文章编号:1672 - 058X(2011)05 - 0496 - 05

遗传规划算法的改进研究

姜 群,陈 宁,张 曼
(重庆理工大学 计算机科学与工程学院,重庆 400054)

摘 要:通过改变遗传规划算法在生成初始群体时的方法,调整变异概率,修正适应度函数,对遗传规划算法进行改进,使生成的初始群体具有良好的性能;并通过两个函数进行符号验证,说明改进后的方法是有效可行的。

关键词:遗传规划;改进算法;符号回归;适应度函数
中图分类号:TP18 **文献标志码**: A

20 世纪 70 年代,Holland[1] 教授受生物学的启发,提出了著名的遗传算法[2];经过 30 多年的应用与发展,遗传算法已经成为非线性优化计算的有效工具,具有重要的现实意义和工程意义,得到了广泛的研究和运用。遗传规划是从遗传算法中派生和发展起来的一种搜索寻优技术。上世纪 90 年代初,美国学者 Koza[3] 在遗传算法的基础上进一步提出遗传规划算法(Genetic Programming)。遗传规划(GP)是一种关于产生问题解的计算机程序或者其他复杂结构的自动方法。遗传规划试图研究计算科学的一个中心问题:计算机怎样在没有明显编程的情况下来解决问题。遗传规划为上述问题的解决提供了一个可能的工具;它在解决人工智能、机器学习、控制技术等领域中的问题时效果显著。同时,发现遗传规划算法本身也存在许多问题,首先,初始种群是随机产生的,遍布于整个解空间,但整体素质一般很差,有许多甚至是不可行的, 要经过几代或更多代后整体素质才有提高。其次,遗传规划算法的效率受多种因素影响,是否能够收敛到全局最优解(或满意解)不仅与初始群体的质量有关,还与参数选取、遗传操作及适应值的测试方式等有很大关系。因此,有必要对遗传规划进行改进,从而提高其收敛性能,加快寻优过程、缩短寻优时间。

1 遗传规划的基本内容

(1) GP 的基本思想。基本过程是在由许多可行的计算机程序组成的搜索空间中,寻找出有最佳适应度的计算机程序。仿效动物界进化和遗传的过程,遵从"优胜劣汰,适者生存"的自然法则[4],应用复制、交换及突变等若干个进化方式。子代计算机程序通过自然选择和遗传机制而产生。

(2) GP 求解问题的操作步骤。① 确定个体的表达方式,表现为确定函数符集 F 和终止符集 T;② 随机产生初始群体;③ 计算群体中各个体的适应度;即根据个体解决问题的好坏赋予一适应度;④ 执行遗传操作,包括(选择)复制、交叉、突变等;⑤ 循环执行③④直到满足终止条件;执行过程如图 1。

(3) GP 的主要特征。产生的结果具有层次化的特点;随着进化的延续,个体不断朝着问题答案的方向动态地发展;不需事先确定或限制最终答案的结构或大小,遗传规划将根据环境自动确定;输入、中间结果和输出是问题的自然描述,无需或少需对输入数据的预处理和对输出结果的后处理;在遗传规划中,个体结构变化是主动的,它们并不是对问题答案的被动式编码,个体结构在遗传时能从当前状态主动地改变结构和大小进化成新的,更优的状态。

收稿日期:2011 - 03 - 05;**修回日期**:2011 - 04 - 25.
作者简介:姜群(1959 -),女,重庆市人,副教授,从事智能计算、数据挖掘及生物信息研究.

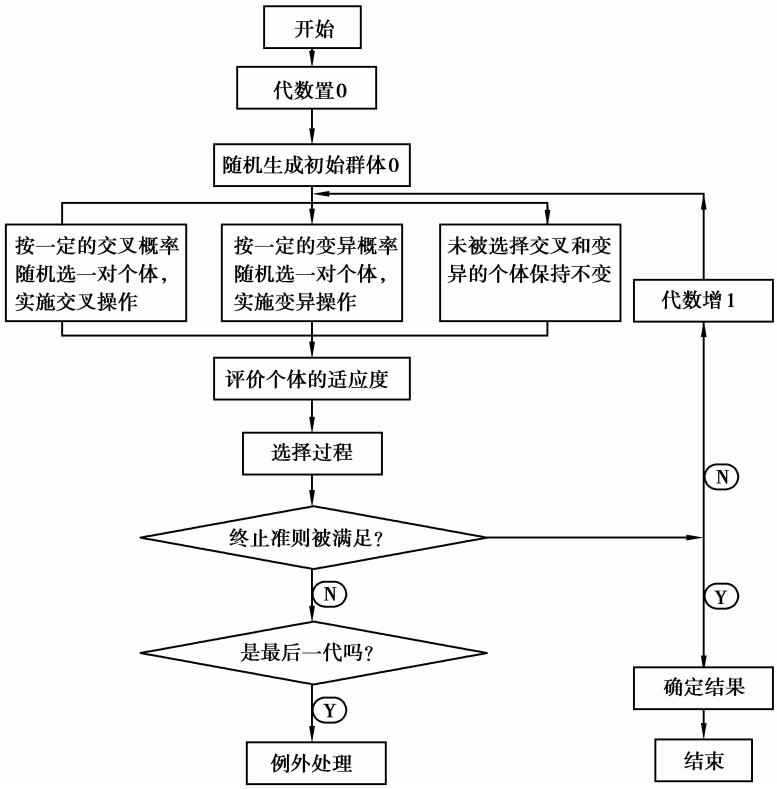

图 1　遗传规划算法的执行过程

(4) GP 的个体描述方法。在遗传规划中,首先要解决的问题是如何用一系列可行的函数对个体进行描述,而这种函数能反复地由 N 个函数集合 F:$F=\{f_1,f_2,\cdots,f_N\}$ 和 N 个终止符集合 T: $T=\{a_1,a_2,\cdots,a_N\}$ 组合而成。函数集 F 中每个特定的函数 f_i 假定有 $z(f_i)$ $(i=1,2,\cdots, Nm)$ 个自变量,则对函数 $f_1,f_2,\cdots,f_{Nm}$ 来说,相应的自变量个数分别为 $Z(f_1),Z(f_2),\cdots,Z(f_{Nm})$。

函数集 T 内的函数可以是:算术运算符,如 +, -, ×,／等;标准数学函数,如 SIN, COS , EXP, LOG 等;布尔运算符,如 AND, OR, NOT 等;条件表达式,如 IF-THEN-ELSE, SWITCH-CASE 等;可迭代函数,如 DO-UNTIL, WHILE-DO 等;可递归函数;也可以是任何其他可定义的函数。终止符集 T 内可以是变量或常量。有时终止符隐含着函数关系,为简化起见将它视为无自变量的函数。

(5) GP 的终止准则。遗传规划并行计算的本质在于它永远不停地进化。但在实际应用中,一旦某个终止准则满足,进化过程就立即停止。终止准则一般有两个:当最大容许进化代数 G 满足时,进化过程立即停止;当预先设定的问题求解成功,条件被满足时(例如,群体中的某个体标准适应度达到),进化过程也立即停止。对于一些无法判明其答案的问题(如最优问题),通常采用一些近似的成功判断条件来终止进化。对于一些难以建立成功判断条件的问题,常常在进化 G 代后,通过分析其结果来决定是否终止进化。

2　遗传规划中存在的问题

2.1　遗传规划中个体大小的“爆炸”[5]现象

在遗传规划中存在着大家所熟知的“规模爆炸”的现象,亦即个体树的大小,包括它的深度和广度,在进化过程中不断地增大,而与此同时问题的解却没有任何改进。这一现象为许多研究者所发现,对于大多数的应用问题来讲,树不能太大,否则很难将它翻译为一个有意义的结果。同时正如许多学者所指出的那样,“爆炸”现象常常会伴随着优化进程的变慢。

2.2　子程序的重用性和共享性

在很多计算机语言中,都是由一些小的程序块来实现一定的功能,而它们都是由若干语句组成的,可以重

复地被程序的其他部分和其他程序所调用,不妨将它们通称为“子程序”。对于一名编程人员来说,经常采用的是自顶向下的分析方法,将一个大的模块分解为多个相对独立的子模块,然后针对每个子模块编写相应的子程序,最后调用这些子程序单元来完成整个大模块的功能。由于子模块还可以分解为相对子模块,即子模块调用也是层次的,一个子程序能调用别的子程序,同时又能被其他子程序调用。能重复使用和为其他程序共享的子程序的特性,在程序设计中起至关重要作用,这些都应当反映在作为自动程序设计的遗传规划中。

Koza 首次用自动定义函数(automatically defined functions, ADFS)方法很好地解决了子程序的重用问题[6]。但是一个个体的函数定义分支中所定义的自动定义函数在进行层次调用时,只能调用本个体函数定义分支中所定义的自动定义函数。类似地,结果产生分支也只能调用本个体的函数定义分支中定义的自动定义函数。换句话说,一个个体不能调用其他个体的任何自动定义函数,同时该个体的任何自动定义函数也不能被其他个体所调用,这在一定程度上限制了子程序的共享性。而当群体进化到接近最优解时,某些能够很好地解决子问题的自动定义函数已经通过进化而产生,若能为整个群体中的个体所调用,将必然加速收敛过程。同时,由于遗传规划所特有的树状表示方法,使得两个自动定义函数在交叉之后基本上不可能保持不变,因此,已经进化产生的优秀的自动定义函数不可避免地遭到破坏,从而可能导致收敛的振荡现象发生。由此可见,适当地将优秀的自动定义函数通过某种方式保护起来,保证其不被遗传算子破坏,是一个值得考虑的问题。所以,有必要对遗传规划算法进行相关方面的改进。

3 遗传规划的改进方法

根据遗传规划算法的执行过程,随机生成初始群体,通过复制、交叉和变异等遗传操作产生新一代个体,利用适应度函数评价新生个体。此处对遗传规划算法的改进也主要体现在这几个方面:有目的的生产初始群体;调整变异概率;动态的设置参数;修正适应度函数。这样的改进可以提高初始群体的平均适应度,降低群体的个体复杂度,提高其收敛性能,加快寻优过程、缩短寻优时间。

3.1 有目的生成初始群体

(1) 平均值法。这种方法是首先生成 M 个个体。求其平均适应度,当个体的适应度大于平均值者将被保留下来,重复此过程;自到保留下来的个体数达到 M 为止。该方法生成初始群体时的运算量不大一般情况,重复 2 ~3 次就可以满足要求。

这种方式生成的初始群体,质量较高,个体之间的相似度较大,但由于是随机生成,个体又不会完全相同;它不会出现象遗传算法那样,当大多数个体相同或相似时,容易出现过早的局部收敛现象。这是因为在遗传规划中,即使进行交换的两个个体完全相同,交换后生成的新个体也不会相同,除非两个交换点完全相同,但这种情况的概率很小。

(2) 分组生成法。采用随机方法生成的初始群体,其平均适应度一般是比较差的,甚至有许多个体是不可行的,需要经过很多代进化以后才能使群体的整体适应度得到提高,而且遗传规划中初始群体的规模一般比较大,如果所研究的问题本身又比较复杂时,就会使算法收敛得很缓慢。因此,为了克服随机生成初始群体的这些缺点,对初始群体的生成方法作了改进,采用分组的方法生成初始群体。这种方法产生的初始群体平均适应度高,可减少进化代数,提高收敛效率。具体产生初始群体的方法为根据初始群体的规模进行适当分组,确定每组个体的数目 $M(M>2)$,按组生成初始群体,每组初始群体的生成方法如下:

第 1 步:随机产生两个个体,计算其平均适应度作为此时本组平均适应度,转第 3 步;第 2 步:随机产生一个新个体,转第 3 步;第 3 步:比较新产生个体的适应度与当前的组平均适应度的大小,当新个体的适应度大于等于组平均适应度时,个体被保留下来,并计算此时本组个体的平均适应度,作为新的组平均适应度,转第 4 步,当新个体的适应度小于组平均适应度时,舍弃该个体,转第 2 步;第 4 步:如果本组个体数目达到 M,则停止,否则转第 2 步 。

这种产生初始群体的方法对于每组个体而言,其产生过程是一个组平均适应度逐步提高的过程,初始群体的整体适应度比较高。对于这种初始群体的生成方法,选取组的大小(即每组中个体的数目)对收敛效率有直接影响,如果每组个体太少,则初始群体的产生接近于随机生成,不能明显提高收敛效率。如果组过

大,则产生初始群体本身运算量过大,而且也会使个体在最初生成时就具有较大的复杂度(即个体树的节点数目,在图 1 中,个体的复杂度为 8),从而加大以后进化过程的运算量,这样反而会对遗传规划的收敛产生不利影响。此处的数值实验中,组大小为 50。

3.2　加大遗传算子中的变异力度

在遗传规划中有 3 种基本的算子:复制,交叉和变异。变异操作属于辅助操作,变异的概率一般取得比较小,通常取值在 0 ~ 0.01。变异操作可以使个体的结构发生变化,从而使个体的适应度发生变化。因为,对于一些适应度比较差的个体,通过变异之后其适应度变好比其适应度变差可能性更大。在进化初期,个体的适应度相对比较差,因此,在进化初期采用较大的变异概率对群体中适应度较差的个体进行变异可以加快进化速度。变异概率随着进化代数的增加而递减,当减小到某一给定的最小值后则变异概率不再变化。在进化后期,当群体的整体适应度都比较好的时候采用随机变异。数值实验发现,这种变异方法在进化初期确实加快了群体的进化速度。

3.3　修正适应度函数

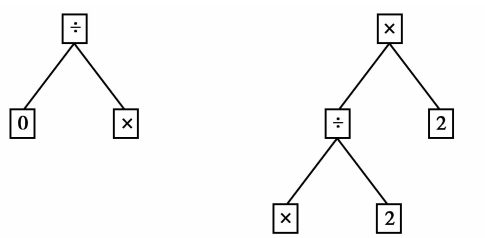

图 2　冗余项示意图

算法效率的高低与个体复杂度的大小有重要关系。复杂度过大的个体,其运算量也比较大,占领大量的内存资源,从而造成算法整体效率的下降。为了避免产生复杂度过大的个体,常用的方法有两种,一种是限制个体树的深度,二是在个体树中增加惩罚项。对于第一种方法,限定个体树的最大深度虽然可以避免巨型个体树的产生,但同时也对搜索空间形成了限制,在某些情况下往往不能得到问题的解。第二种方法是将个体的复杂度引入到适应度函数中,在适应度函数中增加一个惩罚项 δ,δ 随着个体复杂度的增加而增加。复杂度过大的个体,由于惩罚项的原因,其个体适应度变差,在进化工程中就会被淘汰。复杂度较大的个体一般含有较多的冗余项。图 2 就是两个典型的具有冗余个体树的例子。图中左边的个体无论右子树是什么,其结果都是 0,而右边个体的结果是 $(x \div 2) \times 2 = x$。类似这样的冗余项又称为基因内区[7],在遗传规划中,基因内区在进化初期有一定的积极作用,可以在一定程度上保护个体的有效部分不被交换和变异操作破坏,但是在进化后期,由于基因内区大量充斥在群体中,个体复杂度的增加速度远大于适应度改善的速度,增加了大量的冗余计算,进化缓慢,所以惩罚项 δ 应该随着进化代数的增加而增加。因此,δ 的形式应为 $\delta = \delta(d, f_0, t)$,其中 d 表示个体的复杂度,f_0 表示个体的原始适应度,t 表示进化代数,即对个体 I,其适应度为 $f(I) = f_0 + \delta(d, f_0, t)$。即由原来的个体的适应度 $f(I) = \sum_{i=1}^{s} (z_i - y_i)^2$,调整为现在的 $f(I) = (1 + CtCd) \sum_{i=1}^{s} (z_i - y_i)^2$。

4　数值试验

为了验证所提出方法的有效性,选用两个具有相当复杂度的例子进行测试,这两个函数自变量的取值范围分别是 $[-1,1]$ 和 $[0,2\pi]$,在相同的平台和精度要求下,与改进前的算法做了一个比较。在数值试验中,终止符集 $T = \{x, c\}$,c 为随机常数项。函数集为 $F = \{+, -, \times, \div, \sin, \ln, \exp\}$,个体的适应度为:$f(I) = \sum_{i=1}^{s} (z_i - y_i)^2$,通过在此对适应度函数进行调整,得到的新的适应度函数为:$f(I) = (1 + CtCd) \sum_{i=1}^{s} (z_i - y_i)^2$。其中 S 为样本点的个数,Z_i 表示个体在样本点处的函数值,y_i 表示样本点处的原函数值,t 表示进化代数,d 表示个体的复杂度。Ct,Cd 中取 $Ct = Cd = 100$,其他各个参数为 $Pc = 0.1$,$Pe = 0.9$,$P_0 = 0.2$,$g(P_0, t) = P_0\theta$,$\theta = 2/3$,最小变异概率取 $P = 0.02$,群体规模 $N = 500$,分为 10 组,组大小为 50,精度 $\varepsilon = 0.01$,最大迭代次数 $T = 51$,平均收敛时间单位是 s,每个例子运行 20 次,求其结果。

下面给出数值试验结果,其中 TPG、AG、SG 和 IMPG 分别表示改进前的算法、平均值法、分组生产法和从 3 个阶段都进行改进的方法。其中表 1 和表 2 分别表示回归函数 $2\sin(3x) + 3\sin(2x) + 4\sin(x)$ 和 $x^4 +$

$2x^3+3x^2+4x$。

表1 数值结果

	收敛次数	平均迭代代数	平均个体复杂度	平均CPU时间
TPG	13	33.07	43.02	1.68
AG	14	30.69	46.49	1.51
SG	11	31.25	38.71	2.18
IMPG	18	17.72	36.01	1.83

表2 数值结果

	收敛次数	平均迭代代数	平均个体复杂度	平均CPU时间
TPG	16	26.18	36.24	1.36
AG	17	23.53	28.53	1.29
SG	16	26.37	29.42	1.86
IMPG	21	13.07	23.55	1.49

从表1、表2中可以看出,在其他条件都相同的情况下,对于同一个例子相对于TPG、AG和SG,IMPG收敛次数最多,平均迭代代数最少,平均个体复杂度也最小,平均CPU时间虽然不是最少,但是差别不是很大。

5 结 论

由于遗传规划算法存在着"爆炸现象"和子程序的重用性和共享性,从生成初始群体时的方法,调整变异概率,修正适应度函数等方面对遗传规划算法进行改进。数值试验表明,在近乎相当的寻优时间里,全面改进后的遗传规划算法在收敛次数、平均迭代代数以及个体复杂度方面都具有较明显的优势,使得遗传规划算法的收敛性得到明显的提高,说明了改进方面的有效性和合理性。

参考文献:

[1] DAVID E. Genetic Algorithms in Scarch. Optimization and Machine Learning[M]. Addision-Wesley Publishing Company,1989
[2] 云庆夏,黄光球,土战权. 遗传算法与遗传规划[M]. 北京:冶金工业出版社,1997
[3] KOZA J K. Genetic Programming[M]. MIT Press,Cambridge,MA,1992
[4] 王战权,云庆夏,杨东援. 改进的遗传规划研究[J]. 系统工程理论与实践,2000(5):66-69
[5] 王小平,曹立明. 遗传算法理论、应用与软件实现[M]. 西安:西安交通大学出版社,2002
[6] KOZA J. Genetic Programming Ⅱ: Automatic Discovery of Reusable Programs[M]. MIT Press,1994
[7] 林丹,寇纪淞,李敏强. 遗传规划研究与应用中的若干问题[J]. 管理科学学报,1999,2(4):62-69
[8] 候定丕,王惠文,牛爱民. 遗传规划算法的改进及应用研究[J]. 北京:中国科技大学出版社,2006
[9] 李佳泽,李应歧,徐姗. 基于混合算法的无人机路径规划[J]. 四川兵工学报,2011,32(6):105-107

Study on Improvement of Genetic Programming Algorithm

JIANG Qun, CHEN Ning, ZHANG Man

(School of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China)

Abstract: A new method changing genetic programming algorithm on generating initial population is used to adjust variation probability, to modify fitness function and to revise genetic programming algorithm, in order to make generated initial population possess good performance. Two functions are used to conduct symbol verification, which indicate that the improved method is effective and feasible.

Key words: genetic programming; improved algorithm; symbolic regression; fitness function

责任编辑:代小红