

基于改进遗传算法的测试数据自动生成的研究

高雪笛^{1,2} 周丽娟^{1,2} 张树东^{1,2} 柳昊明³

(首都师范大学信息工程学院 北京 100048)¹ (成像技术北京市高精尖创新中心 北京 100190)²
(北京航空航天大学计算机学院 北京 100048)³

摘要 测试数据自动生成是软件测试的基础,也是测试自动化技术实现的关键环节。为了提高测试自动化的效率,在结合测试数据自动生成模型的基础上,提出一种传统遗传算法的改进算法。该算法使用了自适应交叉算子和变异算子,并引入模拟退火机制对其进行改进。同时,该算法还对适应度函数进行了合理的设计,以加速数据的优化过程。通过三角形程序、折半查找和冒泡排序程序,与基本遗传算法、自适应遗传算法进行了比较与分析,并且对改进算法做了性能分析。实验结果表明了该算法的实用性以及在测试数据生成中的可行性和高效性。

关键词 软件测试,遗传算法,哈明函数,测试数据自动生成

中图分类号 TP311 文献标识码 A DOI 10.11896/j.issn.1002-137X.2017.03.044

Research on Test Data Automatic Generation Based on Improved Genetic Algorithm

GAO Xue-di^{1,2} ZHOU Li-juan^{1,2} ZHANG Shu-dong^{1,2} LIU Hao-ming³

(College of Information Engineering, Capital Normal University, Beijing 100048, China)¹

(Beijing Advanced Innovation Center for Imaging Technology, Beijing 100190, China)²

(School of Computer Science and Engineering, Beihang University, Beijing 100048, China)³

Abstract Automatic test data generation is the basis of software testing, and it is also a key link in the process of test automation technology. In order to improve the efficiency of testing automation, a new algorithm was proposed to improve the traditional genetic algorithm based on the combination of test data automatic generation system model. The adaptive crossover operator and mutation operator are used in this algorithm, and the improved simulated annealing mechanism is introduced to improve it. At the same time, the algorithm is also designed to fit the fitness function to accelerate the optimization process of the data. Through the triangle program, binary search and bubble sort program, the basic genetic algorithm and the adaptive genetic algorithm were compared, and the performance test was done for improved algorithm. Experimental results show the practicability as well as feasibility and efficiency of the algorithm in the test data generation.

Keywords Software test, Generic algorithm, Hamming function, Automatic test data generation

1 引言

在测试自动化过程中,测试数据的生成是软件测试自动化的根本,也是测试中极其重要的环节^[1-3]。同时,随着软件功能增强和规模的不断扩大,其逻辑也会更加复杂,如果仍采用传统手动的方式来设计、编写测试数据,不仅会造成不必要的人力、物力或财力上的浪费,而且还很难保证对路径的覆盖率。因此,自动化测试被广泛应用,而研究一种高效的测试数据自动生成的方法对于测试的自动化过程有着十分重要的意义。

目前,测试数据自动生成的方法有很多,基本上可以分为

三大类:静态法、动态法和智能优化法。常见的静态法有随机法、符号执行法等;动态法有迭代松弛法、试探法、程序插桩法等^[2];智能优化算法有遗传算法、模拟退火算法、蚁群算法、禁忌搜索算法等。这些算法为解决测试数据自动生成问题提供了新思路。

遗传算法是模拟生物进化理论的全局优化算法,也是在各个领域应用最广泛的算法。相比于其他智能优化算法,遗传算法不受初始种群大小的限制,具有全局搜索能力强等特点,但也有易早熟、收敛性差等缺陷。因此,本文对遗传算法进行改进,分别对两种算子在自适应的基础上引入模拟退火机制和对适应度函数进行合理设计,3类不同的实验分析证

到稿日期:2016-01-25 返修日期:2016-05-01 本文受国家自然科学基金(31571563),国家科技支撑计划项目(2013BAH19F01),国外访学项目(067145301400),北京市属高等学校创新团队建设与教师职业发展计划项目,高可靠嵌入式系统技术北京市工程研究中心资助。

高雪笛(1989—),女,硕士生,主要研究方向为软件工程、软件测试、人工智能,E-mail:cnugxd@163.com;周丽娟 女,博士,教授,硕士生导师,主要研究方向为数据仓库、商务智能,E-mail:zlj87@139.com(通信作者);张树东 男,教授,主要研究方向为网络与分布式系统、计算机软件,E-mail:13801255795@139.com;柳昊明 男,主要研究方向为数据库、数据挖掘、人工智能。

明了算法的有效性和高效性。

2 基于遗传算法的测试数据自动生成系统框架

遗传算法由美国 Michigan 大学的 Holland J H 教授创建,源于生物进化理论,是一种全局优化搜索方法,也是一种迭代的搜索,具有良好的全局搜索能力^[1]。该算法由初始种群经过选择、交叉、变异等一系列遗传操作,生成由评价值表征的最优个体,根据最大迭代次数等参数信息进行循环搜索。此算法具有隐含的并行搜索,不需要其他辅助信息,不依赖于初始种群。

测试数据的自动生成问题可以描述为:设 P 为待测路径,使用数据生成算法自动生成输入参数变量 $X = \{x_1, x_2, \dots, x_n\}$, n 为输入参数的个数,若输入变量 X 的实际执行路径为 P' ,当 $P' = P$ 时,输入变量 X 为所需的最优测试数据。

由遗传算法理论和测试数据的自动生成技术,得出基于遗传算法的测试数据生成框架由 3 个部分组成(见图 1)。

1) 程序分析模块:对被测程序进行语法、语义和词法等静态分析,生成测试路径集合,从集合中选中一条逻辑路径,并对路径进行插桩后生成桩程序;同时分析程序的输入参数,确定输入变量参数集合。此模块是整个模型框架得以运行的基础。

2) 遗传算法模块:根据上一模块得到的输入参数集合,对参数进行编码组合成对应的染色体串(即遗传算法中的个体),生成初始种群,然后根据被测插桩程序的运行结果反复执行遗传(选择、交叉、变异)操作,直到找到最优数据或者达到最大迭代次数。此模块是模型框架运行的核心。

3) 驱动程序模块:接受测试数据的输入,调用运行第一模块生成的桩程序对每个测试数据进行适应度评价,将评价结果返回给遗传算法模块。此模块是程序分析模块和遗传算法模块的桥梁,在整个模型框架中起到纽带的作用。

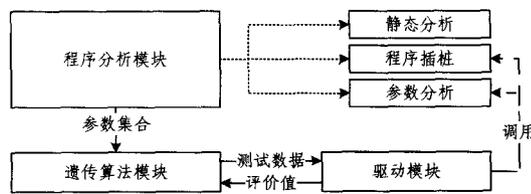


图 1 基于遗传算法的测试用例生成框架

3 改进的遗传算法

遗传算法数据生成框架是将自然界基因的遗传过程与达尔文适者生存原则相结合,不断进化(选择、交叉、变异)与淘汰测试用例,最终得到最优测试用例的过程。下面对将改进的遗传算法应用到测试数据自动生成中的主要问题的处理方法进行讨论。

3.1 编码方式

为了具备更强的搜索能力,保持种群多样性,遗传算法需要对遗传操作的个体进行编码。通常情况下,编码方式有:浮点数编码、二进制编码、十进制编码、符号编码等。为了简化测试数据生成中的实际问题,本文选择二进制编码方式。

根据生物进化原理,二进制编码是用二进制符号 $\{0,1\}$ 构

成一串进化个体的基因型,即所谓的染色体串,它决定了个体的具体表现形式,即实际问题的参数。一般而言,把遗传操作个体的表现型转换成遗传算法迭代中的基因型的过程叫作编码;反之,叫作解码。

对于测试数据生成问题,遗传操作的个体是被测程序的输入参数,当输入参数为多个时,需要对参数进行合并,称为多参数级联编码。具体做法是:把每个参数单独进行二进制编码,形成单个二进制串,然后以多参数级联的方式把每个参数的二进制串合并在一起,构成在遗传算法中进行遗传操作的具体个体。假设被测程序的输入参数有 m 个,每个参数采用定长编码方式,长度为 L ,即由 L 个 $\{0,1\}$ 组成的二进制字符。串长 L 决定了被测程序输入参数的取值范围和精度,连接在一起后形成一个长度为 $m * L$ 的二进制长串,此二进制长串就是遗传的染色体串,也就是遗传个体的基因型。

根据多参数级联编码方式,参数的解码过程如下:对于长度为 mL 的二进制串,每个参数的表示范围为 $[0, 2L - 1]$,把基因型转变为表现型的解码函数为:

$$X_i = f(b_{i1}, \dots, b_{in}, \dots, b_{iL}) = \sum_{k=1}^L b_{ik} * 2^{n-k} \quad (1)$$

其中, $i=1, 2, \dots, m$, 表示第 i 个输入参数, n 为第 i 个参数的第 n 位字符。

相比于其他编码方式,采用二进制方式进行编码和解码,算法易于实现,便于交叉和变异操作,能够缩短算法的优化搜索过程,同时保持种群多样性。

3.2 适应度函数

适应度函数作为评价搜索空间中解的优劣度的准则,在遗传算法搜索优化中扮演着重要的角色,是与现实问题唯一的接口。构造一个好的适应度函数不仅可以增大找到可行解的可能性,而且能够减少系统的资源消耗并提高资源利用率。因此,适应度函数的设计应该与软件测试问题紧密结合,通常情况下,用测试数据的覆盖率来评价数据的优劣。本文根据改进的哈密距离函数和分支函数的构造方式,通过路径覆盖和分支覆盖来设计适应度函数。

在程序分析模块,对被测程序的各个分支点以函数插桩的方式插入标志符号和分支函数: $f(1), f(2), \dots, f(n)$, n 表示分支数,从而形成插桩程序。这样,在程序执行时,由标志符号和分支函数的值就可以得出测试用例是否执行了所选逻辑路径。根据 KOREL 分支函数理论^[4],若某个分支点被执行,则分支函数的值为 0;反之,分支函数的值越小,表示靠近程度越大。因此为了得出测试用例的分支覆盖情况,本文采用“分支叠加法”:

$$F = f(1) + f(2) + \dots + f(n) \quad (2)$$

但是,遗传算法的适应值一般要求为优化最大解,因此,对式(2)进行尺度变换:

$$F = \frac{M}{c + f(1)} + \frac{M}{c + f(2)} + \dots + \frac{M}{c + f(n)} \quad (3)$$

其中, c 的取值是为了保证 $c + f(i)$ 可取最小值,此时 F 能取到最大值; M 为适应度的上限。

利用哈密距离来计算路径覆盖情况,分为偏移量和逼近程度两种指标。偏移量表示实际执行路径的偏离分支点,逼近程度表示测试用例经过的正确节点或者该路径被产生的频

度。因此,路径覆盖函数为:

$$P = offset + subP^2 \quad (4)$$

其中, $offset$ 为偏移点, $subP$ 为指定路径和所走路径共有的最长子路径的长度。

综上所述,遗传算法的适应度函数为:

$$Cost = \frac{\sum_{i=1}^n \frac{M}{c+f(i)}}{n} + P \quad (5)$$

3.3 选择算子

本文采用基本遗传算法中最经典的轮盘赌法、比例选择法和最优个体保留法来选择种群个体。首先,把种群个体(即输入参数)进行解码后传递到被测插桩程序中运行,之后获得适应度函数值,然后把个体按照适应度函数值由大到小排序,选择一定比例的大适应度值的个体直接作为下一代个体而不进行交叉和变异操作。而其他个体具有被随机选中的概率,此概率与种群适应度密切相关,本文通过个体所占总体种群适应度的比例来表示。为完成下一轮的交叉和变异,需要进行多次选择,每次选中两个个体,个体被选中后经过交叉、变异操作遗传到下一代,由此进入下一次迭代。

在个体选择过程中,遵守和模拟了自然界物种遗传过程中的适者生存、优胜劣汰的原则,把优良个体的基因遗传保留给下一代,非优良个体被淘汰或进行其他遗传操作。

3.4 自适应的交叉、变异算子

交叉算子和变异算子影响着遗传算法的进化行为和性能,动态的交叉和变异算子具有一定的方向性和目标性,能够保护适应度高的个体,淘汰适应度低的个体,加快搜索速度,克服过早收敛,保持个体多样性。本文采用改进的 Srinivas^[6]的动态自适应的交叉算子和变异算子,两种算子会随着种群适应度和个体适应度的变化而动态自适应地变化。

由 Srinivas 提出的自适应动态遗传算法具有良好的种群多样性,保证了优良的个体能遗传到下一代,但是在算法运行后期,随着平均种群适应度的增加,平均种群适应度和个体最大适应度的差距越来越小,最后很有可能出现两者相等的情况,但是,最大适应度值对应的个体可能并不是全局最优解,而只是局部最优解,如果这样,算法的搜索就会处于停滞不前的状态。为了保证算法的收敛性和避免陷入局部最优,本文在动态自适应算子的基础上进行优化,确保了算法在未搜索到最优解之前仍具有良好的交叉性和变异性,如式(6)、式(7)所示。

$$P_C = \begin{cases} P_{C1} - \frac{(P_{C1} - P_{C2})(f - f_{avg})}{\exp(f_{max} - f_{avg})}, & f \geq f_{avg} \\ P_{C1}, & f < f_{avg} \end{cases} \quad (6)$$

$$P_m = \begin{cases} P_{m1} - \frac{(P_{m1} - P_{m2})(f_{max} - f)}{\exp(f_{max} - f_{avg})}, & f \geq f_{avg} \\ P_{m1}, & f < f_{avg} \end{cases} \quad (7)$$

其中, f_{max} 表示当前所有个体中的适应度最大值, f_{avg} 表示平均种群适应度值, f 表示要变异的个体的适应度或者要交叉的个体的适应度的较高值。

本文在改进动态自适应的算子的基础上还引入了模拟退火机制来接受新个体,当交叉和变异产生新的个体时,接受新个体的概率由 Metropolis 准则来确定,当新个体更优时,则接受新个体;而当新个体的质量没有提高时,则以一定的概率接

受这个较差的新个体^[16],如式(8)所示。

$$P_{t_{k+1}} = \begin{cases} \exp\left[-\frac{f_{k+1} - f_k}{t_{k+1}}\right], & f_{k+1} < f_k \\ 1, & f_{k+1} \geq f_k \end{cases} \quad (8)$$

其中, f_{k+1} 是新产生个体的适应度值; f_k 是当前个体的适应度值; $P_{t_{k+1}}$ 是在温度为 t_{k+1} 条件下的接受概率。

3.5 遗传算法的测试数据自动生成

遗传算法的选择、交叉和变异操作都是基于适应度评价准则的,适应度高的个体在进化过程中占有很大的优势,适应度低的个体经过自适应的操作后演变成高适应度的个体。每代个体都会产生新的个体作为下次迭代的初始种群个体,按此一代一代地循环迭代下去。迭代次数通常标示着算法优化的收敛状态。本文并不是以末代种群的最优个体作为算法输出的最优解,而是以生成满足测试标准的测试数据为问题的最佳个体,同时以到达最大迭代次数为算法的终止条件。

结合测试用例自动生成模型框架,算法的具体流程步骤如下(见图 2):

- 1)初始化参数。确定种群规模、算法最大迭代次数、交叉算子、变异算子等参数。
- 2)输入被测程序,指定测试路径,确定输入参数个数;程序插装,构造适应度函数。
- 3)按照输入变量的解空间进行种群初始化。
- 4)将种群个体解码成输入变量,运行被测程序,根据评价价值判定是否满足停止准则,若满足,则输出满足准则的个体及其覆盖情况;否则,继续步骤 5)。
- 5)对个体进行最优保留和轮盘赌选择操作。
- 6)对个体进行自适应交叉和变异操作。
- 7)对新产生的种群执行步骤 4)。

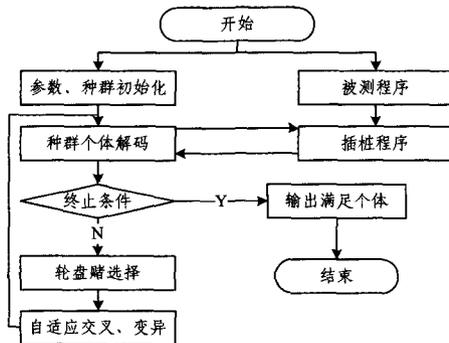


图 2 遗传算法的测试用例自动生成流程图

4 实验结果分析

4.1 模型实验分析

本文采用在测试数据生成领域应用最广泛的三角形判别问题和比较广泛的折半查找程序、冒泡排序程序来验证改进遗传算法(IGA)生成测试数据的优劣,并将其与基本遗传算法(SGA)、基本自适应遗传算法(AGA)进行比较和分析。

1)对于三角形判别问题,因等边三角形和直角三角形的测试数据生成逻辑更复杂和更具有代表性,故本文选定等边三角形和直角三角形进行数据生成。

实验初始时,3种算法的运行参数均按以下数据设置,输入参数为三角形的3条边 A, B 和 C ,每一个参数占 8 位,即

取值范围是[0, 255], 染色体总长度为 24, 最大迭代次数为 1000, 初始种群数为 10~50。自适应遗传算法和改进遗传算法的交叉算子和变异算子的参数设置为: $P_{C1}=0.8, P_{C2}=0.6$, 变异算子中 $P_{M1}=0.05, P_{M2}=0.005$; 基本遗传算法的交叉概率为 0.85, 变异概率为 0.05。

2) 折半查找程序可以描述为: 输入参数是一串带有字符串的按关键字排好序的数组, 在输入需要查找的字符串后, 对整个数组进行折半查找, 根据查找的结果输出被查找字符串在数据中的位置。

实验参数设置: 数组中含有 6 个数据, 且每个数据的取值范围是[0, 15], 占 4 位, 染色体总长度为 24 位。程序具有循环控制, 共循环 5 次, 最大迭代次数为 500, 初始种群分别为 10~50 个, 其他运行参数与三角形判定实验的参数一致。

3) 冒泡排序程序可以描述为: 输入参数是存储数字的未经过关键字排序的一串数组, 经过多次相邻两个元素的比较和交换, 测试输出一串按关键字大小排序的有序数据。程序中含有二重 for 循环和条件控制。

实验参数设置: 数组中含有 6 个数据, 且每个数据的取值范围是[0, 15], 占 4 位, 染色体总长度为 24 位; 最大迭代次数为 500, 初始种群数为 10~50; 基本遗传算法的交叉概率为 0.9, 变异概率为 0.01, 其他参数与三角形实验的参数相同。

实验过程中, 首先输入被测程序, 对程序进行静态分析, 根据分析结果和流程图对选定的指定路径插桩; 然后在种群初始化和参数初始化后分别对 3 类问题运行 3 种遗传算法, 在运行过程中调用插桩程序获得评价函数值, 以诱导进化的方向和进度, 由交叉算子和变异算子驱使 3 种算法迭代寻优, 直到找到满足要求的最优测试数据。每类实验对 3 种遗传算法均分别运行 10 次, 分别记录种群规模为 10~50 时每次搜索到的最优测试数据的最差迭代次数和最好迭代次数, 由统计后的结果计算出 10 次运行的平均迭代次数(见图 3~图 6)。

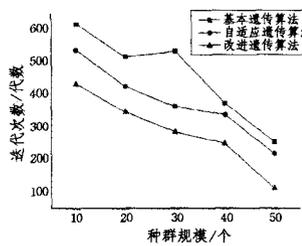
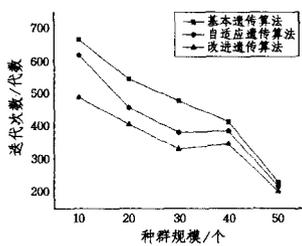


图 3 等边三角形的最优个体迭代次数

图 4 直角三角形的最优个体迭代次数

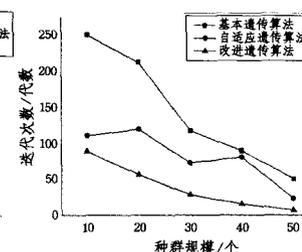
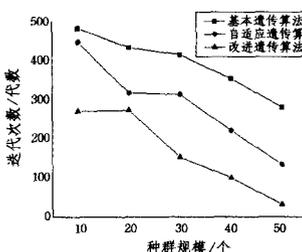


图 5 折半查找程序的最优个体迭代次数

图 6 冒泡排序程序的最优个体迭代次数

从图 3、图 4 中可以看出:

1) 图中显示了生成满足指定路径的测试数据的平均迭代次数, 在其他参数相同的情况下, 随着种群规模的逐步增大, 3 种遗传算法生成最优测试数据的迭代次数均会逐步减少。

2) 图中的横坐标为初始种群规模, 单位为个; 纵坐标为找到最优测试数据的平均迭代次数, 单位为代数。在各种参数相同的情况下, 对于指定路径的测试数据生成, 改进的遗传算法的搜索最优测试数据的迭代次数均少于基本遗传算法和自适应遗传算法的迭代次数。

3) 在运行基本遗传算法和自适应遗传算法时, 均出现两次陷入局部最优、搜索停滞不前的状况, 而改进遗传算法很好地弥补了容易陷入局部最优解的缺陷, 减小了停滞的可能性, 加快了搜索的速度。尤其是在搜索直角三角形的最优测试数据时, 最好的迭代次数为 3, 一般的迭代次数在 40~60 左右, 搜索最优测试数据的效率被大大提高。

从图 5 中可以看出:

1) 图中横、纵坐标的意义和单位与三角形分类问题即图 3、图 4 的实际意义相同。随着种群规模的增大, 3 种遗传算法在利用折半查找搜索最优可行解时的平均迭代次数总体呈减少趋势。

2) 在实验参数相同的情况下, 改进遗传算法的平均迭代次数均优于基本遗传算法和自适应遗传算法。

从图 6 中可以看出:

1) 随着种群规模的增加, 遗传算法利用冒泡排序搜索最佳测试用例的收敛求解速度逐步加快, 平均迭代次数减少, 提高了数据生成效率; 同时, 改进遗传算法的收敛能力均优于基本遗传算法和自适应遗传算法。

2) 由于冒泡排序程序的特殊性, 在寻找最优测试数据时, 基本遗传算法和自适应遗传算法均出现了不稳定性, 在总体趋势降低的情况下, 仍不能稳定搜索, 适用性和全局搜索能力差, 效率不高, 而改进遗传算法具有很好的稳定性和鲁棒性, 收敛速度快。

从实验结果中可以看出, 改进的遗传算法加快了算法的进化过程, 能够快速找到软件测试数据的可行解。因此, 此算法可以很理想地应用到测试数据生成方面。

4.2 算法性能分析

为验证改进遗传算法的性能, 采用 Schaffer F6 函数分别对基本遗传算法和改进遗传算法进行测试, 其表达式如式(9)^[10,18-19]所示:

$$f(x, y) = \begin{cases} 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{(1 + 0.001 * (x^2 + y^2))^2} & -100 \leq x \leq 100, -100 \leq y \leq 100 \\ -100 \leq x \leq 100, -100 \leq y \leq 100 \end{cases} \quad (9)$$

Schaffer 函数是一种二元多峰值函数, 在其定义域内具有无数个局部极值点和一个全局最优点, 此最优解在(0, 0)点取得, 值为 0; 而且全局最优解的附近形成了含有无数局部极值的圆脊, 若算法的性能较差, 则极易收敛于局部极值。因此, 该函数为最常用的优化算法的测试基准函数。

在种群规模为 50、迭代次数分别为 100, 200, 500 的条件下, SGA 算法和 AGA 算法各随机运行 100 次, 其他运行参数均与三角形实验参数一致, 记录两种算法的收敛次数, 如表 1 所列。

表1 性能实验的结果对比

迭代次数	收敛次数	
	SGA	IGA
100	19	28
200	37	54
500	59	79

从表1中可以看出,在基本实验参数相同的条件下,随着迭代次数的增加,两种算法的收敛次数也在增加;并且,IGA算法具有更快的收敛速度,全局搜索效率高于SGA算法。

综上所述,对遗传算法在适应度函数和交叉算子、变异算子上的改进提高了算法的收敛效率,加速了算法的优化进度,具有更好的寻优性能。因此,改进的遗传算法不仅可以很有效地应用到测试数据生成上,还可以应用在其他领域,具有更加广泛的实用性。

结束语 测试套件(test suite)按照各阶段不同的测试目标的一组相关的测试,产生的一系列测试用例,该组测试用例一般都是为了实现同一个测试目的。测试用例是最底层的测试输出,是针对不同的功能点、非功能点的具体测试流程。从测试数据的角度上看,提高测试用例自动生成的效率对测试自动化至关重要。

本文将改进的遗传算法应用到测试数据自动生成中,同时对算法的性能进行测试。实验结果表明,其弥补了基本遗传算法的不足和缺陷,提高了测试数据的生成效率,加速了算法的收敛速度,能够很理想地运用到测试自动化领域或其他领域。

Gordon Fraser 等人^[12]利用 EVOSUITE 针对 5 个开源库和一个工业案例进行了测试套件生成,研究发现,EVOSUITE 在测试套件生成优化中比传统的单一分支覆盖方法提高 18 倍,同时,测试套件集也缩小 44%。在 EVOSUITE 中,利用遗传算法的一系列随机产生、交叉、变异等遗传操作来优化选择的覆盖准则,并且使用限制最大种群数和测试用例的长度、基于排序选择的适应度函数、子代优于父代等策略来控制膨胀问题。

EVOSUITE 是针对测试套件优化生成的一个很好的解决方案,可以生成一组相关的测试用例,更适合于大型面向对象软件。而本文针对遗传算法的改进是基于测试用例产生的方案,也并非传统意义的单一覆盖目标,而是基于路径覆盖和分支覆盖的双重覆盖准则,同时此方案简便,更适合于基于单元测试的白盒测试,不限于面向对象软件和程序软件,不限运行平台,经验证其优于基本遗传算法。但此方案解决大型软件的用例生成问题时比较受限,需要结合 EVOSUITE 这种自动产生测试套件集的测试工具框架做进一步的研究。

参考文献

[1] HOLLAND J H. Genetic algorithms and the optimal allocation of trials [J]. SIAMJ Comput, 1973, 2(2): 89-104.
 [2] NIE P, GENG J, QIN Z G. Survey on automatic test case generation algorithms for software testing [J]. Computer application research, 2012, 29(2): 402-405. (in Chinese)
 聂鹏, 耿技, 秦志光. 软件测试用例自动生成算法综述 [J]. 计算机应用研究, 2012, 29(2): 402-405.

[3] HUANG L F. Simulation Research on Automatically Generate Software Test Data Algorithm [J]. Computer Simulation, 2012, 29(10): 245-247. (in Chinese)
 黄丽芬. 软件测试数据自动生成算法的仿真研究. [J]. 计算机仿真, 2012, 29(10): 245-247.
 [4] KOREL B. Automated software test data generation [J]. IEEE Trans, on Software Engineering, 1990, 16(8): 870-879.
 [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] SRININASM, PAINAIKM. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms [J]. IEEE Tram on Systems, Manand Cybe Rnetics, 1994, 24(4): 656-659.
 [7] CHEN Y, YAO L. Software Test Data Generation Based on an Improved Generic Algorithm [J]. Electronic Science and technology, 2009, 22(7): 9-12. (in Chinese)
 陈雨, 姚砾. 基于改进的遗传算法的测试用例生成. 电子科技, 2009, 22(7): 9-12.
 [8] ARABALI A, GHOFRANI M. Genetic-Algorithm-Based Optimization Approach for Energy Management [J]. IEEE Transactions on Power Delivery, 2013, 28(1): 162-170.
 [9] LIN J C, YEHP L. Using genetic algorithms for test case generation in path testing [C] // 9th Asian Test Symposium (ATS' 00). Taipei, 2000; 241-246.
 [10] LEI H, HAN X. Software Test Data Generation Method Using Hill Climbing Algorithm Combined with a Modified ARPSO [J]. Journal of University of Electronic Science and Technology of China, 2012, 41(6): 885-889. (in Chinese)
 雷航, 韩炫. 采用 HC-MARPSO 算法的软件测试数据生成方法 [J]. 计算机工程与应用, 2012, 41(6): 885-889.
 [11] YU B, JIANG S J, ZHANG Y M. Multiple Paths Test Case Generation Based on Complex System Genetic Algorithm [J]. Computer Science, 2012, 39(4): 139-141. (in Chinese)
 于博, 姜淑娟, 张艳涛. 基于复杂系统遗传算法的多路径覆盖测试用例生成方法 [J]. 计算机科学, 2012, 39(4): 139-141.
 [12] FRASER G, ARCURI A. Evolutionary Generation of Whole Test Suites [J]. International Conference on Quality Software, 2011, 14(1): 31-40.
 [13] ZHONG X M, ZHAO X F. Automated test case generation based on improved tabu search algorithm [J]. Computer Engineering and Design, 2011, 32(7): 2058-2060. (in Chinese)
 仲晓敏, 赵雪峰. 基于改进禁忌搜索算法的测试用例自动生成 [J]. 计算机工程与设计, 2011, 32(7): 2058-2060.
 [14] FRASER G, ARCURI A. Achieving scalable mutation-based generation of whole test suites [J]. Empirical Software Engineering, 2014, 20(3): 783-812.
 [15] RAYADURGAM S, HEIMDAHL M P E. Coverage based test-case generation using model checkers [C] // IEEE International Conference and Workshop on the Engineering of Computer Based Systems. 2015: 83-91.
 [16] WANG Y, WANG C, LIU H L. Application of simulated annealing generic algorithm in multiuser detection technique [J]. Communication and Network, 2011(4): 102-105. (in Chinese)
 王彦, 王超, 刘宏立. 模拟退火遗传算法在多用户检测技术中的应用 [J]. 通信与网络, 2011(4): 102-105.

- [17] 邱菊. 基于蚁群算法的软件测试用例生成方法研究[J]. 软件导刊, 2011, 10(3): 73-74.
- [18] LU H Q, CHEN L, SONG Y S, et al. An improved crossover operator of genetic algorithm[J]. Journal of PLA University of Science and Technology, 2007, 8(3): 250-253. (in Chinese)
卢厚清, 陈亮, 宋以胜, 等. 一种遗传算法交叉算子的改进算法[J]. 解放军理工大学学报, 2007, 8(3): 250-253.
- [19] KONG X L, WANG Y, JU A L, et al. An Improved Quantum Evolutionary Algorithm Based on Regulation Law of Hormone in Endocrine System[J]. Journal of Northwestern Polytechnical University, 2011, 29(6): 978-983. (in Chinese)
孔晓琳, 王毅, 巨安丽, 等. 基于内分泌激素调节机制的量子进化算法[J]. 西北工业大学学报, 2011, 29(6): 978-983.
- [20] MIRZAAGHAEI M, PASTORE F, PEZZÈ M. Supporting Test Suite Evolution through Test Case Adaptation[C]// IEEE Fifth International Conference on Software Testing, 2012: 231-240.

(上接第 201 页)

0.76%的 CPI, 但 TG-BTB 比传统 BTB 减少了 80%的 BTB 查询功耗。TG-BTB 可以很容易地应用在其他流水线结构的处理器中, 只需要在指令提取段和指令执行段更新临时寄存器、剩余轨迹长度计数器和跳转轨迹长度累加器的值。这种方法可以成为低功耗嵌入式处理器或高性能处理器降低功耗的技术。由于多核处理器架构设计师都把精力集中在降低单核的功耗上, 因此这种降低单核功耗的技术显得尤为重要。

参 考 文 献

- [1] PERLEBERG C, SMITH A. Branch target buffer design and optimizations[J]. IEEE Transactions on Computers, 1993, 42(4): 396-412.
- [2] MANNE S, KLAUSER A, GRUNWALD D. Pipeline gating: speculation control for energy reduction[C]// Proceedings of International Symposium on Computer Architecture, 1998: 132-141.
- [3] BONANNO J, COLLURA A, LIPETZ D, et al. Two level bulk preload branch prediction[C]// Proceedings of the IEEE International Symposium on High Performance Computer Architecture, 2013: 71-82.
- [4] KEETON K, PATTERSON D A, HE Y Q, et al. Performance characterization of a Quad Pentium Pro SMP using OLTP workloads[C]// Proceedings of International Symposium on Computer Architecture, 1998: 15-26.
- [5] ANNAVARAM M, DIEP T, SHEN J. Branch behavior of a commercial OLTP workload on Intel IA32 processors[C]// Proceedings of International Conference on Computer Design, 2002: 242-248.
- [6] PYNE S, PAL A. Branch Target Buffer Energy Reduction Through Efficient Multiway Branch Translation Techniques[J]. Journal of Low Power Electronics, 2012, 8(5): 604-623.
- [7] HILGENDORF R B, HEIM G J, ROSENSTIEL W. Evaluation of branch-prediction methods on traces from commercial applications[J]. IBM Journal of Research and Development, 1999, 43: 579-593.
- [8] SUSSENGUTH E H. INSTRUCTION SEQUENCE CONTROL; US, US3559183 [OL]. <http://www.google.com/patents/US3559183>.
- [9] LEE J, SMITH A. Branch Prediction Strategies and Branch Target Buffer Design[J]. Computer, 1984, 17(1): 6-22.
- [10] CASZZA J. First the Tick, Now the Tock; Intel Microarchitecture[R]. Nehalem, 2009.
- [11] DRIESEN K, HÖLZLE U. The cascaded predictor: economical and adaptive branch target prediction[C]// Proceedings of International Symposium on Microarchitecture, 1998: 249-258.
- [12] FAGIN B, RUSSELL K. Partial resolution in branch target buffers[C]// Proceedings of International Symposium on Microarchitecture, 1995: 193-198.
- [13] KOBAYASHI R, YAMADA Y, ANDO H, et al. A Cost-Effective Branch Target Buffer with a Two-Level Table Organization [C]// Proceedings of International Symposium of Low-Power and High-Speed Chips, 1999: 285-285.
- [14] KAEI D R, EMMA P G. Branch history table prediction of moving target branches due to subroutine returns[C]// Proceedings of International Symposium on Computer Architecture, 1991: 34-42.
- [15] JOAO J A, MUTLU O, KIM H, et al. Improving the performance of object-oriented languages with dynamic predication of indirect jumps[C]// Proceedings of International Conference on Architectural Support for Programming Languages and Operating, 2008: 80-85.
- [16] SEZNEC A, FELIX V, KRISHNAN V, et al. Design tradeoffs for the Alpha EV8 conditional branch predictor [C] // Proceedings of International Symposium on Computer Architecture, 2002: 295-306.
- [17] WANG G P, HU X D, YIN F, et al. Research and Design of Hash Indexing Mechanism for BTB[J]. Journal of Computer Research & Development, 2014, 51(9): 2003-2011. (in Chinese)
王国澎, 胡向东, 尹飞, 等. BTB 索引散列算法的研究与设计 [J]. 计算机研究与发展, 2014, 51(9): 2003-2011.
- [18] ORAILOGLU A, PETROV P. Low-Power Data Memory Communication for Application-Specific Embedded Processors[C]// International Symposium on System Synthesis, 2002: 219-224.
- [19] PARIKH D, SKADRON K, ZHANG Y, et al. Power-Aware Branch Prediction; Characterization and Design[J]. IEEE Transactions on Computers, 2004, 53(2): 168-186.
- [20] SMITH J E, GOODMAN J R. A Study of Instruction Cache Organizations and Replacement Policies[J]. Acm Sigarch Computer Architecture News, 1983, 11(3): 132-137.
- [21] KAYNAK C, GROT B, FALSAFI B. Confluence: unified instruction supply for scale-out servers[C]// International Symposium on Microarchitecture, ACM, 2015: 166-177.
- [22] DALLY W J, BALFOUR J, BLACK-SHAFFER D, et al. Efficient Embedded Computing[J]. Computer, 2008, 41(7): 27-32.