遗传程序设计 程序设计范型 自然选择 程序设计

计算机科学 1996 Vol. 23 No. 1

14-18

遗传程序设计(之二)
——基于自然选择的程序设计范型

TP311

吴少岩 罗铁庚 陈火旺
(国防科技大学计算机系 长沙 410073)

摘 要 This paper expounds fundamental views and methods of genetic programming paradigm. Genetic programming is opening up a new direction for coping with one of the central problems in computer science: "can computer learn to solve a problem without explicitly programming?"

关键词 Genetic Programming, Program induction, Genetic Algorithm, Fitness, Reproduction, Crossover, Evolution.

1. 引 论

"物竞天择,适者生存"是达尔文生物进化论阐明的主要原理之一。自然界中,生物体的结构体现了生物对其环境的适应能力(即生存与繁殖的能力);而结构又是由自然选择不断进化的。自然界总是延续适应性强的物种,淘汰不适应的物种。"适应性"驱使遗传操作:异性结合(Crossover)和变异(Mutation)创造出新的适应性更强的生物结构。

计算机程序是人类迄今为止创造出的最复杂的结构之一,同时程序设计也是人们面临的最棘手的任务。不用明确编程,如何让计算机学会求解问题?这是 Arthur Samuel 早在五十年代作为计算机科学的核心问题而提出的。回答这一问题的现有方法,包括机器学习、AI、自改进系统、自组织系统、神经网络、归纳等等都不是去寻求以计算机程序结构表示的答案,而是与程序有相当差距的表示形式,如神经网的加权向量、决策树、形式文法、框架、概念类簇、多项式系数、产生式规则、遗传算法中的染色体串等等。每一种结构仅适用于某类问题的求解,缺乏普遍的适用性。而程序结构本身却具有极大的灵活性:能够表示可在计算机上求解的几乎任何问题的解。程序结构的组成是人们所熟知的几种操作(层次、循环、递归、多类型、子程序等)。

所谓不需明显编程而让计算机解题,实际上是不需要人预先规定解的大小、形状与结构复杂性。有关这些解的属性应该是答案的一部分而非问题的一部分。由于要利用程序的灵活性,于是我们面临的问题变为在浩瀚的程序空间中以自适应的、智能的方式搜寻所需的程序。

John R. Koza 基于自然选择原理创造性地提出了遗传程序设计方法(Genetic Programming, GP),这是一种与领域无关的、机械地搜索程序空间的方法。通过增加染色体结构的复杂性,这种方法拓广了传统遗传算法[6]的应用范围。Koza 提出了两个论点:

- 各领域中许多看起来不同的问题都可看成为寻找一定的计算机程序的问题:给定程序的某个输入则产生所需的输出,换言之,许多不同的问题可形式化为程序归纳问题。
- 遗传程序设计提供了实现归纳的方法,亦即遗传程序设计可搜索程序空间中特别适合求解(或近似求解)所给问题的程序。由遗传程序设计产生的程序(结构)是"适应"的结果,正是适应性导致了所需的程序结构。

遗传程序设计首先随机地产生由数百或数千个任意大小和形状的程序组成的程序群体(Population),称为初始群体代;然后,利用达尔文自然选择原理对群体实施遗传操作:繁殖(Reproduction)和交配(Crossover)。参与遗传操作的程序个体

吴少岩 博士研究生,研究方向为软件进化、自适应开发。陈火旺 教授,博士导师。主要研究领域:软件自动化、AI 等。

是按其解决问题的能力即适应度(Fitness)来选择的，经过若干代的进化之后，群体中的程序便越来越适合解决指定的问题。

然而，读者一定质疑：仅仅凭借对一些问题随机产生的、不正确的程序作性能评估(适应度计算)和几个与问题无关的简单机械操作(遗传操作)就能"培育"出能解决复杂问题的程序吗？

读者的疑虑是自然的，因为计算机科学是基于逻辑的，以往在机器学习、自组织系统、自改进系统、AI等领域的研究无不遵循下述七条原理：正确性、协调性、合理性、确定性、有序性、简洁性和可判定性。因而也就容易忽视还有完全不同的原理指导问题求解的可能性。基于自然选择原理的遗传程序设计相悖于所有这七条原理，反映了遗传程序设计的重要特征。

1. 正确性：GP处理的对象是不确切(或错误的)解(程序)，很少产生精确的解。

2. 协调性：不协调是GP的本质特征。GP并发地使用相悖的程序求解问题，而且程序的差异性越大越有助于求解问题。

3. 合理性：没有基于前提与推导规则的逻辑推理序列来证明GP的结果。

4. 确定性：GP的所有关键步骤都是基于概率的，只有随机性，没有确定性。

5. 有序性：不整齐性、无序性是自然界生物过程的核心特征，也是GP的核心特征。

6. 简洁性：简洁性(或优美性)是科学的指导原则；适应性而非简洁性才是GP的指导原则。

7. 可判定性：GP和自然界中的生物过程一样没有明确定义的终止点。

Koza并未给出任何数学理论，证明遗传程序设计总能成功地求解每个问题，然而，他用大量的实验支持了这一惊人的结论：遗传程序设计可用来求解许多领域内的各种不同问题，它为符号表示增添了一个强有力的学习过程。Koza提供的例子跨越多个领域，包括优化控制、符号回归、规划、解微分方程、寻求博奕策略、进化自发行为、发现经验、分类、模式识别、进化归类结构以及归纳等，所有这些问题都可用同一方式求解：遗传程序设计在所有程序的空间中搜寻一个特别适合求解指定问题的计算机程序。

2. 程序归纳的普遍性

程序归纳是指在程序空间中归纳地发现一个程序，它给定某个输入便产生所需的相应输出。依据Koza的观点，许多不同领域中的问题都可以形式化为程序归纳问题，领域不同，描述程序归纳的术语也不尽相同。但术语之间的差别并未掩盖各种问题作为程序归纳问题的共同特点。表2.1列出了在程序归纳中使用的各种术语。

表2.1

问题领域	程序	输入	输出
优化控制	控制策略	状态变量值	控制变量值
规划	规划	传感器或检测器值	效应器动作
序列归纳	数学表达式	索引位置	序列元素
符号回归	数学表达式	自变量值	因变量值
自动程序设计	公式	值	结果
寻求博奕策略	策略	已知信息	移动
发现和预测经验	模式	自变量	因变量
分类	决策树	属性值	对象类
自发行为的进化	规则集	传感输入	动作
细胞自动机的自动程序设计	细胞状态变迁规则	细胞及近临的状态	细胞的下一状态

3. 遗传程序设计概述

"适应"(或"学习")意指对结构的某些改变，使结构在环境中表现更好的性能。Holland在"Adptation in Natural and Artificial Systems(1975)"一书中指出了所有自适应系统应具备的几个关键特征：适应变化的结构、初始结构、评估结构的适应度(Fitness)量度、改变结构的操作、每一阶段的系统状态(记忆)、终止过程的方法、指定结果的方式、控制过程的参数。在使用Holland的术语阐述遗传程序设计之前，我们先讨论一下表示模式问题。

3.1 表示模式

表示模式是遗传算法[1,2,6]的关键问题，不仅因为遗传算法直接操作问题的编码表示，也因为表示模式限定了系统观察世界的窗口。传统的遗传算法使用定长字符串作为表示模式，Holland等人利用定长字符串的数学技巧构造了一个理论体系来解释遗传算法工作机制的合理性。

然而，定长字符串不具有任何层次性，也不能方便地表示包含了循环、递归的任意计算过程；同时这种表示没有动态可变性，事先选定的长度限定了系

统的内部状态数，限定了系统的学习能力，事先决定解的大小、形状及构成成份是早期机器学习系统不成功的重要根源。

对很多问题而言，最自然的表示解的模式是层次计算机程序，表示解的程序，其大小和形状事先一般是未知的，而程序具有动态改变大小、形状的能力。

遗传程序设计的操作对象（组成群体的个体）是普通的计算机程序。然而出于下述理由，Koza 选择了 Common LISP 作为表示个体的语言：

(1)LISP 的程序和数据具有相同的格式：S 表达式，这样可将遗传群体中的程序作为数据来操作，然后操作的结果又可作为程序来执行。

(2)S 表达式作为程序等价于该程序的分析树，LISP 提供了引用分析树的方便手段，而遗传操作正是对这种树的子树进行的。

(3)LISP 支持结构动态改变大小、形状。

(4)LISP 能方便地处理层次结构。

(5)LISP 的 EVAL 函数是执行程序的有效机制。

(6)LISP 软件环境包含丰富、成熟的工具。

3.2 适应变化的结构

对遗传算法和遗传程序设计而言，适应变化的结构是搜索空间中个体点构成的一个群体而非单个点。遗传方法不同于其它技术在于它同时并行地搜索数百或上千个点。

遗传程序设计中，适应变化的结构是层次结构的程序，其大小、形状和内容是可以动态改变的。令

$F=\{f1,f2,\cdots,fnf\}$

$T=\{a1,a2,\cdots,ant\}$

F 是由 nf 个函数组成的函数集，T 是由 nt 个终结符构成的终结符集。遗传程序设计中所有个体结构都是由 F 和 T 中的元素递归复合而成的，函数集 F 可以包含：算术操作、数学函数、布尔操作、条件操作、循环和递归、面向问题的函数。

终结符集 T 包含变量原子或常数原子。与普通程序设计相对应，F 中的函数相当于组成程序的过程和函数；而 T 中的原子相当于程序员设定的变量和常数。如：F＝{AND，OR，NOT}，T＝{D0，D1}，S 表达式：(OR(AND(NOT D0)(NOT D1))(AND D0 D1)) 对应于分支有序的、结点带标号的树，如图 3.1。

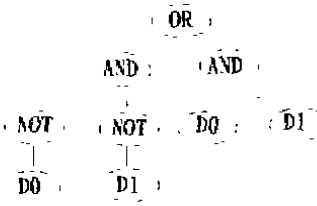

图3·1 由F和T复合而成的一个S表达式

函数集 F 与终结符 T 应满足封闭性与充分性。所谓封闭性是指函数集中的每个函数都是良定义的，对它遇到的任何参数组合都是有定义的（闭的）；而充分性是指 F 和 T 能够表示问题的解。封闭性容易满足，因为可以补充、修改不封闭函数的定义，使其封闭；但充分性较难确定。实际上，确定对求解某个问题具有充分解释能力的变量集（即 F 和 T）是科学中的共同问题。F 和 T 包含的元素通常蕴含在具体问题中，目前并没有一般的原则指导 F 和 T 的选择，这是程序员的工作。

3.3 初始结构

遗传程序设计的初始结构由初始群体中的个体 S 表达式组成。初始群体中的每个 S 表达式都是随机生成的一棵结点带标号的、分支有序的树。

初始随机树的生成方式有多种，但应使初始群体中的树有不同的大小和形状。两种基本方法是："满(Full)"方法和"长(Grow)"方法，"满"方法生成的树，其根与叶结点（标记了终结符）的无回溯路径长等于指定的最大长度；"长"方法生成的树，其根与叶结点的无回溯路径长不大于指定的最大长度。显然"长"方法生成的树，大小与形状有更多的差异。Koza 采用了"长"、"满"相结合的生成方法：设最大的指定深度为 MD，群体大小为 M，则对于 2，…，MD 中的每一深度，用"长"方法和"满"方法各生成 [M/(2*(MD—1))]棵树。

3.4 适应度(Fitness)

适应度是达尔文自然选择的动力，同样也是传统遗传算法和遗传程序设计的动力。自然界中，个体的适应度是个体生存到繁殖期并能繁殖的概率；人为世界中，我们以某种方式度量适应度，并以此控制修改人为群体中个体结构的操作。遗传程序设计一般使用四种形式的适应度：原始适应度、标准化的适应度、调整的适应度和规范化的适应度。

(1)原始适应度(Raw Fitness)。意指适应度的度量是以问题本身的自然术语来陈述的。例如"货郎担"问题中的适应度是推销员通过一条路径的代价，

代价越低，路径越好。适应度通常是在一个适应度实例集(Fitness Cases)上计算的，这个实例集只是整个域空间的有限采样集，但它们对整个域空间应有足够的代表性，因为由它们获得的结果应能推广到整个域空间。

在很多情况下，原始适应度是以误差的形式出现的。如果S表达式的结果值是整型或浮点型的，则个体i在代t时的原始适应度r(i,t)是：

$$r(i,t)=\sum_{j=1}^{N_c}|s(i,j)-c(j)|$$

其中s(i,j)是S表达式i对实例j返回的值，c(j)是实例j的准确值。

如果S表达式的结果值是布尔型或符号型的，则上述距离和为失配数目；当为复数型、向量型或多值型时，r(i,t)是每个分量间的距离和。

(2)标准化的适应度(s(i,t))。用原始适应度定义，总是以较小的值作为较好的值，亦即

$$s(i,t)=\begin{cases} r(i,t) & \text{若较小的值为较好的值} \\ rmax-r(i,t) & \text{否则} \end{cases}$$

其中rmax是原始适应度可取的最大值。

(3)调整的适应度(a(i,t))。由标准化的适应度s(i,t)计算得出：

$$a(i,t)=1/(1+s(i,t))$$

a(i,t)的取值介于0～1之间。当s(i,t)接近0时，调整后的适应度能夸大s(i,t)值之间微小差别的重要性。

(4)规范化的适应度。如果选择参与遗传操作的个体时，采用正比于适应度比例的方式，则需引入规范化的适应度n(i,t)的概念。n(i,t)由a(i,t)计算得出：

$$n(i,t)=a(i,t)/\sum_{k=1}^{M}a(k,t)$$

这里M是群体的大小，n(i,t)的取值介于0～1之间，较大的值为较好的值，并且

$$\sum_{i=1}^{M}n(i,t)=1$$

3.5 变更结构的主要操作

遗传程序设计有两个变更结构的主要操作：达尔文繁殖和交配(异性组合)。

(1)繁殖。是达尔文自然选择和适者生存的基本动因。繁殖操作是单性的，仅有一个S表达式作为父亲参与操作，操作完成时产生一个子孙。繁殖由两步构成：首先根据某种基于适应度的选择方式从群体中选取一个S表达式；其次将所选个体不加改变地复制到新的群体(新一代群体)中。

有多种基于适应度的选择方法，然而最普遍使用的是正比于适应度的选择(这时繁殖仍称为按适应度比例的繁殖)。其它选择方式包括级别选择和竞赛选择。级别选择是基于适应度值的级别而非数值。这种选择取消了高适应度个体在群体中的支配作用，夸大了适应度相近的个体间的差别；竞赛选择中，随机地从群体中选出一组个体(通常为两个)，具有较好适应度的个体被选中。

(2)交配。组合双亲的某些结构成份创造出新的子孙。交配始于两个S表达式(作为双亲)，操作之后产生两个子孙，这是一个异性操作。同繁殖一样，作为双亲的两个个体也是按某种基于适应度的选择方式独立地被选择的。图3.2—3.4用一个例子说明了交配操作的过程：

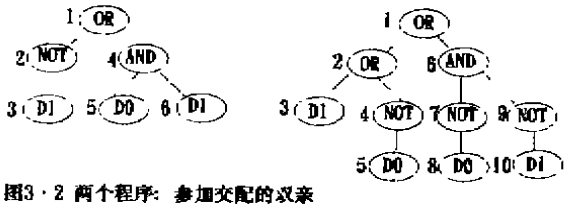

图3·2 两个程序：参加交配的双亲

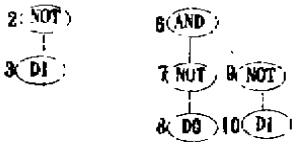

图3·3 两个交配段：第一个双亲选择2为交配点；第二个双亲选择6为交配点

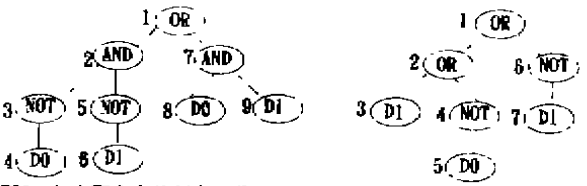

图3·4 交配产生的两个子孙

参与交配的个体，其交配点是随机地、独立选择的。在本例中，两个个体分别选择了2、6为交配点，以交配点为根的两棵子树，即为交配段(图3.3)。交配发生时，双亲的交配段互换，从而产生两棵新树，即交配产生的两个子孙(图3.4)。

参与交配的两个个体相同时，由于交配点不同，仍能产生新的子孙，因此遗传程序设计中的群体不可能收敛于局部优化的"陷井"(未成熟收敛)中。这一点同基于定长字符串的遗传算法形成鲜明对照。在遗传算法中，由于繁殖操作的收敛压力，若干代之

后，群体包括了少数个体。当表示为定长字符串的个体同自身交配时，产生的子孙仍然与双亲相同。这时交配操作蜕化为繁殖操作，从而整个群体容易产生不成熟收敛。

除了繁殖与交配两个主要操作外，在遗传程序设计中，还有几个很少使用的次要操作，即变异(Mutation)、置换(Permutation)、编辑(Edting)、封装(Encapsulation)和"十中抽一(Decimaion)"，限于篇幅，在此不予赘述。

3.6 自适应系统的状态

遗传程序设计中，在任何阶段自适应系统的状态仅由个体的当前群体组成，不需其它的存储或记忆。但在具体实现上，可能要缓冲某些方便运行的控制参数。

3.7 终止准则与结果指定

自然界的进化过程是无终结的，遗传程序设计也是如此。然而作为一次实际的运行，必须给出终止条件。这个终止条件是预先指定的：群体进化的最大代数 G 或者是某个面向问题的成功谓词。

结果指定的方式有两种，其一是当运行终止时指定当前代中的最佳个体为运行结果；其二是在进化过程中保存最佳个体，当运行终结时，保存的个体即为运行结果。

3.8 控制参数

遗传程序设计的控制参数主要有两个：群体大小 M 和最大运行代数 G。下面列出了遗传程序设计的 19 个控制参数及其缺省值。

两个主要数值参数：群体大小 M＝500；最大的运行代数 G＝51

11 个次要数值参数：交配概率 Pc＝90%；繁殖概率 Pr＝10%；选择非叶结点交配的概率 Pip＝90%；运行中 S 表达式的大小上限 Dc＝17；初始随机 S 表达式的大小上限 Di＝6；变异概率 Pm＝0.0%；编辑频率 Fed＝0；封装概率 Pen＝0.0%；"十中抽一"条件＝NIL；"十中抽一"目标百分比 Pd＝0.0%

六个定性变量：初始随机群体的生成方法是"长"、"满"各半；基本选择方法是正比于适应度的选择；配偶选择方法是正比于适应度的选择；使用调整的适应度；500 以下的群体不使用"偏心选择(Over Selection)"[1]，群体大于 1000 时使用；不使用"精英(Elitist)策略"[2]

4. 结束语

遗传程序设计范型是一种与领域无关的方法(弱方法)，它提供了为求解一个问题寻找一个计算机程序的统一途径。对程序员而言，只需完成下述五步，即确定：(1)终结符集 T；(2)函数集 F；(3)适应度计算；(4)控制运行的数值参数值和定性变量；(5)指定结果和终止运行的准则。

总之，遗传程序设计范型通过执行下述三步，进化地产生求解问题的计算机程序：

(1)由函数集和终结符集的随机组合产生初始的程序群体；

(2)循环完成下述各步，直至满足终止准则：

(a)执行群体中的每个程序，根据程序求解问题的能力赋予它一个适应度值；

(b)应用下列两个主要操作创建新的程序群体；操作对象的选择(程序)是基于适应度的概率选择。

(i)繁殖：复制已有的程序到新的群体；

(ii)交配：组合两个程序中随机选取的部分，创造两个新程序；

(3)指定进化过程中出现的最佳程序为遗传程序设计的结果。此结果为问题的解或近似解。

限于篇幅，本文我们未给出遗传程序设计的应用实例，仅仅概述了这种范型的基本观点和方法。我们将在后续文章中讨论遗传程序设计的应用问题。

参考文献

[1]Koza, J. R. Genetic Programming, MIT Press, Cambridge, MA, 1992

[2]Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1992

[3]L. J. Arthur, Rapid Evolutinary Development. Wiley Series in Software Engineering Practice, 1992

[4]John H. Holland, Adaptation in Natural and Artificial System, University of Michigan Press, 1975

[5]C. Darwin, On the Origin of Species by Means of Natural Selection, John Murray, 1859

[6]陈火旺、吴少岩、罗铁庚，遗传程序设计(之一)，计算机科学，Vol. 22, No. 6, 1995

1)偏心选择：让高适应度的个体有高出其适应度的被选机会；2)精英策略：总是将当前最好的个体繁殖到下一代。