# 热路径结合程序切片思想在错误定位中的应用①

肖 燕，缪 力，李 玮

(湖南大学 信息科学与工程学院，长沙 410086)

**摘 要**：回归测试指对修改后的软件进行测试。为提高回归测试错误分析效率，基于目前热路径思想在程序分析里的应用，结合程序切片方法，提出一种高效的回归测试方法。首先找出在程序执行过程中方法级的执行路径频率，结合应用 Dslice 切片算法应用用于回归测试，对已知的错误的程序进行调试，比较准确地进行了方法级的错误定位。实验结果表明通过热路径与程序切片相结合所得的算法可有效提高回归测试的错误分析效率。

**关键词**：热路径；回归测试；动态切片；错误定位

## Application of Fault Location with Hot Path and Program Slicing

XIAO Yan, MIAO Li, LI Wei

(School of Information Science and Project, Hunan University, Changsha 410086, China)

**Abstract:** Regression testing means testing the modified software. Improving the efficiency of error analysis of regression testing, based on the current thinking in the heat path in the application of program analysis, combined with program slicing method, proposed an efficient method for regression testing.Firstly finding out the method path frequencies of execution in the program-level execution, application of combined application Dslice slicing algorithm for regression testing, the procedure known to debug errors, more accurately locate the method error level. The results show that the path through the heat combined with the proceeds of program slicing algorithm can improve the efficiency of regression testing, error analysis.

**Key words:** hot path; regression test; dynamic slice; fault location

程序分析[1]是指对计算机程序行为进行分析的过程，主要应用在软件逆向工程、软件理解、软件优化、安全、和协议验证诸多方面。据报道软件中每年约有1000 个新的漏洞被发现，而且 50％的漏洞是常见漏洞的重复出现，这些都是由具有安全隐患的编程所引起的。因此如何高效准确地检测程序漏洞，是一件很有意义的事情。在研发软件的过程中，由于程序员的疏忽等原因，往往容易给程序引入不同类型的错误。总的来说，这些错误可以分为两类：第一类称为编译时错误，此类错误在程序进行编译时，能由编译器[2]检测到，因此能被较早地发现，并及时纠正过来；第二类错误称为运行时错误，编译器很难检测到此类错误，程序在实际运行过程中遇到一个此类错误后，将会抛出异常并终止。若是有一种方法，既能解决编译时的错误又能解决运行时错误，必会大大提高程序检测的效率。程序分析分为静态和动态两种，静态分析不需要执行程序，只需要分析源代码；动态分析则是要实际运行程序。前者耗时少，准确度低；后者耗时大，准确度高。本文运用准确度较高的动态分析，来进行错误定位。

Java 语言作为日前国际上使用最为广泛的一种程序设计语言，吸引了全球超过 500 万的软件开发人员使用。Java 程序在游戏控制台、科学超级计算机、手机、互联网等领域得到广泛应用。Java 语言具有封装和继承性，而且子类和父类之间为单一的继承关系，在研究过程中会大大减小分析的复杂性。

① 基金项目:安徽省教育厅自然科学基金(2005KJ004ZD);“中央高校基本科研业务费”资助
收稿时间:2011-03-26;收到修改稿时间:2011-05-02

对热路径的定义总结为：①在程序运行中执行次数最多的路径为热路径；②对程序的运行状态所产生的影响最小的也为热路径。根据改进系统性能的阿姆达尔定律，提出了对程序中部分代码进行改动就可以最大化地提高程序性能的理论，深化了热路径[3]概念的应用。在正常情况下，程序运行时的路径其实是比较固定的，此为程序运行时的稳定性。像 rehash 之类的操作指令运行的次数是比较少的，因为它对程序的状态影响很大，在特定的情况下才会运行。程序大部分时间都是在执行有特定功能的部分代码，所以选准了热路径作为研究对象对程序分析起着很大的简化作用。

通过对国内外热路径的发展状况进行分析，精确地搜集到语句级别的热路径目前是比较困难的，目前的发展程序还仅限于静态地预测上，动态的过程很复杂，时间冗余度高。而对于方法级的执行路径来说，通过方法间的调用关系，找到出错的方法，对单个的方法再进行错误定位检测，对分析过程也有简化作用。

试想，能否通过切片技术找出热路径？

程序切片理论在 1979 年由 mark weiser[4]首次提出，之后这一切片技术已经广泛应用于程序分析方面，对程序分析起到了大大的简化作用。对切片技术来说，后来发展出了动静态，前后向，里外向等一系列的切片方法和工具。通过分析各种依赖关系做出程序依赖图，再根据程序依赖图用算法进行切片。对最常用的动静态切片来说，静态切片是静态的二元切片，动态切片是依据三元切片准则在有特定输入下得到的切片，相比之下，动态切片比静态切片的结果要准确的多。

鉴于以上分析，程序分析可以才用切片理论进行分析，而切片方法可以和热路径理论相结合，本文采用热路径结合程序切片思想在 java 程序中进行错误定位，动态切片将用 Dslice 算法进行实现。

## 1 面向回归测试的错误定位技术

回归测试为软件生命周期的一个重要组成部分，在整个软件测试过程的工作量中占有很大的比重，软件开发的各个阶段都会进行多次回归测试。回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护升级等阶段的成本。几乎在每个项目中都不可缺，回归测试后产生的结果有以下几种情况：①程序原有的错误消失；②程序原有的错误消失，但是出现了新的错误；③原有的错误没有消失，又出现了新的错误。针对后面两种情况，本文都运用面向回归测试的错误定位技术进行调试。图 1 为本文进行研究的系统框架。

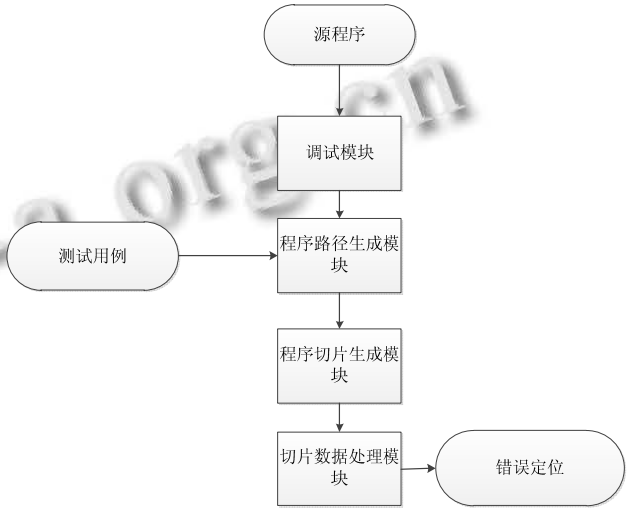


图 1 系统流程框架

文献中[5]提出了 chopping 切片方法，实际是将切片方法中的前向和后向两种切片方法相结合，提出的一种语句级别的静态猜测性的切片方法。

文献中[2]提到了静态热路径枚举方法，此方法在不考虑循环的情况下，通过逻辑回归定位理论分析特征的 feature，枚举出 feature 所在的语句，作为热路径的组成语句。然后用动态路径枚举结果做基准进行效率比较，得到了有效的静态分析热路径方法。

文献中[6]提出了基于经典的数学概率模型隐马尔可夫（HMM）进行动态二进制的热路径优化方法，与文献[2]不同，其主要针对频繁执行的循环块，先分割出程序的执行快，然后分析程序的块级别的依赖关系，再划分出能满足隐马尔可夫数学模型的计算条件。通过 trigger 模型预测出了将要执行的多条路径，在满足模型演算条件后，进行概率计算，计算出最热的路径。

目前的研究工作大多为静态分析技术。从静态分析的特征上说，静态方法都属于不需执行程序的预测，而在程序的实际执行中需要考虑到变量，对象，方法，以及它们之间的各种依赖和控制关系，所以对于单纯的静态分析来说，准确度并不是很高。因此，本文采

用了准确度较高的动态分析方法，动态切片[7]与热路径结合进行回归测试[8]的应用，计算出热路径[9]进行错误定位。

基于以上分析，本文提出面向回归测试的错误分析框架：输入错误程序，然后进行调试，再进行回归测试，评估其结果，如果有错误，再对有错误的源程序进行 Dslice 切片，实现过程为：

①先生成方法调用依赖图（图 3 所示），ABC 分别表示代码中的各个方法；

②再利用三维的动态切片准则(x,sk ,V）进行程序切片，切出对输入 x 在执行点[9]切出所有集合变量 V 有关的方法路径切片；

③切出方法的执行路径后，再对错误程序进行调试，直到得到无错误的程序为止。

在运用切片法则对程序进行动态切片以前，先经过动态运行程序得到所有方法级的执行路径，然后再通过计算切片集合，计算出频率最高的方法，进行错误定位。

## 2　算法和演示实例

对于程序而言，多个类里共有多个不同的方法，本文主要分析方法之间最常见的调用关系，先得到动态的方法执行路径，然后进行动态程序切片。

### 2.1 D slice 算法

动态方法切片 Dslice

输入：动态方法调用路径

输出：执行位置 q 处变量集合 V 的动态方法切片。

第一：始化

1 for all v∈V　do Slice(v)= ∅ ;

2 for all st ∈P　do Dslice(st)={s}；

第二：行每个 sk (s<q)后;

3 计算子节点中方法的出现次数

对于每个调用点 a.method( )

为切片源点

4 if 切片源点的方法中有循环 $f_k$

then Slice(f 后)=Nslice(f)+slice(a) N 表示循环次数

Slice(f 前)=slice(a)

Return

5 else Slice(v)=slice(a)

6 Slice(v)=v 最新修改过的切片

7 slice(v)= slice(a)∪slice(f)

Then

return 0;

8 Slice=Slice(befor)∪Slice(after)∩Slice(f)

### 2.2 算法运用实例分析

假定要调试图 2 所示的程序，若在输入 i=3 时为 P 全（1,3,7,83,93,263,103,123,4,14,26,5,18,20,26,22,24)。

```
1   public static void main( string [ ] args) {
2   -----
3       a = F.my_sacnf();
4       b=tis.soe();
5       c=nam.tim();
6    ------
7   public static int my_scanf( fis, int i, int j, int m){
8   for(i=0,i<3,i++) {
9         int t=dyn.push();
10        int k = fis.read();
11   ------
12  public static long read (){
13  --------
14   public static int soe () {
15  --------
16   f=j.push();
17  --------
18   public static int tim () {
19  -------
20   ej=d.push();
22  dd=wd.run();
23  -------
24  public static int run () {
25  --------
26  public static int push () {
    ----
```

图 2　代码实例图

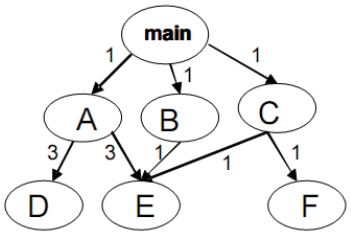


图 3　方法调用依赖图

对不能揭示错误的测试用例“2 65 3”运行的结果进行切片，计算出 E 为调用数最多的方法，取 E 方法里的调用点为切片源点，得到切片 $S_1$（E-A-main）和 $S_2$（E-B-main),$S_3$（E-C-main)对于能揭示错误的测试用例“1 100 65”的运行结果，切片结果计算 A 为最热方法，切片结果表示从 A 开始切片得到 $S_4$（D-A-main）和 $S_4$（E-A-main）集合这五条切片，对比各测试用例的期望值，分析出错和不出错的结果，推出此错误包含在方法 D 里面，然后继续对方法 D 里的语句进行检测，找到出错语句。切片越多越详细，计算结果会越准确。

## 3 试验结果分析

为了验证上述方法的有效性，本文开发了 Dslice 算法，输入测试用例得到所有方法调用路径的 xml 文件，分析正确和失败的测试用例得到的不同 xml 文件的结果。先计算出调用次数最多的方法，能切出跟此方法有关的所有的方法调用路径，输出结果，计算出所有调用方法的最热路径。该算法也可以得到所有程序的调用方法集，以及每个测试用例执行时的方法调用路径等。

本文实验中采用的 Siemens 程序集是错误定位技术研究领域得到广泛使用的基准程序集，实验程序集描述如表 4 所示。

表 4 实验程序信息

| Program | loc | Ver | Meth | Test | Speed_Inc |
|---|---|---|---|---|---|
| totinfo | 330 | 15 | 20 | 2504 | 630 |
| tcas | 159 | 34 | 9 | 1200 | 2569 |
| replace | 590 | 8 | 24 | 2300 | 2631 |
| printtoken | 610 | 6 | 23 | 1500 | 3302 |

对程序 totinfo 来说，计算出最热方法的路径为(A-C-F-H-K)其中，方法 C 和 H，K 都是程序已经给出的存在错误的方法，从而得出本文计算得到的错误定位准确度很高。

## 4 结语

本文通过热路径与动态程序切片相结合进行回归测试里的错误定位，对已知的错误程序进行调试，运用 Dslice 切片算法进行最热方法的路径切片，得出方法级的热路径，通过实验验证，此算法为有效的错误定位算法。下一步的工作重点是利用更大的实验程序来验证我们方法的有效性，以及对 Dslice 算法的进一步完善。

## 参考文献

1 Winstead J, Evans D. Towards Differential Program Analysis. Workshop on Dynamic Analysis., ICSE 2003. 37–40.

2 Buse RPL, Weimer W. The Road Not Taken: Estimating Path Execution Frequency Statically. ICSEIEEE, 2009, 144－54.

3 Weiser M, Program Slicing, IEEE Transactions on Software Engineering,no.4,1984,10:352–357.

4 Gupta R, Mehofer E, Zhang Y. Profile-guided compiler optimizations. In The Compiler Design Handbook, 2002, 143–174.

5 蒋曹青.一种回归测试后的错误定位的算法.计算机工程与科学,2005,26–30.

6 刘魁.基于隐马尔可夫模型的热路径算法预测研究.计算机应用研究,2010,49–53.

7 R.V.R. et al. Soot-a java optimization framework. Proc. of CASCON, 1999, 125–135.

8 Tip F. A survey of program slicing techniques. J Progr. Lang., 1995,3(3):121–189.

9 Baba T, Masuho T, Yokota T, Ootsu K. Design of a two-level hot path detector for path-based loop optimizations. Advances in Computer Scienceand Technology, 2007, 23–28.

10 Wagner D, Dean D. Intrusion detection via static analysis. Proc. of the 2001 IEEE Symposium on Security and Privacy. May 2001, 34–100.

(上接第 166 页)

## 参考文献

1 张雅鹏.主动数据仓库基于规则的事件匹配机制的研究与实现.北京邮电大学,2006.3:3–6.

2 徐焕良,李绪蓉,丁秋林.基于规则库的业务构件重组的实现.计算机集成制造系统,2003,(10):911–913.

3 Gamma E, Helm R.李英军,等译.设计模式:可重复面向对象软件的基础.北京:机械工业出版社,2008.23–45.

4 高华,张宏军,陈刚,等.作战仿真数据集成框架研究及其实现.火力与指挥控制,2009,2:150–153.

5 Sperley E. The Enterprise Data Warehouse: Planning, Building and Implementation. Prentice Hall PTR, 1999.

6 支凤丽.数据移植过程中的数据质量控制方法的研究.同济大学,2007.48–62.

7 陈伟.数据清理关键技术及其软件平台的研究与应用[博士学位论文].南京:南京航空航天大学,2004.70–76.