第 33 卷 第 3 期 计 算 机 工 程 2007 年 2 月
Vol.33 No.3 Computer Engineering February 2007
·博士论文· 文章编号：1000—3428(2007)03—0021—03 文献标识码：A 中图分类号：TP311

基于面向路径的遗传算法的测试用例自动生成

金 虎[1,2]，李志蜀[1]，张 磊[1]，李宝林[1]，李勇军[1]
(1. 四川大学计算机学院，成都 610064；2. 成都信息工程学院计算机系，成都 610041)

摘 要：采用遗传算法用于自动生成测试用例的设计，算法面向路径测试作了下列工作：(1)以控制路径表 CFDPATH_T 为基础分析测试用例与执行路径的关系；(2)设计面向路径的遗传算法实现测试用例的自动生成，证明了该方法能实现测试用例空间上对路径的等价类划分；(3)从统计角度对该方法的错误检测能力进行了量化分析；(4)在该算法的基础上设计了试验和结果分析，结果表示该算法较随机方法有更好的性能。

关键词：自动软件测试；遗传算法；自适应；测试用例

Test Cases Automatic Generation Based on Path-oriented Algorithm Generation

JIN Hu[1,2], LI Zhishu[1], ZHANG Lei[1], LI Baolin[1], LI Yongjun[1]
(1. School of Computer, Sichuan Univ., Chengdu 610064; 2. Dept. of Computer Sci., Chengdu Univ. of Info. Tech., Chengdu 610041)

【Abstract】 Applying GA in automatic test cases generation, followed is the main work: (1)Using control flow path table to analyze the relation between test cases and the executing path; (2)Implementing the GA for automatic test cases generating which is path-oriented, and proving this method can accomplish the equivalent-class subdividing in all the test cases set; (3)In the point of statistics to analyze the error-detecting capability of this method; (4)Doing experiment for validating, which shows a far better performance than a random way.

【Key words】 Automatic software test; Genetic algorithm; Self-adaptive; Test case

测试用例的生成是软件测试的重要内容，程序在测试用例上执行，以发现软件中存在的缺陷是种基本的测试方法。传统的人工构造测试用例方式工作量大、测试周期长，且易于出现测试遗漏。测试用例自动构造技术可以较好弥补这些问题。随机测试用例自动生成方法由于实现简单曾被大量使用，它在程序的输入域上随机生成测试用例，这种方式容易受数据分布的影响，导致大量测试冗余。为此，启发式学习算法被进一步研究、融合遗传算法的测试用例自动生成技术是一个研究的热点。

遗传算法[1]是模拟生物进化的优化算法，由Holland在 20 世纪 60 年代提出。将遗传算法应用于测试技术，大多是对特定测试目标进行优化，这些方法较随机测试方式普遍有更好的性能。其中D.Berndt[2]、RP.Pargas[3]等对遗传算法用于测试用例自动生成进行了详细研究，但仍有可改进之处。(1)算法以单一优化为目标，侧重一次性优化计算。而测试是连续和反复的系统过程，动态连续优化计算更利于提高测试性能。(2)传统研究主要采用覆盖率指标的经验方式分析错误检测能力，量化分析做得相对较少。

1 本文的主要工作

采用了面向路径的测试用例生成方法，应用遗传算法对测试用例空间进行等价划分，通过进化计算指导测试用例生成。(1)以控制流路径表 CFDPATH_T 为基础分析测试用例与执行路径的关系；(2)使用遗传算法设计测试用例的自动生成，根据触发路径对测试用例空间进行等价划分；(3)将算法每次执行结果作为样本，用统计方法对算法的错误检测能力进行分析；(4)设计试验并对结果进行分析。主要创新为使用遗传算法对测试用例空间进行等价划分，根据适应度动态调整测试用例生成以减少测试冗余。

2 遗传算法用于测试用例生成

2.1 遗传算法

遗传算法[1]使用生物学中物种进化的基本概念来描述进化计算，运用自适应搜索技术来解决优化问题。遗传算法对求解问题进行编码，表现成染色体形式，染色体决定了生物的性状，有时也被称为个体。染色体中每一位代表一个基因。若干个个体组成的集合形成一代群体，群体的个体数表示要求解问题的规模。群体进行自适应进化，进化通过个体间的繁衍完成，主要有交叉和突变两种方式。交叉即选择群体中的两个个体x_1、x_2，用它们染色体上的部分基因位进行互换，用以产生两个不同于x_1、x_2的新个体。变异则针对种群中的个体，改变其染色体上部分基因位来产生新个体。交叉和变异体现了群体产生新个体的过程。选择操作则为了产生优于父辈的新个体，实现群体的进化。用个体对环境的适应度表示个体的性能，个体适应度越大竞争能力就越强，其染色体中存在优良的基因。选择只使用具有高适应度的个体进行遗传，这样就增大了产生优良子代的几率。进化是反复迭代的过程，进化最后具有最高适应度的个体包含了问题的解。

2.2 程序控制路径

结构化程序设计中，程序由一系列指令节点组成。程序

基金项目：四川省重点科技攻关基金资助项目（05GG021）
作者简介：金 虎(1974－)，男，博士生，主研方向：软件工程，网络与信息系统；李志蜀，教授、博导；张 磊、李宝林，博士生；李勇军，讲师
收稿日期：2006-08-07 **E-mail：**cn_jh@cuit.edu.cn

的控制流图[5]定义了指令执行路径，其中节点是程序语句，有向边表示所连接两个节点之间的控制流。

定义 1 路径P：设结构化程序G，指令节点为$s_i \in$ Node(G)，若存在有序序列<s_0,s_1,...,s_n>，并且s_{i+1}节点可以在s_i节点后立即执行，则该序列组成程序G的一条路径。

定义 2 触发路径：设测试用例为T，路径P=<s_{i0},s_{i1},..., s_{in}>是程序在T作用下，指令节点执行的有序序列，则称路径P为测试用例T的触发路径，记为TP(T,P)。

定义 3 触发路径测试用例集：设触发路径为 P，若所有能产生该执行路径的测试用例组成集合 T，则称为触发路径测试用例集 T(P)。

程序的控制流路径表 CFDPATH_T 是执行路径的参照基础，测试用例产生执行路径为触发路径 TP，若正常执行则 TP∈CFDPATH_T，否则表示该路径为执行异常；不可达路径则为不能触发的控制流路径。不同测试用例如果触发相同执行路径，则为该路径的测试冗余。

定理 1 设所有触发路径的测试用例集合T(P)形成被测试程序输入域空间上的一个划分，T(P_i)集合中的元素是面向触发路径Path(x)的等价关系。

证明 对于$\forall x, y, z \in T(Pi)$，且$x \neq y \neq z$

(1)∵ Path(x)=Path(x)，∴$x \equiv$ Path(x)(满足自反性)

(2)当 Path(x)=Path(y)时，则 Path(y)=Path(x)(满足对称性)

(3)当 Path(x)=Path(y)，Path(y)=Path(z)时，则 Path(x)= Path(z) (满足传递性)

综上所述，对输入域空间的划分是面向测试用例触发路径的等价类划分，同类测试用例触发相同的执行路径。

定义 4 触发路径测试用例的测试贡献度：设单条触发路径 P 的测试用例集为 T(P)，集合中元素的个数为 N，N≥0, 则该集合中元素对此触发路径 P 的测试贡献度为

$$C=\begin{cases}1/N, & N>0\\ 0, & N=0\end{cases}$$

同类测试用例对测试的贡献度相等，该类测试用例集合的大小为测试冗余，越大则这类测试用例越易出现，单个测试用例发现程序错误的能力下降，不利发现新的路径错误。

2.3 遗传算法生成测试用例

(1)第一代种群为随机方式产生。以路径为目标设计初始权值时，触发路径集初始化为 CFDPATH_T。假定集合中每条触发路径的执行几率相同，则有相等的初始权值$\frac{1}{n(P)}$，n(P)为路径数。而实际测试用例空间的划分并不均匀，不同路径的测试用例集大小通常也不相同。

(2)当前测试用例集 T 作为父辈种群，驱动程序执行后，记录触发路径 P。第一代随机生成的测试用例受输入域分布影响，生成较多相似或相同的易执行的触发路径。合并同类测试用例，生成当前触发路径的测试用例集。

(3)根据单个测试用例的贡献度重新计算适应度。同类测试用例集的所有测试用例具有相同的测试贡献度$C_i=\frac{1}{n(P)*N(T_i)}$，适应度$F_i$=$C_i$。计算是自适应的，易于触发的路径对应的测试用例集增大，其测试贡献度下降，适应度也随之下降。相反，测试用例个数较少的集合具有较高适应度，算法将奖励这类测试用例的生成。

(4)选择算子采用适应度比例的概率方法，根据个体在群体中的适应度作为概率进行选择。具有最大适应度的个体被保留，并以所占比例复制后进行交叉和突变操作。$F=\sum_{i=0}^{n} c_i$，$i \in$触发路径且$T(P_i)>0$。

(5)交叉算子采用单点交叉，可能破坏种群中的高适应度模式，影响全局搜索性能。变异算子用于提高局部搜索的能力，当 P ∉ CFDPATH_T 时，需对导致错误的测试用例作更多的局部搜索。

(6)算法的停止条件是：在所有触发路径的测试用例集合中，非空集合中最大适应度低于某个阈值时算法停止。阈值保证了最难出现的触发路径的测试用例生成个数达到一定的数量。

算法 GAGenTestCase：

输入：被测试程序 G,被测试程序的 *CFDPATH_T*

输出：程序 G 的触发路径测试用例集合 T

符号：程序触发路径表 CFDPATH_T，触发路径集 TP(G)，算法停止阈值 v

步骤：

(1) Generating CurPopulation randomly; Copy CFDPATH_T to TP(G);

(2) Add CurPopulation to T;

(3) For each $P_j \in$ TP(G) do compute the Fitness of T(P_j);

(4) For each CurPopulation ∈ T do {

Using T_i in CurPopulation to run G to record TP(T_i, P_j); Add T_i to T(P_j); }

(5) For each P_j do recomputed the Fitness of T(P_j);

(6) Compute Fit Ratio to select the fittest T(P);

(7) Crossover and Mutation to generate NewPopulation;

Copy NewPopulation to CurPopulation;

(8) If (MAX(Fit(T_i)) < <v) then go to step (2)

Return T

算法的时间复杂度主要由(2)到(8)中的循环决定。步骤(3)计算时间为程序的触发路径数 $n(P)$；(4)为种群数量(S)×程序执行时间(E)；步骤(5)与步骤(3)的时间相似；(6)对当前种群的适应度进行一遍选择排序；(7)为在种群数量(S)上的交叉 C 和突变 M 操作耗费的时间。循环体内单次执行时间 $T(N)=O(S\times(E+C+M))$，整个算法的时间为 $T(N)$×进化代数，进化代数受阈值影响。

2.4 测试用例的错误检测能力

遗传算法产生一组测试用例集称为一次测试，则可看作是测试过程的一个样本，给出下述符号进行描述。设算法在第i次测试中生成的触发路径集为$P_i=P_{i1}\cup P_{i2}\cup\ldots\cup P_{in}$，$P_{ij}\cap P_{ik}=\Phi$，触发路径测试用例集为$T_i=T(P_{i1})\cup T(P_{i2})\cup\ldots\cup T(P_{in})$，$T(P_{ij})\cap T(P_{ik})=\Phi$。使用$P=P_1\cup P_2\cup\ldots\cup P_n$来表示整个过程中的所有触发路径集，则测试用例集形成程序输入域空间上的一个划分。把$T(P_i)$简记为T_i，则$\sum T_i=T$，$T_i\cap T_j=\Phi$，$i\neq j$。

定义5 错误密度 设$N(T_i)$表示测试用例子域T_i的域空间中元素个数，$n(T_i)$表示测试用例子域的实际测试用例生成个数，$m(T_i)$表示在该测试子域上导致出现程序错误的元素个数，则在测试用例子域T_i上的错误密度可表示为$\frac{m(Ti)}{n(Ti)}$。

遗传算法自动生成测试用例方法的错误检测能力可表示为$FD(G)=1-\prod_{i=1}^{n(P)}(1-\frac{m(i)}{N(i)})^{n(i)}$，$N(P)$表示触发路径集 P 中元素的个数。

定理 2 判定满足性定理 设测试过程要求满足未暴露的程序执行错误低于特定的标准，采用遗传算法的测试用例自动生成方法能够以统计方法检测此测试要求是否满足。

证明 设程序中各个测试子域的错误密度$E_i=\frac{m(i)}{n(i)}$，最大错误密度可表示为$Emax$，$E_i \le Emax$。将$Emax$带入上式中可得:$Emax$=1-(1-FD)$^{\frac{1}{n\times n(P)}}$。若要$Emax \le \alpha$，则需满足1-(1-$FD$)$^{\frac{1}{n\times n(P)}} \le \alpha$，推得$FD \ge 1-(1-\alpha)^{n\times n(P)} = \beta$。$FD$是关于实际错误检测个数$m(i)$与用例个数$n(i)$的统计量，在给定显著性水平条件下假设$FD \ge \beta$成立，使用假设检验可对$FD$的样本数据进行分析，以判断测试是否能够满足测试要求。

3 试验结果分析

三角形分类问题是测试中的典型示例，问题描述为：接收了个整数 A、B 和 C 输入为三角形的边，根据边的关系输出三角形类型。试验首先解决对解空间的编码，设定三角形边的范围为 A、B、C=[1，500]，可直接使用二进制编码方式。例如 A=256，B=200，C=300，编码后为 0100000000 011001000 100101100。这种编码方式并非完全编码，例如区间(500，512)范围，但这正好可以触发程序中的输入域错误。表 1 是种群数量为 100，阈值为 0.000 6 时，分别使用遗传算法和随机生成方法的试验结果。

表 1 遗传算法和随机测试用例自动生成方式的比较

	CFDPATH	RUNS	Test Case Num.				
			RAN ERR	INVALID	TRI	ISO	EQU
GGEN	5	65	401	2 533	2 596	613	349
RGEN		547	12	27 871	26 550	187	1

表中 RAN ERR，INVALID，TRI，ISO，EQU 分别表示触发输入域限制错误，非三角形，一般三角形，等腰以及等边三角形各路径的测试用例集。随机方式满足 CFDPATH 被覆盖时，运行了 547 次，对于输入域上分布较窄区域生成的测试用例个数很少，例如 RAN.ERR.以及 EQU。遗传算法方式则能在这些路径上生成较多的测试用例，使得对该路径的测试充分。

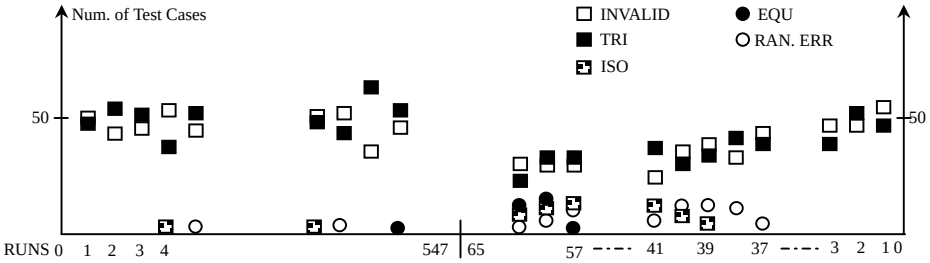

图 1 进化过程中的不同测试用例数量的分布

图 1 表示随机方法和遗传算法在进化过程中不同的触发路径测试用例集大小的变化。图左为随机方法，每轮测试用例的生成符合独立同分布，触发路径测试用例集的分布与输入变量空间数据分布相似。遗传算法采用了适应度进行优化，有好的测试用例生成时，其性状能较好得到遗传，在随后连续进化的子代中，仍能保证该类测试用例有较大的产生概率，如图右 37 代处产生的 RAN.ERR.。

可见，采用遗传算法实现测试用例的自动生成，比随机方式能更好地避免程序输入变量域分布特性的影响，促进小几率测试用例生成，极大提高测试有效性。

4 结语

本文使用遗传算法实现测试用例的自动生成，根据程序执行时的触发路径，将生成的测试用例划分为不同的等价类，并根据其适应度动态调整测试用例的数量。算法既能为每条执行路径生成一定数量的测试用例以保证测试充分性，又能降低冗余测试用例的生成，较之随机方法的测试用例生成有明显的改进。

参考文献

1 Holland J. Adaptation in Natural and Artificial Systems[M]. Michigan: University of Michigan Press, 1975.

2 Berndt D, Fisher J, Johnson L. Breeding Software Test Cases with Genetic Algorithms[C]//Proceedings of the 36th Hawaii International Conference on System Sciences, 2003.

3 Pargas R P, Harrold M J. Test-data Generation Using Genetic Algorithms[J]. The Journal of Software Testing, Verification and Reliability, 1999, 9(4): 263-282.

4 Yan Pingfan, Zhang Changshui. Artificial Neural Networks and Evolutionary Computing[M]. Peking: Tsinghua University Press, 2005.

5 Jorgensen P C. 软件测试[M]. 第 2 版. 韩 和, 杜旭淘, 译. 北京: 机械工业出版社, 2003.

（上接第 3 页）

2 Guarino N. Formal Ontology and Information Systems[C]// Proceedings of FOIS'98, Trento, Italy. Amsterdam: IOS Press, 1998: 3-15.

3 Uschold M, King M. Towards A Methodology for Building Ontologies[C]//Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995-08.

4 Grüninger M, Fox M S. The Logic of Enterprise Modeling[M]// Brown J, O'Sullivan D. Reengineering the Enterprise. Chapman & Hall, 1995: 83-98.

5 Gómez-Pérez A. Knowledge Sharing and Reuse[M]//Liebowitz J. The Handbook of Applied Expert Systems. CRC Press, 1998.

6 Vipul K. Design and Creation of Ontologies for Environmental Information Retrieval[C]//Proc. of the 12th International Conf. on Knowledge Acquisition, Modeling and Management, Banff, Canada, 1999-10.

7 Wang Hongwei, Jiang Fu, Wu Jiachun. Extended Ontology Model and Ontology Checking Based on Description Logics[J]. 上海交通大学学报(英文版), 2004, E-9(1): 34-41.

8 Wang Hongwei, Jiang Fu, Wu Jiachun. Study on Formal Ontology Model: Constructing Customer Ontology in CRM Context[C]//Proc. of the 9th Americas Conference on Information System, Tampa, Florida, USA, 2003-08-04.