Android 应用程序 GUI 模型生成方法研究

李振南[1]，戚晓芳[1]

（1.东南大学计算机科学与工程学院，南京，211189）

摘　要：Android 应用程序 GUI 模型可用于自动生成 Android 应用程序测试用例。现有的 GUI 模型生成方法未有效处理界面间回路问题，导致所生成的 GUI 模型不完整，从而影响测试的充分性。为此我们提出一种基于边标记的 GUI 模型生成方法，并开发了自动化测试工具 FectDroid。FectDroid 采用基于边标记的 GUI 模型生成方法保证 Android 应用程序所有可触发事件均可被执行，生成较完整的 GUI 模型。实验结果表明 FectDroid 生成的 GUI 模型可以实现较高的 Activity 覆盖率和 GUI 搜索数量。

关键词：Android 应用程序；自动化测试；GUI 模型

The GUI Model Generation Method Research of Android Application

Li ZhenNan[1], Qi XiaoFang[1]

(1. School of Mechanical Engineering, Southeast University, Nanjing, 211189)

Abstract: The GUI model of Android application can be used to automatically generate test cases for Android application. Existing GUI model generation method is not effective to handle the problem of the loop between interfaces, which leads to a incompletion of the generated GUI model, thus affecting the adequacy of the test. In this paper we put forward a edge marking-based GUI generation method and develope the automated testing tool named FectDroid, which using edge marking-based GUI model generation method to ensure all trigger events can be implemented in Android application and generate a complete GUI model. The experimental results show that the GUI model that FectDroid generate can achieve higher Activity coverage and GUI search amount.

Key words: Android application; Automated testing; GUI model

1 引言

随着智能手机的快速发展，拥有丰富图形界面的移动应用程序得到广泛应用。与一般的桌面应用相比，移动应用程序具有操作多样、界面丰富、更新速度快等特点，针对具有图形界面的应用程序，传统的人工测试方法已经不再适用，如何保障移动应用的质量引起学者们高度关注，其中移动应用的自动化测试已成为该领域的一个重要研究方向[1-5]。

一般来说，移动应用程序的自动化测试方法主要有基于黑盒的随机测试方法[6-7]、记录重放的测试方法[8-10]以及基于模型的测试方法[11-15]。

作者简介：李振南，（1993-），男，硕士研究生，E-mail: 765819256@qq.com；戚晓芳，（1972-），女，副教授，E-mail: xfqi@seu.edu.cn.

基于黑盒的随机测试方法通过向移动应用发送随机的用户事件流(如按键输入、触摸屏输入、手势输入等)来对应用进行压力测试，是一种测试移动应用稳定性的快速有效的方法，但该方法生成的随机事件流会出现冗余和功能重叠的问题。记录重放的方法首先记录用户的操作过程并生成相应的脚本文件，之后通过修改和回放脚本文件来模拟用户操作，以达到自动运行移动应用程序并检测潜在错误的目的。但该方法的脚本文件只能覆盖移动应用程序的部分功能，无法充分地对其进行测试。基于模型的测试方法首先要生成移动应用程序对应的 GUI 模型，之后根据 GUI 模型生成相应的测试用例，但生成的 GUI 模型以 Activity 为状态节点，不能对 Activity 中多个不同的 GUI 进行描述，导致测试用例生成的准确度不高，降低测试效果。

针对解决 GUI 模型中以 Activity 为状态节点的

问题，Choi W 等人提出的 SwiftHand 工具以 GUI 为模型的状态节点，并通过启发式的搜索策略来减少建立模型过程中应用的重启次数[16]。Peng W 等人提出的 DroidCrawle 工具通过对比执行触发事件前后的屏幕界面来建立 GUI 模型，并后通过回退算法找到下一个含有未触发事件的界面继续进行探测[17]。但上述工具没有考虑界面间存在回路的情况，因此会导致搜索过程中界面的缺失，造成建立的 GUI 模型不完整。

为了进一步提高移动应用程序 GUI 模型的完整性，本文提出自动化测试工具 FectDroid，FectDroid 利用一种基于边标记的 GUI 模型搜索方法进行建模。该方法通过在搜索过程中判断界面是否含有未触发事件来决定下一个界面的选择，以此确保每个界面中的所有可触发事件均可被执行，进而保证生成的 GUI 模型没有界面的缺失。

本文第 2 部分介绍采用该方法实现的工具 JAET 的框架，第 3 部分给出新的 GUI 搜索方法的具体思想，第 4 部分对 JAET 和其他工具进行对比实验，第 5 部分对全文进行总结。

2 基于边标记的 GUI 模型搜索方法

针对现有的 GUI 模型搜索方法无法正确处理界面回路的问题，本文提出一种基于边标记的 GUI 模型探测方法。该方法通过计算安卓应用程序界面中剩余的可触发事件的数量来决定下一个界面状态的选择，以此确保每一个界面状态中的所有可触发事件均可被执行，进而保证生成的 GUI 模型没有界面状态的缺失。

基于边标记的 GUI 模型探测方法如图 1 所示，方法会对 Android 应用程序进行两次搜索。第一次搜索通过执行 Android 设备屏幕界面可触发事件来模拟真实用户行为，从而对搜索到的界面进行 GUI 建模。第二次搜索对已生成的 GUI 模型再遍历，对包含未触发事件的界面继续进行搜索，从而完善 GUI 模型。通过两次搜索过程，便可以确保 Android 应用程序的 GUI 模型更加完整。

第一次搜索过程如图 2 所示，首先我们要提供 Android 应用程序的入口信息，这里的信息包括应用程序包名以及入口 Activity 名，函数 startApp 根据入口信息启动应用程序并获取当前屏幕界面 CurGUI，之后将 CurGUI 保存到 GUITree 中（4-5 行）。进入 while 循环后，函数 selectEvent 从 CurGUI 中选取一个可触发事件 e，并通过函数 execute 执行该事件 e，行完成后保存当前屏幕界面到 CurGUI，将执行事件 e 之前的屏幕界面保存到 PreGUI，最后将 e 标记为已触发事件并保存到 RecordList 中（7-11 行）。接下来算法将通过比较事件执行前后的界面来决定下一步的操作。

输入： Android 应用程序入口信息 entryInfo；
输出： Android 应用程序模型 GUITree；
guiCrawl(entryInfo)
1. GUITree ← generalCrawl(entryInfo)；
2. GUITree ← expandCrawl(GUITree)；
3. 输出 GUITree；

图 1 基于边标记的 GUI 模型搜索方法

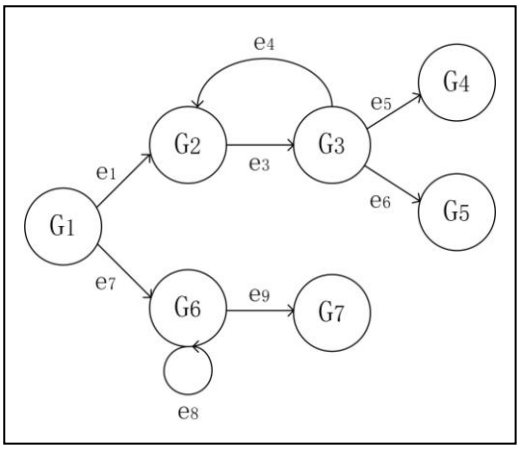

图 3 GUI 模型图

如果 CurGUI 和 PreGUI 相同，那么方法将继续从 CurGUI 进行搜索。例如图 3 所示的 GUI 模型中，界面 G_6 执行事件 e_8 后状态并没有发生改变，所以继续从 G_6 进行搜索。如果 CurGUI 和 PreGUI 不同且 CurGUI 不存在于 GUITree 中，那么说明 CurGUI 是新界面，将 CurGUI 保存到 GUITree 中，并继续从 CurGui 进行搜索（12-14 行）。如果 CurGUI 和 PreGUI 不同且 CurGUI 已经是 GUITree 的节点，那么说明 CurGUI 是已经被搜索到的界面。此时算法将判断当前界面 PreGUI 是否含有未触发事件。如果 PreGUI 没有未触发事件，那么将继续从 CurGui 进行搜索（15-16 行）。例如在图 3 中，假设 G_3 的事件 e_5，e_6 已被触发，此时会执行事件 e_4 返回已存在于 GUITree 的界面 G_2，由于 G_3 的所有事件均被触发，因此算法会在 G_2 中继续探测。如果 PreGUI

```
输入：Android 应用程序入口信息 entryInfo；
输出：Android 应用程序模型 GUITree；
generalCrawl(entryInfo)
1.  GUITree ← Φ；
2.  RecordList ← Φ；
3.  PreGUI ← Φ，CurGUI ← Φ；
4.  CurGUI ← startApp(info)；
5.  GuiTree ← CurGUI；
6.  while(true)
7.      e ← selectEvent（CurGUI）；
8.      PreGUI ← CurGUI；
9.      CurGUI ← execute(e)；
10.     将 e 标记为已触发事件；
11.     RecordList ← e；
12.     if(CurGUI != PreGUI)
13.         if(CurGUI not in GUITree)
14.             GUITree ← CurGUI；
15.         else if(PreGUI 没有未触发事件)
16.             CurGUI ← CurGUI；
17.         else if(PreGUI 含有未触发事件)
18.             p ← findPath(CurGUI, PreGUI)；
19.             CurGUI ← execute(p)；
20.         end if；
21.     end if；
22.     if(CurGUI 没有可触发事件)
23.         CurGUI ← rollBack(GUITree)；
24.         if(CurGUI == Φ)
25.             break；
26.         end if；
27.     end if；
28. end while；
29. 输出 GUITree；
```

图 2 第一次搜索过程

还有未触发事件，那么函数 findPath 将从 GUITree 中查找一条从 CurGUI 到 PreGUI 的转换路径，并通过函数 execute 执行该路径返回到 PreGUI 继续进行搜索（17-19 行）。例如在图 3 中，G_3 执行事件 e_4 返回到 G_2（假设 G_3 中事件 e_5，e_6 未被触发），由于 G_3 仍有未触发事件，因此函数 findPath 找到一条从 G_2 到 G_3 的路径，使屏幕界面返回 G3 从而继续进行搜索，确保 G_4、G_5 不会缺失。

```
输入：Android 应用程序模型 GUITree；
输出：屏幕界面状态；
rollback(GUITree)
1.  RollBackGUI ← Φ，GUI ← Φ；
2.  while(true)
3.      RollBackGUI ← 模拟 back 按键回退；
4.      if(outOfBound() == true)
5.          break；
6.      end if；
7.      GUI ← findGUI (RollBackGUI)；
8.      if(GUI 含有未触发事件)
9.          break；
10.     end if；
11. end while；
12. 输出 GUI；
```

图 4 回退算法

在探测过程中，如果 CurGUI 没有可触发事件，那么执行回退算法 rollBack（22-23 行）。如图 4 所示，函数 outOfBound 会在执行 back 操作后对应用进行边界检查，如果应用程序退出，则 rollBack 返回空值，第一次搜索结束（24-25 行），返回生成的 GUI 模型（26 行），否则函数 findGUI 将从 GUITree 中找到与回退后的界面 RollBackGUI 对应的且含有未触发事件的界面状态 GUI 并将其返回，从而继续进行搜索。

```
输入：GUITree
输出：GUITree
expandCrawl(GUITree)
1.  for GUI in GUITree
2.      if(GUI 含有未触发事件)
3.          对 GUI 进行搜索；
4.      end if；
5.  end for；
6.  输出 GUITree；
```

图 5 第二次搜索过程

二次搜索会在常规搜索生成的 GUI 模型基础上进行。由于常规搜索过程中的回退算法对同一 Activity 中存在多个 GUI 的情况可能会失效，例如在具有 TabLayout 布局的 Activity 中，执行物理返回按键并不能实现界面的回退，从而导致部分界面

的可触发事件遗漏。因此二次搜索会再次遍历 GUI 模型中的界面，找到界面中遗漏的可触发事件并模拟执行。图 5 所示的是二次搜索算法，首先遍历 GUITree 中的界面 GUI，若 GUI 中含有可触发事件，则对 GUI 进行搜索，并将搜索到的新界面放入 GUITree 中。所有界面遍历完后返回更新的 GUITree。

3 FectDroid 框架

FectDroid 工具连接 Android 设备，提取屏幕界面的所有可触发事件，如点击按钮、输入文本等（以下简称用户行为），选择一个用户行为，通过模拟该行为实现与 Android 设备的交互，根据 Android 设备的界面响应继续分析。FectDroid 分为 5 个功能模块：GUI 模型搜索模块、设备连接器、界面解析器、事件选择器以及事件执行器，其框架如图 6 所示。

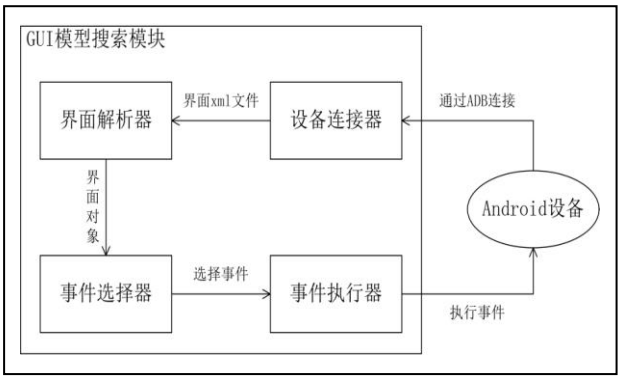

图 6 FectDroid 框架

3.1 GUI 模型搜索模块

GUI 模型搜索模块是 FectDroid 工具的主模块，它的核心便是第二部分提出的基于边标记的 GUI 模型搜索方法。该模块通过模型搜索方法将其包含的四个子模块连接起来，使它们协调运作，帮助选择需要处理的界面状态。

3.2 设备连接器

设备连接器的主要功能是通过 MonkeyRunner 工具中核心包 chimpchat.jar 来建立 Android 设备与底层 ADB（Android Debug Bridge） 服务器的通信，从而实现设备连接器对 Android 设备的操作与控制。

3.3 界面解析器

界面解析器的主要功能是通过设备连接器向 Android 设备索取当前屏幕界面的布局文件，解析该布局文件来得到当前界面中所有可触发事件并返回界面对象。目前 FectDroid 可处理的可触发事件为单击行为、长按行为以及输入行为。

```
<hierarchy rotation="0">
    <node index="0" text="" resource-id=""
        class="android.widget.LinearLayout"
        package="com.jyy" content-desc=""
        checkable="false" checked="false"
        clickable="false" enabled="true"
        focusable="false" focused="false"
        scrollable="false" long-clickable="false"
        password="false" selected="false"
        bounds="[0,0][720,1280]">
        <node index="0" text=""
            resource-id="com.jyy:id/image_view"
            class="android.widget.ImageView"
            package="com.jyy" content-desc=""
            checkable="false" checked="false"
            clickable="false" enabled="true"
            focusable="false" focused="false"
            scrollable="false" long-clickable="true"
            password="false" selected="false"
            bounds="[260,380][460,580]" />
        <node index="1" text=""
            resource-id="com.jyy:id/phone"
            class="android.widget.EditText"
            package="com.jyy" content-desc=""
            checkable="false" checked="false"
            clickable="true" enabled="true"
            focusable="true" focused="true"
            scrollable="false" long-clickable="false"
            password="false" selected="false"
            bounds="[32,612][688,712]" />
        <node index="2" text=""
            resource-id="com.jyy:id/login_button"
            class="android.widget.Button"
            package="com.jyy" content-desc=""
            checkable="false" checked="false"
            clickable="true" enabled="true"
            focusable="true" focused="false"
            scrollable="false" long-clickable="false"
            password="false" selected="false"
            bounds="[32,912][688,1008]" />
        </node>
    </node>
</hierarchy>
```

图 7 界面布局文件

单击行为即用户单次点击的动作，Android 设备会根据用户单击的区域进行不同的界面响应，在图 7 所示的界面布局文件中，包含一个 resource-id 为 com.jyy:id/login_button 且 clickable 为 true 的单击按钮事件。

长按行为是移动应用程序中独有的特殊操作，

用户可以通过长按屏幕区域进行相应的功能交互。在图 2 所示的界面布局文件中，包含一个 resource-id 为 com.jyy:id/image_view 且 long-clickable 为 true 且的长按事件。

输入行为就是用户在操作过程中给文本编辑框输入字符串的行为。在图 7 所示的界面布局文件中，包含一个 resource-id 为 com.jyy:id/phone，class 为 android.widget.EditText 的文本编辑框。

界面解析器通过解析图 7 所示的界面文件后，得到的用户行为如表 1 所示。

表 1 用户行为列表

	单击事件	长按事件	输入事件
resource-id	login_button	image_view	phone

3.4 事件选择器

事件选择器的第一个功能是在界面的可触发事件集中选取一个事件，并对选取的事件需要的一些条件参数进行初始化。第二个功能是根据以建立的 GUI 模型来查找两个界面之间的转换路径，并返回两个界面转换所需的所有可触发事件。

3.5 事件执行器

事件执行器的主要功能是执行事件选择器中已选择的可触发事件。事件执行器首先会通过设备连接器获得对设备的操作以及控制权限，提取选择的可触发事件类型和触发需要的条件参数，执行相应的 ADB 操作命令来模拟用户对 Android 设备的真实行为。

利用 FectDroid 对 Android 应用程序的建模过程中，FectDroid 首先会通过设备连接器建立与 Android 设备的 ADB 服务器的通信，之后界面解析器会向 ADB 服务器发送获取屏幕界面布局文件请求，ADB 服务器接受请求后会返回界面布局文件到界面解析器，界面解析器提取布局文件中的所有可触发事件，生成界面对象并返回给状态决策器，状态决策器利用本文提出的 GUI 模型搜索方法（在第 2 部分介绍）来决定当前屏幕界面的选择，之后事件选择器会在当前屏幕界面中选择一个可触发事件并返回给事件执行器进行模拟操作，模拟操作执行后界面解析器会向 ADB 服务器请求界面布局文件，从而继续对 Android 应用程序进行建模，直到所有的界面全部被搜索。

4 实验结果

为验证 FectDroid 生成 Android 应用程序 GUI 模型的能力，我们从 Android 应用商店选取了几种不同类型的 Android 应用程序进行测试，选取的 Android 应用程序如表 2 所示。JNews 用于新闻浏览的客户端，Bequick 是对大脑、眼睛以及手指速度的测试和训练的游戏，Mileage 用于里程计费，Tippy-Tipper 用于费用计算，whohasmystuff 用于行程备忘，AlarmClockXtreme 用于闹钟提醒。

表 2 待测 Android 应用程序

No	应用名称	代码行数
1	JNews	9204
2	Bequick	11530
3	Mileage	1023
4	Tippy-Tipper	4207
5	whohasmystuff	1537
6	AlarmClockXtreme	45492

由于 Android 系统本身的硬件资源有限，Android 应用程序的界面状态并不是很多，所以实验利用上述 GUI 模型搜索方法对 Android 应用程序进行不限时测试，力求探测到 Android 应用程序中所有的界面。本文提出的 FectDroid 工具将与 DroidCrawle、A3E、GUI crawler 工具的 GUI 模型搜索方法进行对比，用探测得到的 GUI 模型中 Activity 覆盖率以及 GUI 总数进行比较。实验结果如表 3 和表 4 所示。

Activity 总数可以通过分析应用的配置文件得到，从表 3 的数据可以看出，FectDroid 可以探测到更多的 Activity，实现了较高的 Activity 覆盖率。对于应用 AlarmClockXtreme 而言，FectDroid 虽然实现的 Activity 覆盖率比其他工具高，但也只达到了 57%，原因有两点：

1) FectDroid 不支持对控件 ListView 的滚动事件处理，超出屏幕的可触发事件不能执行，从而导致部分界面的丢失。
2) AlarmClockXtreme 的部分可触发事件由于请求外网服务得不到响应，界面无法根据响应结果做出反应，导致界面丢失。

表 2 Activity 覆盖率

（A_N 表示搜索到的 Activity 数，A_C 表示 Activity 覆盖率）

Android 应用	Activity 数	DroidCrawle		A3E		GUI crawler		FectDroid	
		A_N	A_C	A_N	A_C	A_N	A_C	A_N	A_C
JNews	7	4	57.1%	4	57.1%	2	28.6%	7	100%
Bequick	1	1	100%	1	100%	1	100%	1	100%
Mileage	4	3	75%	4	100%	2	50%	4	100%
Tippy-Tipper	5	3	60%	4	80%	2	40%	5	100%
whohasmystuff	2	2	100%	2	100%	2	100%	2	100%
AlarmClockXtreme	19	5	26.3%	2	10.5%	1	5.2%	11	57.9%
总数	38	18	47．4%	17	44.7%	10	26.3%	30	78.9%

表 3 GUI 搜索数量

Android 应用	GUI 数	GUI 搜索数量			
		DroidCrawle	A3E	GUI crawler	FectDroid
JNews	-	13	2	2	25
Bequick	-	8	1	1	17
Mileage	7	6	4	2	7
Tippy-Tipper	6	4	2	2	6
whohasmystuff	12	4	2	2	12
AlarmClockXtreme	-	12	2	1	34
总数	-	47	13	10	101

由于 Android 应用程序界面会根据用户行为进行动态变换，因此现有的工具无法统计 Android 应用程序的界面总数，我们只能进行手工统计。但手工统计对于那些规模较大，界面变化较多的 Android 应用程序并不适用，所以我们无法手工统计出 JNews、Bequick 以及 AlarmClockXtreme 的界面总数。表 3 的数据可以看出，即使各工具在 Activity 覆盖率相近的情况下，FectDroid 建立的 GUI 模型搜索到的界面数量也远多于 DroidCrawle、A3E、GUI crawler 工具探测到的数量，这反映出 FectDroid 具有较出色的界面搜索能力。

5 结论

本文提出了移动应用程序自动化测试工具 FectDroid，并利用基于边标记的 GUI 模型搜索方法对 Android 应用程序进行建模。与现有的自动化测试工具相比，FectDroid 在提高 Activity 覆盖率和界面搜索数量方面效果明显，不足之处在于无法处理界面中滚动等特殊事件以及没有考虑到界面中参数不同组合对界面搜索能力的影响[13]，后期我们会进行进一步研究。

参考文献

[1] Android Developers. The Developer ’ s Guide. https://developer.android.google.cn/guide/, last accessed onDecember 10th, 2016

[2] Baek Y M, Bae D H. Automated model-based Android GUI testing using multi-level GUI comparison criteria[C]. IEEE/ACM International Conference on Automated Software Engineering. ACM, 2016:238-249.

[3] Gudmundsson V, Lindvall M, Aceto L, et al. Model-based Testing of Mobile Systems -- An Empirical Study on

QuizUp Android App[J]. 2016, 208:16-30.

[4] 赵耀宗, 程绍银, 蒋凡. Android 应用程序 GUI 遍历的自动化方法[J]. 计算机系统应用, 2015(9):219-224.

[5] Amalfitano D. Using GUI ripping for automated testing of Android applications[C]. Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2012:258-261.

[6] Android Developers. UI/Application Exerciser Monkey,.http://developer:android:com/tools/help/monkey.html.

[7] Machiry A, Tahiliani R, Naik M. Dynodroid: an input generation system for Android apps[C]. Joint Meeting on Foundations of Software Engineering. 2013:422-434.

[8] Gomez L, Neamtiu I, Azim T, et al. RERAN: Timing- and touch-sensitive record and replay for Android[C]. International Conference on Software Engineering. IEEE, 2013:72-81.

[9] Hu Y, Azim T, Neamtiu I. Versatile yet lightweight record-and-replay for Android[J]. Acm Sigplan Notices, 2015, 50(10):349-366.

[10] Hu Y, Neamtiu I. VALERA: an effective and efficient record-and-replay tool for android[C]. International Conference on Mobile Software Engineering and Systems. ACM, 2016:285-286.

[11] Amalfitano D, Fasolino A R, Tramontana P. A GUI Crawling-Based Technique for Android Mobile Application Testing[C]. IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops. IEEE, 2011:252-261.

[12] Takala T, Katara M, Harty J. Experiences of System-Level Model-Based GUI Testing of an Android Application[C]. IEEE International Conference on Software Testing. 2011:377-386.

[13] Mirzaei N, Garcia J, Bagheri H, et al. Reducing combinatorics in GUI testing of android applications[C]. ACM ICST. 2016:559-570.

[14] Azim T, Neamtiu I. Targeted and Depth-first Exploration for Systematic Testing of Android Apps[J]. Acm Sigplan Notices, 2013, 48(10):641-660.

[15] Zheng C, Zhu S, Dai S, et al. SmartDroid: an automatic system for revealing UI-based trigger conditions in android applications[J]. ACM SPSM. 2012.

[16] Choi W, Necula G, Sen K. Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning[J]. Acm Sigplan Notices, 2013, 48(10):623-640.

[17] Wang P, Liang B, You W, et al. Automatic Android GUI Traversal with High Coverage[C]. International Conference on Communication Systems & Network Technologies. IEEE Computer Society, 2014:1161-1166.