基于镜像选择序优化的MART算法

李志博,李清宝,兰明敬,孙剑帆
(中国人民解放军战略支援部队信息工程大学,河南郑州 450001)

摘 要: 镜像自适应随机测试(Mirror Adaptive Random Testing,MART)算法将输入空间划分为多个不相交的相等子域,源域中使用自适应随机测试(Adaptive Random Testing,ART)算法生成测试数据,剩余子域中使用镜像函数生成镜像测试数据.镜像策略的引入减少了ART算法的计算开销,但是算法的检错有效性也随之降低.通过研究MART算法的特征,分析如何提升MART算法的检错有效性.针对镜像函数将源测试数据镜像到各子域时的镜像顺序,对比分析镜像选择序与镜像函数对MART算法的影响,本文提出了基于镜像受限选择序的MART算法,通过约束镜像选择序,使镜像测试数据分布更均匀.在仿真实验与实例实验结果中均显示,针对镜像策略中镜像选择序的优化,提高了MART算法的检错有效性.

关键词: 软件测试;随机测试;自适应随机测试;镜像自适应随机测试;镜像选择序
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112(2022)02-0314-12
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20201352

MART Algorithm Based on Mirror Selection Order Optimization

LI Zhi-bo, LI Qing-bao, LAN Ming-jing, SUN Jian-fan
(*PLA Strategic Support Force Information Engineering University, Zhengzhou, Henan* 450001, *China*)

Abstract: The input space is divided into several disjoint equal subdomains by mirror adaptive random testing (MART). The adaptive random testing(ART) algorithm is used to generate the test data in the source domain, and the mirror function is used to generate the mirror test data in the remaining subdomains. The introduction of mirror strategy reduces the computational cost of ART algorithm, but the effectiveness of failure detection is also reduced. By studying the characteristics of MART algorithm, this paper analyzes how to improve the effectiveness of MART algorithm. In this paper, we compare and analyze the influence of mirror selection order and mirror function on MART algorithm, and propose an algorithm based on mirror limited selection order, which makes the test data more evenly distributed by restricting the mirror selection order. The results of simulation and empirical experiments show that the optimization of the mirror selection order in the mirror strategy improves the effectiveness of the failure detection of the MART algorithm.

Key words: software testing; random testing; adaptive random testing; mirror adaptive random testing; mirror selection order

1 引言

被测程序通常有较大的输入空间,因此选择一部分可以有效识别软件失效的测试输入作为测试数据具有重要意义.测试数据生成技术指导如何生成高效的测试数据,如随机测试(RT)数据生成方法[1]、基于符号执行的测试数据生成方法[2]、基于有限状态机的测试数据生成方法[3]、基于智能搜索的测试数据生成方法[4,5]等.文献[6]提出了一种基于变异分支优势度排序的弱变异测试数据生成方法,在提高了测试集的错误检测能力的同时提高了效率.对于MPI程序的路径覆盖测试,文献[7]提出了一种将集成代理模型(ESM)的估计集成到测试数据生成过程中的方法,以此来降低适应度值的计算量.文献[8]提出了一种基于代理辅助进化优化的MPI程序路径覆盖测试数据生成方法,该方法能有效地生成高质量的测试数据.

RT是一种简单、易于实现的测试方法.它不涉及到复杂的软件需求或程序的内部结构信息,只需在输入域内随机地生成测试数据即可.RT没有利用被测软

收稿日期:2020-11-30;修回日期:2021-10-29;责任编辑:李勇锋
基金项目:国家自然科学基金(No.61402525);国家社会科学基金(No.15AJG012);国家"核高基"科技重大专项(No.2013JH00103)

件的任何信息,且生成的测试数据具有冗余度高,覆盖度低,盲目性等缺点,曾被Myers认为可能是最糟糕的测试方法. 但由于RT具有简单、易实现、成本低、无偏性、速度快等优点,易与其它测试方法结合使用,因此被广泛应用在各个领域的软件测试和可靠性评估中. 同时,所有测试方法生成的测试数据理论上均可由RT生成,因此RT具有发现一切错误的潜力[9].

经实验研究发现,引发失效的输入易于聚集在连续的区域. 依此结论,Chen提出了ART(Adaptive Random Testing)算法[10]. 与RT不利用任何信息随机产生测试数据相比,ART算法利用已执行成功测试数据的信息,以及引发失效的输入易于集中的特征,使生成的测试数据尽可能均匀的分布在输入空间内. 实验表明ART的检错有效性较RT有明显改善,发现第一个错误所需的测试数据数量更少. 学术界提出了各种基于ART思想的改进算法,例如,基于距离的ART算法(DART)[11],基于限制的ART算法(RRT)[12],基于划分的ART算法[13]等. ART也是一种基础测试方法,应用领域非常广泛,易于同其他测试数据生成方法结合,提升测试数据生成方法的有效性,如ART与程序的路径覆盖信息相结合用来提升测试数据生成的有效性[14]、ART用于安全测试生成检测XSS漏洞的测试数据[15]、ART与基于搜索的测试数据生成方法相结合,在蚁群算法的局部搜索中引入了ART[16]、ART用于测试数据优先级排序,将FSCS算法用于测试数据优先级排序,检错能力优于随机法与贪婪法[17]. 文献[18]在不同测试场景下对组合测试、RT、ART进行了对比实验,实验结果显示ART优于RT,并且在96%的测试场景下检错能力与组合测试相近.

但是ART算法需要额外的开销使生成的测试数据更加均匀. 尤其是基于距离的ART算法,距离计算是算法较大的开销,这些开销减少了ART算法的可用性,降低了ART算法生成测试数据的效率. 镜像策略是一种有效提高测试数据生成效率的手段. 通过镜像策略,将已生成测试数据通过镜像函数生成新的测试数据,减少了ART算法的计算开销,提高了ART算法的效率,但是降低了ART算法的有效性[19].

2　镜像策略

根据镜像策略在ART算法中的实施流程划分,MART算法主要包括三个部分:镜像划分、镜像函数和镜像选择序.

2.1　镜像划分

镜像划分指将输入空间划分子域的方式. 根据各维度划分是否均等,分为相等划分和非等划分.

假设N为输入空间D的维度数量,即$|D|=N$,每个维度用$X_i(i=1,2,3,\cdots,N)$表示,将维度X_i平均划分的数量用$k_i(i=1,2,\cdots,N)$表示. 若对输入空间D进行镜像划分后,存在两个维度X_i与X_j划分的数量不等,即$k_i\neq k_j$,称为非等划分. 若对输入空间D进行镜像划分后,任意两个维度X_i与X_j划分的数量都相等,即$k_i=k_j$,称为相等划分.

在相等划分中,若每个维度划分后的个数$k_i(i=1,2,\cdots,N)$可以用$2^{\omega}(\omega=1,2,3,\cdots)$表示,如

$$\omega=\log_2 k_i\left(\omega\in N_+\right) \tag{1}$$

称此类相等划分为对等划分,ω为镜像划分的深度,子域数量$m=\left(2^{\omega}\right)^N$.

2.2　镜像函数

通过镜像划分方法,将输入空间划分为m个子域. 选择其中一个子域作为源域,源域中应用ART算法生成测试数据,其余$m-1$个镜像子域中如何对应生成测试数据,则是通过镜像函数将源域中的测试数据映像到镜像域中.

常见的镜像函数有平移和对称.

假设在n维输入空间中,源域空间为D_0,镜像子域为D_k,其中,

$$D_0=\left\{\left(x_1,\cdots,x_n\right)\middle|\, x_i\in\left[x_{i\_\min},x_{i\_\max}\right],i\in[1,n]\right\}$$

$$D_k=\left\{\left(y_1,\cdots,y_n\right)\middle|\, y_i\in\left[y_{i\_\min},y_{i\_\max}\right],i\in[1,n]\right\}$$

在源域空间D_0中,测试数据生成算法生成的测试数据为$\mathrm{tc}_0=\left(t_1,t_2,\cdots,t_n\right)$,则测试数据$\mathrm{tc}_0$在镜像子域$D_k$中的镜像测试数据$\mathrm{tc}_k$为:

(1)若镜像函数为平移,则:

$$\begin{aligned}\mathrm{tc}_k=\mathrm{tc}_0+\left(D_k-D_0\right)=\left(t_1,\cdots,t_n\right)+\left[\left(y_{1\_\min}-x_{1\_\min}\right),\cdots,\right.\\ \left.\left(y_{n\_\min}-x_{n\_\min}\right)\right]=\left(\left(x_1+y_{1\_\min}-x_{1\_\min}\right),\cdots,\\ \left(x_n+y_{n\_\min}-x_{n\_\min}\right)\right)\end{aligned} \tag{2}$$

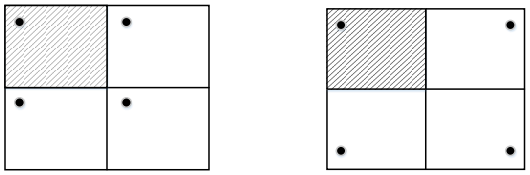
(2)若镜像函数为对称,则:

$$\mathrm{tc}_k=\left(\left(y_{1\_\min}+x_{1\_\max}-t_1\right),\cdots,\left(y_{n\_\min}+x_{n\_\max}-t_n\right)\right) \tag{3}$$

如图1所示,在2维输入空间中,镜像划分深度为1时,划分后的镜像子域的数量为4. 假设阴影部分子域作为源域,源域中通过算法生成测试数据,根据镜像函数将源域中的测试数据镜像到其余的各子域中,通过平移镜像函数生成镜像测试数据如图1(*a*)所示,通过对称镜像函数生成镜像测试数据如图1(*b*)所示.

对称镜像函数可能出现如图2所示的情况,使测试数据在整个输入域中分布不够均匀.

相对来说,平移镜像函数较对称镜像函数在镜像子域中生成测试数据更均匀.



(*a*) 平移 (*b*) 对称

图1 两种镜像函数示意图

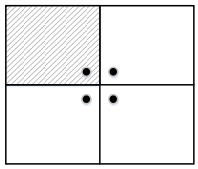


图2 对称镜像可能存在的问题

2.3 镜像选择序

源域中的测试数据先执行,然后是镜像域中的测试数据. 当镜像子域的数量较多时,不同镜像子域内测试数据执行顺序对算法有效性也有影响.

镜像选择序是不同镜像域中的测试数据执行顺序,有顺序选择序、随机选择序等.

顺序选择序指选择镜像域的顺序按从左到右、从上到下的固定顺序依次选择. 将2维输入空间划 n 行 m 列,每个子域顺序编序,则选择镜像域的顺序即 x_i 的下标序.

随机选择指随机在 $m \cdot n-1$ 个镜像域中选择一个,根据镜像函数生成测试数据进行执行.

3 基于镜像策略的ART算法

基于镜像策略的ART的思想[19]:将输入空间划分为多个不相交的相等子域,源域中使用ART算法生成测试数据,剩余子域中使用镜像函数生成镜像测试数据. 详见算法1.

算法1 MART算法

输入:输入域维度 n,划分深度 ω,失效域F随机分布在输入域中.

输出:生成测试用例集 E.

1. 初始时,$E=\varnothing$;
2. 根据镜像相等划分策略划分输入域,一共划分成 $2^{n\cdot\omega}$ 个子域;
3. 任选一个子域 D_1 作为源域;
4. 在源域 D_1 中依据ART算法生成一个测试用例tc;
5. 如果tc检测到失效,转到步骤10;否则,将tc存入 E 中.
6. 根据镜像域的选择序确定测试用例tc待镜像的下一个子域 m;
7. 根据镜像函数,计算出镜像子域 m 中的镜像测试数据tc';
8. 判断tc'是否检测到失效,如果检测到失效转向步骤10;否则,将tc'存入 E 中.
9. 若所有 $2^{n\cdot\omega}-1$ 镜像子域中已生成测试数据tc的镜像测试数据,则转向步骤4;否则,转向步骤6.
10. 算法终止.

将输入域空间 D 等分为 D_1 和 D_2 两部分. 若取 D_1 为源域,D_2 为镜像域,仅需在 D_1 中使用ART算法生成测试数据,通过镜像函数平移到 D_2 中,使 D_1 与 D_2 中测试数据具有相同分布,而只在 D_1 中进行了距离的计算. 假设输入域空间 D 等分为 m 个子域,若ART算法与MART算法生成相同数量测试数据,MART只需要ART距离计算的 $1/m^2$[20].

4 基于镜像选择序优化的MART算法

4.1 问题提出

镜像随机选择序是随机选择未生成镜像测试数据的镜像子域,生成镜像测试数据. 镜像域的随机选择序简单,易实现,一定程度上较均匀的选择镜像子域,从而生成的镜像数据过程也是随机的.

镜像随机选择序可能会出现已生成测试数据的子域与被选择镜像子域在某个维度下是属于同一个分区的情况,如图3所示.

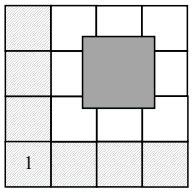


图3 镜像随机选择序可能存在的问题

如图3所示,假设在二维输入空间中,镜像深度 $\omega=2$ 时,镜像划分后,输入空间被平均划分成16个子域,图中深色正方形区域为失效域. 选取标号"1"的域为源域,源域中生成测试数据后,选择镜像子域生成镜像测试数据,如果选择的镜像子域在阴影子域内,则无法触发缺陷,因为失效域在白色子域所在范围内. 镜像子域随机选择序是随机的选择子域,不做任何限制,可能会出现选择的镜像子域与已生成测试数据的子域在相同行或相同列的现象,也就是选择生成测试数据的镜像子域在某个维度或多个维度上取值范围相同.

针对镜像随机选择序可能出现的这种情况,考虑对镜像选择序进行约束,本文通过限制镜像子域选择序,增强镜像子域选择序的多样性,从而提高MART算法的检错有效性.

4.2 镜像受限选择序

镜像受限选择序是每次选择镜像子域生成镜像测试数据时,选择镜像子域不是完全随机的,是受限制的随机选择方式.

假设 n 维输入空间为 $D=\left\{\left(d_0,d_1,\cdots,d_{n-1}\right)\middle|\, d_0\in D_0,d_1\in D_1,\cdots,d_{(n-1)}\in D_{(n-1)}\right\}$,镜像划分深度为 ω,则每个维度被划分为 2^ω 个区域,整个输入空间被划分为 $2^{n\cdot\omega}$ 个镜像子域. 第 i 维空间 d_i 被划

分的区域用d_{ij}表示,其中$0\leqslant j\leqslant 2^{\omega}-1$.

定义$\sum_{ij}$表示第i维空间d_i的第j个区域中被选中的镜像子域的数量. 当$\sum_{ij}=2^{\omega(n-1)}$时,表示第$i$维空间$d_i$的第$j$个区域中所有镜像子域都被选出.

对于某个子域$X=\left\{\left(x_0,x_1,\cdots,x_{n-1}\right)\right\}$,定义$\delta\left(x_i\right)$表示子域$X$在第$i$维空间中所属的子域位置. 镜像受限选择算法描述如算法2所示.

算法2　镜像受限选择序算法

输入:输入空间D;

输出:子域选择序;

1. 镜像划分深度为ω,划分子域数量$m=2^{n\cdot\omega}$,初始$\sum_{ij}=0$,其中$0\leqslant i\leqslant n-1, 0\leqslant j\leqslant 2^{\omega}-1$.

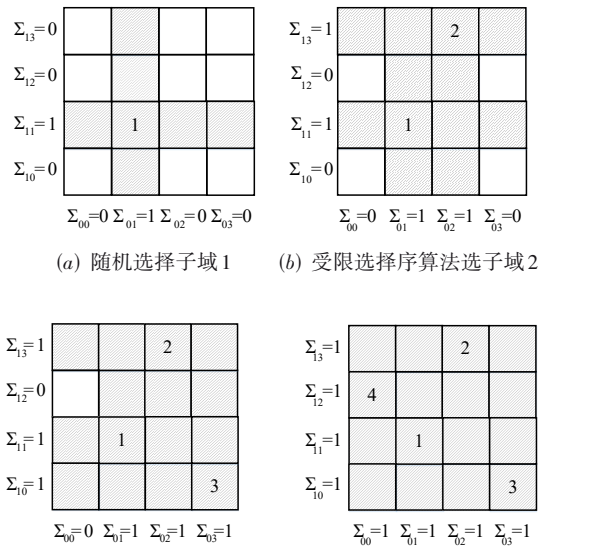
2. 随机选择一个子域作为源域$D_s=\left\{\left(d_0',d_1',\cdots,d_{n-1}'\right)\middle|\, d_0'\in D_{0s},d_1'\in D_{1s},\cdots,d_{n-1}'\in D_{(n-1)s},0\leqslant s\leqslant 2^{\omega}-1\right\}$将源域空间对应每个维度的区域数量加1,即$\sum_{ij}$=1,其中,$i\in\{0,1,\cdots,n-1\}, j=\delta\left(d_i'\right)$.

3. 在所有$\sum_{ij}$=0的子域内,随机选择一个子域D',同理,将该子域对应的每个维度的区域数量加1,$\sum_{ij}$=1,其中$j=\delta\left(D_i'\right)$.

4. 当所有子域全部对应的$\sum_{ij}$=1时,重复此操作,在所有$\sum_{ij}$=0的子域内,随机选择子域. 直至所有子域全部被选出.

5. 返回子域选择出的顺序.

如图4所示,在二维输入空间中,镜像划分深度为2,则每个维度被划分成4部分,二维空间被划分为16个镜像子域. 随机选择一个子域作为源域,如图4(a)中,标号为1的子域,则$\delta\left(x_0\right)=1,\delta\left(x_1\right)=1$,故$\sum_{01}=1,\sum_{11}=1$. 即阴影区域为源域"1"在每个维度上的子域所占的区域,下一个镜像子域的选择不在阴影区域内选择. 也就是在图4(a)中的空白子域中,随机选择一个域作为镜像子域"2",如图4(b)所示,则"2"子域的$\delta\left(x_0\right)=2,\delta\left(x_1\right)=3$,故$\sum_{02}=1,\sum_{13}=1$. 即"2"子域所在的行列子域转换为阴影区域. 同理,在空白子域中随机选择"3"、"4"子域(如图4(c)、(d)),第一轮子域选择结束后,每个维度的每个区域都有选出的镜像子域,即$\forall\left(i,j\right)\sum_{ij}=1$,如图4($d$)中所有子域全部为阴影填充.

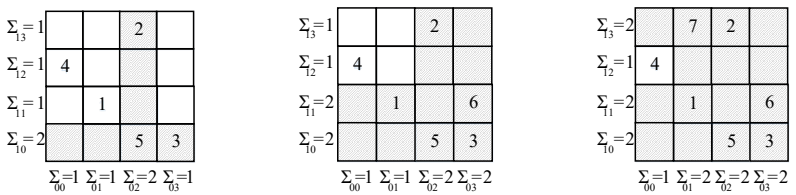


(a) 随机选择子域1　(b) 受限选择序算法选子域2　(c) 受限选择序算法选子域3　(d) 受限选择序算法选子域4

图4　镜像受限选择序

然后依此规则,开始第二轮迭代,选出相应的镜像子域,第二轮迭代结束后,每个维度的每个区域都有选出的2个镜像子域,即$\forall\left(i,j\right)\sum_{ij}=2$,第$k$轮次迭代后,$\forall\left(i,j\right)\sum_{ij}=k$,直至$k=2^{\omega(n-1)}$时(图中示例$k=4$时),全部镜像子域被选出.

有时可能会出现镜像子域受限选择冲突现象,如图5所示.



(a) 一轮迭代后随机选子域5　(b) 受限选择序算法选子域6　(c) 受限选择序算法选子域7

图5　镜像受限选择序冲突现象

当第一轮迭代结束后,$\forall\left(i,j\right)\sum_{ij}=1$,此时清空所有阴影底纹,恢复为白色,开始第二轮迭代,从剩余的空白镜像子域中随机选择一个,如图5(a)所示,选择镜像子域"5". 如图5(a)所示,则"2"子域的$\delta\left(x_0\right)=2$,$\delta\left(x_1\right)=0$,故$\sum_{02}=2,\sum_{10}=2$. 即"5"子域所在的行列子域转换为阴影区域. 同理,在空白子域中随机选择"6"、"7"子域(如图5(b)、(c)),此时,仅剩$\sum_{00}=1,\sum_{12}=1$,其余$\sum_{ij}=2, i\in\{0,1\}$. 而仅剩的白色子域"4"已经被选出,此现象为镜像子域受限选择冲突现象. 算法针对此情况的冲突解决方案是,清空所有阴影底纹,恢复为白色,然后从所有空白且没有标号的子域中随机选择一个子域,进入下一轮迭代,直至所有子域均为选出.

4.3 基于镜像受限选择序的MART算法

根据镜像受限选择序思想，本文提出了基于镜像受限选择序的MART算法. 通过限制待镜像的镜像域，使得镜像域的选择序更加均匀，保证了测试数据尽可能的在每个维度上均匀分布.

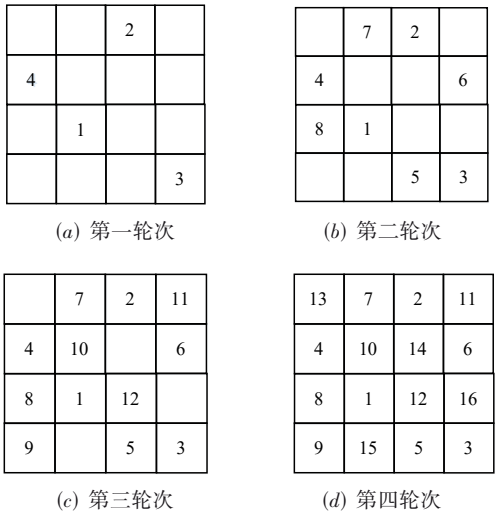
算法3 基于镜像受限选择序的MART算法

输入：n维输入空间，镜像深度ω；

输出：测试数据集E.

1. 对于n维输入空间，镜像深度ω，将输入域划分为$2^{n\cdot\omega}$个子域.
2. 从子域中任选一个作为源域，其余子域为镜像域.
3. 源域中使用ART算法生成测试数据tc，放入E中.
4. 测试数据tc执行被测程序若触发缺陷，转向步骤6，否则通过镜像函数映像生成镜像测试数据tc′.
5. 镜像测试数据生成顺序取决于镜像选择序（参考算法2：镜像受限选择序算法），其中检验生成的镜像测数据tc′（放入E中）是否触发缺陷，若触发缺陷，转向步骤6，否则到步骤3.
6. 算法终止，返回已生成测试数据集E.

如图6所示，在二维输入空间中，镜像划分深度为2时，整个输入空间被划分为16个子域，假设选取标号为“1”的子域空间为源域，即在1号子域中根据ART算法生成测试数据. 根据镜像受限选择序MART算法的基本思想，依次选择镜像子域通过镜像函数生成镜像测试数据t. 第2号子域被随机选出，只须保证与1号子域不同行不同列. 同理，第3号子域被随机选出，只须保证与1、2号子域不同行不同列. 第一轮次选出的4个子域如图6(a)所示，依此原则，第二、三轮每轮次内依次选择不同行不同列的子域，如图6(b)(c)所示. 直至第4轮次执行完毕，如图6(d)所示，所有16个镜像子域被选出，生成子域序列s. 在源域中使用ART算法生成的测试数据根据镜像函数，按照子域序列s的顺序依次生成测试数据，每次生成一个镜像测试数据，执行被测程序检验是否触发缺陷，如果触发缺陷则算法终止，否则依次执行完序列s中所有的测试数据，若还未发现缺陷，则需在源域中根据ART算法生成第二个测试数据，以上述原则产生镜像子域的选择序，生成镜像测试数据，直至发现缺陷或达到迭代终止条件.



(a) 第一轮次 (b) 第二轮次

(c) 第三轮次 (d) 第四轮次

图6 基于镜像受限选择序的MART算法

5 实验设计与结果分析

5.1 仿真实验

5.1.1 实验设计

仿真实验涉及的参数主要为维度、失效率、失效模式、测试方法、实验次数等，具体如下：

(1)维度. 表示待测程序输入参数的个数. 假定以二、三、四、五维输入域进行分析对比，每个维度坐标取值范围为[1,1 000]，如二维输入域为正方形，三维输入域为正方体，以此类推.

(2)失效率. 为引发失效的输入域占整个输入域的比例，显然，失效率$\theta\in[0,1]$. 失效域大小由失效率确定，失效率=块状失效模式大小/整个输入空间大小，失效域的位置随机出现在输入域内. 失效率取值范围为$\theta\in[0.000\,1,0.1]$.

(3)失效模式. 使用代码模拟输入空间中的块状失效模式、条状失效模式和点状失效模式. 根据实验设置的输入空间大小D，失效率θ，计算失效区域$F=D\times\theta$. 块状失效模式模拟在输入空间内随机生成一个面积或体积为F的正方形（二维）或立方体（三维）区域；条状失效模式模拟在输入空间中的一个长条区域，通过在X轴、Y轴正向坐标轴上各自随机选取一个点，两个点连接后的直线作为长条失效域的长度，宽度由失效域面积F确定，使得长宽比例围成的条状区域面积为F；点状失效模式模拟在输入空间内随机生成k个不相交的相等大小的正方形区域，假定每个正方形的面积为S，则k个正方形区域面积之和$k\times S$等于失效域面积F，本实验中k取值为10，即模拟10个离散点状失效域. 失效模式的影响分析时，考虑三种失效模式，除此之外，无特殊说明时，失效模式均为块状失效模式.

(4)测试数据生成算法. 实验算法为ART算法、镜像顺序选择序MART算法、镜像随机选择序MART算法、镜像受限选择序MART算法. 三类MART算法均采用对等划分的方式对输入空间进行子域划分，镜像函数包括平移和对称两种. 源域中测试数据生成算法均采用FSCS算法. 详见表1.

随机测试（RT）：随机测试算法，F_{measure}理论值为$1/\theta$（其中θ为失效率）.

ART算法：为ART算法中的经典DART算法之一FSCS算法，基于距离计算的自适应随机测试算法，其中

表 1　实验中的算法列表

算法缩写	算法名	算法描述
RT	Random Testing	随机测试
ART	Adaptive Random Testing(FSCS)	自适应随机测试(FSCS算法)
MART_T	Mirror Adaptive Random Testing(FSCS)+Translate	镜像自适应随机测试+平移函数+顺序选择镜像域
MART_R	Mirror Adaptive Random Testing(FSCS)+Reflect	镜像自适应随机测试+对称函数+顺序选择镜像域
MART_T_R	Mirror Adaptive Random Testing(FSCS)+Translate+Random	镜像自适应随机测试+平移函数+随机选择镜像域
MART_R_R	Mirror Adaptive Random Testing(FSCS)+Reflect+Random	镜像自适应随机测试+对称函数+随机选择镜像域
MART_T_N	Mirror Adaptive Random Testing(FSCS)+Translate+New	镜像自适应随机测试+平移函数+受限选择镜像域(本文新算法)
MART_R_N	Mirror Adaptive Random Testing(FSCS)+Reflect+New	镜像自适应随机测试+对称函数+受限选择镜像域(本文新算法)

候选测试数据集大小设为10.

镜像顺序选择序MART算法:镜像子域选择方式为顺序方式. 镜像随机选择序MART算法:镜像子域选择方式为顺序方式. 镜像受限选择序MART算法:为4.3节提出的新算法.

(5)检错有效性度量指标. 使用各类算法生成测试数据,当测试数据落入模拟的失效域中时,触发缺陷,记录第一个发现缺陷时已生成测试数据的数量. 每次执行算法发现第一个缺陷所需的测试数据的数量为F_{count},则本实验中$F_{measure}$值为10 000次F_{count}的平均值. 算法有效性度量采用$F_{measure}$值,是首次触发缺陷生成测试数据数量的期望值. 为了便于对比不同失效率下ART算法相对RT算法的$F_{measure}$改进程度,定义$F_{ratio}=F_{ART}/F_{RT}$,F_{ratio}反映了ART算法相对RT算法的改进程度. 若$F_{ratio}<1$,则说明ART算法检错有效性比RT算法强,当$F_{ratio}\geqslant 1$时,说明ART算法相对RT算法检错有效性无优势.

(6)数据展示标识. 实验结果图形展现时,ART算法使用黑色实线标识,使用平移镜像函数的各类MART算法(MART_T、MART_T、MART_T_N)使用实线标识(分别对应品红色、蓝色与红色),使用对称镜像函数的各类MART算法(MART_R、MART_R_R、MART_R_N)使用虚线标识(分别对应品红色、蓝色与红色).

5.1.2　实验结果分析

5.1.2.1　失效模式的影响分析

二维输入空间中,镜像划分深度为2,划分子域数量为16时;以及镜像划分深度为3,划分子域数量为64时,分析各类MART算法在块状失效模式、条状失效模式和点状失效模式下检错有效性.

如图7(*a*)(*b*)所示,块状失效模式中,镜像深度为2时,各类MART算法中,镜像平移函数的MART算法优于镜像对称MART算法. 镜像深度为3时,同类MART算法中镜像平移函数的MART算法与镜像对称MART算法差异不大. MART_T_N算法在各类MART算法中检错有效性最强.

如图7(*c*)(*d*)所示,条状失效模式下,镜像顺序选择序算法在各类MART算法中表现优异,F_{ratio}值小于其他算法. 分析镜像顺序选择算法在条状失效模式下的优势,主要由模拟仿真方式决定. 因为条状模式代码中失效区域与X轴、Y轴的正向轴相交,而镜像顺序选择算法的顺序也是从坐标原点开始,沿X轴依次顺序选择,然后向Y轴正向方向扩展. 这种条状失效模式模拟方式有利于镜像顺序选择序触发缺陷,所以该类算法F_{ratio}值较小.

如图7(*e*)(*f*)所示,点状失效模式下,各类ART算法检错有效性增强. 随着深度增加,各类算法检错有效性均有所降低. 整体来看,MART_T_N算法检错有效性略强于其他MART算法.

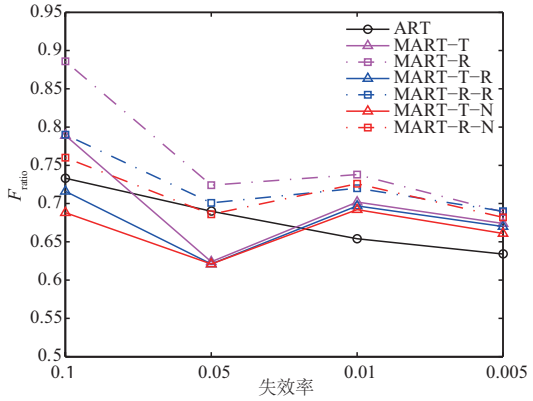
三种失效模式下,各类ART算法都随着镜像子域增多,检错有效性有所降低. MART_T_N算法在块状失效模式与点状失效模式中较其他MART算法相比检错能力略强.

5.1.2.2　镜像划分的影响分析

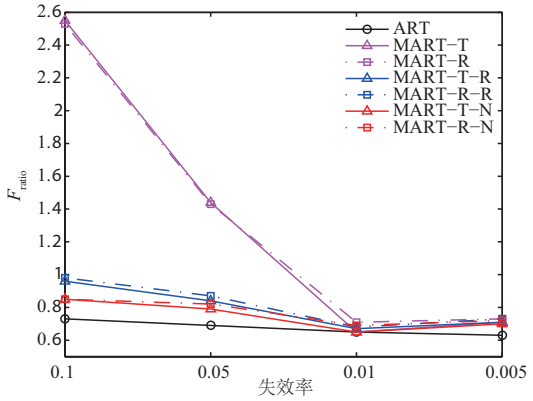
分析镜像划分深度不同时,ART算法及各类MART算法的有效性对比. 失效率为0.01时,镜像深度分别取ω=1,2,3,4,二维空间时子域数量为4、16、64、256,三维空间时子域数量为8、64、512、4 096. 对比实验F_{ratio}值如图所示.

图8中,不同镜像深度时,平移镜像函数普遍优于对称镜像函数,三维空间中,平移镜像函数的优势比在二维空间中更大. MART_T_N算法在三维空间,深度为4时,较MART_T_R算法优势明显.

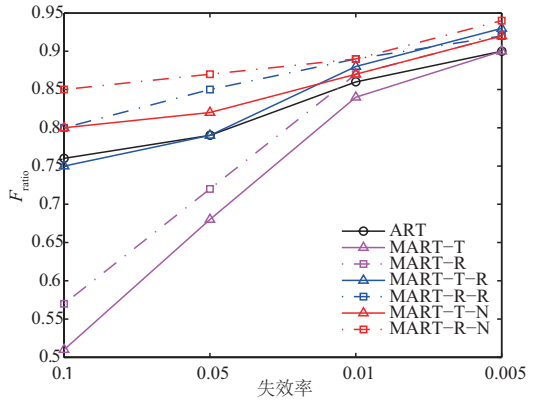
总得来说,当镜像划分的子域数量与RT算法的理论$F_{measure}$值相近时,各类MART算法的检错能力最强. 镜像划分的子域数量增大时,各类MART算法的检错能力都下降. 镜像受限选择序算法检错有效性优于其他MART算法. 随着深度的增加,MART_T_N算法优势更加明显.



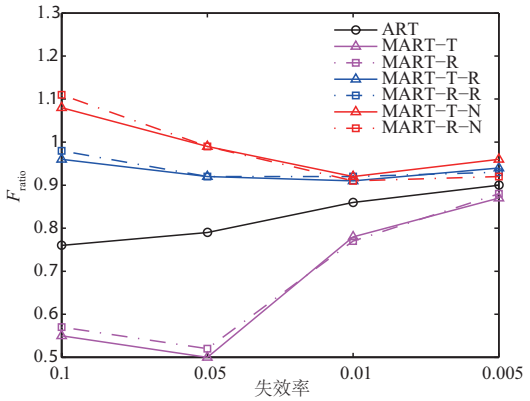
(*a*) 块状失效模式,镜像深度为2



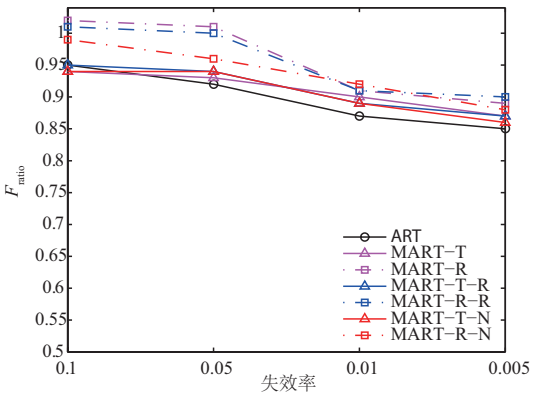
(*b*) 块状失效模式,镜像深度为3



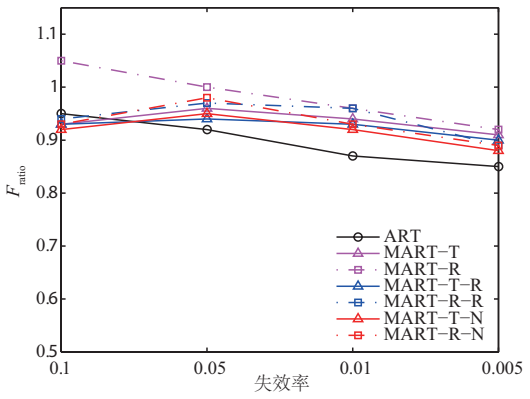
(*c*) 条状失效模式,镜像深度为2



(*d*) 条状失效模式,镜像深度为3



(*e*) 点状失效模式,镜像深度为2



(*f*) 点状失效模式,镜像深度为3

图7 不同失效模式,2~3镜像深度,各类算法F_{ratio}值

5.1.2.3 维度的影响分析

(1)镜像划分深度相同,维度不同

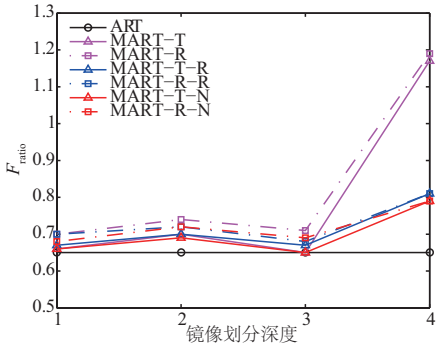
失效率为0.01(RT算法理论$F_{measure}$值为100),2~6维输入,镜像划分2,各算法F_{ratio}值如图9.

在低维空间中,不同镜像序的MART算法差异不大,随着维度的增大,MART_T_N算法比MART_T_R算法的优势增大,除了当维度和深度确定的子域数量与RT的理论$F_{measure}$值近似时,各类MART算法的F_{ratio}值最低.

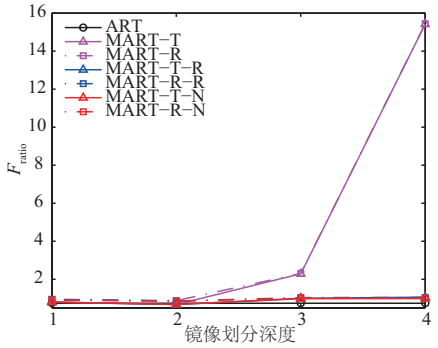
(2)划分子域数量相同,维度不同

分别考虑失效率为0.01时与0.001时,不同维度时而相同子域数量进行实验验证.

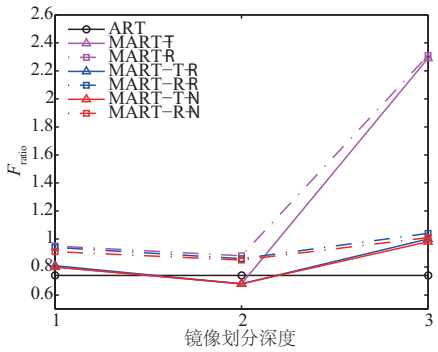
本次实验设置失效率为0.01时(RT算法理论$F_{measure}$值为100),子域数量最接近的值为64,分别对2、3、6维输入空间进行镜像划分3、2、1时得到64个子域,各算法的F_{ratio}值如图10(*a*)所示. 实验设置失效率为



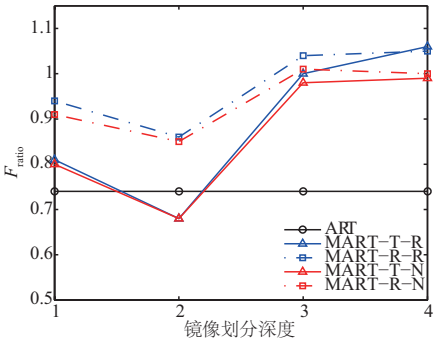
(*a*) 2维空间,深度1~4



(*b*) 3维空间,深度1~4

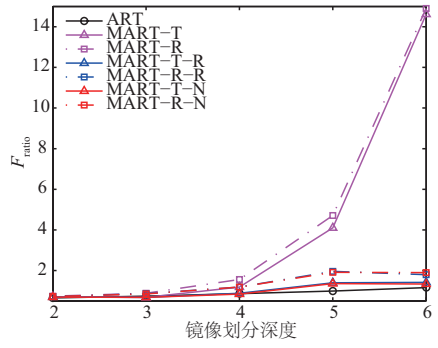


(*c*) 3维空间,深度1~3

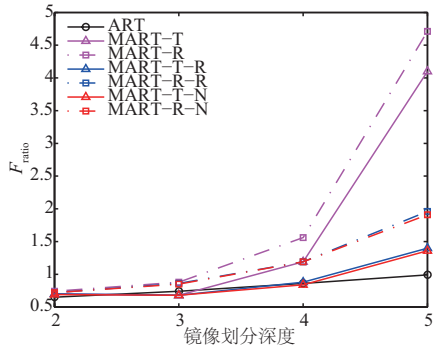


(*d*) 3维空间,深度1~4(除顺序镜像)

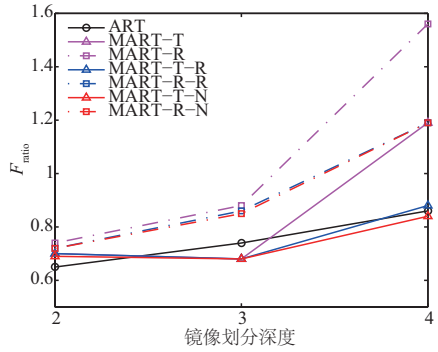
图8　失效率0.01,2~3维空间,划分深度1~4,各算法F_{ratio}值



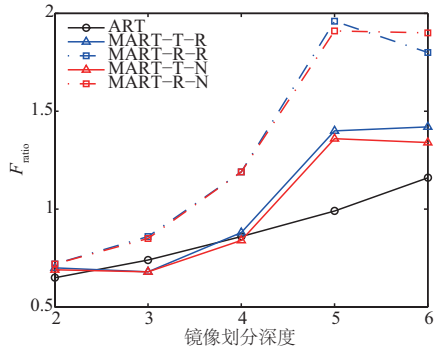
(*a*) 2~6维度各算法F_{ratio}值



(*b*) 2~5维度各算法F_{ratio}值

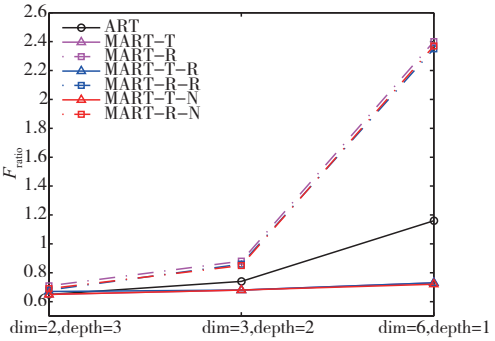


(*c*) 2~4维度各算法F_{ratio}值

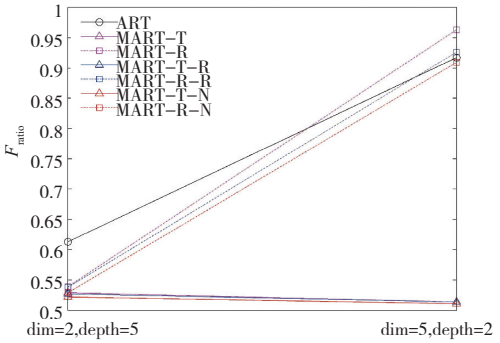


(*d*) 2~6维度各算法F_{ratio}值(除顺序镜像序)

图9　失效率0.01,镜像深度为2,2~6维度各算法F_{ratio}值



(a) 失效率0.01，子域数64



(b) 失效率0.001，子域数1024

图 10 子域数量与RT理论$F_{measure}$相近时，各算法F_{ratio}值

0.001时(RT算法理论$F_{measure}$值为1 000)，子域数量最接近的值为1 024，分别对2、5维输入空间进行镜像划分5、2时得到1 024个子域，各算法的F_{ratio}值如图10(*b*)所示.

如图10所示，在失效率0.01与0.001时，在子域数量与RT算法的理论$F_{measure}$值相近时，各类平移镜像函数MART算法(MART_T、MART_T_R、MART_T_N)的检错有效性最强，并且对维度变化不敏感. ART算法、各类对称镜像函数MART算法(MART_R、MART_R_R、MART_R_N)受维度变化影响大，随着输入空间维度增大，F_{ratio}值增大.

5.1.2.4 失效率的影响分析

由于镜像顺序选择序MART算法在子域数量大于RT的理论$F_{measure}$值时，检错有效性迅速下降. 所以下面分析只涉及到镜像随机选择序MART算法、镜像受限选择序MART算法.

分析在三维输入空间中，失效率分别为0.1、0.05、0.01、0.005、0.001时(RT的理论$F_{measure}$值为10、20、100、200、1 000)，镜像深度分别取ω=1，2，3，4，子域数量分别为8、64、512、4 096，ART算法、镜像随机选择序MART算法、镜像受限选择序MART算法的F_{ratio}值如图11所示.

(1)不同失效率时，F_{ratio}值分析

深度为1、2、3对应的子域数量(8、64、512)与失效率0.1、0.01、0.005(RT的理论$F_{measure}$值为10、100、200)相近. 如图11(*a*)、(*c*)、(*e*)所示，失效率为0.1、0.01、0.001时，镜像平移函数的MART算法分别在深度为1、2、3时F_{ratio}值最低，且MART_T_R、MART_T_N算法F_{ratio}值相近. 随着深度的增大，MART_T_N算法比MART_T_R算法的优势增大，F_{ratio}值差距增大.

如图11(*a*)、(*b*)、(*d*)所示，各类MART算法的F_{ratio}值都高于ART算法，随着深度的增加，子域数量的增多，MART_T_N算法与MART_T_R算法的F_{ratio}值差异增大，优势增大.

(2)不同失效率时，F_{ratio}值分析

分析MART_T_N算法在镜像划分深度分别为1、2、3、4时的表现，以便指其在实际应用中镜像划分深度参数的设置.

如表2所示，镜像划分深度与RT算法的理论$F_{measure}$值相近时，MART_T_N算法具有较高的检错能力，甚至优于ART算法的F_{ratio}值. 整体来看，随着镜像划分深度的增加，子域数量的增大，MART_T_N在各失效率下检错能力下降.

5.2 实例实验

5.2.1 实验设计

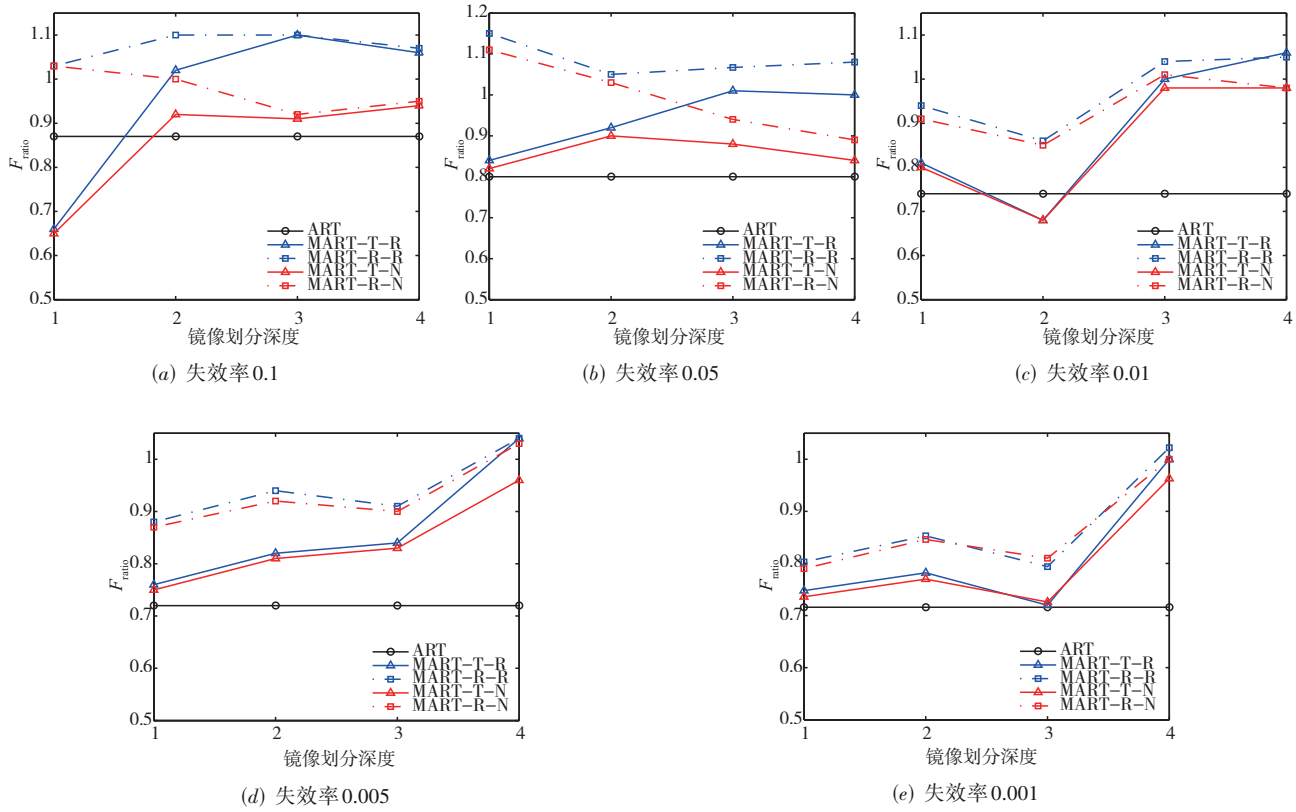
5.2.1.1 测试基准程序

本次实验有四个真实程序，其中Trityp程序和TCAS程序是来源于Software artifact Infrastructure Repository(SIR)的被测程序. Integer程序源于JDK中java.lang.Integer类. DSCompiler类来源于org.apache.commons.math3.analysis.differentiation包. 详情如表3所示.

Trityp程序为三角形分类程序，3个整型输入作为三边长度，输出是否能构成三角形以及属于哪类三角形. Mujava变异测试工具用来生成被测程序的变异体，每个变异体是在源程序中注入一个错误来生成的. Trityp程序一共生成462个变异体. 去除72个等价变异体，剩余390个变异体. 假定整型输入域范围为[1，100]，每个变异体的失效率通过程序输入参数遍历整个输入域计算得到. 实验选择失效率在(0.000 1，1)范围内的368个变异体.

TCAS是经典的西门子程序之一，是有12个输入参数的防撞系统源代码. 输入域范围为[0，1 000]. 变异体来源于SIR，这些变异体种的失效源于真实的错误. 变异体失效率在[0.000 01，0.04]范围内.

Integer程序源于JDK中java.lang.Integer类，2个输



(a) 失效率0.1　(b) 失效率0.05　(c) 失效率0.01

(d) 失效率0.005　(e) 失效率0.001

图11　在3维空间,各算法1~4深度的F_{ratio}值(除顺序镜像序)

表2　不同失效率,不同深度下MART_T_N算法的F_{ratio}值

失效率	F_RT	ART	MART_T_N			
			Depth=1	Depth=2	Depth=3	Depth=4
0.1	10	0.87	**0.64**	0.91	0.90	0.94
0.05	20	0.8	0.82	0.89	0.905	0.84
0.01	100	0.74	0.95	**0.69**	0.98	0.98
0.005	200	0.72	0.93	0.83	0.825	0.96
0.001	1 000	0.716	0.861	0.902	**0.726**	0.93

表3　四个实例被测程序的详情

被测程序	类	参数个数	取值范围	变异体数量	失效率范围
Trityp	1	3	[1,100]	368	[0.001,0.01]
Tcas	1	12	[0,1 000]	20	[0.000 01,0.04]
Integer	1	2	[1,1 000]	139	[0.000 9,1)
DSCompiler	1	1~14	[-100,100]	7294	[0.000 01,1)

入参数取值范围为[1,1 000]. Integer程序使用mujava生共160个变异体,去除21个等价变异体,剩余139个变异体的失效率在[0.000 9,1)范围内.

DSCompiler类来源于org.apache.commons.math3.analysis.differentiation包,Apache Commons类库中math3类库为轻量级数学和统计组件. 使用Mujava生成共计7 294个变异体,失效率为[0.000 01,1)范围内.

5.2.1.2　实验参数设置

本实验对比的镜像随机选择序算法、镜像受限选择序算法. 通过第5.1节仿真实验分析,平移镜像函数在MART算法中较对称镜像函数更有效,所以本实验只考虑平移镜像函数.

镜像随机选择序MART算法(MART_T_R):采用对等划分的方式对输入空间进行子域划分,镜像函数为平移镜像函数,镜像子域选择方式为随机序方式,源域中测试数据生成算法采用FSCS算法.

镜像受限选择序MART算法(MART _T_N):为本文提出的方法,镜像函数为平移镜像函数.

实验执行过程如下:分别使用MART_T_R和MART_T_N算法生成测试数据,执行源程序及变异体. 若变异体与源程序运行结果不同,则此算法杀死该变异体,记录当前已生成测试数据数量.

针对每个有效变异体,分别记录MART_T_R和MART_T_N杀死该变异体所需测试数据数量F-count,实验重复2 000次均值作为$F_{measure}$. 令F_{random}与F_{new}分别表示MART_T_R和MART_T_N的$F_{measure}$,F_{ratio}为MART_T_N与MART_T_R的$F_{measure}$比值,即MART_T_

N算法相对MART_T_R算法检错有效性改进程度. 若$F_{ratio}<1$,说明MART_T_N算法优于MART_T_R算法.

$$F_{ratio}=F_{new}/F_{random}$$

5.2.2 实验结果分析

四个程序使用MART_T_R和MART_T_N运行后的F_{ratio}值统计结果如表4所示. 表中列出每个程序变异体数量,统计出MART_T_N算法与MART_T_R算法的F_{ratio}值大于等于1和小于1的情况. 在Trityp程序的368个变异体中,$F_{ratio}<1$的变异体数量为294个,即79.9%的变异中,MART_T_N算法的检错能力优于MART_T_R算法. Integer程序的139个变异体中,$F_{ratio}<1$的变异体数量为112个,即80.6%的变异中,MART_T_N算法的检错能力优于MART_T_R算法. TCAS程序的20个变异体中,$F_{ratio}<1$的变异体数量为16个,即80%的变异中,MART_T_N算法的检错能力优于MART_T_R算法. DSCompiler类的7 294个变异体中,$F_{ratio}<1$的变异体数量为5 568个,即76.3%的变异中,MART_T_N算法的检错能力优于MART_T_R算法.

在实例实验中,从MART_T_N算法对各变异体检错能力来看,MART_T_N算法在实际程序代码中,具有高于MART_T_R算法的检错能力.

表4 实例程序变异体F_{ratio}结果

被测程序	变异体数量	变异体数量	
		$F_{ratio}<1$	$F_{ratio}\geqslant 1$
Trityp	368	294	74
Integer	139	112	27
TCAS	20	16	4
DSCompiler	7 294	5 568	1 753

6 小结

镜像策略将输入域划分多个镜像子域,通过镜像函数将源域中测试数据生成方法生成的测试数据镜像到各镜像子域中,从而生成整个输入域内的测试数据. 将镜像策略用于ART算法的测试数据生成过程中,减少了ART算法的计算开销,通过镜像函数即可生成镜像测试数据.

但镜像策略融入ART算法后,虽然提高了测试数据生成的效率,但降低了算法的有效性. 本文对MART算法进行了研究,针对镜像函数将源测试数据镜像到各子域时的镜像顺序,对比分析镜像选择序与镜像函数对MART算法的影响,提出了基于镜像受限选择序的MART算法. 实验结果显示,针对镜像策略中镜像选择序的优化,提高了MART算法的检错有效性,得到以下结论:

(1)镜像平移函数的MART算法多数情况下优于镜像对称函数的MART算法,或与其相当;

(2)当镜像划分的子域数量与RT算法理论$F_{measure}$值相近时,各类MART算法检错能力最强.

(3)当划分子域数量大于RT算法理论$F_{measure}$值,镜像选择序对MART算法影响较大. 镜像顺序选择序的MART算法的检错能力急速下降,所以,不适宜用于镜像划分子域数量较多的场景.

(4)本文提出的镜像受限选择序MART算法检错有效性高于镜像随机选择序MART算法. 随着镜像划分深度、输入空间维度越大,镜像受限选择序MART算法的优势越明显.

参考文献

[1] WU Huayao, NIE Changhai, PETKE Justyna, et al. An empirical comparison of combinatorial testing, random testing and adaptive random testing[J]. IEEE Transactions on Software Engineering, 2020, 46(3): 302-320.

[2] 谢肖飞, 李晓红, 陈翔, 等. 基于符号执行与模糊测试的混合测试方法[J]. 软件学报, 2019, 30(10): 3071-3089.
XIE XiaoFei, LI XiaoHong, CHEN Xiang, et al. Hybrid testing based on symbolic execution and fuzzing[J]. Journal of Software, 2019, 30(10): 3071-3089. (in Chinese)

[3] 潘雄, 等. 基于无关变量分离的EFSM测试数据进化生成[J]. 北京航空航天大学学报, 2019, 45(5): 919-929.
PAN Xiong, et al. Evolutionary generation of test data for EFSM based on irrelevant variable separation[J]. Journal of Beijing University of Aeronautics and Astronautics, 2019, 45(5): 919-929. (in Chinese)

[4] SIMONE Scalabrino, GIOVANNI Grano, et al. OCELOT: A search-based test-data generation tool for C[C]//2018 IEEE/ACM International Conference on Automated Software Engineering. Montpellier, France: IEEE, 2018: 868-871.

[5] 冯霞, 郝慧敏. 基于遗传算法的IMX系统测试数据自动生成研究[J]. 电子与信息学报, 2015, 37(10): 2501-2507.
FENG Xia, HAO Hui-min. Research on automatic generation of test data in MX based on genetic algorithms[J]. Journal of Electronics and Information Technology, 2015, 37(10): 2501-2507. (in Chinese)

[6] YAO X, ZHANG G, PAN F, et al. Orderly generation of test data via sorting mutant branches based on their dominance degrees for weak mutation testing[J]. IEEE Transactions on Software Engineering, DOI:10.1109/TSE.2020.3014960.

[7] SUN B, GONG D, TIAN T, et al. Integrating an ensemble surrogate model's estimation into test data generation[J]. IEEE Transactions on Software Engineering, DOI:10.1109/TSE.2020.3019406.

[8] GONG D, SUN B, YAO X, et al. Test data generation for path coverage of MPI programs using SAEO[J]. ACM Transactions on Software Engineering and Methodology, 2021, 30(2): 1-37.
[9] 李志博, 李清宝, 于磊, 等. 基于划分的自适应随机测试综述[J]. 计算机科学, 2019, 46(3): 25-35.
LI Zhibo, LI Qingbao, YU Lei, et al. Survey on adaptive random testing by partitioning[J]. Computer Science, 2019, 46(3): 25-35. (in Chinese)
[10] CHEN T Y, LEUNG H, MAK I K. Adaptive random testing[C]//Ninth Asian Computing Science Conference. Oxford, UK: ACM, 2004: 320-329.
[11] LI ZhiBo, LI QingBao, YU Lei. An enhanced adaptive random testing by dividing dimensions independently[J]. Mathematical Problems in Engineering, 2019: 1-15.
[12] CHAN K P, CHEN T Y, TOWEY D. Restricted random testing[J]. International Journal of Software Engineering & Knowledge Engineering, 2002, 16(4): 553-584.
[13] LI Zhibo, LI Qingbao, LI Renjie, WANG Ling. An enhanced ART in high dimensional input domain[C]//2019 IEEE 10th International Conference on Software Engineering and Service Science. Beijing, China: IEEE, 2019: 495-497.
[14] NIKRAVAN E, FEYZI F, PARSA S. Enhancing path-oriented test data generation using adaptive random testing techniques[C]//Proceedings of the International Conference on Knowledge-based Engineering & Innovation. Tehran, Iran: IEEE, 2015: 510-513.
[15] LV C, ZHANG L, ZENG F, et al. Adaptive random testing for XSS vulnerability[C]//Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference(APSEC). Putrajaya, Malaysia: IEEE, 2019: 63-69.
[16] BIDGOLI A M, HAGHIGHI H. Augmenting ant colony optimization with adaptive random testing to cover prime paths[J]. The Journal of Systems and Software, 2020, 161 (3): 110495. 1-110495. 17.
[17] WANG R, LI Z, JIANG S, et al. Regression test case prioritization based on fixed size candidate set ART algorithm[J]. International Journal of Software Engineering and Knowledge Engineering, 2020, 30(3): 291-320.
[18] 吴化尧. 组合测试方法及其有效性研究[D]. 南京: 南京大学, 2018 .
WU Huayao. Research on Combinationorial Testing and Its Fault Detection Ability[D]. Nanjing: Nanjing University, 2018. (in Chinese)
[19] KUO F Q. An indepth study of mirror adaptive random testing[C]//Proceedings of the Ninth International Conference on Quality Software. Jeju, Korea: IEEE, 2009: 51-58.
[20] KUO F C. On Adaptive Random Testing[D]. Melbourne: Swinburne University of Technology, 2006.

作者简介

李志博　女,1982年生,副教授,工学博士. 主要研究方向为软件测试、大数据、知识图谱.
E-mail: lizhibo1019@163.com

李清宝　男,1967年生,教授,博导,工学博士. 主要研究方向为信息安全、软件测试.
E-mail: 13653716702@139.com

兰明敬(通讯作者)　男,1982年生,副教授,硕导,工学博士. 主要研究方向为知识图谱、大数据测试.
E-mail: lanmingjing@126.com