算法框架



- **种群初始化**在解空间中随机、均匀地产生 M 个个体，每个个体由 n 个染色体组成，作为第 0 代种群，标记为

$$X_i(0) = (x_{i,1}(0), x_{i,2}(0), \cdots, x_{i,n}(0))$$

$$i = 1, 2, \cdots, M$$

- **变异、交叉、选择**三步操作迭代执行，直到算法收敛。第 g 次迭代的第 i 个个体标记为

$$X_i(g) = (x_{i,1}(g), x_{i,2}(g), \cdots, x_{i,n}(g))$$

$$i = 1, 2, \cdots, M$$

种群初始化

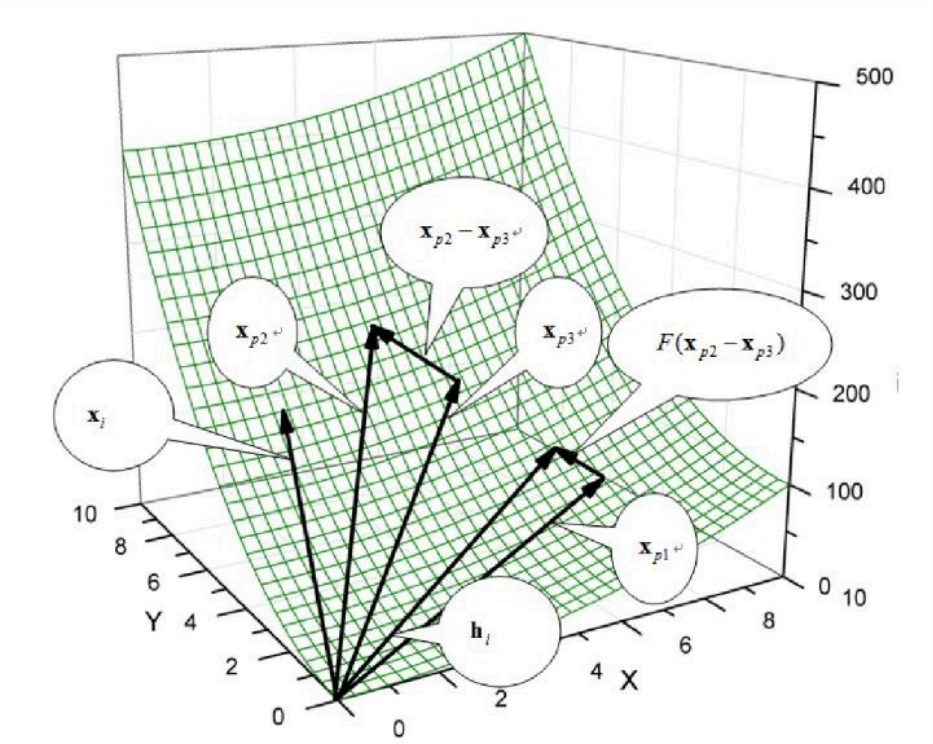
在 n 维空间里随机产生满足约束条件的 M 个染色体，第 i 个染色体的第 j 个维取值方式如下 (rand(0,1) 产生 0 到 1 的均匀分布的随机数)：

$$x_{i,j}(0) = L_j + rand(0,1)\,(U_j - L_j)$$
$$i = 1, 2, \cdots, M$$
$$j = 1, 2, \cdots, n$$

变异算子

在第 g 次迭代中，对个体 $X_i(g) = (x_{i,1}(g), x_{i,2}(g), \cdots, x_{i,n}(g))$，从种群中随机选择 3 个个体 $X_{p1}(g), X_{p2}(g), X_{p3}(g)$，且 $p1 \neq p2 \neq p3 \neq i$，则

$$H_i(g) = X_{p1}(g) + F \cdot (X_{p2}(g) - X_{p3}(g))$$

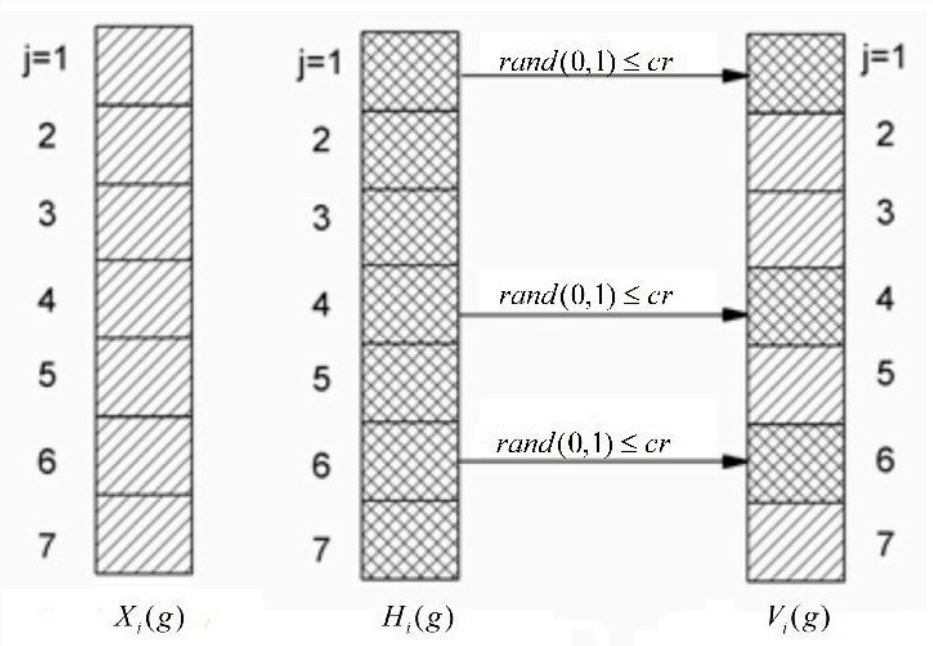


交叉算子

交叉操作可以增加种群的多样性，方法如下：

$$v_{i,j}(g) = \begin{cases} h_{i,j}(g), rand(0,1) \leq cr \\ x_{i,j}(g), else \end{cases}$$

其中 $cr \in [0,1]$ 为交叉概率，$rand(0,1)$ 是 $[0,1]$ 上服从均匀分布的随机数。



选择算子

首先查看根据评价函数选择 $V_i(g)$ 或 $X_i(g)$ 作为 $X_i(g+1)$

$$X_i(g+1) = \begin{cases} V_i(g), & \text{if} \quad f(V_i(g)) < f(X_i(g)) \\ X_i(g), & else \end{cases}$$

可以看出：

- 对每个个体，$X_i(g+1)$ 要好于或持平 $X_i(g)$。
- 肯定会收敛于最优点（可能是局部最优）。
- **变异、交叉** 操作有助于突破局部最优到达全局最优。

差分进化算法寻找函数最优解

定义关于参数 x, y 的函数表达式如下：

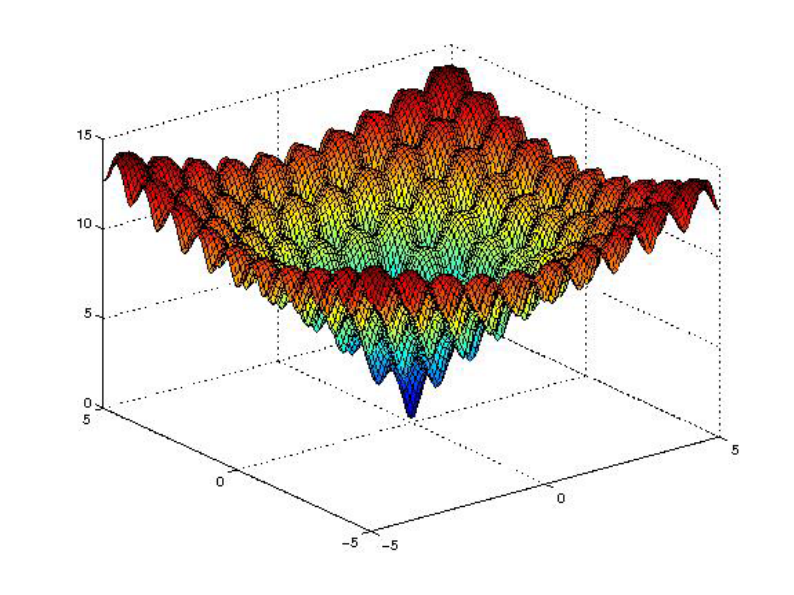
$$f(x,y) = -20e^{-0.2\sqrt{\frac{x^2+y^2}{2}}} - e^{\frac{\cos 2\pi x + \cos 2\pi y}{2}} + 20 + e$$

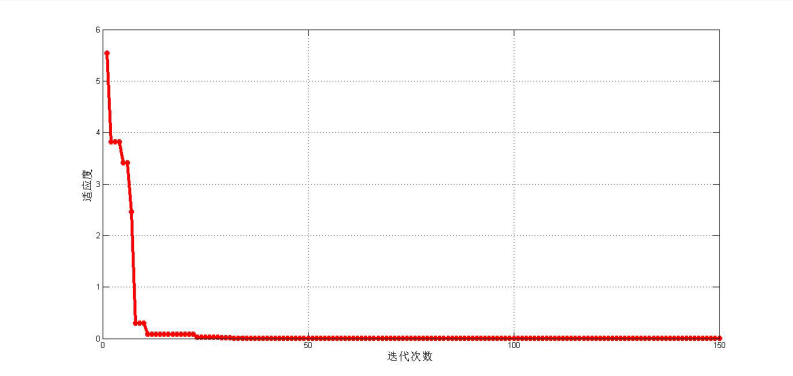
求函数的最小值。

- 个体定义：P, $P[0]$ 表示 x, $P[1]$ 表示 y。
- 取值范围：$-5 < x, y < 5$。
- 适应度定义：$f(x, y)$。

差分进化算法寻找函数最优解

用差分进化算法求解，效果如右图所示（参数设置：$N = 20, F = 0.5, cr = 0.5$，迭代次数 $T = 300$）





优缺点

和其他进化算法相比，差分进化算法具有以下优点：

1. 在非凸、多峰、非线性、连续不可微函数优化问题上表现出极强的稳健性。
2. 收敛速度快。
3. 操作简单，容易实现。

缺点：

1. 算法后期个体间差异逐渐缩小，收敛速度慢，容易陷入局部最优。
2. 控制参数和学习策略对算法性能有着重要的影响，并且高度依赖于优化问题的本质。
3. 有时需要过多的迭代才能搜索到全局最优。

参数的选取

- M：一般介于 $5 \times n$ 与 $10 \times n$ 之间，但不能少于 4，否则变异算子无法进行；
- F：一般在 $[0, 2]$ 之间选择，通常取 0.5；
- cr：一般在 $[0, 1]$ 之间选择，比较好的选择应在 0.3 左右。cr 取值偏大，收敛速度会加快，但易发生早熟现象。

参数的自适应调整 (F)

将变异算子中随机选择的三个个体进行从优到劣的排序，得到 X_b, X_m, X_w，对应适应度 f_b, f_m, f_w，则变异算子改为：

$$V_i = X_b + F_i (X_m - X_w)$$

同时，F 的取值根据生成差分向量的两个个体自适应变化，平衡全局搜索和局部搜索之间的矛盾。

$$F_i = F_l + (F_u - F_l) \frac{f_m - f_b}{f_w - f_b}$$

其中，

$$F_l = 0.1, F_u = 0.9$$

参数的自适应调整 (cr)

对于适应度好的解，取较小的 cr，使得该解进入下一代的机会增大；对于适应度差的解，则取交大的 cr，加快改变该个体的结构，使该解被淘汰掉。

$$cr_i = \begin{cases} cr_l + (cr_u - cr_l)\frac{f_i - f_{min}}{f_{max} - f_{min}} & \text{if} \quad f_i > \bar{f} \\ cr_l & \text{if} \quad f_i < \bar{f} \end{cases}$$

其中 f_i 是个体 X_i 的适应度，f_{min} 和 f_{max} 分别是当前种群中最差和最优个体的适应度，$\bar{f}$ 是当前种群适应度平均值，cr_l 和 cr_u 分别是 cr 的下限和上限，一般 $cr_l = 0.1, cr_u = 0.6$。

变异策略

变异策略表示为 $DE/a/b$，其中 a 表明被变异个体的选择方式，b 表明差向量的个数。

1. **DE/rand/1:**
$$V_i = X_{p1} + F(X_{p2} - X_{p3})$$
2. **DE/best/1:**
$$V_i = X_{best} + F(X_{p1} - X_{p2})$$
3. **DE/current to best/1:**
$$V_i = X_i + F(X_{best} - X_i) + F(X_{p1} - X_{p2})$$
4. **DE/best/2:**
$$V_i = X_{best} + F(X_{p1} - X_{p2}) + F(X_{p3} - X_{p4})$$
5. **DE/rand/2:**
$$V_i = X_{p1} + F(X_{p2} - X_{p3}) + F(X_{p4} - X_{p5})$$