文章编号： 1671-7449(2025)01-0020-07

基于改进优序图法的读时故障注入设计

范烁阳[1,2,3]，王明亮[1,2*]，施敏华[1,2]，常　亮[1,2]，王国华[1,2]

（1. 中国科学院微小卫星创新研究院，上海 201203；2. 上海微小卫星工程中心，上海 201203；
3. 中国科学院大学，北京 101408）

摘　要： 软件故障注入是可靠性验证的重要保障，与系统的安全性和鲁棒性密切相关。提出了一种基于改进优序图法的读时故障注入方法。相比于传统方法，所提方法能够提高故障注入的有效性，缩小故障空间。首先，对小尺度程序故障注入的结果进行理论分析，得到特征值权重向量；然后，对大尺度程序的汇编代码进行解析，得到汇编代码的特征值；之后，根据特征值权重向量，计算大尺度程序汇编代码的权重；最后，在MiBench数据集上验证方法的有效性。将大尺度程序各条汇编代码分别按照权重和有效故障率进行降序排列，各取前50%的元素，二者的相似度可达到81%。

关键词： 软件故障注入；读时注入；优序图法；可靠性；软件测试
中图分类号： TP311.56　　**文献标识码：** A　　**doi:** 10.62756/csjs.1671-7449.2025004
引用格式： 范烁阳，王明亮，施敏华，等. 基于改进优序图法的读时故障注入设计[J]. 测试技术学报，2025，39(1)：20-26.
FAN Shuoyang，WANG Mingliang，SHI Minhua，et al. Read-time fault injection design based on improved priority graph method[J]. Journal of Test and Measurement Technology，2025，39(1)：20-26.

Read-Time Fault Injection Design Based on Improved Priority Graph Method

FAN Shuoyang[1,2,3]，WANG Mingliang[1,2*]，SHI Minhua[1,2]，CHANG Liang[1,2]，WANG Guohua[1,2]

（1. Innovation Academy for Microsatellites of Chinese Academy of Sciences，Shanghai 201203，China；
2. Shanghai Engineering Center for Microsatellites，Shanghai 201203，China；
3. University of Chinese Academy of Sciences，Beijing 101408，China）

Abstract： Software fault injection is an important guarantee for reliability verification and is closely related to the security and robustness of the system. This paper proposes a read-time fault injection method based on an improved priority graph method. Compared with traditional methods，the new method can improve the effectiveness of fault injection and reduce the fault space. Firstly，the results of small-scale program fault injection are theoretically analyzed to obtain the eigenvalue weight vector. Then，the assembly code of the large-scale program is analyzed to obtain the characteristic values of the assembly code. Then，based on the eigenvalue weight vector，the weight of the large-scale program assembly code is calculated. Finally，the effectiveness of the method is verified on the MiBench data set. Arrange each assembly code of the large-scale program in descending order according to weight and effective failure rate，and take the

收稿日期： 2024-04-30
作者简介： 范烁阳(2000—)，男，硕士生，主要从事软件故障注入研究。E-mail：fanshuoyangtyut@163.com。
***通信作者：** 王明亮(1981—)，男，高级工程师，博士，主要从事空间目标轨道预报性能评估与改进、航天软件缺陷检测理论与方法研究。E-mail：13761306729@139.com。

top 50% of the elements. The similarity between the two can reach 81%.

Key words: software fault injection; read-time injection; priority diagram method; reliability; software test

0 引 言

随着计算机系统在各个领域的广泛应用，其安全性和关键性变得尤为重要。因此，开发者会为系统设计容错机制，使得系统即便在出现故障的情况下，也可以提供正确的服务[1]。为了验证容错机制是否满足设计要求，需要对系统进行可靠性测试。软件故障注入作为一种可靠性验证方法，在系统测试中起到了重要作用，其优势在于成本低且易于控制[2]。如果在测试环节中缺少软件故障注入，将会导致测试不充分、成本高、可移植性差等问题。

软件故障注入大致可以分为两类：基于程序代码的故障注入和基于存储单元的故障注入。

基于程序代码的故障注入是通过改变源代码或二进制代码来模拟单粒子效应。Yayan等[3]提出了一种基于突变的定制软件故障注入工具（IM-FIT），用于评估安全关键系统软件的鲁棒性，节省了26%的系统运行时间。Ahmad等[4]提出了一种可配置位置感知故障注入技术（CAFI），CAFI导致的程序崩溃率比高层次软件故障注入导致的程序崩溃率高18%以上。Park等[5]提出了一种基于汽车开放系统架构（AUTOSAR）的故障注入方法，应用于ECU软件开发过程，程序运行时间较原始运行时间仅增加了1.24%。Ahmad等[6]提出了一种基于轻量级动态软件的故障注入（LDSFI）技术，不需要重新编译源代码，不会增加程序的运行时开销。基于程序代码的软件故障注入能够在代码的指定位置注入故障，验证系统的可靠性，可是这类方法往往与目标系统高度相关，可移植性较低。

针对软件故障注入可移植性低的问题，一些研究者又提出了基于存储单元的故障注入。该故障注入技术通过改变寄存器或内存的值来模拟单粒子效应。林灏铨[7]设计与实现了一款基于KM22智能处理平台的软件故障注入及测试系统，通过实验发现，代码段和栈段对故障注入较为敏感，堆段最不敏感。Bodmann等[8]通过软件故障注入提供了更准确的设备FIT率估计方法。Folkesson等[9]提出了一种称为信号错误空间修剪的新技术，可以使故障空间减少约30%～43%。基于存储单元的软件故障注入能够模拟瞬时故障或持久故障，但此类方法故障注入的有效性较低，大部分故障注入未能影响程序的输出结果。

针对软件故障注入有效性低的问题，本文提出了基于改进优序图法的读时故障注入方法，提高了故障注入的有效性。首先，对小尺度程序进行故障注入；然后，结合汇编代码和故障注入结果进行分析，构建判断矩阵，得到特征值权重向量，权重计算方法在传统优序图法上进行了改进，降低了主观性；之后，对大尺度程序进行解析，得到汇编代码的各项特征值，根据上一步得到的特征值权重向量，计算大尺度程序汇编代码的权重；最后，在MiBench数据集上进行实验，验证方法的有效性。

1 汇编代码特征

读时故障注入模型通过分析小尺度程序汇编代码的特征值和有效故障率构建判断矩阵，最终得到特征值权重向量。提取汇编代码特征是其中的关键环节，下面给出了一些具有代表性的特征。

1）写入周期：表示某寄存器最近一次写入和当前本次读出之间相隔的周期。在理想状态下，“周期”一词指的是“时钟周期”，但是在实际情况中，由于环境的不同，时钟周期会存在些许差异，无法保证输出结果的一致性。因此，本文采用指令数来代替时钟周期。换句话讲，写入周期指的是某寄存器最近一次写入和当前本次读出之间相隔的指令数。图1给出了示例汇编代码。地址2544处xmm0被写入，地址254a处xmm0被读出，地址254a与地址2544相隔的指令数为1，地址254a处xmm0的写入周期为1。同理，地址2556和255a处的xmm0写入周期分别为4和5。

2）生命周期：表示某寄存器上一次读/写和当前本次读出之间相隔的周期。需要注意的是，生命周期的定义与写入周期的定义略有不同。写入周期指的是“最近一次写”与当前读相隔的周期。生命周期指的是“上一次读/写”与当前读相隔的周期。例如，地址2544处xmm0被写入，地址254a处xmm0被读出，地址254a与地址2544相隔的指令数为1，

地址254a处xmm0的生命周期为1。同理，地址2556和255a处的xmm0生命周期分别为3和1。

```
2544:  movsd    0x18(%rsp),%xmm0
254a:  comisd   %xmm0, %xmm1
254e:  jae      0x4025f0<SolveCubic+304>
2554:  fstp     %st(0)
2556:  ucomisd  %xmm0, %xmm1
255a:  movapd   %xmm0, %xmm2
```

图 1 示例汇编代码

Fig. 1 Sample assembly code

3）读取进程：指从某寄存器最近一次写入开始，当前读取次数与总读取次数之比。地址2544处xmm0被写入，在后续汇编代码中，xmm0总共被读取3次。地址254a处xmm0被第一次读取，读取进程为1/3。同理，地址2556和255a处的xmm0读取进程分别为2/3和3/3。

2 小尺度程序和大尺度程序的划分标准

对源程序进行不同程度的修改，生成两个变体，分别为小尺度程序和大尺度程序。在数据层面对程序进行修改，不会改变程序的整体结构，小尺度和大尺度程序属于同构程序；在算法层面对程序进行修改，会改变程序的整体结构，小尺度和大尺度程序属于异构程序。因此，本文在数据层面对程序进行修改，生成小尺度与大尺度程序。

在数据层面对程序进行修改，可以采用如下3种方法：

1）设置不同的迭代次数：在小尺度程序中，设置更少的迭代次数；在大尺度程序中，设置更多的迭代次数。

2）设置不同的步长：在小尺度程序中，设置更大的步长；在大尺度程序中，设置更小的步长。

3）设置不同大小的数据集：在小尺度程序中，采用小数据集；在大尺度程序中，采用大数据集。

3 读时故障注入模型

3.1 改进优序图法架构

优序图法是一种基于主观判断的权重计算方法，用来分析各个特征值对结果的重要性。如果把写入周期、生命周期等特征的特征值放在一起同时考虑，将会是一件非常困难的事情。这时，可以转变思路，将所有特征值两两进行比较，构建判断矩阵，然后通过后续计算，最后给出重要性次序。

本文在优序图法的基础上进行了改进：新方法不再依赖专家打分，而是将汇编代码按照写入周期、生命周期、读取进程等特征的特征值进行分组，并结合汇编代码的有效故障率构建判断矩阵。改进后的优序图法相比于层次分析法，降低了对主观判断的敏感性。此外，改进后的优序图法不再需要进行一致性检验，降低了操作的复杂性。

3.2 改进优序图法流程

改进优序图法流程如图2所示。首先，将小尺度程序反汇编，对汇编代码进行特征分析，统计特征值。汇编代码具有多种特征，如写入周期、生命周期、寄存器类型等。每种特征具有多个特征值。例如，寄存器类型分为8位通用寄存器、16位通用寄存器、XMM寄存器等。

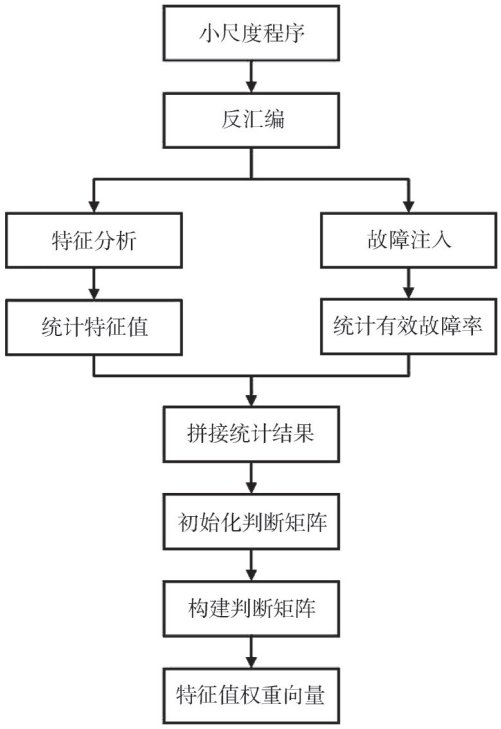

图 2 改进优先序图法流程

Fig. 2 Improved priority diagram method process

然后，根据反汇编结果进行故障注入，统计有效故障率。具体操作为：针对各条汇编代码，对源寄存器中的每一位分别进行单比特翻转，收集输出结果。输出结果分为两类：无效故障和有效故障。无效故障指故障注入的输出结果与正常的输出结果相同，有效故障指故障注入的输出结果与正常的输出结果不同。有效故障的产生原因有静默数据损坏、堆栈溢出、程序超时等。根据收集到的输出结果，

统计有效故障率。将上述步骤得到的特征值统计结果与有效故障率统计结果进行拼接。

接着，进行整个过程中最重要的一步：构建判断矩阵 J。构建判断矩阵流程如图 3 所示。传统的判断矩阵构建方法是利用专家打分的方式来实现的。本文对传统方法进行了一些改进：依据各个特征值的有效故障率来构建判断矩阵。

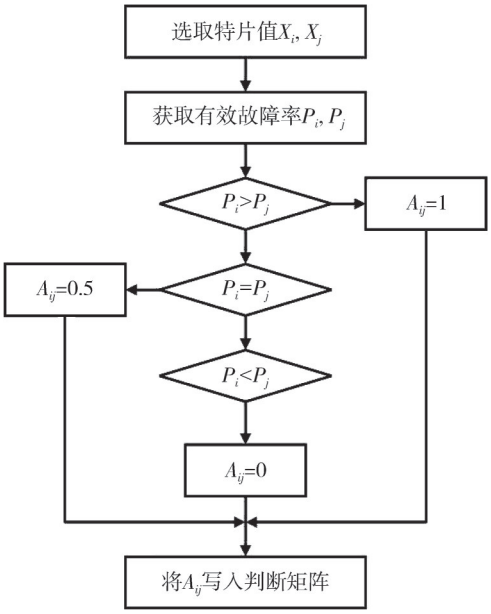

图 3　构建判断矩阵流程

Fig. 3　Build judgment matrix process

设X_1, X_2, …, X_n为某特征的n个特征值。根据各个特征值的有效故障率对所有特征值进行两两比较。式(1)中A_{ij}为X_i与X_j的有效故障率比较结果。若X_i比X_j有效故障率高，则A_{ij}的值为1；若X_i与X_j有效故障率相同，则A_{ij}的值为0.5；若X_i比X_j有效故障率低，则A_{ij}的值为0。例如，若X_1比X_2有效故障率高，则A_{12}的值为1；若X_1与X_2有效故障率相同，则A_{12}的值为0.5；若X_1比X_2有效故障率低，则A_{12}的值为0。至此，判断矩阵构建完成。

$$J=\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1j} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2j} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{i1} & A_{i2} & \cdots & A_{ij} & \cdots & A_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nj} & \cdots & A_{nn} \end{bmatrix}。 \quad (1)$$

接着，对判断矩阵 J 各行进行求和。S_i 为矩阵第 i 行的行和，计算公式为

$$S_i=\sum_{j=1}^{n}A_{ij}。 \quad (2)$$

判断矩阵一行对应一个特征值，所以矩阵的行和代表对应特征值的得分。W_i 为第 i 个特征值的权重，计算公式为

$$W_i=\frac{S_i}{\sum_{i=1}^{n}S_i}。 \quad (3)$$

特征值权重向量 V 为

$$V=(W_1 \quad W_2 \quad \cdots \quad W_i \quad \cdots \quad W_n)^{\mathrm{T}}。 \quad (4)$$

根据上述步骤，对其他特征进行同样的操作，最终得到所有特征的特征值权重向量。

3.3　大尺度程序静态分析

首先，对大尺度程序进行反汇编得到汇编代码。然后，分析各条汇编代码，发掘每条汇编代码各项特征所对应的特征值。最后，根据小尺度程序得到的所有特征的特征值权重向量，计算大尺度程序各条汇编代码的权重。

根据1.1节内容，图1中地址2556处的汇编代码为ucomisd %xmm0, %xmml。其写入周期为4，生命周期为3，读取进程为2/3。ucomisd属于SSE2指令集中的比较指令（SSE2 Compare Instructions，SSE2CmpInst），是用于比较两个双精度浮点数大小关系的指令，是结果通常存储在标志位寄存器中。源寄存器只有一个，即xmm0。源寄存器类型为XMM寄存器。XMM寄存器是Intel处理器中的一个128位寄存器，主要用于存储和操作数据，支持单指令多数据（SIMD）运算和浮点数据处理。表1为图1地址2556处汇编代码各项特征的特征值以及对应的权重。将上述特征值对应的权重相加，结果为1.518 8，即该条汇编代码的权重。

表 1　示例汇编代码各项特征值权重

Tab. 1　The weight of each feature value in the sample assembly code

特征	特征值	权重
写入周期	4	0.058 6
生命周期	3	0.013 9
读取进程	2/3	0.041 3
指令类型	SSE2CmpInst	0.030 0
源寄存器个数	1	1.000 0
源寄存器类型	XMM registers	0.187 5
寄存器位数	128	0.187 5

4　故障注入实验

4.1　实验设置

数据集：在本文中使用MiBench数据集。

MiBench是一个开源的嵌入式基准测试集合，包含多个嵌入式程序。这些嵌入式程序被分为六类：汽车和工业控制、网络、安全、消费设备、办公自动化、通信。在本文实验中，使用了汽车和工业控制类的嵌入式程序，包括basicmath、bitcount、qsort。basicmath是一些简单的数学计算测试；bitcount通过计算整数数组中的位数来测试处理器的位操作能力；qsort是字符串快速排序程序。

故障模型：本文采用的是单比特翻转模型。理想情况下，应该同时考虑单比特翻转模型和多比特翻转模型，但是只有大约2%的多比特翻转引起的静默数据损坏（SDC）比单比特翻转引起的SDC高出5百分点以上[10]。因此，为了研究问题的方便，只考虑单比特翻转模型。

故障位置：本文选择在寄存器中注入故障。理想条件下，应该同时考虑寄存器故障注入和内存故障注入，但是从另一个角度考虑，在程序运行的过程中，内存中的数据最终会被送到寄存器中。因此，从某种意义上来讲，内存故障注入可以等效为寄存器故障注入。

故障时间：按照故障时间，可以将故障注入分为：读时注入和写时注入。本文选择读时注入。读时注入指的是在寄存器被读取之前注入故障，即在汇编代码中的源操作数中注入故障。所以，本文选择在各条汇编代码的源寄存器中注入单比特翻转故障。

尺度划分：通过设置不同的迭代次数、步长或数据集，控制程序的运行时长。分析较短运行时长的程序运行结果，预测较长运行时长的程序脆弱性。在MiBench数据集中，basicmath程序共有4个函数：SolveCubic、usqrt、deg2rad、rad2deg，每个函数采用不同的迭代次数或步长来划分小尺度程序和大尺度程序；bitcount程序设置了不同的迭代次数；qsort程序采用了不同大小的数据集。具体的尺度划分方式如表2所示。

表2 尺度划分方式

Tab. 2 Scale division method

程序		小尺度	大尺度
basicmath	SolveCubic	步长1	步长0.451
	usqrt	迭代数1001	迭代数100 000
	deg2rad	步长1	步长0.001
	rad2deg	步长PI / 180	步长PI/5 760
bitcount		迭代数75 000	迭代数11 25 000
qsort		输入为小数据集：单词列表	输入为大数据集：表示数据点的三元组的集合

4.2 实验步骤

实验设计思路如图4所示。这里需要准备两个程序：小尺度程序和大尺度程序。小尺度程序和大尺度程序功能相似，区别在于小尺度程序的输入数据规模要小于大尺度程序。

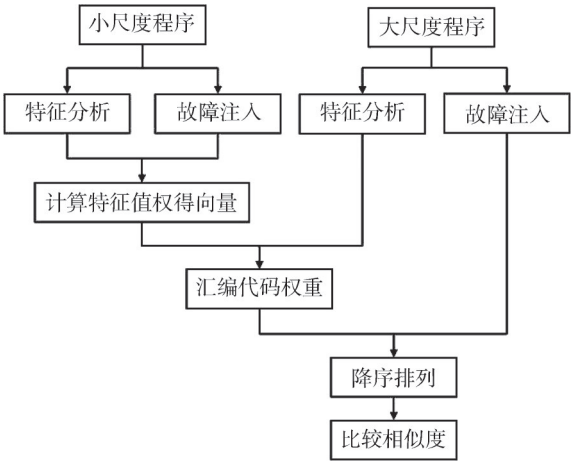

图4 实验设计思路

Fig. 4 Experiment design

首先，对小尺度程序进行特征分析和故障注入。根据特征分析结果和故障注入结果，计算特征值权重向量。此外，对大尺度程序进行特征分析和故障注入。然后，根据小尺度程序得出的特征值权重向量，结合大尺度程序得出的特征分析结果，计算大尺度程序各条汇编代码的权重。最后，对于大尺度程序各条汇编代码，分别按照权重和有效故障率降序排列，各取前x%的元素构成两个集合，比较两个集合的相似度。如果相似度较高，则说明汇编代码的权重可以一定程度上反映其脆弱性。

4.3 实验结果

在大尺度程序中共注入10 832条故障，其中有效故障5 253条，无效故障5 579条。根据图5，权重降序序列和有效故障率降序序列各取前50%的元素，二者的相似度可以达到81%。为了方便理解，这里进行举例说明。假设程序总共有100条汇编代码，先选取通过理论计算得到的权重较大的前50条汇编代码，再选取通过实际测试得到的有效故障率较大的前50条汇编代码，二者的相似度达到了81%，也就是说二者有40条汇编代码是相同的。实验结果表明，新方法可以在一定程度上预测汇编代码的脆弱性，提高故障注入有

效性，缩小故障空间。

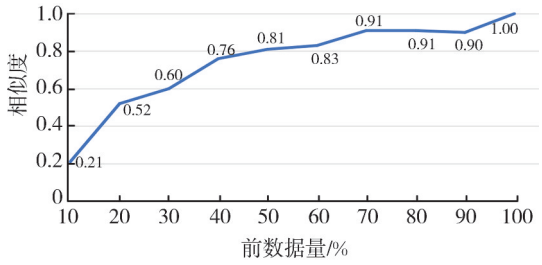

图 5　序列相似度

Fig. 5　Simalarity of sequence

选取有效故障率较大的前 50% 汇编代码，统计各个寄存器的占比。根据图 6 可以看出，rax、rbx、r12 这些寄存器的占比较高。这些频繁使用的寄存器更容易受到单比特翻转故障的影响。因此，可以赋予这些高利用率寄存器更高的权重。

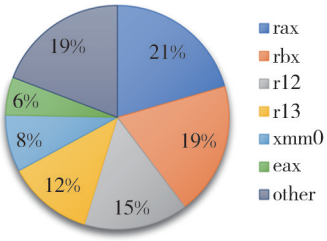

图 6　寄存器占比

Fig. 6　Register ratio

根据大尺度程序的故障注入结果，统计寄存器各个比特位的有效故障率。通过分析统计结果，可以更好地了解寄存器的脆弱性。图 7 绘制了浮点数寄存器 xmm0 各比特位有效故障率的热度图。

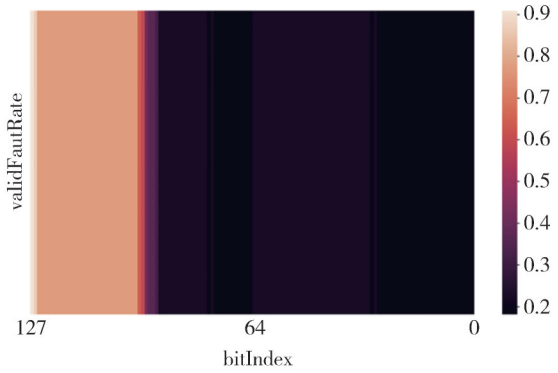

图 7　寄存器 xmm0 各比特位有效故障率热度图

Fig. 7　Heat map of valid fault rate of each bit of register xmm0

热度图的左边代表寄存器高位，右边代表寄存器低位。颜色越浅，该比特位有效故障率越高；颜色越深，有效故障率越低。由此可以发现，对于浮点数寄存器来说，高位的有效故障率要大于低位的有效故障率，这可能与程序结构或硬件设计有关。

以上实验均基于 basicmath 程序。为了验证方法的普适性，本文在其他两个程序 bitcount、qsort 上进行了测试。对于 bitcount 程序，有效故障率降序序列和权重降序序列分别取前 50% 的元素，二者的相似度可以达到 78%；对于 qsort 程序，相似度可以达到 74%。

SAFIRE 是一个用于并行、多线程应用程序的快速准确的故障注入框架[11]。LLTFI 是一款允许用户在较低级别（LLVM IR 级别）对应用程序进行故障注入实验的工具[12]。为了检验本文所提方法的效果，将其分别与 SAFIRE、LLTFI 进行对比实验。本文方法选择权重较大的汇编代码进行故障注入，而 SAFIRE、LLTFI 随机选择汇编代码进行故障注入。实验结果如表 3 所示，本文所提方法的有效故障数更高（有效故障指影响程序输出的故障注入操作），提高了故障注入效率。

表 3　本文故障注入方法与 SAFIRE、LLTFI 的对比实验

Tab. 3　Comparative experiment between the fault injection method proposed in this paper and SAFIRE, LLTFI

故障注入次数	有效故障数		
	本文方法	SAFIRE	LLTFI
848	796	234	563
1 808	1 665	700	1 198
2 704	2 352	1 025	1 786
3 792	3 153	1 379	2 493
5 712	3 746	2 148	3 743

5　结　语

本文提出了一种基于改进优序图法的读时故障注入方法，采用该方法可以提高故障注入的有效性，缩小故障空间。首先，在小尺度程序上进行故障注入，构建判断矩阵，计算特征值权重向量；然后，对大尺度程序进行特征分析，统计特征值；最后，根据特征值权重向量和大尺度程序特征值，计算大尺度程序各条汇编代码的权重。汇编代码权重的大小代表了其脆弱性的高低。在实验中，对于大尺度程序的各条汇编代码，分别按照有效故障率和权重进行降序排列，各取前 50% 的元素形成两个集合，两者的相似度可以达到 81%。

参考文献：

[1] 程义，庄毅，曹子宁. 利用随机森林的单粒子翻转软件故障注入方法[J]. 小型微型计算机系统，2021，42（11）：2452-2458.

CHENG Yi, ZHUANG Yi, CAO Zining. Fault injection for single event upset simulation based on random forest [J]. Journal of Chinese Computer Systems, 2021, 42(11): 2452-2458. (in Chinese)

[2] LI Y, LIU J, LI G, et al. Airborne software testing technology analysis based on fault injection[C]//2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), 2020: 279-282.

[3] YAYAN U, BAGLUM C. Tailored mutation-based software fault injection tool (IM-FIT) [J]. SoftwareX, 2023, 23: 101463.

[4] AL-HAJ AHMAD H, SEDAGHAT Y. CAFI: a configurable location-aware fault injection technique for software reliability assessment against soft errors [J]. Microprocessors and Microsystems, 2022, 94: 104648.

[5] PARK J, CHOI B. ASFIT: AUTOSAR-based software fault injection test for vehicles [J]. Electronics, 2020, 9(5): 850.

[6] AHMAD H A, SEDAGHAT Y, MORADIYAN M. LDSFI: a lightweight dynamic software-based fault injection [C]//2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), 2019: 207-213.

[7] 林灏铨. 面向智能处理平台的软件故障注入及测试系统设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2021.

[8] BODMANN P R, OLIVEIRA D, RECH P. Accurate FIT rate estimation through high-level software fault injection [J]. IEEE Transactions on Nuclear Science, 2022, 69(9): 2018-2026.

[9] FOLKESSON P, SANGCHOOLIE B, KLEBERGER P, et al. On the evaluation of three pre-injection analysis techniques for model-implemented fault-and attack injection[C]//2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC), 2022: 130-140.

[10] SANGCHOOLIE B, PATTABIRAMAN K, KARLSSON J. An empirical study of the impact of single and multiple bit-flip errors in programs[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(3): 1988-2006.

[11] GEORGAKOUDIS G, LAGUNA I, VANDIERENDONCK H, et al. SAFIRE: scalable and accurate fault injection for parallel multithreaded applications[C]//2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2019: 890-899.

[12] AGARWAL U K, CHAN A, PATTABIRAMAN K. LLTFI: framework agnostic fault injection for machine learning applications (tools and artifact track) [C]//2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE), 2022: 286-296.