http://bhxb.buaa.edu.cn jbuaa@buaa.edu.cn
DOI: 10.13700/j.bh.1001-5965.2014.0211

基于程序变异的 Simulink 模型测试方法

周艺斌，殷永峰*，李骁丹，王明威

（北京航空航天大学 可靠性与系统工程学院，北京 100191）

摘 要：为解决当前 Simulink 模型变异测试中测试执行开销大、测试用例生成效率低等问题，首先根据当前的 Simulink 模型变异算子集，基于程序变异技术提出了 Simulink 模型的变异测试过程和一组改进变异算子集. 实验表明，在不影响测试用例集变异评分的情况下，该组变异算子集能够有效减少变异模型的生成数量，从而降低测试开销. 其次，设计了一种基于搜索的 Simulink 模型变异测试用例生成方法，该方法将变异模型的测试用例生成问题转换为目标函数极小化问题，通过模拟退火算法对目标函数寻优，最终搜索出能够杀死该变异模型的测试用例. 最后，将该方法应用于典型案例，验证了方法的正确性和有效性.

关 键 词：软件测试；程序变异；Simulink 模型测试；测试用例生成；模拟退火算法
中图分类号：TP311
文献标识码：A **文章编号**：1001-5965(2015)03-0391-07

随着模型驱动软件设计思想(MBD)的广泛应用，软件开发的重心已由传统的代码设计转移到建模及模型的转换上[1]. 如果能在完成软件初步设计的同时，及早发现并修复模型中的错误，不仅能缩短后期的代码测试周期，还能提高模型的可靠性，改善软件产品的质量. 因此，越来越多的研究开始关注高层次的模型验证与测试工作.

Simulink 是 Matlab 提供的一个用于对动态系统进行建模、仿真和分析的工具. 它为用户建模提供了一个图形化的用户界面，通过不同类型模块库中的功能模型来完成系统的建模. 目前，许多航空机载安全关键软件已运用 Simulink/RTW 技术进行开发，但仍存在缺乏完善的模型测试充分性准则及自动高效的测试用例生成等问题[2]. 不同于基于控制流和数据流的测试充分性准则，程序变异是一项用于评价测试优良程度的有效技术，它为测试评价和测试增强提供了准则. 将程序变异技术应用于 Simulink 的模型测试，不仅可以为 Simulink 模型提供测试充分性准则，还可以用来指导设计较强发现故障能力的测试用例生成. 但是由于 Simulink 模型的变异测试过程中存在的测试执行开销大和测试用例生成效率低两个问题，是将变异测试技术从学术界研究转化为工业界应用所面临的主要技术难题.

本文研究设计了针对 Simulink 模型测试的改进变异算子集，在不影响测试用例集变异评分的情况下，该组变异算子集能够减少变异模型的生成数量，从而有效降低测试开销. 在此基础上设计了一种基于搜索的 Simulink 模型变异测试用例生成方法.

1 相关工作

程序变异(program mutation)是一种面向缺陷的测试技术，最早由 DeMillo 等在文献[3]中提

收稿日期：2014-04-17；**录用日期**：2014-05-21；**网络出版时间**：2014-07-01 17:01
网络出版地址：www.cnki.net/kcms/doi/10.13700/j.bh.1001-5965.2014.0211.html
基金项目：航空科学基金资助项目(20095551025)；中央高校基本科研业务费专项资金资助项目(YWF-11-03-Q-114)
作者简介：周艺斌(1990—)，男，山西运城人，博士生，zhouyibin@buaa.edu.cn
*通讯作者：殷永峰(1978—)，男，山东潍坊人，副教授，yyf@buaa.edu.cn，主要研究方向为软件可靠性工程.

引用格式：周艺斌，殷永峰，李骁丹，等. 基于程序变异的Simulink 模型测试方法[J]. 北京航空航天大学学报，2015，41(3)：391-397. Zhou Y B，Yin Y F，Li X D，et al. Simulink model testing method based on program mutation[J]. Journal of Beijing University of Aeronautics and Astronautics，2015，41(3)：391-397 (in Chinese).

出,主要应用于单元测试,在接口测试、面向方面测试及面向对象测试中也有相关的理论研究. 它依赖于两个基本原则:其一是称职程序员假设,即假设熟练的程序员写出的是一个接近于正确的程序 P;另一个是耦合效应假设,即若测试用例可以检测出简单缺陷,则该测试用例也易于检测到更为复杂的缺陷.

程序变异的基础是变异算子集. 变异算子是在符合语法规则前提下,从原有程序生成差别极小程序(变异体)的转换规则[4]. 文献[5]于1987 年针对 Fortran77 语言首次定义了 22 种变异算子,这组算子集为后来的其他编程语言变异算子的设计提供了重要的依据. 该组变异算子主要分为 4 种类型,即常量变异、操作法变异、语句变异及变量变异,实际中每种分类下面都有很多个变异算子. 每种分类下面的变异算子的类型和数量依赖于针对的编程语言.

程序变异技术虽然已有较多研究成果,但其应用却存在分析过程中计算开销过大的技术难题. 大量变异模型的生成使得测试分析工作的开销及其高昂,为此需要有效的优化方法来减小计算开销. 文献[6]中首先提出了选择性变异的方法,即忽略 ASR 和 SVR 两个可以生成 30%~40% 变异体的变异算子. 这种策略被称为“2-selective mutation”,文献[7]将这种策略延伸为“N-selective mutation”. 实验结果表明,使用“N-selective mutation”策略后的变异评分均值仍可以保持较高,并明显降低了变异体数量. 在以上的实验研究分析下,文献[8]将文献[5]中提出的 22 种变异算子进一步分为了操作数类算子、表达式类算子和语句类算子. 通过对每一类变异算子的分析,最终确定了 5 个最为重要的变异算子:ABS 算子、AOR 算子、LCR 算子、ROR 算子、UOI 算子,实验结果表明,使用这 5 个变异算子仅将其变异评分降低了 0.5%. 文献[9]中针对 Proteum 测试工具为 C 语言的变异算子设计了优化的方法,提出了选择出充分变异算子集的 6 条指导策略. 即:①考虑能获得高变异评分的变异算子. ②考虑每个变异算子类中的一个算子. ③依据实验将包含在高效算子中的算子除去. ④建立增量策略. ⑤考虑能够在变异评分上提供增量的变异算子. ⑥考虑具有高可信度的算子.

在上述研究分析的基础上,本文拟在不影响变异评分的前提下,以上述 6 条策略为指导,通过对 Simulink 变异算子进行约简优化来大规模减少变异体数量,从而减小变异测试的计算开销.

2 基于程序变异的模型测试过程

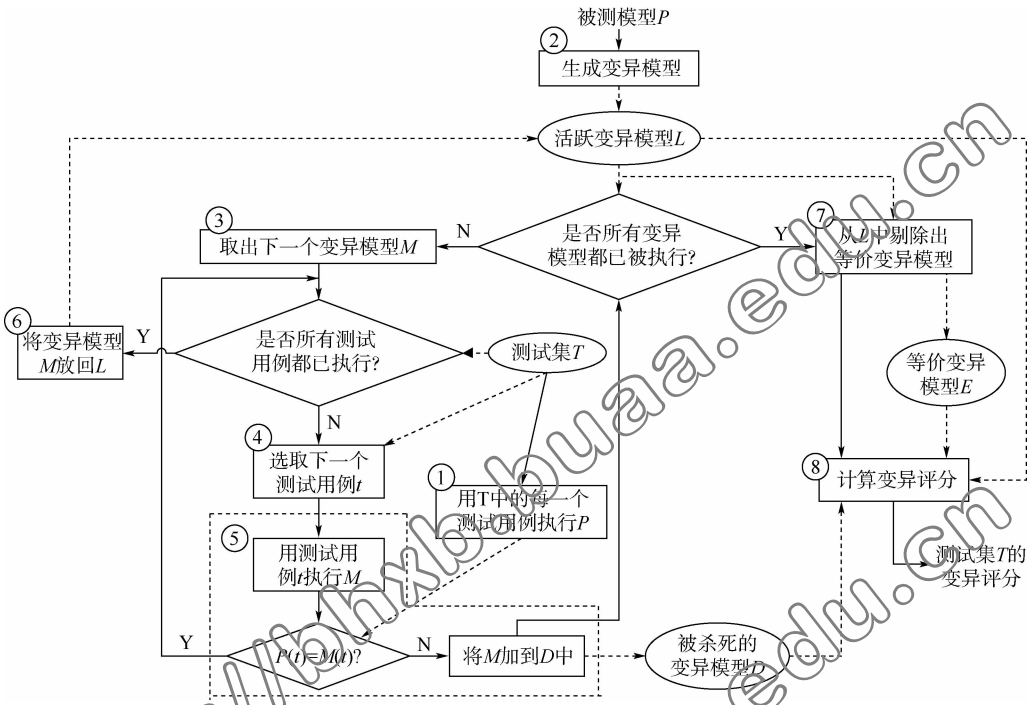
许多航空机载安全关键软件已运用 Simulink/RTW 技术进行开发,但仍存在以下问题:①仍缺乏完善的测试充分性度量证明模型的测试是充分的. 不同的软件测试人员使用不同的充分性准则标准,没有统一的标准. ②如何自动高效生成满足 Simulink 模型测试所需的测试用例集,仍是一个亟待解决的问题. ③虽然已有众多工具应用于模型驱动的软件开发过程领域,但是对Simulink模型的测试工具研究尚属于起步阶段.

考虑 Simulink 环境特点,设置被测单位为一个系统模型,在生成变异模型的过程中不对子系统的模块进行变异. 这时子系统类似于代码中的调用函数,在以子系统为被测单位时再对其模块进行变异. 本文提出了基于程序变异的 Simulink 模型测试过程,如图 1 所示.

过程具体说明如下:①用已有的测试集 T 执行原始 Simulink 模型 P. ②根据设定好的变异算子生成活跃变异模型 L 集合. ③选取一个未考虑过的活跃变异模型 M. ④选取未执行过的测试用例 t. ⑤使用测试用例 t 执行变异模型 M,检查针对测试用例 t 执行 M 产生的结果与执行 P 产生的结果是否相同. 若相同则返回步骤④,选取下一测试用例;若不相同则称为 t 杀死了变异模型 M,将 M 添加到被杀死的变异模型 D 集合中. ⑥当测试集 T 中没有测试用例能将变异模型 M 杀死时,将 M 放回到活跃变异模型 L 中. ⑦检查 L 集合是否为空. 若不为空,测试其与原始模型 P 的等价性. 从 L 中剔除出等价变异模型 E. ⑧计算测试用例集 T 的变异评分(mutation score). 给定集合 L,D 和 E,用 $S_M(T)$ 表示 T 的变异评分,则

$$S_M(T)=\frac{|D|}{|M|-|E|}=\frac{|D|}{|L|+|D|}$$

正如上式所示,一个测试集的变异评分总是介于 0~1 之间. 如果测试集 T 能够杀死除等价变异模型外的所有变异模型,则 $|L|=0$,变异评分 $S_M(T)$ 为 1,该测试集 T 的发现错误能力较强. 反之 T 不能杀死任何一个变异模型,则 $|D|=0$,变异评分 $S_M(T)$ 为 0,测试集 T 的发现错误能力较弱. 测试用例 t 杀死变异模型为当且仅当测试用例 t 使得变异模型的最终输出与原模型的最终输出不同.



实线—指向测试过程的下一步骤;虚线—数据在数据库、步骤间的流动.

图 1　基于程序变异的 Simulink 模型测试方法流程

Fig. 1　Process of Simulink model testing method based on program mutation

3　Simulink 模型的改进变异算子集

Simulink 包含很多现成的模块库,主要有输入源库、离散及连续系统库、数学运算库及非线性系统库.根据以上通用的模块库,考虑 Simulink 建模过程中的典型错误,文献[10]中首次研究了 Simulink 模型的故障模型,包括类型故障、变量故障、常量故障、连续离散时间故障、语句故障和表达式故障,并基于上述故障模型,针对 Simulink 模型提出了一组变异算子集.不同于传统程序变异的算子,该组算子集针对 Simulink 中常用的不同模块进行变异,如 Switch 模块的门限发生变异、积分及延时模块的变异.文中将其提出的变异算子集应用于一个二次方程模型的测试过程,通过计算 4 组测试用例集的变异评分,从而选择出最优用例集.文献[10]针对一组典型 Simulink 模型设计了变异测试实验,本文对实验结果进行了统计.该实验共生成 59 个变异模型,其中,TRO 算子生成的变异模型数量最多,且该实验的等价变异模型全部由 TRO 算子产生,根据 N-selective mutation 策略,本文首先将 TRO 算子忽略,不仅有效减少生成 27.11% 的变异模型,还可以减少等价变异模型的生成,从而大幅降低测试开销.随后的实验验证了优化方案的必要性.

基于充分变异算子的指导准则,上述算子集可以分为如下 5 类:数据类型变异(TRO)、变量变异(VCO,VNO)、常量变异(CCO,CRO,DCO)、Switch 模块变异(SCO,SSO)和表达式变异(ROE,AOR,ASR,LOR).考虑文献[9]中的 6 条变异算子优化策略,依次对每一类里的变异算子进行分析.本文选择出 6 种重要的变异算子:VCO,CRO,SCO,DCO,ROR,AOR.图 2 是一个包含基本模块的 Simulink 模型,其主要功能是判断二次方程解的类型,通过对该模型的变异测试实验表明,以上 6 种变异算子组成的针对 Simulink 模型的改进变异算子集比原有的变异算子集,在变异模型生产数量和变异评分方面都有进步.

在图 2 的二次方程模型解类型检验模型中,本文选择了 6 组测试用例集(每组包含 6 个测试用例),并执行于由上面 6 种算子生成的 29 个变异模型,得出了如表 1 所示的实验结果.本文中变异模型均通过修改模块参数以实现算子对模型的变异操作.

从表 1 中的结果可看出,改进后的变异算子集能够显著减少变异模型的数量,并且略微提高了测试用例集的变异评分.由于其中随机生成的测试用例的广泛性不够,测试用例集 T_3 的变异评分偏低.对于未被杀死的非等价变异模型,在第 4 节中将介绍基于搜索的用例生成方法,在改进变异算子集的基础上,主要关注于如何生成能够杀死上面提到的变异模型的测试用例.

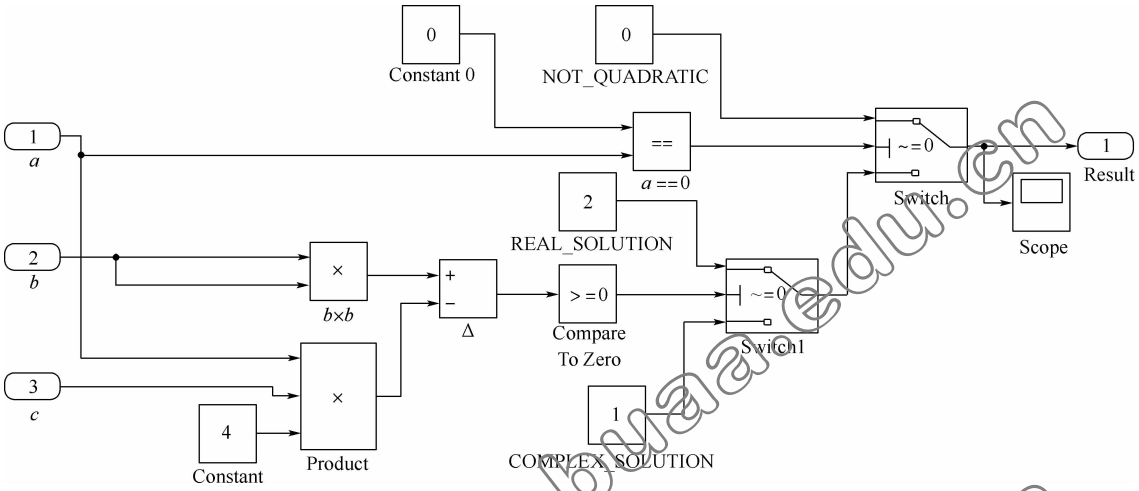


图 2 二次方程式模型

Fig. 2 Quadratic model

表 1 用例集的测试结果

Table 1 Testing results of test sets

测试用例集	原变异算子集				改进后的变异算子集			
	杀死变异模型	存活变异模型	等价变异模型	变异评分/%	杀死变异模型	存活变异模型	等价变异模型	变异评分/%
T_1	52	3	4	94.55	29	0	0	100
T_2	48	7	4	87.27	26	3	0	89.65
T_3	51	4	4	92.73	26	3	0	89.65
T_4	54	1	4	98.15	29	0	0	100
T_5	51	4	4	92.73	28	1	0	96.55
T_6	50	5	4	90.91	27	2	0	93.1
平均	51	4	4	92.73	27.5	1.5	0	94.83

4 基于搜索的测试用例生成

由图 1 可看出,当用例集 T 的测试用例都不能杀死变异模型,则需要向 T 中添加用例,其中测试用例生成是最花费人力和时间的过程. 文献[11]中提出了一个自动化的测试用例生成框架,该框架采用搜索算法为结构化的模型生成了测试用例,实验证明该组用例能够达到较高的结构化覆盖准则,但是该实验仅选用了 3 个变异算子,对于变异测试的充分性不足. 本文通过采用第 3 节中的改进算子集,更真实地模拟了 Simulink 仿真过程中可能出现的错误. 文献[12]研究了基于搜索的 Simulink 模型测试数据生成法,针对 Simulink 模型复杂性的特点,采用模拟退火算法对目标函数求优. 但是,满足结构化覆盖标准的测试用例有时并不能够发现 Simulink 模型中的一些错误,如逻辑模块故障、Switch 模块故障等. 由于结构化覆盖标准下的测试用例仅能保证覆盖特定路径,如果上述模块发生故障(如设计人员将 Switch 模块的门限“>0”错写为“>=0”)的情况下,该用例仍然可覆盖该特定路径,则该故障不能被发现. 因此如何高效地从 Simulink 模型产生符合高变异评分的测试用例是本节研究的重点.

基本思想是:反复执行被测程序,根据执行过程中搜集到的数据判断当前输入满足特定测试需求的程度,借助反馈机制,逐渐调整输入直到满足测试需求为止. 本文考虑对于指定路径上不满足要求的分支,将测试数据生成问题转化为函数极小化问题,其重点在于如何选择合适的目标函数和函数极小化方法.

4.1 构造代价函数

根据文献[13]提出的代价函数,本文设计了代价函数定义基本规则和变异模型的代价函数构造规则,其中基本规则如表 2 所示.

表 2 代价函数

Table 2 Cost functions

断言	代价函数值
布尔型	满足时为 0,否则为 K
$E_1 < E_2$ 或 $E_1 \leqslant E_2$	满足时为 0,否则为 $E_1 - E_2 + K$
$E_1 = E_2$	满足时为 0,否则为 $\lvert E_1 - E_2 \rvert + K$
$E_1 \neq E_2$	满足时为 0,否则为 K
$E_1 \vee E_2$	$\dfrac{\cos(E_1) \times \cos(E_2)}{\mathrm{cost}(E_1) + \mathrm{cost}(E_2)}$
$E_1 \wedge E_2$	$\mathrm{cost}(E_1) + \mathrm{cost}(E_2)$

表 2 中,K 表示当断言不满足时的代价函数值,可用于比较各断言之间的差距. 例如,对于断言来说 $A<5$,$A=6$ 比 $A=10$ 更接近于该断言,$A=6$的代价函数值更小. 为了构造 Simulink 不同模块之间的代价函数,文献[14]中提出为了使得原模型与变异模型的输出产生差异,必须满足两个条件:

1) 变异模块的输出必须发生改变;

2) 该输出的改变必须影响到整个模型的输出.

一般模块变异后,条件 1 能够满足. 但是条件 2 则需要研究模型的结构,并保证从变异模块到系统输出之间的路径上,每个模块的输出与原模型的不同. 如果系统模型包含多个输出,则至少 1 个输出不同就认为被杀死. 针对本文提出的优化变异算子集,不同位置的不同类型变异均有不同的代价函数,下面将逐一介绍代价函数的构造.

4.1.1　常量模块与变量模块

如果模块的变异类型为常量变异与变量变异,则其代价函数应该为:$C=C_{CV}+C_A$. 其中,C_{CV} 为常量模块与变量模块变异后需不同于原模块的代价,C_A 为该模块变异后其改变需影响到整个模型的输出的代价.

4.1.2　Switch 模块

图 3(a)代表受影响的信号输入 Switch 模块的第 1 或第 3 输入,此时其代价函数应为:$C=C_D+C_S+C_A$. 其中,C_D 表示受影响的信号需不同于原模型中信号的代价,C_S 表示若受影响的信号输入 Switch 模块的第 1 输入时,第 2 输入信号达到门限的代价;若受影响的信号输入 Switch 模块的第 3 输入时,第 2 输入信号未达到门限的代价.

图 3(b)代表受影响的信号输入 Switch 模块的第 2 输入,此时其代价函数应为:$C=C_D+(C_{S1S3}+C_{\bar{P}M})\vee(C'_{S1S3}+C_{P\bar{M}})+C_A$. 其中,$C_{S1S3}$ 为原 Switch 模块的第 3 输入与变异 Switch 模块的第 1 输入不同的代价,C'_{S1S3} 为原 Switch 模块的第 1 输入与变异 Switch 模块的第 3 输入不同的代价.

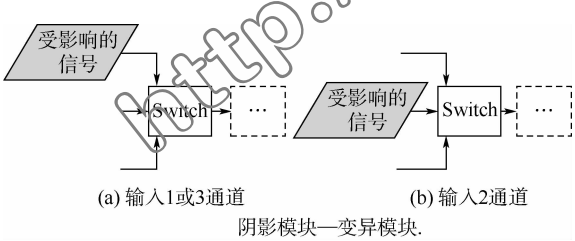


图 3　受变异的信号输入 Switch 模块

Fig. 3　Mutated signal input Switch blocks

4.1.3　表达式模块

如果模块的变异算子为 ROR,AOR 或 LOR 时,则其代价函数应为:$C=C_{P\neq M}+C_A$. 其中,$C_{P\neq M}$ 为输入经过变异表达式操作的输出值不同于原模块的代价.

图 4(a)代表受影响的信号输入表达式模块的任意输入端口,其代价函数为:$C=C_D+C_{P\neq M}+C_A$. 图 4(b)代表受影响的信号输入两个表达式模块的任意输入端口,其代价函数为:$C=C_D+(C_{O1}\vee C_{O2})$,其中 $C_{O1}=C_{P1\neq M1}+C_{A1}$,$C_{O2}=C_{P2\neq M2}+C_{A2}$.

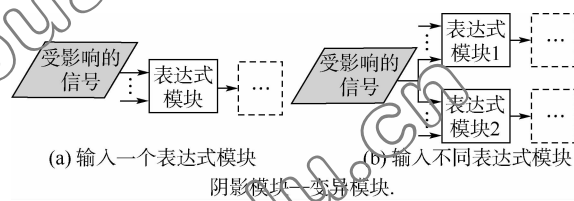


图 4　受变异的信号输入表达式模块

Fig. 4　Mutated signal input expression blocks

结合表 2 中的代价函数构造规则,可以递归地从变异模块到最终输出来计算,从一组随机测试用例中搜索出能够杀死该变异模型的测试用例的代价函数. 假设图 3 中模型受变异算子 ROR 影响生成变异模型,其中 Compare To Zero 模块的“>=”被替换为“<=”. 如果该变异模型在图 1 中步骤⑤中没有被杀死,则其代价函数应为

$$C=C_{P\neq M}+C_A=$$
$$C_{P\neq M}+(C_{S1S3}+C_{\bar{P}\bar{M}})\vee(C'_{S1S3}+C_{P\bar{M}}+C_S)=$$
$$\mathrm{cost}(b^2-4ac)\neq 0)+(\mathrm{cost}(1\neq 2)+0)\vee$$
$$(\mathrm{cost}(2\neq 1)+0)+\mathrm{cost}(a\neq 0)=$$
$$\mathrm{cost}(b^2\neq 4ac)+\mathrm{cost}(a\neq 0)$$

4.2　搜索杀死特定变异模型的测试用例

在 4.1 节中将测试用例生成问题转化为了代价函数值极小化问题的基础上,采用模拟退火算法来解决代价函数极小化问题[15]. 首先,模拟退火算法能够跳出初始输入的局部最优解,搜索到目标输入的稳定性较高,对 Simulink 模型中各种复杂情况具有更高适应性[12]. 另外,由于从 R2009a 版本开始,Matlab 自带的优化工具箱集成了模拟退火算法,因此用 Matlab 环境来验证本文的测试数据生成算法方便易行. 在第 5 节中将通过典型实例来简要介绍该算法的应用和变现.

5　实例验证

本节通过设计 4 组图 2 中模型的变异模型来验证第 4 节的测试数据生成方法. 表 3 为图 2 中模型的 4 个变异模型,具体描述如表 3 所示.

表 3 实验变异模型描述

Table 3 Description of mutated models in experiment

编号	变异模块	算子类型	变异描述	系统代价函数
1	Compare To Zero	ROR	“ > = ”被替换为“ > ”	$\mathrm{cost}(b^2=4ac)+\mathrm{cost}(a\neq 0)$
2	Switch1	SCO	“ ~ = 0”被替换为“ ~ = 1”	$\mathrm{cost}(a\neq 0)$
3	REAL_SOLUTION	CRO	“2”被替换为“1”	$\mathrm{cost}(b^2\geqslant 4ac)+\mathrm{cost}(a\neq 0)$
4	$b\times b$	AOR	“ × ”被替换为“ + ”	$\mathrm{cost}(b^2\neq 2b)+\mathrm{cost}((b^2\geqslant 4ac)\wedge(2b<4ac))\vee$ $\mathrm{cost}((b^2<4ac)\wedge(2b\geqslant 4ac))+\mathrm{cost}(a\neq 0)$

这里设置代价函数的 $K=1$，由定义可知，目标函数的最优值为 0. 即可使目标函数为 0 的测试输入能够杀死对应的变异模型. 对于系统的代价函数及以上 4 组变异模型的目标函数，可用 Matlab 脚本编写. 用 Matlab 编写的测试数据生成脚本如下：

```
ObjectiveFunction = @ obj_function;% 目标函数句柄设定
startingPoint = [010];% 初始值设定
lb = [ -10 -10 -10];% 输入值下限
ub = [101010];% 输入值上限
options = saoptimset('InitialTemperature', 1 000, 'TemperatureFcn', @ temperatureexp, 'AnnealInterval', 500, 'PlotFcns', {@ saplotbestx, @ saplotbestf, @ saplotx, @ saplotf});% 模拟退火算法参数设定
[x, fval, exitFlag, output] = simulannealbnd(ObjectiveFunction, startingPoint, lb, ub, options);% 执行模拟退火算法
```

这里设置模拟退火算法的终止条件为目标函数达到 0，起始温度为 1 000，冷却率为 0.95. 通过分别编写目标函数，执行算法生成测试数据，运行上述脚本，4 组模型的执行结果如图 5 所示. 最优目标函数表示在横轴的迭代次数下得到纵轴的目标函数值（最优目标函数为 0），最优变量值表示当目标函数达最优时的测试数据.

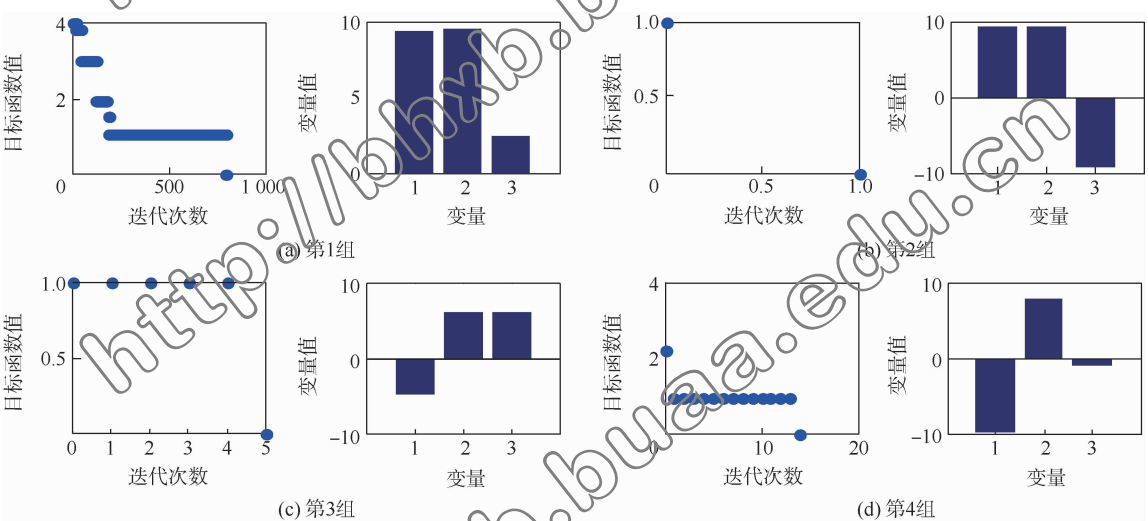


图 5 搜索迭代次数与最优变量输入值

Fig. 5 Searching iteration number and optimal input value of variables

由图 5 可看出，通过采用模拟退火算法经过一定迭代次数均可使目标函数达到最优值，即可以找到需要的测试数据. 除第 1 组的迭代次数为 761 次外，另 3 组的平均迭代次数不超过 30 次，就可以找到满足代价函数为 0 的测试输入. 因此，当变异模型不能被现有测试用例杀死而需要额外添加测试用例时，使用本文方法能够有效实现.

6 结 论

本文基于程序变异技术提出了 Simulink 模型的变异测试过程和一组改进变异算子集，并相应设计了一种基于搜索的 Simulink 模型变异测试用例生成方法，经实验验证表明：

1）该组改进变异算子集能够减少变异模型的数量，并且保证测试用例集的变异评分基本不变；

2）对于 Simulink 基本模块库中的不同变异模块，基于搜索的 Simulink 模型变异测试用例生成方法能够快速准确地生成满足测试要求的测试用例.

今后研究将从以下几个方面展开:本文采用手工方式构造实验变异模型的代价函数,未实现代价函数构造自动化.仅考虑了 Simulink 中的基本模块库,对于其他特定领域的模块库,其变异算子的设计有待继续研究.等价变异模型判定、子系统模块处理等问题的优化,也是未来值得研究的方向.

参考文献(References)

[1] Molina J M, Pan X, Grimm C, et al. A framework for model-based design of embedded systems for energy management[C]// Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES). Piscataway, NJ: IEEE, 2013: 1-6.

[2] He N, Rümmer P, Kroening D. Test-case generation for embedded Simulink via formal concept analysis[C]//Proceedings of the 48th Design Automation Conference. New York: ACM, 2011: 224-229.

[3] DeMillo R A, Lipton R J, Sayward F G. Hints on test data selection: help for the practicing programmer[J]. Computer, 1978, 11(4): 34-41.

[4] Jia Y, Harman M. An analysis and survey of the development of mutation testing[J]. IEEE Transactions on Software Engineering, 2011, 37(5): 649-678.

[5] King K N, Offutt A J. A fortran language system for mutation-based software testing[J]. Software: Practice and Experience, 1991, 21(7): 685-718.

[6] Mathur A P. Performance, effectiveness, and reliability issues in software testing[C]//15th Annual International Computer Software and Applications Conference. New York: IEEE, 1991: 604-605.

[7] Offutt A J, Rothermel G, Zapf C. An experimental evaluation of selective mutation[C]//Proceedings of the 15th International Conference on Software Engineering. Piscataway, NJ: IEEE Computer Society Press, 1993: 100-107.

[8] Offutt A J, Lee A, Rothermel G, et al. An experimental determination of sufficient mutant operators[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 1996, 5(2): 99-118.

[9] Barbosa E F, Maldonado J C, Vincenzi A M R. Toward the determination of sufficient mutant operators for C[J]. Software Testing, Verification and Reliability, 2001, 11(2): 113-136.

[10] Binh N T. Mutation operators for Simulink models[C]// Knowledge and Systems Engineering (KSE), 2012 Fourth International Conference on. Piscataway, NJ: IEEE, 2012: 54-59.

[11] Zhan Y, Clark J. Search based automatic test-data generation at an architectural level[C]//Genetic and Evolutionary Computation-GECCO 2004. Berlin: Springer, 2004, 1413-1424.

[12] 邓绍鹏,杨志义,王宇英. 基于搜索的 Simulink 测试数据生成[J]. 计算机应用研究, 2012, 29(7): 2527-2530.
Deng S P, Yang Z Y, Wang Y Y. Search-based test-data generation for Simulink[J]. Application Research of Computers, 2012, 29(7): 2527-2530(in Chinese).

[13] Bottaci L. Predicate expression cost functions to guide evolutionary search for test data[C]//Genetic and Evolutionary Computation—GECCO 2003. Berlin: Springer, 2003: 2455-2464.

[14] Zhan Y, Clark J A. Search-based mutation testing for Simulink models[C]//Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. New York: ACM, 2005: 1061-1068.

[15] McMinn P. Search-based software test data generation: a survey[J]. Software Testing, Verification and Reliability, 2004, 14(2): 105-156.

Simulink model testing method based on program mutation

ZHOU Yibin, YIN Yongfeng*, LI Xiaodan, WANG Mingwei

(School of Reliability and Systems Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract: In order to solve the current problems (expensive testing cost and low efficiency of test case generation) in mutation test for Simulink models, a mutation testing process and an optimized set of mutation operators were proposed for Simulink models based on program mutation according to the current mutation operators for the Simulink models. Experiments show that this set of mutation operators can effectively reduce the generation number of mutation models without prejudice to the mutation score of testing case set, thus it will effectively save the testing cost. Then a search-based test case generation method for Simulink models mutation testing was described. The test case generation problem was transformed into the objective function minimization problem, and the test cases which can kill the mutation models were ultimately obtained through the optimization of objective function by algorithm of simulated annealing. Finally, the application of a typical case for the method verified the correctness and effectiveness.

Key words: software testing; program mutation; Simulink model testing; test case generation; algorithm of simulated annealing