第 38 卷 第 4 期
Vol.38 No.4

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

·人工智能及识别技术· 文章编号：1000—3428(2012)04—0158—04 文献标识码：A 中图分类号：TP311.5

基于改进遗传算法的面向路径测试数据生成

王 林，尤 枫，赵瑞莲

(北京化工大学计算机系，北京 100029)

摘 要：在遗传算法中，面向路径测试数据自动生成存在迭代次数多、效率低的问题。为此，提出一种改进型的遗传算法。通过分析被测源程序得到其结构信息，并利用该结构信息，控制遗传算法中交叉、变异操作发生的位置及范围，提高遗传操作的精确性和目的性。实验结果表明，与传统遗传算法相比，该算法具有更快的收敛速度，测试数据生成效率更高。

关键词：遗传算法；面向路径；测试数据生成；程序结构信息；分支表达式；交叉；变异

Path-oriented Test Data Generation Based on Improved Genetic Algorithm

WANG Lin, YOU Feng, ZHAO Rui-lian

(Dept. of Computer Science, Beijing University of Chemical Technology, Beijing 100029, China)

【Abstract】For the problem that Genetic Algorithm(GA) suffers from large iteration times and low efficiency in path-oriented test data generation, this paper proposes a Modified Genetic Algorithm(MGA), through analyzing the source code, structural information is gained and used to control the crossover and mutation point and range in order to make the genetic operation more accurate and purposeful. Experimental result shows that MGA has faster convergence speed and higher test data generation efficiency compared with traditional genetic algorithm.

【Key words】Genetic Algorithm(GA); path-oriented; test data generation; program structural information; branch expression; crossover; mutation

DOI: 10.3969/j.issn.1000-3428.2012.04.051

1 概述

随着软件规模的日益庞大，软件测试的重要性显得愈发明显。根据统计，在完整的软件开发过程中，测试所占的工作量已经达到了软件开发全部工作量的 40%以上。特别对于一些安全关键类软件，测试过程的花费高达开发花费的 3~5 倍[1]。在软件测试过程中，测试数据生成是最重要的一环，如何在给定的测试准则下自动生成测试数据是一个既具有理论意义又具有实用价值的研究方向。

面向程序路径的测试数据生成可描述为：对于被测程序 P，设其输入向量空间为 S，给定一条路径 pa，寻找一个输入向量 $\boldsymbol{x}(x \in S)$，使得程序 P 在以 $\boldsymbol{x}$ 为输入时执行路径 pa。现有面向程序路径测试数据生成方法可以分为随机法、静态法和试探性算法[2]。随机法通过随机方式生成测试数据，其优点是简单易行，但当被测程序的输入空间较大时，随机法通常效率低下。静态法只对被测程序进行静态分析，并不运行被测程序，典型的方式是符号执行法[3]。符号执行法以符号代替数值作为被测程序的输入，收集路径中的约束集，通过对约束集进行求解[4]来达到生成测试数据的目的。当被测程序较简单时，符号执行具有较好的执行效果。但当存在非线性约束时，其效率会急剧下降，甚至出现求解失败的情况。试探性算法[2]是指从程序的输入空间中随机选择输入，运行被测程序，根据执行结果并结合概率论的思想产生新的输入继续执行被测程序，直至生成满足要求的测试数据。试探性算法主要包括遗传算法和模拟退火算法[5-6]。其中遗传算法是一种模拟自然界生物进化规律(优胜劣汰、适者生存)的算法。最初由美国 Michigan 大学 Holland J[7]教授于 1975 年首先提出。在求解复杂非线性问题时，遗传算法采用概率化的搜索方式，具有全局搜索优化能力强，能自学习、自适应的特点。故其在软件测试数据自动生成中具有广泛的应用[8-9]。大量研究表明，遗传算法在面向路径的测试数据自动生成中取得了较好的效果，但也存在如下问题：遗传算法在进行交叉、变异操作时，交叉、变异点可能选取在个体基因中的任意位置，交叉、变异操作产生优良基因的同时也存在破坏已有优良基因的可能。这导致了算法生成覆盖指定路径的测试数据时，迭代次数较多，算法效率较低。

本文提出了一种改进型的遗传算法(Modified Genetic Algorithm, MGA)，通过利用被测程序的结构信息来确定个体交叉、变异操作发生的位置，约束交叉、变异操作的范围。

2 基于 MGA 算法的测试数据生成方法

遗传算法应用于面向路径的测试数据自动生成时，适应度函数用来评价个体的优劣，即以个体作为输入时，程序实际执行路径与目标路径的逼近程度。若能得到实际执行路径上不满足目标路径的分支语句中与输入变量相关的信息，即该分支表达式的取值与哪些输入变量有关，则可有针对性地对这部分输入变量进行调整，以尽快满足目标路径对该分支表达式的取值(真/假)要求。

例如在如下程序中，$x1$，$x2$，$x3$ 是其输入变量，target 所在的分支为目标路径应当执行的分支。若要生成测试数据，即确定输入变量 $x1$，$x2$，$x3$ 的取值，使程序执行 target 部分的代码，那么如果能知道与语句④中表达式($v1>5$)相关的输入变量是 $x1$ 和 $x3$，则可针对变量 $x1$ 和 $x3$ 进行调整，尽快使

基金项目：国家自然科学基金资助项目(61073035,60903002)

作者简介：王 林(1986－)，男，硕士研究生，主研方向：软件可靠性测试；尤 枫，副教授；赵瑞莲，教授

收稿日期：2011-07-22 **E-mail：**wanglin0823@gmail.com

分支表达式($v1>5$)成立，生成满足路径要求的测试数据。

被测程序部分代码如下：

```
Procedure
{
① scanf("%d,%d,%d",&x1,&x2,&x3);
    …
②  v1=2×x1+5×x3;
③  v2=x2×x2+v1;
     //v1,v2…是程序变量
    …
④ if(v1>5)
⑤      //target
    else
    …
}
```

因此，改进型遗传算法在传统 GA 的基础上，有针对性地对个体中与不满足目标路径的分支语句相关的输入变量进行调整，即利用与分支表达式相关的输入变量信息来确定交叉、变异点的位置及其操作的边界，避免对优良基因的破坏，以提高测试生成的效率。但 MGA 算法生成测试数据前，需要识别被测程序各分支表达式中与输入变量相关的信息。

3　被测程序结构信息的识别

被测程序的结构信息主要是指程序分支点对应的判断表达式中涉及的输入变量信息。如果能识别出与分支表达式中每一个变量相关的输入变量，则可得到与整个分支表达式相关的输入变量信息。

程序变量与输入变量之间的关系可以分为以下 2 类：

(1)程序变量本身即是输入变量

输入语句中涉及的变量即为输入变量。如 C 语言 scanf 语句中涉及的变量。如被测程序部分代码中的 scanf("%d,%d,%d",&x1,&x2,&x3)语句，其变量 $x1$，$x2$，$x3$ 即为输入变量。

(2)程序变量不是输入变量，其取值同一组输入变量有关

程序变量通过赋值语句与输入变量发生关联。如被测程序部分代码中，变量 $v1$ 不是输入变量，但其通过赋值语句②与输入变量 $x1$，$x3$ 相关联。程序变量 $v2$ 通过赋值语句③与输入变量 $x2$，程序变量 $v1$ 相关，而 $v1$ 又与 $x1$，$x3$ 相关，故可知变量 $v2$ 同输入变量 $x1$，$x2$，$x3$ 相关。

对于一个被测源程序，可通过逐行扫描源代码来识别各分支表达式相关的输入变量信息。扫描识别过程如下：逐行扫描被测源程序，用 R_x 表示 x 相关的输入变量集合。若扫描到 scanf("%d",&x)语句，可以确定变量 x 是一个输入变量，通过记录该 scanf 语句的出现次序及其参数列表中参数的个数，可以确定 x 是程序的第几个输入变量。例如扫描语句①后可确定 $x1$，$x2$ 和 $x3$ 是程序的输入变量，其集合 $R_{x1}=\{x1\}$，$R_{x2}=\{x2\}$，$R_{x3}=\{x3\}$，分别表示与 $x1$，$x2$ 和 $x3$ 相关的输入变量。若扫描到赋值语句，根据赋值号右边变量的相关输入变量可计算出赋值号左边变量的相关输入变量。如程序中标号②的赋值语句，由语句①可知变量 $x1$，$x3$ 的相关输入变量集合为 $R_{x1}=\{x1\}$，$R_{x3}=\{x3\}$，则可计算出与 $v1$ 相关的输入变量集合 $R_{v1}=R_{x1}\cup R_{x3}=\{x1\}\cup\{x3\}=\{x1, x3\}$。同理可得到语句③中程序变量 $v2$ 的相关输入变量集合 $R_{v2}=R_{x2}\cup R_{v1}=\{x2\}\cup\{x1, x3\}=\{x1, x2, x3\}$。若扫描到分支语句，则根据表达式中变量的相关输入变量集合，来计算与该表达式相关的输入变量集合。如标号④处的分支表达式($v1>5$)中只涉及变量 $v1$，因此与该分支表达式相关的输入变量集合即是与变量 $v1$ 相关的输入变量集合。用 R_{Related} 表示与某一分支表达式相关的输入变量集合，则表达式($v1>5$)的 $R_{\text{Related}}=R_{v1}=\{x1, x3\}$。

通过对被测源程序的逐行扫描分析，可以构造出一个如表 1 所示的结构信息表，对于程序中的每一个分支表达式，表中都有一个唯一的表项与之对应，即与该分支表达式相关的输入变量集合 R_{Related}。

表 1　分支表达式相关的输入变量信息

分支表达式	与分支表达式相关的输入变量集合
E^1	$R_{\text{Related}}{}^1$
E^2	$R_{\text{Related}}{}^2$
…	…
E^n	$R_{\text{Related}}{}^n$

4　基于 MGA 算法的测试数据自动生成

对于一条给定程序路径，利用 MGA 算法生成测试数据的基本流程为：首先随机生成一组初始解(初始种群)，分别以种群中的个体作为输入运行被测程序，若存在满足要求的个体(解)则算法结束，否则计算个体的适应度值并对个体进行评估，从中选取一组适应度较强的个体进行交叉、变异操作以生成新的个体构成新的种群。新种群中的个体将再次作为输入运行被测程序。如此重复直至生成覆盖指定路径的测试数据或达到预定迭代次数后退出，即测试数据生成失败。

利用 MGA 算法生成面向路径的测试数据时需要解决个体编码方式、适应度函数选取、遗传策略设计等问题。

4.1　个体的编码方式

应用 MGA 算法求解具体问题时，应将被求解问题的解空间映射成适合遗传操作的长度有限的编码。因此，需要将被测软件的输入转化成某种编码表示的个体，以便对个体进行选择、交叉、变异等遗传操作。目前常用的编码方式有二进制编码和实数编码。由于被测软件的输入多为实数类型，因此本文采用实数编码方式，直接使用被测软件的输入数据作为个体。例如，如果一个被测软件有 3 个输入变量，其一组输入[21, 75, 98]即为一个实数编码的个体。

4.2　适应度函数的选取

适应度函数的设计是 MGA 算法的关键。本文采用面向路径测试数据生成中常用的适应度函数，具体计算公式如下：

$$Fitness=ApproachLevel+normalize(d)$$
$$normalize(d)=1-1.001^{-d}$$
$$d=BranchDistance$$

其中，*ApproachLevel* 值反映了以个体作为输入运行被测程序时，程序实际执行路径同目标路径的总体相符程度，可定义为程序实际执行路径中不满足目标路径的分支结点个数。*ApproachLevel* 值越小表明实际路径同目标路径越接近。*BranchDistance* 值描述了程序实际执行路径不同于目标路径的第 1 个分支结点从真(假)分支到假(真)分支的距离。*BranchDistance* 值越小表明从真(假)分支到假(真)分支的距离越小。

4.3　选择策略

MGA 算法选取父代中 K 个适应度最好的个体，并对已选取的个体，按其适应度值进行分组，同一组中的个体具有相同 *ApproachLevel* 值。由 *ApproachLevel* 值的定义可知：以同组个体作为输入运行被测程序时，程序实际执行路径将在同一分支点处偏离目标路径，它们拥有相同的“第 1 个不满足目标路径的分支结点”，即其表达式相关的输入变量集合 R_{Related} 是一样的。同组中的个体可用图 1 表示，其中，斜线部分表示需要调整的输入变量(R_{Related})，其他部分表示与分支表达式取值不相关的输入变量。

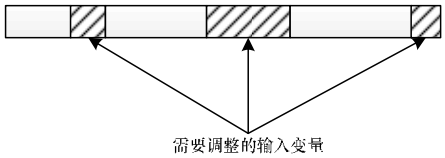


图 1 同组中的个体

4.4 交叉操作

由于同组个体在同一分支结点处偏离目标路径，它们拥有相同的相关输入变量集合。如前所述，调整这部分变量可加速测试数据的生成，因此 MGA 算法的交叉操作在同组个体之间进行。但组内个体可能有多个，也可能仅有一个。为此，MGA 算法的交叉操作分 2 种情形。

对于组内个体多于 1 个的情形，采取相邻个体两两进行交叉，具体交叉操作如图 2 所示。

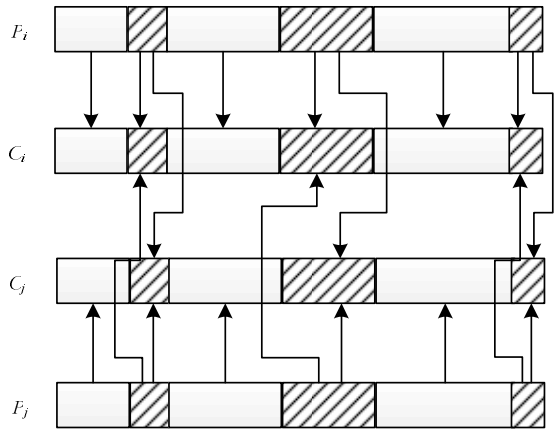


图 2 MGA 算法交叉操作

假设图 2 中 P_i，P_j 是同组中的 2 个父代个体，C_i，C_j 是交叉操作生成的子代个体。为避免交叉操作生成优良基因的同时破坏已有的优良基因，MGA 算法只对斜线区域的输入变量进行多点交叉操作，非斜线区域输入变量的取值将在子代中予以保留，即：子代个体中属于斜线区域的输入变量随机选取父代个体 P_i 或 P_j 中对应输入变量的取值，对于非斜线区域的输入变量，C_i 将完整保留父代个体 P_i 中对应输入变量的取值，C_j 将完整保留父代个体 P_j 中对应输入变量的取值。

对于由单一个体组成的组，由于无法进行通常意义上的两两交叉操作，因此采取复制个体的方式来模拟交叉操作(等同于自己同自己交叉)。

4.5 变异操作

变异操作模拟了自然界中的生物突变现象。为不破坏已有优良基因，MGA 算法只针对个体中需要修改的输入变量(斜线区域)进行变异操作。具体操作为：对交叉操作生成的子代个体进行多点变异，即随机对每个需要修改的输入变量进行增减。

5 实验结果与分析

为检验 MGA 算法在测试数据生成中的效率，本文针对一组被测程序分别利用 2 种算法进行了面向路径的测试数据生成实验，并从算法的收敛速度、执行时间两方面对传统 GA 算法和 MGA 算法进行了比较。实验结果表明，MGA 算法较 GA 算法具有更快的收敛速度，测试数据生成效率更高。

5.1 被测程序

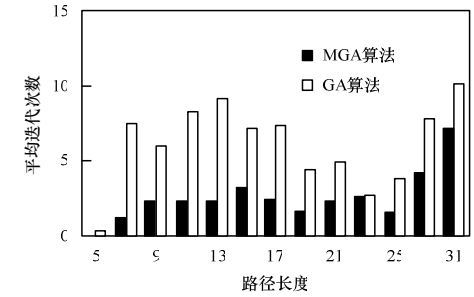
为提高实验有效性，避免因被测程序的同质化而削弱实验结论的普遍性，本文选择了功能及代码结构差异显著的若干程序作为实验对象，表 2 给出了被测程序的一组相关属性值，如代码行数、独立路径数(独立路径是指至少引入了一个新语句的路径)、输入变量个数、最长路径长度等。

表 2 被测程序相关信息

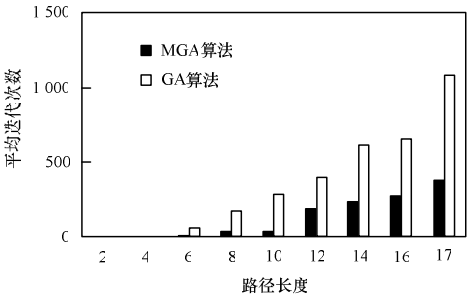
程序属性被测程序	代码行数	独立路径数	输入变量个数	最长路径长度
Next Date 程序	233	32	3	31
密码锁程序	56	9	8	17
车辆贷款程序	173	288	5	23
存款利率程序	584	96	4	19

5.2 基于 MGA 算法和 GA 算法的测试生成结果及分析

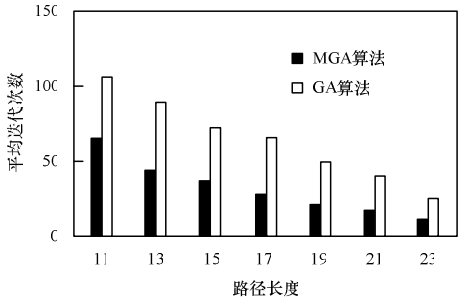
算法迭代次数对比如图 3 所示。



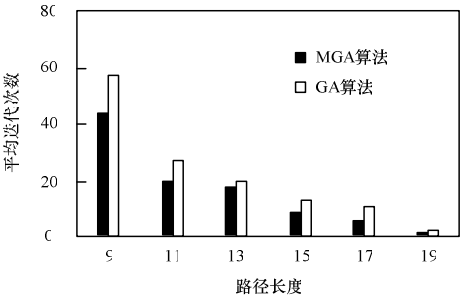
(a)Next Date 程序



(b)密码锁程序



(c)车辆贷款程序



(d)存款利率程序

图 3 算法迭代次数对比

针对上述 4 个被测程序，利用 MGA 和 GA 算法生成面向路径的测试数据时，种群数量设置为 20，每代选取 10 个

最优个体进行交叉、变异操作，对于每条路径分别利用 2 种算法生成 10 次测试数据。实验中将统计算法所用的迭代次数以及生成测试数据所消耗的时间。不同长度的路径，其测试生成所需的迭代次数不同，4 个被测程序路径长度与迭代次数的关系见图 3，纵坐标为针对长度相同的一组路径各生成 10 次测试数据时算法所需的平均迭代次数。从中可以看出，对于相同长度路径生成测试数据，MGA 算法需更少的平均迭代次数。

为进一步分析比较 MGA 算法和 GA 算法在测试生成时的效率，本文以密码锁程序中一条最长且最复杂的路径为例，即如图 4 所示程序控制流图中的一条路径：0→1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→16，分析 2 种算法适应度函数值的变化。分别利用 MGA 算法和 GA 算法生成测试数据，并每隔 10 代记录 2 种算法最优个体适应度值的变化趋势，如图 5 所示。

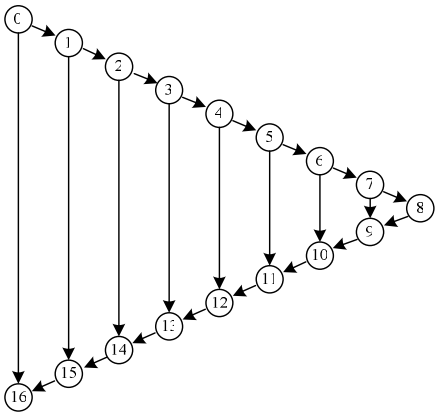


图 4　密码锁程序控制流图

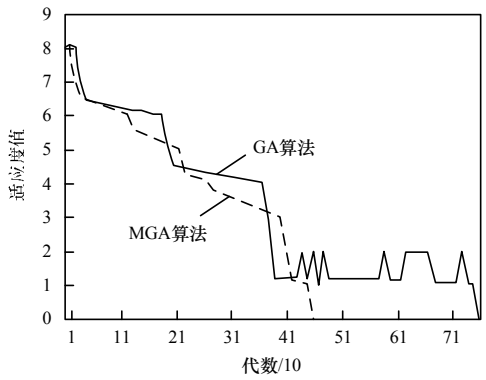


图 5　适应度值变化趋势对比

从图 5 中可以看出 GA 算法在迭代过程中，适应度值并非随迭代次数的增加单调递减。当迭代次数较大时，适应度值反复波动，从而导致 GA 算法的测试生成效率较低。MGA 算法在测试生成过程中，适应度值随迭代次数的增加单调递减，具有更快的收敛速度。

此外，生成测试数据所需的时间也是衡量算法效率的一个重要参数。针对上述每个被测程序，算法生成测试数据所需的时间消耗如图 6 所示。

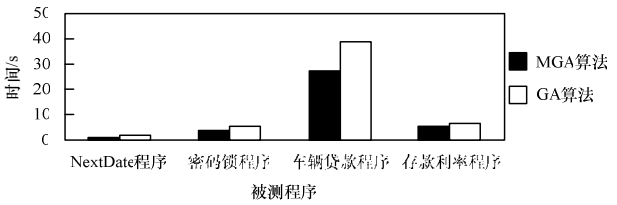


图 6　算法消耗时间对比

图 6 中纵坐标表示对于被测程序的每条路径均生成 10 次测试数据时，算法消耗的总时间。从中可以看出，MGA 算法在生成测试数据时需要更少的时间消耗，具有更好的执行效率。

6　结束语

本文提出了一种利用被测程序结构信息来指导交叉、变异操作的改进型遗传算法，并将其应用到面向路径的测试数据生成当中。实验数据表明，本文提出的 MGA 算法同传统 GA 算法相比降低了迭代次数，缩短了生成测试数据所需的时间，提高了测试数据生成的效率。在下一步工作中，将考察程序分支表达式中出现复合数据类型变量的情况，以扩大 MGA 算法的适用范围。

参考文献

[1] 齐治昌, 谭庆平, 宁　洪. 软件工程[M]. 北京: 高等教育出版社, 2001.

[2] 单锦辉, 王　戟, 齐治昌. 面向路径的测试数据自动生成方法述评[J]. 电子学报, 2004, 32(1): 109-113.

[3] James C. King. Symbolic Execution and Program Testing[J]. Communications of the ACM, 1976, 19(7): 385-394.

[4] Chen Jifeng, Zhu Li, Shen Junyi. An Approach on Automatic Test Data Generation with Predicate Constraint Solving Technique[J]. International Journal of Information Technology, 2006, 12(3): 132-141.

[5] Tracey N, Clark J, Mander K. Automated Program Flaw Finding Using Simulated Annealing[C]//Proc. of International Symposium on Software Testing and Analysis. New York, USA: [s. n.], 1998: 231-251.

[6] 郭　巍, 桂小林. 模拟退火多亲遗传数据生成算法研究[J]. 计算机工程, 2010, 36(11): 69-72.

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

[8] Lin J C, Yeh P L. Automatic Test Data Generation for Path Testing Using GAs[J]. Information Sciences, 2001, 131(1-4): 47-64.

[9] 李　军, 李艳辉, 彭存银. 基于自适应遗传算法的路径测试数据生成[J]. 计算机工程, 2009, 35(2): 203-205.

编辑　索书志

(上接第 157 页)

参考文献

[1] 张祖勋, 张剑清. 数字摄影测量学[M]. 武汉: 武汉测绘科技大学出版, 1997.

[2] 张永生, 巩丹超, 刘　军, 等. 高分辨率遥感卫星应用: 成像模型、处理算法及应用技术[M]. 北京: 科学出版社, 2004.

[3] Lowe D G. Distinctive Image Features from Scale-invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 91-110.

[4] 刘焕敏, 王　华, 段慧芬. 一种改进的 SIFT 双向匹配算法[J]. 兵工自动化, 2009, 28(6): 89-91.

[5] 周志强, 汪　渤, 吕　冀. 不同分辨率图像的角点匹配方法[J]. 北京理工大学学报, 2008, 28(7): 598-601.

[6] 于丽莉, 戴　青. 一种改进的 SIFT 特征匹配算法[J]. 计算机工程, 2011, 37(2): 210-212.

编辑　顾逸斐