文章编号:1673-0062(2017)01-0072-05

基于 GEP-RNC 的指数对数型程序不变量发现方法

李玉燕,阳小华*,吴取劲

(南华大学 计算机科学与技术学院,湖南 衡阳 421001)

摘　要:程序不变量的发现是一种提高软件质量的有效方法.不变量发现工具 Daikon 可以发现程序中蕴含的简单不变量形式,但不包括复杂的函数型不变量.本文基于 GEP-RNC 算法对指数对数型不变量发现方法进行研究,通过实验证明 GEP-RNC 算法可以有效的发现指数对数形式的不变量,解决了基因表达式编程算法在复杂函数形式发现中稳定性不佳,精度不高的问题,扩展了 Daikon 不变量测试库中程序不变量的类型.

关键词:程序不变量;GEP-RNC;对数;指数

中图分类号:TP311.5　　**文献标志码**:B

Method of Finding Program Invariants of Exponential and Logarithm Based on GEP-RNC

LI Yu-yan, YANG Xiao-hua*, WU Qu-jing

(School of Computer Science and Technology, University of South China, Hengyang, Hunan 421001, China)

Abstract: The discovery of program invariants is an effective method to improve the quality of software. We can find the simple invariants in the program through the test library by using the invariant discovery tool Daikon, but not including the complex function invariants. In this paper, we study the method of finding the invariants of exponential and logarithm type based on GEP-RNC algorithm. It is proved that GEP-RNC algorithm can effectively find the invariants of exponential and logarithmic form, which solves the problems of poor stability and low precision of gene expression programming algorithm in the form of complex function, and extends the type of program invariants in the invariant test library in Daikon.

key words: program invariants; GEP-RNC; logarithm; exponential

收稿日期:2017-01-23

基金项目:湖南省哲学社会科学基金(14YBA335)

作者简介:李玉燕(1992-),女,硕士研究生,主要从事软件测试方向的研究.E-mail:838690337@qq.com. * 通讯作者:阳小华,E-mail:875153468@qq.com

0 引　言

程序不变量是描述程序在运行时保持不变性质的逻辑断言.根据关系数据库理论,程序不变量分为函数依赖程序不变量和非函数依赖程序不变量.函数依赖程序不变量是指那些存在某种函数关系的变量之间的规则,可以包含变量间的线性关系,以及其他函数关系等.所有不是函数依赖的关系统称为非函数依赖关系,例如大小关系、次序关系、不等关系、时序关系和范围关系等.程序不变量可以反映程序的运行行为,具有广泛的用途[1].程序不变量不仅可以用于辅助程序编码过程,还运用于软件测试之中,用于软件缺陷检测、软件错误定位以及过滤无效测试用例等[2-3].

程序不变量广阔的应用前景和应用需求,吸引了许多专家对其进行研究.不变量的获取有静态分析和动态发现两种方法[4].静态分析可以通过检查程序代码和文档来发现程序中隐藏的不变性质,但通常只能应用于中小型的程序.而动态发现可以通过程序实际运行中获得的轨迹数据,对其进行分析来发现程序中的不变量.

动态发现方面主要研发了一系列的不变量发现工具,有美国的 Ernst 等人研究出的 Daikon 工具,乔治亚理工学院的 Christoph Csallner 等人开发的 DySy 工具,斯坦福大学开发的动态不变量发现与检测引擎 DIDUCE.其中最具代表性的是 Daikon 工具.Daikon 通过预置不变量测试库,用运行轨迹进行检验测试库中的每个不变量,根据可信度统计,通过的就是程序中所蕴含的不变量.这些工具在程序不变量动态发现上取得了显著的成绩,通过预置不变量的形式,以探测的方法来获得大多数的不变量.获得的结果并不完整,只能检测出不变量测试库预置的不变量,而对于较复杂的函数型不变量,因不在测试库中而无法检测出.但 Daikon 提供了一个接口来添加和扩展不变量形式,相关研究者利用该接口实现了一些不变量的发现.Cobb 定义了关系数据库的元素和 Daikon 的程序位置和对变量的观察之间的映射,可以挖掘数据库表的行级别和列级别的不变量[5].许欢利用动态不变式运行程序的特点和后置断言与循环不变式间的联系,分析得到相应的规律,使用 daikon 发现循环不变式[6].本文针对对数函数型不变量的动态发现进行研究.

基因表达式编程 GEP(gene expression programming)是 Gandida Ferreira 在 2001 年提出的一种新型自适应演化算法.借鉴自然界生物进化的规律,在遗传算法(genetic algorithm,GA)和遗传编程(genetic programming,GP)的基础上提出的基于表现型和基因型的优化算法.自该算法提出后,引起了国内外众多学者的关注,对 GEP 展开了深入地研究,获得了大量的理论与实践的研究结果.GEP 已经成功运用解决了许多如分类、函数发现、数据挖掘、时间序列等问题[7].文献[8]提出一种函数型似然程序不变量的发现方法:第一步利用关系数据理论检测是否存在函数型程序不变量;第二步利用 GEP 对函数的发现能力,获得较客观的程序不变量的预置形式;第三步根据第二步获得的启发式信息,寻找相应的数据理论获得精确的函数表达式.文献[9]对指数函数型程序不变量发现方法进行研究,将 GEP 的理论应用于程序不变量的研究中,发现 GEP 对于发现线性指数函数形式的程序不变量具有很好的效率,增大了从轨迹数据中发现更多不变量的可能性.GEP 算法在复杂函数形式发现中存在稳定性不佳,精度不高的等问题,本文在此方法的基础上,在文献[9]的基础上,基于 GEP-RNC 算法,进行了线性指数和对数函数型不变量的动态发现.在不降低标准 GEP 算法的发现效率的前提下,可以有效提高函数不变量发现的稳定性和精确度.

1 GEP-RNC 简介

带随机数值常数的基因表达式编程(GEP with random numerical constants, GEP-RNC)算法[10]是 Ferreira 提出了在基因尾部附加 Dc 域随机创建常数的方法.GEP-RNC 算法是比 GEP 算法更复杂的算法,它是在 GEP 染色体的基因尾部添加一个长度等于尾部长度 t 的用符号表示对应常数的 Dc 域,该数组保存一组既定数目的候选随机常量,基因中出现的常数项的基因位用“?”代表.GEP-RNC 算法有一组额外的遗传算子,包括 RNC 变异、Dc 变异、Dc 倒串、Dc IS 插串、Dc 置换、常数校正、常数插入、常数范围查找.GEP-RNC 算法的 Dc 域和候选常量集中的内容也可以参加 GEP 的各种遗传操作,候选常量在进化过程中也随着进行进化,保证了随机常数可以有效地在运行时产生恰当的多样性,并在以后的进化过程中保持这种多样性.GEP-RNC 算法不需要对函数形式进行假设,只需要设定函数符号集合和终结符号集合,通过遗传、变异、重组等操作,通过不断的进化获得函数关系.

GEP 算法可以通过相同的终结符的算术运算产生常数,如 x/x 产生常数 1,$x-x$ 产生常数 0,即通过算法在演化过程中的某些函数符合终结符的特定连接运算操作产生所需要的常数,并在程序执行过程中根据适应度函数不断的进行自适应的调整.这使得对常数的处理变得复杂,也增加了算法的复杂度.而对于含有浮点数的模型等复杂性问题上,在基因中引入附加常数域的方法可以获得更高的成功率和精度.

2 基于 GEP-RNC 程序不变量发现方法

GEP-RNC 算法进化过程与 GEP 的过程一样,都是使用函数集和终结符集随机生成染色体个体,通过遗传、变异、重组等操作,依据适应度算法来确定函数的拟合程度,通过不断的进化获得函数关系.

根据蕴含了某种函数关系的轨迹数据,GEP-RNC 算法可以很好的发现其中的函数关系.发现程序不变量的流程是:先利用关系数据理论发现程序中的函数依赖关系,然后通过 GEP-RNC 的函数发现能力,从大量的轨迹数据中进行函数形式的挖掘,再寻找相应的数学理论精确参数,确定并检验函数表达式即为获取的程序不变量形式.本文着重于研究对数型函数的程序不变量的发现,旨在研究 GEP-RNC 算法对于对数型函数的发现能力.

2.1 基于 GEP-RNC 指数对数型程序不变量发现实验

2.1.1 函数集和终结符集合

构造一个 GEP,首先要选择合适的函数集和终结符集合来产生染色体.集合的选择要满足两个要求[11]:1)充分性:选择的集合要足以能够表示问题的解.不能选择过大的函数集合,否则会极大的扩大 GEP 的搜索空间,降低搜索效率.2)封闭性:寻找的公式可能以任何的方式组合且子节点的输出一定要能够被父节点接受,因而,所有终结符、函数的值域以及函数的每一个参数的定义域都要相同.此次实验选择的函数集 $F=\{+,-,*,/,\ln,\exp\}$,终结符集 $T=\{x,?\}$,? 表示随机产生的常量,随机常量的取值范围为[-10,10].

2.1.2 适应度函数选择

进化算法需要对新产生的染色体进行环境适应能力评价.一般用适应度来度量物种对环境适应能力的强弱.本次选择 R-square 作为实验的适应度函数.其公式如下所示:

$$R_i=\frac{n\sum_{j=1}^{n}(T_jP_{i,j})-(\sum_{j=1}^{n}T_j)(\sum_{j=1}^{n}P_{i,j})}{\sqrt{(n\sum_{j=1}^{n}T_j^2-(\sum_{j=1}^{n}T_j)^2)(n\sum_{j=1}^{n}P_{i,j}^2-(\sum_{j=1}^{n}P_{i,j})^2)}}$$

$$f_i=1\ 000\times R_i\times R_i$$

其中,R_i 表示第 i 个个体所对应的函数模型的复相关系数,f_i 表示第 i 个个体的适应度函数,T_j 表示第 j 组数据的输入,$P_{i,j}$ 表示第 i 个个体对于第 j 个样本的返回值.

2.1.3 运行参数和遗传算子参数的设置

运行参数的确定主要来源于实验的经验值范围,遗传参数的设置选择最优的进化策略,可以提高 GEP-RNC 的收敛速度和发现效率,确定进化终止条件是为了在不收敛的情况下可以终止操作.设置结果如表 1 所示.

表 1 部分参数

Table 1 Partial parameters

基本参数		GEP-RNC 部分遗传算子概率	
初始染色体个数	50	RNC 变异	0.002 06
头部长度	15	Dc 变异	0.002 06
基因个数	1	Dc 倒串	0.005 46
进化终止条件	最大迭代次数=500	Dc IS 插串	0.005 46
变异	0.001 38	Dc 置换	0.005 46
倒串	0.005 46	常数校正	0.002 06
IS 插串	0.005 46	常数插入	0.001 23
RIS 插串	0.005 46	常数范围查找	0.000 085

2.1.4 实验过程

实验环境:Pentium(R) Dual-Core CPU, 2.60 GHz

实验平台:GeneXproTools 5.0

1)选择实验函数:选取以下函数作为实验对象

F_1: $y = \ln x$

F_2: $y = \ln(2x)$

F_3: $y = \log_2(3x + 7)$

F_4: $y = 3\log_5(x + 7)$

F_5: $y = \log_3(x^2 + 2x + 3)$

F_6: $y = \exp(x)$

F_7: $y = 5 * \exp(2x + 3)$

F_8: $y = 3 * x * \exp(x)$

F_9: $y = \exp(x) + x^3 + 1$

2)获取实验数据:F_1 到 F_5 选取[0,100]的随机值,F_6 到 F_9 选取[-10,10]的随机值,获取以上函数的轨迹点(x,y)作为测试数据.训练样本数为 500,测试样本数为 1 000.

3)运行获得结果:将每组测试数据在测试平台上运行 100 次,记录适应度大于 999 生成的表达式,经过简化后进行统计分析并得出结果.实验结果如表 2 所示.

表 2　实验结果

Table 2　Experimental results

实验对象	生成表达式形式	出现次数/次
F_1	$y = \ln x$	100
	$y = \ln(2x)$	92
F_2	$y = \ln(ax + t)$	8
	$y = k * \ln(ax + t)$	80
F_3	$y = k * \ln(ax + t) + b$	18
	$y = k * \ln(ax + t) + \ln(\ln(b + x)$	2
F_4	$y = k * \ln(ax + b)$	67
	$y = k * \ln(ax + b) + c$	25
F_5	$y = k * \ln(x^2 + bx + c)$	45
	$y = k * \ln(ax^2 + bx + c)$	14
	$y = k * \ln(ax^2 + bx + c) + d * \ln x$	4
F_6	$y = \exp(x)$	100
F_7	$y = k * \exp(ax + b)$	100
F_8	$y = a * x * \exp(x)$	24
	$y = a * x * \exp(x) + b$	63
	$y = a * x * \exp(x) + bx + c$	11
	$y = a * x * \exp(x) + bx^2 + c$	2
	$y = \exp(x) + x^3 + b$	20
F_9	$y = \exp(x) + ax^3 + b$	78
	$y = \exp(x) + x^3 + ax + b$	2

从实验中可以看出,利用 GEP-RNC 算法可以发现程序中蕴含的函数形式.根据对数函数的性质,对于对数函数的研究,函数 $y = \alpha \cdot \log_b(f(x))$ 可以转换成 $y = (a/\ln b) \cdot \ln(f(x))$ 形式,简化为 $y = k \cdot \ln(f(x))$ 的形式,对于指数函数 $y = k \cdot \exp(ax + b)$ 可等价于 $y = \exp(ax + c)$,$y = \exp(ax)/d$ 等函数,其中 a,b,c,d,k 为有理数.因为函数的性质导致函数形式多变,GEP-RNC 算法生成的函数形式大部分都符合换算后的实验的函数形式.对于简单的指数或对数函数形式,它的发现效率甚至达到 100%,函数形式较为复杂时,经过简化和转换后,发现效率依然很高.GEP-RNC 算法可以很好的发现指数或对数型函数形式,且具有较好的发现效率.因此,通过 CEP-RNC 算法可以获得指数对数函数形式的启发信息,将此形式预置到 Daikon 中,可由 Daikon 完成指数对数型程序不变量的发现.

2.2　GEP-RNC 算法和 GEP 算法的比较

为了 GEP-RNC 算法和 GEP 算法的直接比较,我们使用 2.1 中 F_3、F_4、F_8、F_9 进行实验,GEP-RNC 算法与 GEP 算法共有的参数取相同值,与表 2 相同,最大迭代次数为 500,在 100 次独立运行的基础上,通过平均适应度、最优 R-Square 及成功率来观察算法的性能.实验结果如表 3 所示,得到的最优表达式如表 4 所示.

表 3　GEP-RNC 算法和 GEP 算法的比较结果

Table 3　Comparison of GEP-RNC with GEP algorithms on benchmark

	项目名	平均适应度	适应度最优值	成功率
F_3	GEP	985.605 4	999.999 9	79%
	GEP-RNC	996.079 5	999.999 9	98%
F_4	GEP	988.182 9	999.999 9	80%
	GEP-RNC	995.212 0	1 000	92%
F_8	GEP	990.312 6	1 000	81%
	GEP-RNC	995.310 4	1 000	87%
F_9	GEP	991.084 6	999.999 9	95%
	GEP-RNC	997.601 6	1 000	98%

从实验结果可以得知,GEP-RNC 算法得到的平均适应度比 GEP 算法高,且成功率也高于 GEP 算法.对于对数型函数,标准的 GEP 算法得到的最优表达式经过化简后依然复杂,而 GEP-RNC 算法得到的表达式要简单些,得到的函数型式更接近真实表达式;对于指数型函数,GEP-RNC 得到的表达式相对简单,获得最优表达式均与原型相同.从整体而言,GEP-RNC 算法同样具备发现指数对数函数形式的能力,GEP-RNC 算法相比于 GEP 算法,具有更高的精确度和稳定性.

表 4 最优表达式

Table 4 Optimal expression

实验对象	算法	最优表达式	化简式
F_3	GEP	$y = \ln((\ln x + x/x + (x + x)/(x * x)) * (x + x + x + x + x + x))$	$y = \ln(6x\ln x + 6x + 6)$
	GEP-RNC	$y = \ln 4.20 * \ln 4.20 * \ln(x + x + 6.77 + x)$	$y = 1.43 * \ln(3x + 6.77)$
F_4	GEP	$y = (x * x/(x + x) + x/x + x)/x * \ln(x + x + x + x/x)$	$y = (1.5 + 1/x)\ln(3x + 1)$
	GEP-RNC	$y=1.86 * \ln(x-(-7.06+4.42/(4.42(-7.01)-(x/(-7.06)+(9.97 * 7.01)))))$	$y=1.86 * \ln(x+7)$
F_8	GEP	$y = x/x * x/(x/x) * (\exp(x) - (x - x)) * ((x + x + x)/x)$	$y = 3 * x * \exp(x)$
	GEP-RNC	$y = (\exp(2.17) - (5.62) * (x + x - x) * \exp(x)$	$y = 3 * x * \exp(x)$
F_9	GEP	$y = \exp(x) + x - \exp(x - x) - (x - (x/x)) + (x - x) - x * x * x)$	$y = \exp(x) + x^3$
	GEP-RNC	$y = x * x * x + (\exp(1.02) + 1.62) * 0.23 + \exp(x)$	$y = \exp(x) + x^3 + 1$

3 结束语

对于程序不变量的发现,要求高效找出程序中蕴含的所有不变量.本文是通过 GEP-RNC 发现指数对数型函数形式的不变量.相比于标准的 GEP 算法,具有更高的精确性和稳定性.此方法为 Daikon 扩充了不变量发现的形式,增加了程序不变量发现的可能性.

参考文献:

[1] 周辉.基于程序不变量的并发软件可靠性计算[D].杭州:浙江理工大学,2014.

[2] 刘鑫.范围不变量软件错误定位优化方法研究[D].哈尔滨:哈尔滨工业大学,2016.

[3] 胡觉亮.测试方法对软件可靠性计算的影响分析[D].杭州:浙江理工大学,2014.

[4] RAO A A,RAJU P R.Optimization of program invariants [J].Acm Sigsoft Software Engineering Notes,2014,39(1):1-9.

[5] COBB J,JONES J A,KAPFHAMMER G M,et al.Dynamic invariant detection for relational databases[C]//Proceedings of the Ninth International Workshop on Dynamic Analysis.New York:ACM,2011:12-17.

[6] 许欢,王以松.用 Daikon 发现循环不变量式[J].贵州大学学报,2012,29(4):77-81.

[7] PENG Y Z,YUAN C A,QIN X,et al.An improved gene expression programming approach for symbolic regression problems[J].Neurocomputing,2014,137(15):293-301.

[8] 吴取劲.基于 GEP 的函数型似然程序不变量发现方法[D].衡阳:南华大学,2009.

[9] 黄彩霞.基于 GEP 的指数函数型程序不变量发现方法研究[D].衡阳:南华大学,2012.

[10] FERREIRA C.Gene expression programming:Mathematical modeling by an artificial intelligence[M].2nd.Gewerbestrasse:Springer,2006.

[11] 元昌安,彭昱忠,覃晓,等.基因表达式编程算法原理与应用[M].北京:科学出版社,2010.

(上接第 71 页)

[4] 李松芹,张寄洲.跳扩散模型下重置期权的定价[J].高等学校计算数学学报,2005,27(增刊 1):182-187.

[5] 刘邵容,王恒太.O-U 过程模型下一种改进的亚式再装期权定价[J].经济数学,2014,31(1):117-120.

[6] 易小兰,张庆华,闫理坦.带泊松跳分数市场的欧式幂期权定价[J].苏州科技学院学报,2015,29(2):173-178.

[7] 孙玉东,师义民,谭伟.分数布朗运动环境下亚式期权定价的新方法[J].工程数学学报,2012,23(1):27-31.

[8] 卢炜迪,韩月才.双指数链式平方期权的定价研究[J].数学的实践与认识,2015,45(18):15-19.

[9] 张翠娥,徐云.随机利率下服从 O-U 过程的二元期权定价[J].数学理论与运用,2012,23(1):27-31.

[10] 周清,李超.分数利率模型下几何平均亚式期权的定价公式[J].应用数学学报,2014,37(4):662-675.

[11] JUN D,KU H.Pricing chained options with cured barriers [J].Mathematics Finance,2013,23(4):763-776.