



基于组合混沌遗传算法的最小测试用例集生成

申情, 蒋云良, 沈张果, 楼俊钢

(湖州师范学院信息工程学院, 浙江 湖州 313000)

摘要: 最小测试用例集生成是软件测试的重要研究领域之一。将具有均匀分布特性的 Chebyshev 和 Logistic 混沌映射相结合的混沌序列引入遗传算法的选择、交叉和变异操作, 并在遗传测试用例选择方法中添加混沌扰动, 实现全局最优, 以解决遗传算法用于测试用例集约简时局部搜索能力弱、易早熟收敛等问题。在随机生成的测试用例需求对应关系及 Siemens 测试套件等实例上进行了实验研究, 并与现有的经典方法在测试用例集生成规模和算法执行时间上进行了比较, 实验结果表明, 在保持算法执行时间的基础上, 在遗传测试用例方法中引入混沌映射有助于生成规模更小的测试用例集。

关键词: 软件测试; 测试用例最小化; 混沌遗传算法; 测试用例

中图分类号: TP311

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2016178

Test case minimizing based on combination chaos genetic algorithm

SHEN Qing, JIANG Yunliang, SHEN Zhangguo, LOU Jungang

School of Information Engineering, Huzhou University, Huzhou 313000, China

Abstract: Test case minimizing is one of the most important research fields in software testing. Uniformly distributed Chebyshev and Logistic chaos sequence were introduced in the selection, crossover and mutation of genetic algorithm. Chaos disturbance was also added in genetic testing suite to address the common problems of weak ability in local search and premature convergence, thus to optimize the test result. Experiments were conducted in randomly generated test suites and Siemens test suites. Comparisons were also made with classical methods regard to the scale of production of test suite and the execution time of the algorithms. The results of the experiment indicate that based on the same execution time of the algorithms, a smaller scale test suite can be produced by introducing chaotic sequence in genetic testing suite selection.

Key words: software testing, test cases minimizing, chaos genetic algorithm, test case

收稿日期: 2016-02-23; 修回日期: 2016-06-14

通信作者: 蒋云良, jyl@zjhu.edu.cn

基金项目: 国家自然科学基金资助项目 (No. 61103051); 浙江省自然科学基金资助项目 (No. LY15F020018); 浙江省公益性技术应用研究计划基金资助项目 (No. 2015C33247); 湖州市科技计划基金资助项目 (No. 2014GZ02)

Foundation Items: The National Natural Science Foundation of China (No. 61103051), Natural Science Foundation of Zhejiang Province (No. LY15F020018), Zhejiang Provincial Science and Technology Plan of China (No. 2015C33247), Science and Technology Program of Huzhou City (No. 2014GZ02)



1 引言

计算机系统已被应用到社会生活各方面,软件形式更加复杂多样,规模越来越大。在这样的背景下,保证软件质量,使其准确完成预期目标变得尤为重要。软件测试是软件质量保证中必不可少的一部分,为满足一定测试需求覆盖率,生成的测试用例数目往往异常庞大;另一方面,软件系统开发过程的迭代需要频繁地进行回归测试,测试冗余严重。为提高测试效率、降低测试成本,减少测试用例的执行、管理与维护的开销,测试用例集的约简是极为必要的^[1-3]。

若用 $R=\{r_1, r_2, \dots, r_k\}$ 表示测试需求集, $T=\{t_1, t_2, \dots, t_n\}$ 表示测试用例集, (其中, n 代表测试用例数量, k 代表需求数量), 可用一个 $n \times k$ 矩阵 $g_{[n][k]}$ 来表示测试用例 t_i 与测试需求 r_j 的覆盖关系为:

$$g = \begin{bmatrix} g_{[1][1]} & g_{[1][2]} & \cdots & g_{[1][k]} \\ g_{[2][1]} & g_{[2][2]} & \cdots & g_{[2][k]} \\ \vdots & \vdots & \ddots & \vdots \\ g_{[n][1]} & g_{[n][2]} & \cdots & g_{[n][k]} \end{bmatrix} \quad (1)$$

其中, $g_{[i][j]}$ 的取值为 0 或 1, $g_{[i][j]}=1$ 则表示用例 t_i 可以覆盖需求 r_j , $g_{[i][j]}=0$ 表示用例 t_i 不能覆盖需求 r_j 。测试用例最小化生成问题可以描述如下, 给出一个测试用例集 T 、一个测试用例需求集 R , R 满足给定程序要求的某种测试覆盖准则, T 可以完成对 R 中所有 r_i 的测试, 要求从 T 中找到一个测试用例优化集, 可以满足所有 r_i 的测试, 需要在大量的测试用例中筛选出最优的测试用例, 即能用最少的测试用例覆盖最多的测试需求^[1]。

现有的最小测试用例生成方法主要包括启发式方法, 如 Harrold 和 Gupta 使用的贪心算法和启发式算法^[4], Chen 提出的 GE 和 GRE 算法^[5,6]; 智能搜索方法, 如邢颖^[7]采用的分支限界搜索以及爬山法, Windisch^[8]、陈翔等人^[9]、王令赛^[10]、史娇娇^[11]采用的粒子群优化算法, Harman、Jones、Lin、Berndt 以及 Michael 使用的遗传算法^[12-18]等; 其他方法还包括整数规划算法^[19]、扩张集算法^[20,21]、有限状态机^[22,23]等。

其中, 遗传算法(genetic algorithm, GA)由于其良好的全局搜索能力, 在测试用例最小化生成中取得了较好的应用, 该算法借鉴达尔文的遗传选择和自然淘汰的生物进化理论完成对测试用例集约简。遗传算法最早由美国 Michigan 大学^[24]的 Holland J 教授提出, 是一种概率性全局优化方法, 其优化对象不需可导或者连续, 有较强的全局和并行性, 能自动识别搜索过程中的有益信息以指导优化

过程, 作为一种高效、并行、全局搜索寻优的方法, 具有很强的顽健性与适应性, 能解决传统优化方法难以解决的复杂优化问题。基于遗传算法的测试用例生成主要思想如下: 首先对原始测试用例集进行编码, 在此基础上产生若干个测试用例选择方案, 构成初始化种群, 种群个体即一种可能的测试用例选择方案; 然后从种群中选出两个解, 使用交叉算子生成新的解, 利用测试覆盖率、测试运行代价等评价个体优劣; 随后不断地用新个体替代种群中适应值较差的个体, 最后得到最优的个体及最优的测试用例选择方案。

在遗传算法应用于测试用例最小化问题研究上, Ma^[14]把程序分为若干的块或者段, 用 0-1 关系表示被测程序块和测试用例之间的满足情况, 设计个体编码, 然后利用测试覆盖率与测试开销等信息构造适应度函数; Jones^[15]用实验研究表明, 遗传算法生成的测试数据比随机法小 1~2 个数量级; Lin^[16]使用广义海明距离来优化遗传测试用例方法中的适应度函数, 进一步提高方法的性能; Berndt^[17]将输入空间划分成多个区间, 根据待选输入的多种特性创建最小化函数; Michael^[13]结合覆盖表生成一个最小化函数, 采用微分遗传算法进行全局寻优; Khor^[18]将遗传算法和正规概念分析(formal concept analysis)相结合, 进行测试用例集的生成; Berndt^[17]通过模拟实验等方法分析了各种基于遗传算法的测试用例生成方法的性能。

然而普通遗传算法用于测试用例生成也存在局限性, 如接近全局最优解时搜索速度会变慢, 个体多样性减少较快导致早熟收敛等。针对此, 利用混沌理论的遍历性, 提出一种基于组合混沌遗传算法的最小测试用例集生成方法(minimized test case generation based on combination chaos genetic algorithm, TCGCG)。采用一种结合 Chebyshev 混沌映射和 Logistic 混沌映射的组合混沌序列产生方法^[25], 在种群初始化、交叉和变异的全过程中使用组合混沌序列, 利用混沌序列增加各种运算过程的随机均匀性, 以降低同种群间个体的相似度, 从而提升算法全局搜索能力; 同时结合适应度水平设计自适应的交叉概率及变异概率选取策略, 既保证适应度较高的优秀个体保留至下一代, 保证了种群的多样性, 又提高适应度较低个体的变异概率, 加快算法收敛速度, 降低时间消耗。

2 基于混沌遗传算法的测试用例集约简方法

提出的 TCGCG 方法利用组合混沌映射生成初始种

群,每个种群个体包括 n 个基因,种群个体即一种可能的测试用例选择方案,并将具有均匀分布特性的组合混沌序列引入遗传算法的选择、交叉和变异操作,有效地避免了未成熟收敛,提升算法的全局搜索能力和计算效率,混沌系统产生初始种群基因,然后对各个混沌变量附加混沌小扰动进行种群优化,通过不断进化收敛到一个最适合的个体上。从而将测试用例最小化转化为在种群中寻找基因个数最少的个体覆盖所有测试需求。

基于 TCGCG 的测试用例集最小生成方法的基本步骤如图 1 所示,主要包括采用 Chebyshev 映射和 Logistic 映射相结合生成混沌序列、利用混沌序列初始化种群、设定个体适应度函数、计算个体适应度、定义遗传算子(选择、交叉、变异)、添加混沌扰动等步骤。

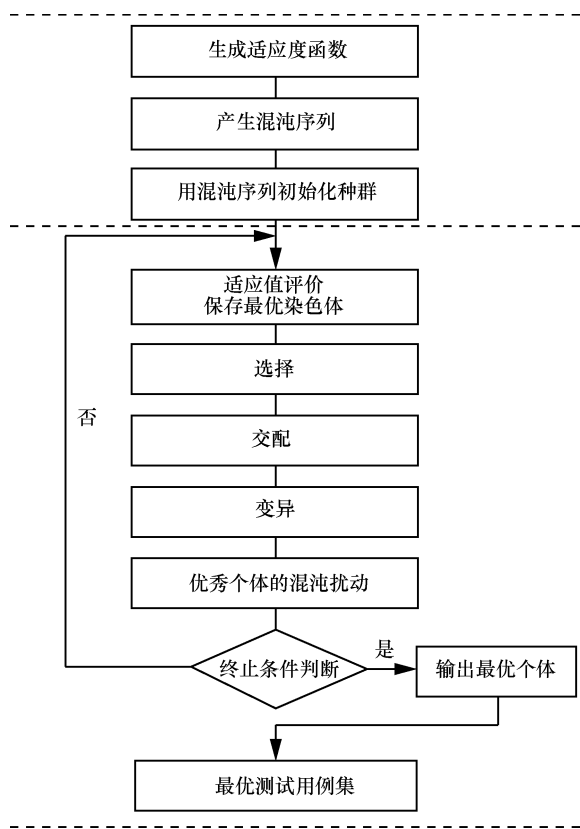


图 1 TCGCG 用于测试用例集最小化生成的基本步骤

2.1 产生组合混沌序列

Chebyshev 映射: $x_{i+1} = \cos(k \arccos x_i)$, $x_i \in [-1, 1]$, 当映射阶数 $k \geq 2$ 时,系统处于混沌状态,不同初值产生的序列间具有自相关性与零值互相关性。

Logistic 映射: $x_{i+1} = \mu \cdot x_i \cdot (1 - x_i)$, 对初值敏感,非周期,不收敛, $\mu=4$ 时处于完全混沌状态。

采用 Chebyshev 和 Logistic 混沌映射结合的生成方法: 令 $y_0=x_0$ 为初始值,使用 Chebyshev 映射 $y_{i+1} = \cos(k \arccos y_i)$ 得到一个伪随机序列 $\{y_0, y_1, y_2, \dots, y_l\}$,然后将序列取绝对值之后与 Logistic 映射求和 $temp = \mu x_i(1 - x_i) + |y_{i+1}|$, 每次迭代后采用映射 $x_{i+1} = \text{mod}(temp, 1)$ 保证混沌序列取值为 $(0, 1)$, 其中, $\text{mod}(x, 1)$ 算子代表取实数 x 的小数部分。生成混沌序列时,取 $\mu=4, k=4$,可以保证序列完全混沌。该映射将 Chebyshev 映射和 Logistic 映射中的两个变量联系起来,以父代混沌映射的结果作为子代混沌序列的种子值,提高了伪随机数分布的均匀性,并形成统一的输出序列。

2.2 利用混沌序列初始化种群

假设初始种群中个体数目为 M ,未约简前总测试用例数量为 n ,则初始种群及种群中个体可分别表示为:

$$P_0 = \{W_1, W_2, \dots, W_i, \dots, W_M\}, W = \{g_1, g_2, \dots, g_j, \dots, g_n\} \quad (2)$$

其中, P_0 表示初始种群, W 表示种群中的个体, W_i 表示初始种群中的第 i 个初始个体, g_j 表示该个体上的第 j 号染色体。采用第 2.1 节的方法由 M 个不同初值产生 M 个长度均为 n 的混沌序列,对 $W_1, W_2, \dots, W_M$ 上的染色体进行赋值,方法如下:

$$\begin{cases} g_j = 1, \text{对应序列值} \in (0.5, 1] \\ g_j = 0, \text{对应序列值} \in [0, 0.5] \end{cases} \quad (3)$$

由此完成初始种群中个体的初始化。 $g_j=1$ 表示第 j 号染色体对应的测试用例被选中, $g_j=0$ 表示未被选中,一个个体基因编码可以表示所有测试用例是否被选中。

2.3 个体适应度函数

当个体编码改变时需重新计算其适应度,使用传统适应度函数计算式,针对本文的基因编码方式将适应度计算的对象变为对个体 W_i 按式(4)进行适应度计算:

$$F(W_i) = \text{Cov}(W_i) / \text{Cost}(W_i) \quad (4)$$

其中, $\text{Cov}(W_i)$ 指个体的测试覆盖度, $\text{Cost}(W_i)$ 是个体的测试运行代价。覆盖程度 $\text{Cov}(W_i)$ 为计算父体编码 W_i 中覆盖测试需求集的个数。实验中暂不考虑不同测试需求的测试代价不同的影响,对所有的测试用例的代价都设置为 1。

2.4 定义遗传算子

遗传算子包括选择、交叉、变异 3 步,主要对个体进行遗传变异,对其进行优化,最终得到新的个体,新个体的生产可能会增加向最优解变异的机会,因此在遗传算子结束后需再对新的个体进行适应值评价,判断是否满足输出条件。



(1) 选择

采用改进的轮盘赌选择算法,利用各个体适应度的概率决定其遗留可能性。若某个体 W_i 的适应度为 $F(W_i)$,依次计算种群中每个个体的适应度,则第 i 个个体的适应度累计值 $C_i = \sum_{j=1}^i F(W_j)$, 假设需要选择 S 个个体直接遗留到下一代个体,利用第 2.1 节中的混沌映射产生长度为 S 的序列 $\{y_0, y_1, y_2, \dots, y_S\}$, 由于组合混沌映射生成的组合混沌序列在统计特性上呈现均匀分布, 序列值均匀分布在区间 $(0,1)$ 上,选择所有满足 $C_i \leq y_j \leq C_{i+1}, j=0,1,2,\dots,S$ 的 S 个个体 W_i , 利用该序列可增加选择运算的随机均匀性,有效提升算法的全局搜索能力, 另外 W_i 被选择的概率为 $P_{W_i} = \frac{F(W_i)}{\sum_{i=1}^M W_i}$, 从而可以保证适应度越大的个体遗留的可能性也越大。

(2) 交叉

对 2 个个体 $W_1=\{x_1, x_2, \dots, x_n\}, W_2=\{y_1, y_2, \dots, y_n\}$ 进行交叉操作时,如果交叉点选择得不适合,则可能出现与父代个体一样的子代个体,导致交叉操作无效。因此,针对遗传算法中交叉算子的不足,首先确认个体交叉点的有效区域,然后在该区域内随机选择交叉点,确保交叉点操作能生成新的和父代个体不同的子代。交叉点的有效区域如下:

$$\begin{aligned} A_{\min} &= \min\{alk_m \neq k_{j0}, a \in 1, 2, \dots, m\}, \\ A_{\max} &= \max\{alk_m \neq k_{j0}, a \in 1, 2, \dots, m\} \end{aligned} \quad (5)$$

交叉点有效区域为 $(A_{\min}, A_{\max})$, 例如 2 个个体 $X=(110101), Y=(111010)$, 其交叉点有效区域为 $(3,6)$ 。

在交叉与变异操作中,交叉概率和变异概率的选取对提高搜索性能尤为重要,交叉概率和变异概率越大,则产生新个体的速度就越快,但若选取过大,可能会影响算法中一些重要的数学特性和搜索能力。对于适应度较高的个体,为使其顺利进入下一代,应该给予比较低的概率,而对于适应度较低的个体则刚好相反,因此采用 Srinivas^[20]提出的个体适应度自适应调整办法,具体如式(6)、式(7)所示。

$$P_{\text{cross}} = \begin{cases} k_1 \frac{2F_{\max} - F(W_i) - F(W_j)}{F_{\max} - F_{\text{avg}}}, & F(W_i) - F(W_j) > 2F_{\text{avg}} \\ k_2, & F(W_i) - F(W_j) \leq 2F_{\text{avg}} \end{cases} \quad (6)$$

$$P_{\text{mutation}} = \begin{cases} k_3 \frac{F_{\max} - F}{F_{\max} - F_{\text{avg}}}, & F > 2F_{\text{avg}} \\ k_4, & F \leq F_{\text{avg}} \end{cases} \quad (7)$$

式(6)中, P_{cross} 为参与交叉运算的两个个体的交叉概率, $F_{\max}$ 为群体中适应度的最大值, $F(W_i)$ 与 $F(W_j)$ 分别为交叉的两个个体的适应度值, F_{avg} 为群体内所有个体的适应度均值;式(7)中, P_{mutation} 为个体的变异概率, F 为参加变异运算个体的适应度值, k_1, k_2, k_3, k_4 为常量系数。采用自适应的交叉概率及变异概率选取策略,既保证了适应度较高的优秀个体保留至下一代,保证了种群的多样性,又提高了适应度较低个体的变异概率,加快了算法的收敛速度。

2.5 添加混沌扰动

对当前种群中适应度后 90% 的个体,利用混沌系统,对其进行一定程度的微小扰动,从而提高其适应度。将选中的个体所指代的二进制的每一位都加一混沌扰动,按式(8)进行添加,然后按式(9)映射为优化变量,进行迭代计算。

$$W_k' = (1-\beta) \cdot W^* + \beta \cdot W_k, 1 \leq k \leq n \quad (8)$$

$$W_k'' = c_i + d_i \cdot W_k' \quad (9)$$

其中, W^* 为当前最优个体, W_k 为当前个体, k 为迭代次数,在式(9)中, c_i, d_i 为变换常数。通过式(8)可得到一组新个体:

$$W_k' = (g_{k1}', g_{k2}', \dots, g_{kn}'), 0 < \beta < 1 \quad (10)$$

对于式(8)中 β 的取值,按照式(7)进行自适应选取:

$$\beta = \begin{cases} \exp\left(-\frac{k^2}{2\sigma^2}\right), & 2 \leq k \leq 22 \\ 1 - \left(\frac{k-1}{k}\right)^2, & k > 22 \end{cases} \quad (11)$$

在搜索阶段的初期,希望 W_k 取值变动较大,随着搜索逐渐接近最优解,需将 β 逐渐缩小,以保证在小范围内找到搜索最优解。

由于 $g_{ki}'' \in \{0,1\}, i=1,2,\dots,n$, 因此需将式(9)得到的 g_{ki}'' 通过式(11)变换到相应的取值:

$$g_{ki}'' = \begin{cases} 1, & g_{ki}'' > 0.5 \\ 0, & \text{其他} \end{cases}, i=1,2,\dots,n \quad (12)$$

3 实验分析

为验证提出的基于混沌遗传算法测试用例约简方法的有效性,开发相关的测试用例约简工具进行实验研究,比较了 TCGCG 方法与一些其他的遗传算法用于测试用例集约简方法在随机生成的测试用例需求对应关系及 Siemens 测试套件等实例上的测试用例集规模和算法执行时间。实验所用计算机处理器配置为 Intel Core i7-4600U@2.10 GHz 2.70 GHz,内存为 4 GB,操作系统为 Windows 7

旗舰版 64 位。实验中设定最大迭代次数为 1 000,若迭代 10 代后无明显改进则退出迭代,种群中个体数目 $M=50$, $k_1=k_3=1,k_2=k_4=0.5$ 。

3.1 用于比较的遗传测试用例生成算法

实验中用于比较的方法包括:基于一般遗传算法的测试用例生成方法 (test case generation based on genetic algorithm,TCGG)^[17];在 TCGG 基础上的对变异算子进行优化后的改进遗传算法测试用例最小生成方法(modified test case generation based on genetic algorithm,MTCGG),对变异算子优化采用 Srinivas 提出的个体适应度自适应调整办法,如式(6)、式(7)所示。

3.2 实验一

通过随机函数生成 0-1 矩阵,表示测试用例与测试需求之间的关系,矩阵的规模分 10 组,分别为 20×20、40×40、60×50、60×60、80×80、90×80、100×100、150×100、200×150 以及 300×200,每组中包含 30 个矩阵,矩阵内各点通过随机的方式取值为“1”或者“0”,并且确保每个需求至少被 5 个测试用例覆盖,每个矩阵各不相同,从而保证实验随机性。对每组对应关系分别运行 TCGG、MTCGG 以及 TCGCG,并记录每个矩阵约简花费时间(单位为 ms)以及约简后的大小,各种情况下每组 30 个矩阵的平均值见表 1。表 1 中小括号中的百分比表示总覆盖率 $\left(\sum_{i=1}^{30} \frac{\text{第 } i \text{ 个被覆盖的需求数}}{\text{需求数} \times 30}\right)$ 未达到 100%。

表 1 中列出了在每组数据上使用 TCGG、MTCGG 以及 TCGCG 3 种方法约简后平均最小测试用例数、平均花费时间。从表 1 中可以看出,除了使用 TCGG 方法的 100×100

矩阵,MTCGG 方法的 60×60、100×100 矩阵,TCGCG 方法的 60×60 矩阵,其余情况下使用 3 种方法都能得到 100%的覆盖率,说明遗传算法具有不错的搜索能力。10组实验中,在平均最小用例数指标上,TCGCG 方法在其中 7组表现最优,包括 20×20(并列)、40×40、80×80、90×80、150×100、200×150、300×200;而 MTCGG 方法有 3 组最优,分别是 60×50、60×60 和 100×100;而采用 TCGG 方法只有在 20×20(并列)矩阵上表现最优。在花费时间指标上,TCGCG 方法在其中 8 组的表现要优于 TCGG 方法和 MTCGG 方法。

图 2 中列出了在不同矩阵规模下使用 TCGG、MTCGG 以及 TCGCG 3 种方法约简后平均最小测试用例数的比较。从图 2 中可以看出,大部分情况下使用 TCGCG、MTCGG 方法优化得到的最小测试用例集数量要少于 TCGG。说明通过优化变异算子有助于提高遗传算法的搜索效率,提高寻优能力。

图 3 中列出了在不同矩阵规模下使用 TCGG、MTCGG 以及 TCGCG 3 种方法约简后平均花费时间的比较。从图 3 可以看出,矩阵规模较小的时候,TCGCG 与 MTCGG 方法可以得到相近的平均最小测试用例数量以及平均花费时间,然而随着矩阵规模变大,TCGCG 方法的优势越来越明显。

3.3 实验二

选用 space 程序以及 Siemens 提供的 5 个 C 程序和相关的测试套件进行约简算法的比较^[27,28],每个程序均设计了大量测试用例并保证每个需求至少被 30 个测试用例覆盖,表 2 中给出了每个程序的总需求数(包括分支、定义引用对)以及对应的可执行测试用例数。

表 1 不同方法所花费时间及平均最优用例大小比较

序号	矩阵规模 (测试需求数×用例数)	TCGG		MTCGG		TCGCG	
		花费时间/ms	平均最优用例/个	花费时间/ms	平均最优用例/个	花费时间/ms	平均最优用例/个
1	20×20	326	3.7	292	3.8	334	3.7
2	40×40	1 001	4.63	877	4.73	629	4.33
3	60×50	1 573	5.37	956	5.23	922	5.27
4	60×60	1 856	5.30	1 033	5.03(99.3%)	1 042	5.13(99.9%)
5	80×80	3 397	6.13	2 899	5.67	2 375	5.57
6	90×80	4 229	6.23	3 784	6.07	3 651	5.87
7	100×100	5 424	6.57(99.9%)	4 616	5.97(99.8%)	4 096	6.20
8	150×100	8 743	7.23	6 341	6.67	5 722	6.27
9	200×150	23 262	6.87	20 453	6.87	12 569	6.23
10	300×200	46 723	7.53	33 274	7.37	21 415	6.63

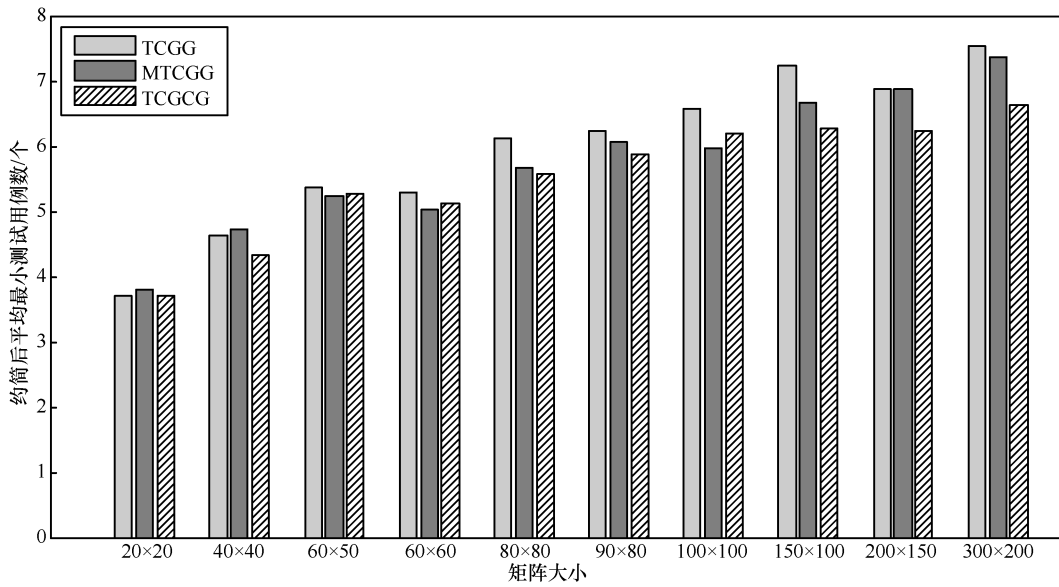


图2 不同矩阵规模下3种方法最小测试用例数比较

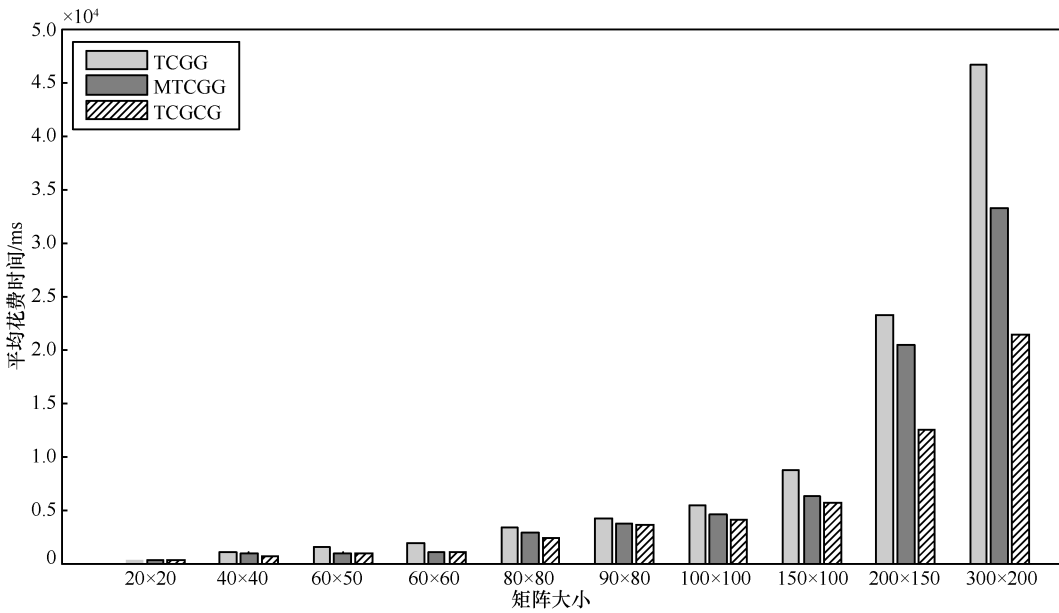


图3 不同矩阵规模下3种方法平均花费时间比较

表2 实验评测程序

被测程序	描述	可执行测试用例数(约简前)		总需求数	
		分支覆盖	定义—引用覆盖	分支	定义—引用对
space	解释器	533	3 585	1 356	5 179
tcas	航空器碰撞系统	46	57	41	51
print tokens	词法分析器	97	268	127	275
print tokens2	词法分析器	159	240	154	235
schedule	优先级调度器	62	294	84	148
replace	模式匹配及替换	191	664	155	759
totinfo	数据统计系统	83	292	83	287

为确保随机性,实验设计如下:设待测程序测试用例集为 $\{t_1, t_2, \dots, t_N\}$,选择数量分别为 N_1, N_2, N_3, N_4, N_5 的测试用例进行约简,在不同的初始值 $y_{j0}, j=1, 2, \dots, 20$ 的情况下,采用第4.1节的方法可产生20组不同的混沌序列 $\{y_{j0}, y_{j1}, y_{j2}, \dots, y_{jL_i}\}, j=1, 2, \dots, 20$ ($\lceil \cdot \rceil$ 表示向上取整运算),其中 L_i 表示每个序列的长度,计算方式为 $L_i = \lceil 0.6 \times N_i \rceil, i=1, 2, 3, 4, 5$,接着在 $\{t_1, t_2, \dots, t_N\}$ 中选取编号分别为 $\lceil N \times y_{j0} \rceil, \lceil N \times y_{j1} \rceil, \lceil N \times y_{j2} \rceil, \dots, \lceil N \times y_{jL_i} \rceil$ 的测试用例,剩余的 $N_i - L_i$ 个测试用例选取要保证所有测试需求能被100%覆盖。实验中各待测程序在分支覆盖

及定义引用对覆盖时的 N 值以及 N_1, N_2, N_3, N_4, N_5 值的选择见表3。

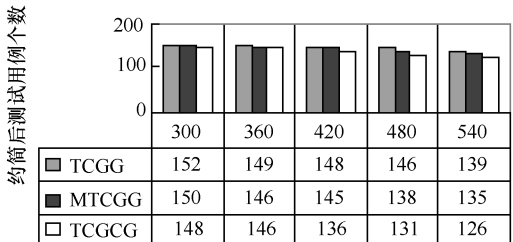
图4中给出了各待测程序在不同初始测试用例集数量情况下20组测试用例集的平均约简结果,从图4中可以看出,无论是分支覆盖还是定义—引用覆盖,在大部分情况下,本文提出的TCGCG方法要优于TCGG方法和MTCGG方法,可以得到更少的测试用例精简集合。

3.4 实验结果讨论

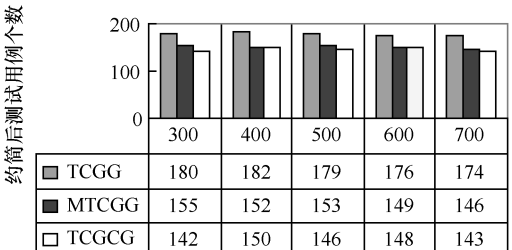
从两个实验中可以看出,TCGCG方法和MTCGG方法

表3 实验评测程序选取的测试用例数量情况

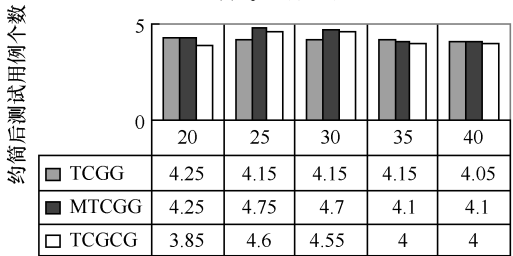
被测程序	N		N_1, N_2, N_3, N_4, N_5	
	分支覆盖	定义—引用覆盖	分支	定义—引用对
space	874	3 585	300,360,420,480,540	300,400,500,600,700
tcas	46	57	20,25,30,35,40	20,25,30,35,40
print tokens	97	268	30,40,50,60,70	60,70,80,90,100
print tokens2	159	240	40,50,60,70,80	40,50,60,70,80
schedule	62	294	40,50,60,70,80	60,70,80,90,100
replace	191	664	40,50,60,70,80	60,70,80,90,100
totinfo	83	292	20,30,40,50,60	30,40,50,60,70



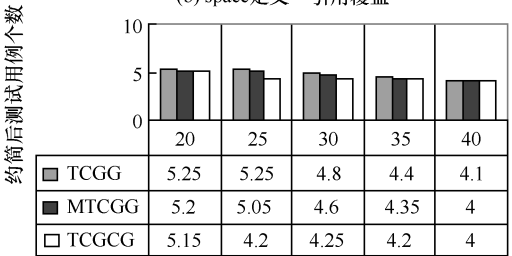
(a) space分支覆盖



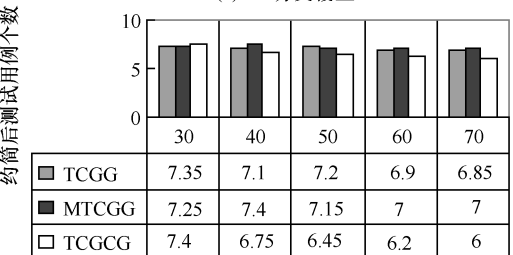
(b) space定义—引用覆盖



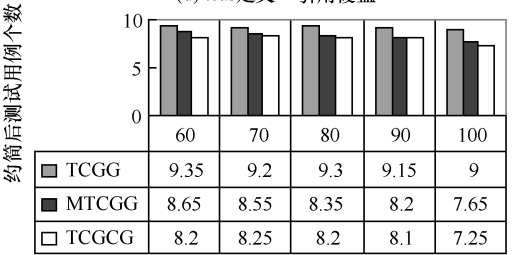
(c) tcas分支覆盖



(d) tcas定义—引用覆盖



(e) print tokens分支覆盖



(f) print tokens定义—引用覆盖

图4 各待测程序不同初始测试用例集数量情况下的平均约简结果

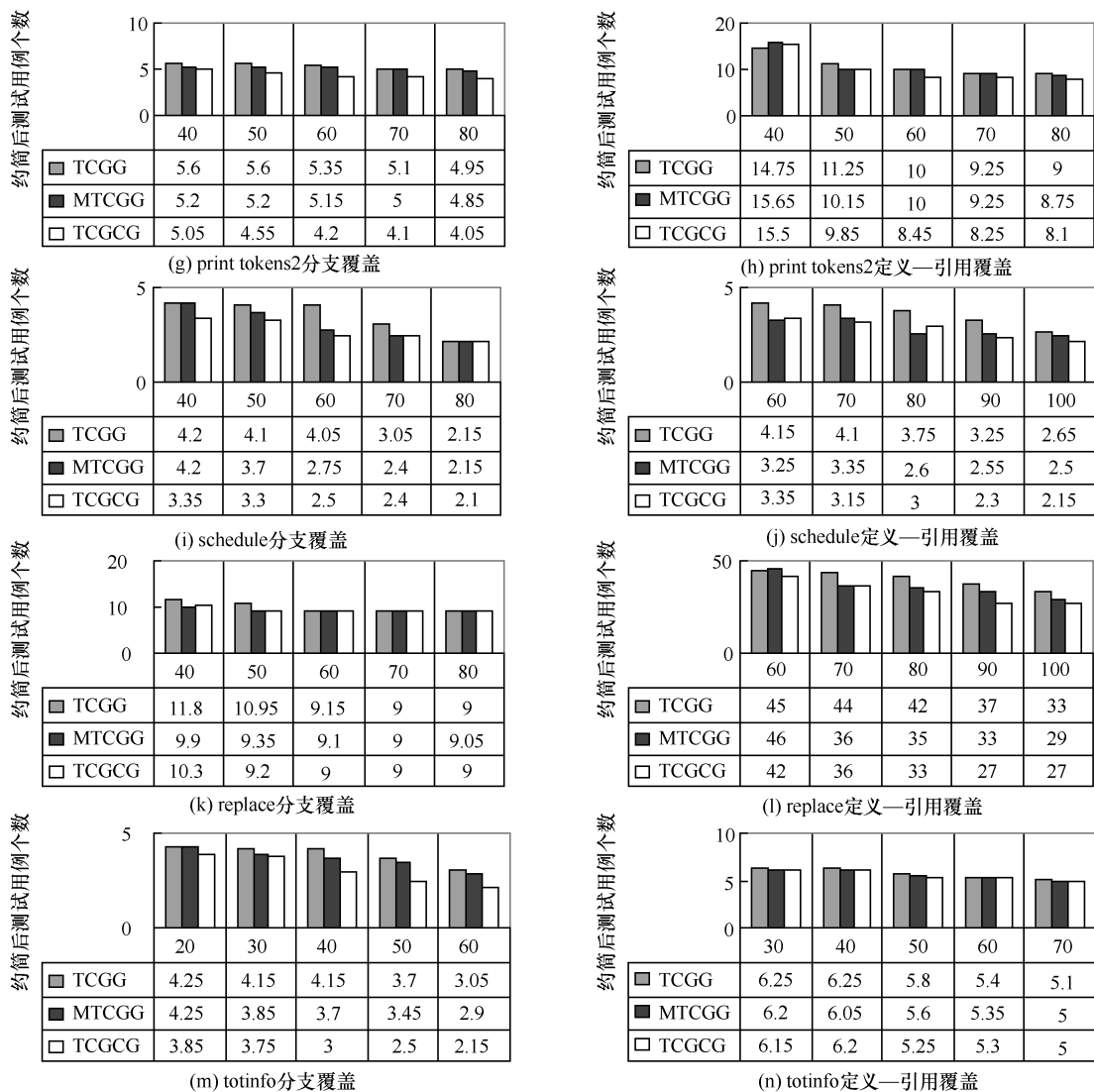


图4 各待测程序不同初始测试用例集数量情况下的平均约简结果

相比 TCGG 方法,可以在更短时间内获得最好的约简效果。这主要是因为它们通过采用自适应的交叉概率及变异概率选取策略,保证了适应度较高的优秀个体保留至下一代,从而提高了适应度较低个体的变异概率,加快算法的收敛速度;另外一方面,通过将 Chebyshev 和 Logistic 混沌映射结合的混沌序列引入基于遗传算法的测试用例生成方法的选择、交叉和变异操作,使得种群构造更具均匀分布特性,降低了同种群间个体的相似度,可以提升算法全局搜索能力,因此提出的 TCGCG 方法比 MTCGG 方法以及 TCGG 方法在大部分情况下可以得到更好的约简效果。

4 结束语

针对测试用例约简的特点,介绍了一种基于组合混沌

遗传算法的测试用例约简方法,它将混沌机制引入遗传算法中,并且改进了交叉、变异算子,使得最终能快速搜索到全局最优解。实验证明,这种新的方法能够有效克服遗传算法的早熟收敛等缺陷,对测试用例生成效率有一定的提高。主要工作为:具有均匀分布特性的 Chebyshev 和 Logistic 混沌映射结合的混沌序列引入了基于遗传算法的测试用例生成方法的选择、交叉和变异操作;采用了一种基于个体适应度的自适应调整的交叉、变异算子,并设计了一种混沌扰动设定的自适应方法;与传统遗传算法在最小测试用例集生成规模和算法执行时间上进行了比较,以验证本文提出方法的有效性。

进一步的研究工作包括: k_1 、 k_2 、 k_3 、 k_4 等参数以及初始种群规模 M 取值对算法性能、收敛效率的影响;利用已有

的测试数据优化初始群体以提高算法收敛速度。

参考文献:

- [1] TODD L. An empirical study of regression test selection techniques [J]. *ACM Transactions on Software Engineering and Methodology*, 2001, 10(2): 185-207.
- [2] NIE C, LEUNG H. The minimal failure-causing schema of combinatorial testing [J]. *ACM Transactions on Software Engineering and Methodology*, 2011, 20(4): 1-38.
- [3] BIBLE J, ROTHERMEL G, ROSENBLUM D. A comparative study of coarse- and fine-grained safe regression test selection techniques [J]. *ACM Transactions on Software Engineering and Methodology*, 2001, 10(2): 149-183.
- [4] HARROLD M, GUPTA R, SOFFA M. A methodology for controlling the size of a test suite [J]. *ACM Transactions on Software Engineering and Methodology*, 1993, 2(3): 270-285.
- [5] CHEN T, LAU M. A new heuristic for test suite reduction [J]. *Information and Software Technology*, 1998, 40(5): 347-354.
- [6] CHEN T, LAU M. On the divide and conquer approach towards test suite reduction [J]. *Information Sciences*, 2003, 152 (1): 89-119.
- [7] 邢颖, 宫云战, 王雅文. 基于分支限界搜索框架的测试用例自动生成[J]. *中国科学: 信息科学*, 2014, 44(10): 1345-1360.
XING Y, GONG Y Z, WANG Y W. Based on branch threshold search framework of automatic generation of test cases[J]. *China Science: Information Science*, 2014, 44(10): 1345-1360.
- [8] WINDISCH A, WAPPLER S, WEGENER J. Applying particle swarm optimization to software testing [C]//The 9th Annual Conference on Genetic and Evolutionary Computation, April 4-8, 2007, London, UK. New Jersey: IEEE Press, 2007: 1121-1128.
- [9] 陈翔, 顾庆, 王子元. 一种基于粒子群优化的成对组合测试算法框架[J]. *软件学报*, 2011, 22(12): 2879-2893.
CHEN X, GU Q, WANG Z Y. A paired combination test algorithm based on particle swarm optimization framework [J]. *Journal of Software*, 2011, 22(12): 2879-2893.
- [10] 史娇娇, 姜淑娟, 韩寒. 自适应粒子群优化算法及其在测试数据生成中的应用研究[J]. *电子学报*, 2013, 41(8): 1555-1559.
SHI J J, JIANG S J, HAN H. Adaptive particle swarm optimization algorithm and its application in test data generation[J]. *Electronic Journals*, 2013, 41(8): 1555-1559.
- [11] 王令赛, 姜淑娟, 张艳梅. 基于正交搜索的粒子群优化测试用例生成方法[J]. *电子学报*, 2014, 42(12): 2345-2351.
WANG L S, JIANG S J, ZHANG Y M. Particle swarm optimization based on orthogonal search method for generating test cases[J]. *Electronic Journals*, 2014, 42(12): 2345-2351.
- [12] YOO S, HARMAN M. Regression testing minimization, selection and prioritization: a survey [J]. *Journal of Software Testing, Verification and Reliability*, 2012, 22(2): 67-120.
- [13] MICHAEL C C, MCGRAW G E, SCHATZ M A. Generating software test data by evolution[J]. *IEEE Transactions on Software Engineering*, 2001, 27(12): 1085-1110.
- [14] MA X Y, SHENG B K, YE C Q. Test-suite reduction using genetic algorithm [C]//The 6th International Workshop on Advanced Parallel Processing Technologies, April 5-10, 2005, Hong Kong, China. New Jersey: IEEE Press, 2005: 253-262.
- [15] JONES B, EYRES D. A strategy for using genetic algorithms to automate branch and fault-based testing [J]. *The Computer Journal*, 1998, 41(2): 98-107.
- [16] LIN J C, YE H P L. Using genetic algorithms for test case generation in path testing [C]//The 9th Asian Test Symposium, October 3-5, 2000, Washington D C, USA. New Jersey: IEEE Press, 2000: 241-246.
- [17] BERNDT D, FISHER J, JOHNSON L. Breeding software test cases with genetic algorithms [C]//The 36th Hawaii International Conference on System Sciences, May 5-8, 2003, Hawaii, USA. New Jersey: IEEE Press, 2003: 17-26.
- [18] KHOR S, GROGONO P. Using a genetic algorithm and formal concept analysis to generate branch coverage test data automatically [C]//The 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), IEEE Computer Society, November 6-10, 2004, Lawrence, KS, USA. New Jersey: IEEE Press, 2004: 346-349.
- [19] LEE J, CHUNG C. An optimal representative sets election method [J]. *Information and Software Technology*, 2000, 42(1): 17-25.
- [20] MARRE M, BERTOLINO A. Using spanning sets for coverage testing [J]. *IEEE Transactions on Software Engineering* 2003, 29(11): 974-984.
- [21] JEFFREY D, GUPTA N. Improving fault detection capability by selectively retaining test cases during test suite reduction [J]. *IEEE Transactions on Software Engineering*, 2007, 33 (2): 108-123.
- [22] 张涌, 钱乐秋, 王渊峰. 基于扩展有限状态机测试中测试输入数据自动选取的研究[J]. *计算机学报*, 2003 26(10): 1295-1303.
ZHANG Y, QIAN L Q, WANG Y F. Based on extended finite state machine test input data in the automatic selection of research[J]. *Journal of Computer*, 2003, 26(10): 1295-1303.
- [23] 杨瑞, 陈振宇, 张智轶. 一种基于扩展有限状态机的自动化测试用例生成方法 [J]. *中国科学: 信息科学*, 2014, 44(5): 588-609.
YANG R, CHEN Z Y, ZHANG Z Y. Based on extended finite state machine automation test case generation method [J]. *China Science: Information Science*, 2014, 44(5): 588-609.
- [24] HOLLAND J. *Adaption in natural and artificial systems* [M]. Boston: MIT Press, 2015: 126-137.



[25] 殂云霄, 周杰. 基于组合混沌遗传算法的认知无线电资源分配[J]. 物理学报, 2011, 60(7): 079501.

ZU Y X, ZHOU J. Based on the combination of chaos genetic algorithm in cognitive radio resource allocation [J]. Journal of Physics, 2011, 60(7): 079501.

[26] SRINIVAS M, PATNAIK L M. Adaptive probabilities of crossover and mutation in genetic algorithms [J]. IEEE Transactions on System, Man and Cybern, 1994, 24(4): 656-667.

[27] HARMAN M. The current state and future of search based software engineering[C]//The 29th IEEE International Conference on Software Engineering. Minneapolis, May 23-25, 2007, Minneapolis, MN, USA. New Jersey: IEEE Press, 2007: 342-357.

[28] University of Nebraska-Lincoln. Computer science& engineering [EB/OL]. (2016-02-11)[2016-03-12]. <http://www.cse.unl.edu>.

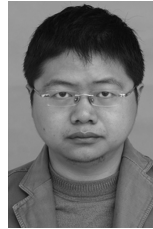
[作者简介]



申情(1982-),女,湖州师范学院讲师,主要研究方向为软件测试、人工智能、系统性能评估等。



蒋云良(1967-),男,博士,湖州师范学院教授,主要研究方向为智能信息处理、GIS等。



沈张果(1982-),男,湖州师范学院讲师,主要研究方向为大数据分析、软件可靠性建模、智能交通等。



楼俊钢(1982-),男,博士,湖州师范学院副教授,主要研究方向为软件可靠性建模、软件测试、大数据分析等。