基于遗传算法的测试用例生成技术①

宋 倩

(江苏自动化研究所, 连云港 222006)

摘 要: 在计算机软件的生命周期中, 由于各种各样的原因, 留给软件测试的时间往往不够执行全面而充分的测试. 覆盖率和数量作为测试用例质量的主要影响因子, 一直受到人们的关注. 然而测试用例的执行顺序作为影响测试效率的重要因素, 却一直未引起足够的重视. 提出了一种基于遗传算法的测试用例生成技术, 综合考虑覆盖率、数量和用例的执行顺序三个因子, 生成高效的测试用例集. 使测试人员能够按照既定的顺序执行用例集, 在最短的时间内, 完成最重要的软件功能测试.

关键词: 软件测试; 覆盖率; 遗传算法; 用例执行顺序; 用例生成技术

Test Case Generation Based on Genetic Algorithm

SONG Qian

(Jiangsu Automation Research Institute, Lianyungang 222006, China)

Abstract: In the actual life cycle of software, due to various reasons, the time that spending on software testing is often not enough to perform the comprehensive test. Coverage ratio and the number of test cases as the main quality factors, have been payed more attention to. However, the execution order of test case as one of the important factors that affect test efficiency, has not caused enough attention. This paper presents a test case generation technology based on genetic algorithm, which considering Coverage ratio, the number of test cases and the execution order. Testing personnel can execute test case in the established order and accomplish the most important software function test in the time as shorter as possible according to the method referred in this paper.

Key words: software testing; coverage; genetic algorithm; exection sequence of test case; case generation technology

随着计算机技术的普及以及在一些重要领域的应用, 软件测试逐渐受到重视, 但在实际的产品生命周期中, 由于各种各样的原因, 留给软件测试的时间往往不够执行全面而充分的测试. 高覆盖率和低数量的测试用例可以有效地减少测试时间, 提高测试效率, 如何设计高效的测试用例, 使得软件测试工作在有限的时间内执行最充分的测试是现阶段软件测试工作的难点[1]. 本文综合考虑了测试用例的覆盖率、数量以及执行顺序三方面的因素, 采用遗传算法, 对相关算子和运行参数进行改进, 利用遗传算法的特点设计了一种新的测试用例生成模型, 以达到进一步提高测试效率的目的.

1 基于遗传算法的测试用例生成

1.1 测试用例生成模型

本文将遗传算法用作核心算法来进行测试用例自动生成的研究, 首先要建立测试用例自动生成模型, 并对影响求解效率的相关算子进行研究设计. 围绕遗传算法的主要设计思想为: 根据相关被测软件的文档确定被测软件的运行状态因素的初始化状态可用编码; 对各状态之间的转换所需时间进行代价估算; 种群规模确定; 适应度函数选择.

具体应用到测试用例的自动生成按照以下步骤进行:

1) 分析被测软件的需求规格说明, 提取其所有的输入数据和工作条件;

① 收稿时间:2014-04-21;收到修改稿时间:2014-05-16

2) 将输入数据按照等价类划分的方法进行分类, 并将工作条件按照枚举的方法列出;

3) 根据输入和工作条件确定软件的初始工作状态;

4) 计算初始种群中每一条染色的适应度值, 从而确定种群的优劣度;

运用初始的测试用例集合执行遗传操作, 获得相应的适应度值, 如果遗传代数达到要求, 则转向步骤 7), 否则转向步骤 6);

根据种群的优劣度, 执行改进后的遗传算法包, 生成新的测试用例集合, 并转向步骤 5);

运行结束, 输出合适的测试用例.

根据上述步骤, 将遗传算法的测试用例自动生成模型分为三部分: 测试环境的构造, 根据对软件相关文档和具体软件执行情况的分析, 确定一个稳定的被测试软件的初始运行状态; 遗传算法包, 按照前面步骤生成初始的种群, 通过对种群中的染色体进行反复的评价、运算, 引导种群向目标值靠近, 找到最终合适的解或者达到预期限定的迭代次数; 测试仿真运行环境, 对种群的优劣进行评价, 返回一个适应度值供核心算法包使用[2].

1.2 模型算子设计

1.2.1 染色体编码方案和初始种群构造

遗传算法的编码方法可以分为三大类: 最小字符集编码方法、浮点数编码方法、符号编码方法. 本文选择用最小字符集编码方法[3].

1.2.2 适应度函数设计

假设某系统软件各状态之间的切换代价表如表 2 所示. 一条染色体基因序列为 i 的编码为(0,0,0,0,0,0,0,0), 基因序列为 i+1 的编码为(1,1,1,1,1,1,1,1), 按照表 1 中所列时间代价值 i 和 i+1 之间进行切换的时间值为: Ti=t1+ t2+ t31+ t4+ t5+ t6+ t71+ t81. 则该染色体的适度函数值为:

表 1 时间代价表

	工作模式	系统时间	运行状态			系统当前坐标	当前方位	目标距离	目标属性						目标种类		
时间代价	t1	t2	0-1	0-2	1-2	t4	t5	t6	0-1	0-2	0-3	1-2	1-3	2-3	0-1	0-2	1-2
			t31	t32	t33				t71	t72	t73	t74	t75	t76	t81	t82	t83

$$P(T)=1000/\sum_{i=1}^{i=n-1} Ti \quad (1)$$

其中 n 为一条染色体的基因数量.

1.2.3 遗传操作算子设计

1)选择

采用适应度比例和最佳个体保存相结合的方法进行选择: ①根据式(1)计算各个个体的适应度比例值将适应度比例值最高的个体保存到缓冲区中; ②根据选择概率 Ps 选择 $P_i > P_s$ 的个体进入交配池作为待交叉的个体. 其中, Ps 为种群规模 N 的倒数; ③设 P(t+1)为经过交叉和变异操作形成的新种群, 用缓冲区中的个体和子代种群 P(t+1)中的个体一一进行比较. 若存在相同的, 则选择操作结束. 若不存在则转入下一步. ④用缓冲区中的个体去替代种群 P(t+1)中适应度函数值最低的个体, 构成新的规模为 N 的群体 P(t+1).

2)交叉

为了保证交叉操作的正确性, 对染色体中存在互为逆序基因的个体进行交叉, 把互为逆序的基因对在进行交叉的个体中进行互换. 假设染色体 i 的基因数量为 6, 父体 P1 和 P2 中存在互为逆序的基因. P1 和 P2 的交叉过程如图 1 所示.

3)变异

依据个体编码表示方法的不同, 采用逆转变异算子. 具体操作步骤如下: ①产生一组随机数, 随机数的数目为种群规模的 0.1 倍, 随机数的范围为[0, N-1]

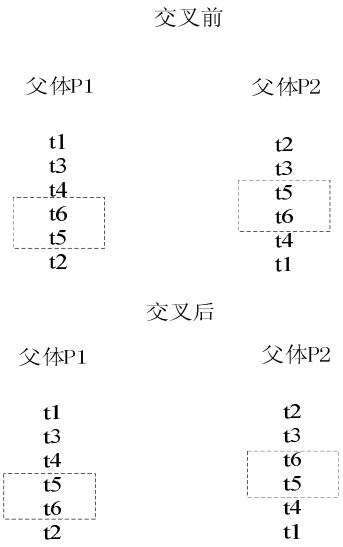

图 1　交叉操作示意图

(N 为种群规模), 随机数的值即为产生变异的个体编号. ②对于每一个被选中的个体, 随机地选取两个基因座的位置 i, j 然后将 i, j 之间的基因进行逆转. 变异的操作过程如图 2 所示.

图 2　变异过程示意图

1.2.4 运行参数设计

1)种群大小 N

将群体规模设计成一个公式(2)所示的分段函数, n 为全部软件状态的逻辑组合数. 文中软件的 n=2*2*3*2*2*2*4*3=1152, 其中 N<n!.

$$N=\begin{cases} n, n \le 20 \\ 20, 20 < n \le 400 \\ \text{转换}\ n, 400 < n \le 10000 \\ 100, n > 10000 \end{cases} \quad (2)$$

2)选择概率

选择概率为适应度比例的平均值, 高于平均值的个体进入交配池, 保证每次都有个体被选中, 同时被选中的个体不至于很多[4].

3)交叉概率

本文摒弃交叉概率的概念, 设计一种符合具体情况的交叉规则, 将交配池中的个体放入一个二维数组中, 设此二维数组一共有 n 行, 则行下标之和等于 n 的两行进行交叉操作.

4)变异概率

本文的变异概率 Pm 也是由一组随机数决定的, 首先根据种群的规模 N 产生$\left\lceil \frac{N}{10} \right\rceil$个范围在闭区间[0, N-1]中的随机数置于数组 a 中, $\left\lceil \frac{N}{10} \right\rceil$组范围在[0, n-1]中的随机数置于数组 b 中. 其中 a[0]=k, b[0]={i,j}, 且 i<j, 表示要进行变异的第一条染色体为当前群体中下标为 k 的染色体, 变异部分为编号为 i 和 j 之间的所有基因, 随后将基因编号 i 和 j 之间的所有基因逆序排列.

1.2.5 算法终止条件

由于实际应用遗传算法时不允许让其无停止地进行搜索, 同时问题的最优解也通常未知, 因此必须设计一些近似收敛准则来终止算法进程[4]. 即给定一个最大进化步数为终止条件, 如公式(3)所示, 这里最大进化步数和种群的初始规模是相等的.

$$t(n)=N=\begin{cases} n, n \le 20 \\ 20, 20 < n \le 400 \\ \text{转换}\ n, 400 < n \le 10000 \\ 100, n > 10000 \end{cases} \quad (3)$$

2　实验与分析

本文采用算法模型生成测试用例, 在覆盖率相同的情况下, 对比传统测试过程的时间代价和基于遗传算法生成的测试用例的时间代价, 验证生成的测试用例是否提高了测试效率.

传统测试过程的时间代价取经验值的平均值, 单位为人分, 即当 1 人执行该测试过程时, 执行完该测试所需的分钟数. 为兼顾实验的普遍性, 分别选取了软件状态因素为 6、10、20、50、100 和 1000 时进行实验分析. 两者的时间代价对比结果如表 2 所示, 当软件状态因素越多时, 基于遗传算法生成的测试用例在执行过程中节省的时间代价越明显.

表 2 时间代价对比表

	6	10	20	50	100	1000
传统测试	8134	42144	90678	114689	389076	5876442
基于遗传算法生成的用例	7818	39848	80074	98769	289048	3979889
时间代价缩减	316(3.9%)	2296(5.4%)	10604(11.7%)	15920(13.9%)	100028(25.7%)	1896553(32.3%)

3 结语

本文从提高测试效率的角度提出一种基于遗传算法的测试用例生成模型，并对相关算子和算法运行参数进行设计. 与传统的测试用例生成算法相比，本文的算法不仅在保证覆盖率的前提下使生成的测试用例数量尽可能地少，而且兼顾到优化测试用例的执行顺序，从而进一步缩短测试执行过程的时间，提高测试效率.

参考文献

1 李欣.基于贝叶斯网络和遗传算法的测试用例生成模型[硕士学位论文].重庆:重庆交通大学,2012.
2 刘瑞. 基于改进 PSO 算法的测试用例生成方法研究[硕士学位论文].郑州:河南大学,2011.
3 费雯悦.基于改进遗传算法的测试用例自动生成及质量评价研究[硕士学位论文].北京:北京化工大学,2012.
4 李柱.用于测试用例生成的遗传算法改进[硕士学位论文].成都:西南大学,2011.