arXiv:2301.00530v1 [cs.SE] 2 Jan 2023

基于自适应语义匹配的测试复用：面向安卓移动应用

Shuqi Liu[1], Yu Zhou[1], *, Tingting Han[2],和Taolue Chen[2]1南京航空航天大学计算机科学与技术学院, 中国南京2伦敦大学伯贝克学院计算机科学与数据科学系, 英国{liushuqi, zhouyu}@nuaa.edu.cn, {t.han,t.chen}@bbk.ac.uk *通讯作者

摘要——自动测试生成有助于验证和开发移动应用的行为。利用同一类别应用之间的语义相似性进行测试复用，已被用于减少图形用户界面（GUI）测试所需的人工工作。然而，现有研究大多未能解决事件匹配中的语义问题，导致测试复用失败。为应对这一挑战，我们提出 TRASM（基于自适应语义匹配的测试复用），这是一种基于自适应策略的测试复用方法，旨在为安卓移动应用找到更好的事件匹配。TRASM 首先对通过测试生成获得的初始测试集进行 GUI 事件去重，然后采用自适应策略寻找更好的事件匹配，从而实现对现有测试的复用。与基线方法在 15 个应用上的初步对比实验表明，TRASM 能够提高 GUI 事件匹配的精确率，同时降低测试复用失败率和所需的运行时间。

关键词—自适应语义匹配；安卓移动应用；图形用户界面事件；测试复用；测试预言生成

一、引言

图形用户界面（GUI）测试通常通过设计并执行应用程序的图形用户界面测试用例 [1]，来验证和开发应用程序的行为。然而，随着移动应用功能不断增加，开发人员手动设计 GUI 测试用例（简称 GUI 测试） [2–4], 需要投入更多精力，进而降低了测试流程的效率。

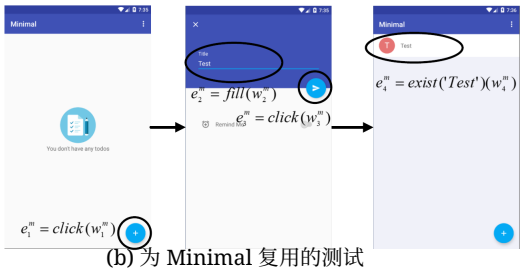
考虑到减少时间消耗的必要性，许多研究人员围绕图形用户界面测试中的自动测试生成 [5–15]开展了一系列研究。最近，一些研究人员发现，可以利用相似应用之间图形用户界面的语义相似度实现测试复用 [16–24]，从而自动生成测试。图 1 展示了一个简单示例：应用待办事项列表 的现有测试（a）成功复用于应用*Minimal*，并得到复用后的测试（b）。如图 1 所示，测试（b）中的事件 e_1^m、 e_2^m、 e_3^m和 e_4^m分别与测试（a）中的e_1^t、 e_2^t、 e_3^t和 e_4^t 相似。

现有研究主要关注如何准确选择图形用户界面中控件的特定特征，例如“文本”和“资源 ID”。研究人员结合所选特征，设计控件之间的语义相似度计算方法，以生成有意义的测试。他们尝试在相似的应用程序中选择相似度较高的控件，以匹配现有测试中的每个事件。然而，对于如何优化匹配过程，关注仍然较少。以图 1 为例，待办事项列表的控件 w_3^t 与应用程序 *Minimal*中语义上正确匹配的控件 w_3^m 在图形用户界面中的布局不同。将测试 (b) 复用到应用程序 待办事项列表时，采用现有方法可能会始终将控件 w_3^m错误地匹配到其他控件。这可能导致后续事件匹配错误，甚至造成测试复用失败。如果现有方法无法奏效，就有必要采取其他相应措施。生成事件的相似度越低，匹配不恰当的可能性就越大。因此，我们考虑挖掘此类事件，并探索语义上更相似的其他控件，以构成替代事件。

此外，在图 1 中，应用程序待办事项列表 在进入主页之前需要跳过启动页，而应用程序*Minimal* 则不需要。并且假设事件步骤为 $e_0^t = click(w_0^t)$。在此假设下，控件 w_0^t 将与应用程序*Minimal*主页上相似度最高的控件匹配。显然，该步骤产生的事件在生成的测试中是冗余的。这个简单的例子说明，在测试复用过程中，我们需要解决现有测试中的某些特定功能所导致的事件冗余问题。



(a) 待办事项列表的现有测试



(b) 为 Minimal 复用的测试

图 1. 测试复用的一个简单示例。测试 (b) 是通过复用现有测试 (a) 得到的。

受上述观察启发，本文提出一种新颖的方法 TRASM（基于自适应语义匹配的测试复用），用于在安卓

移动应用之间复用现有测试。此外，我们还与最先进的基线方法进行了对比实验，以评估我们的工作。总的来说，我们的主要贡献如下：

1）我们提出了一种名为 TRASM 的新方法，它采用自适应策略复用更多现有测试。TRASM 能够在生成的测试中获得更多语义匹配。2）TRASM 包含一种 GUI事件去重方法，可消除复用现有测试中某项特定功能所导致的重复事件，从而提升生成测试的质量。3）我们开展了大量实验，结果证实，TRASM 在提高 GUI事件匹配准确性的同时，减少了测试复用失败，并缩短了测试复用所需的运行时间。

本文其余部分安排如下：第二节介绍相关工作。第三节详细阐述本文的主要思路和所提出的方法。第四节进行实验评估。最后，第五节总结全文并展望未来研究。

II. 相关工作

A. 测试生成

为提高开发人员的效率，研究人员基于不同的探索策略提出了多种自动测试生成研究，为此奠定了坚实的基础。

Sapienz [5] 结合了随机模糊测试、系统化探索和基于搜索的探索，并利用种子注入和多级插桩自动探索和优化测试序列。Gu等人 [7] 利用基于决策树的表示对模型进行动态抽象，并通过演化机制更新模型，从而兼顾模型的准确性和规模。ConmboDroid [15] 获取用于验证应用程序独特功能的用例，然后对其进行系统枚举和组合，以生成质量更高的输入。这些工作的优点在于能够挖掘更多隐藏缺陷，或尽可能提高覆盖率。然而，这些方法生成的测试很少经过标准化处理，难以用于验证应用程序的功能。

不同于它们的目的并受其探索方法启发，我们专注于基于语义信息生成更有意义的测试。

B. 测试复用

测试复用作为一种自动生成GUI测试的替代方法，充分利用现有资源，为验证应用程序的行为提供便利。

Lin等人提出了 CRAFTDROID [19], ，一种跨应用程序的测试迁移方法。该方法利用通过静态分析提取的图形用户界面模型，按顺序匹配与现有测试语义相似的事件序列。他们成功迁移了图形用户界面和预言机事件，为改进测试迁移提供了指导。为了更准确地表示测试事件中控件的相似度，Mao等人 [20] 提出了一种基于语义的事件模糊映射策略，用于在生成目标事件时匹配候选控件。他们总是采用贪心策略，优先探索并匹配相似度最高的控件。遗憾的是，当他们的相似度计算方法效果不佳时，事件匹配的正确性就会受到威胁。考虑到测试复用的成功与否在很大程度上取决于测试事件的语义匹配，因此仍可从应用功能的角度采用适当策略，提高复用测试的质量。

Leonardo等人 [21] 开展了广泛研究，并指出，一些用于表示控件的属性会产生负面作用，而如何设计语义匹配过程是影响匹配结果的最重要因素。他们的主要发现为更好地进行测试复用提供了切入点。到目前为止，仍然没有有效的方法解决语义问题 [25, 26]。尝试优化生成的测试序列，以确保复用测试的质量，应当成为一种可选策略。

III. 我们的方法

图2展示了所提出的测试复用方法TRASM的概览。基于事件语义匹配，TRASM将现有应用程序（源测试）的测试（源应用）和新应用程序（目标应用）作为输入，并输出目标测试。TRASM通过两个重要阶段实现测试复用：初步准备和源测试复用。对于前者，通过测试增强和模型提取处理现有数据，以便实施源测试复用。对于后者，利用前一阶段获得的处理后数据，在目标应用上复用源测试。

测试增强和模型提取是沿用已有工作的初步准备步骤 [19], ，我们仅作简要介绍。具体而言，我们重点介绍自己的主要贡献。

A. 测试增强

测试增强的主要任务是在执行收集到的源测试时，提取控件的语义信息。语义化表示的控件与操作共同构成增强测试，用于匹配目标应用图形用户界面中的控件。

源应用执行每个事件后，会根据到达的图形用户界面状态，使用 *adb* 工具[1] 提取已执行事件中对应控件的语义信息。多个属性（包括类、资源 ID、文本、内容描述、可点击、密码、父级文本、同级文本、活动和包）可唯一标识图形用户界面中的一个控件。这些非空属性及其值共同构成控件的语义信息。例如，对于图 1 中测试 (a) 的控件 w_1^t，所收集的语义信息如表 1 所示。

[1]https://developer.android.com/studio/command-line/adb.html

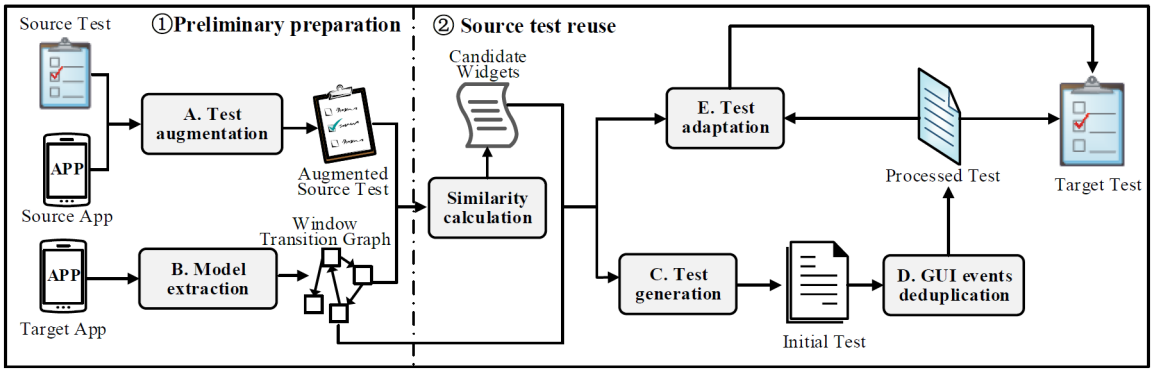


图2. TRASM 概述。

表1. 控件 w_1^t的语义信息。

属性	值
类	android.widget.ImageButton
资源 ID	fab_新建任务
可点击	true
密码	false
活动	.view.MainActivity
包	org.secuso.privacyfriendlytodolist

B. 模型提取

模型提取旨在对目标应用获取的源代码进行静态分析，以获得窗口转换图（WTG）。与现有工作一致，我们使用工具[2]获取 WTG。构建 WTG 的步骤可参见文献 [19]。

窗口转换图（WTG）可以直观地表示应用程序活动之间的交互，由节点集和边组成。在 WTG 中，节点表示应用程序的活动，有向边表示事件可触发的活动转换。例如，对于图 1 中应用程序 *Minimal* 的测试 (b)，执行事件所触发的窗口转换如图3所示，其中节点 Main 和 AddToDo 分别表示应用程序 *Minimal*的两个活动。触发边上的事件后，活动之间会发生状态转换。获得的 WTG 可以为源测试中的控件提供匹配的候选控件。不过，获得的 WTG 可能并不完整。更具体地说，我们会在执行应用程序时，根据反馈运行信息更新 WTG。

C. 测试生成

测试生成的目的是根据增强测试的语义信息以及目标应用的窗口转换图，在目标应用上生成初始测试。

增强测试中的每个事件都会迭代地与目标应用中的相应事件匹配。首先通过相似度计算获得候选控件，以匹配该事件。

[2]https://drive.google.com/file/d/1HEFS96c5nNKnzBPkWlRdwBiunOHgOs-/ view?usp=sharing

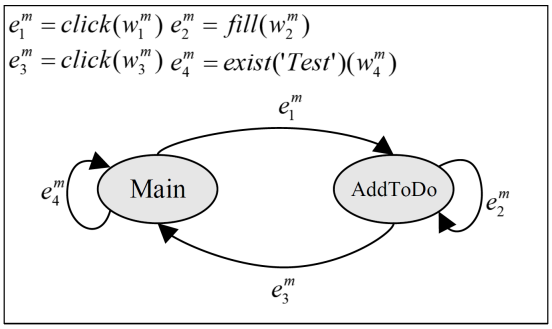


图3. 测试(b)触发的窗口转换。

相似度计算。对于每个控件，从目标应用当前 GUI 页面中提取语义信息，以构建属性词表。例如，控件 w_2^t 和控件 w_2^m的“资源 ID”属性分别为“*et new task name*”和“*userToDoEditText*”。经过预处理 [10, 19],后，我们得到两个词表$\bar{a}$=[‘edit’、‘text’、‘new’、‘task’、‘name’]和 a'=[‘user’、‘todo’、‘edit’、‘text’]。对于任意单词 $w \in a$ 和 $w' \in a'$，单词 w和 w'之间的最高相似度 max $sim(w, w')$被综合用作属性的相似度：

$$sim(a, a') = \frac{\sum_{w \in a} \max_{w' \in a'} sim(w, w')}{|a|} \quad (1)$$

其中， a和 a' 分别表示源测试中控件 S 和目标应用中控件 T对应的属性，$sim(w, w')$表示通过 Word2Vec 模型[27]得到的词向量$\overrightarrow{V_w}$ 和 $\overrightarrow{V_{w'}}$之间的余弦距离：

$$sim(w, w') = \frac{\overrightarrow{V_w} \cdot \overrightarrow{V_{w'}}}{|\overrightarrow{V_w}||\overrightarrow{V_{w'}}|} \quad (2)$$

根据方程（1），我们可以得到控件S 和控件 T之间的相似度：

$$sim(S, T) = \frac{\sum_{a \in S} sim(a, a') * wg(a)}{|S|} \quad (3)$$

其中， $wg(a)$ 表示属性 a 在所有属性中的权重。根据上述计算，我们通过选取目标应用中相似度较高的若干控件，构建候选

通过选择目标应用中相似度较高的多个控件。

我们根据获得的候选控件识别可达控件，并为其分配操作以形成事件。可以从窗口转换图中查询从当前活动到每个候选控件所在活动的所有路径。执行这些路径以识别可达控件并返回前导事件。此外，为避免重复探索路径，我们采用保留已探索路径及其对应前导事件的策略。例如，在测试（a）中匹配事件 e_2^t 时，应用程序 *Minimal* 如图3所示，在执行事件 e_1^m 后到达 AddToDo 活动，随后收集当前页面上的候选控件。通过存储的已探索路径可以发现，活动 AddToDo 与活动 AddToDo 之间存在一条可达路径。控件 w_2^m 位于该可达路径中，因此被识别为可达控件。最后，根据源事件将操作分配给控件 w_2^m。

*D. GUI*事件去重

无效的重复事件会增加测试执行的复杂性。尽管测试中的重复事件不会影响应用程序行为的触发，但GUI事件去重旨在减少此类事件所占用的时间。

由于两个应用的图形用户界面不同，目标应用可能不具备源测试中包含的特殊功能。如第 I 节所述，在测试 (a) 中复用特殊功能，也就是在应用程序 *Minimal* 上匹配事件e_0^t ，毫无意义。为从测试序列中移除此类图形用户界面事件，我们会执行去重。然而，识别测试中无意义的事件颇具挑战。我们将此操作视为检测并删除与生成的初始测试无关的重复事件。考虑到重复事件可能存在多种模式，我们制定了两条规则来区分它们。首先，如果初始测试中只有单个事件重复出现，则删除测试序列开头处的重复事件。其次，如果测试序列以 $\langle e_{n_0}, e_{n_1} \rangle$ 开头，且还包含此类 $\langle e_{n_1}, e_{n_0} \rangle$ 事件，则删除这对事件。完成此操作后，为确保正确性，我们会检查去重后的测试（即处理后的测试）能否保持与初始测试相同的功能。如果不能，我们将放弃 GUI 事件去重。

E. 测试适配

测试适配的目标是探索通过所设计的适配策略，能否找到比处理后的测试更优的测试序列。

测试生成始终优先选择相似度最高的控件进行匹配。当相似度计算方法效果不佳时，可能无法区分最适合匹配的控件，从而影响结果的准确性。测试适配的设计思路是，在处理后的测试中寻找可能包含语义更相似事件的索引，然后重新匹配。不过，要在处理后的测试序列中确定这些索引是一项颇具挑战的任务。本文首先记录处理后的测试中控件与增强测试中除当前匹配事件对应控件之外的其他控件相比相似度更高的索引。随后，我们选取处理后的测试中匹配事件相似度最低的索引，尝试找出匹配错误的事件。经过这两个处理阶段后，我们得到可执行重新匹配的事件索引集。基于以上结果，我们依次从通过第三节 III-C 节获得的候选控件中，对每个索引集中的事件重新匹配。我们设置了提前终止条件，以获得一个在语义上比原始测试序列更相似的新测试序列。

我们以图 1 中应用程序待办事项列表 为例，说明测试适配如何解决测试（b）的复用问题。如第 I 节所述，不同的图形用户界面设计会降低正确控件与源控件之间的相似度，导致事件 e_3^m 匹配错误。通过上述策略，我们获得需要重新匹配的事件 e_3^m 的索引。随后，结合模型提取获得的窗口转换图，在该索引对应的候选控件中重新搜索正确的可达控件 w_3^t ，以构成事件 e_3^t。最后，完成预言机事件 e_4^t 的匹配后，整个过程结束。

IV. 实验评估

我们将方法 TRASM 实现为一个工具。此外，我们将 TRASM 与基线方法 CRAFT-DROID [19], 进行比较，该基线方法通过语义映射在移动应用间进行测试迁移，以验证 TRASM 的有效性和效率。本节将介绍实验设置和实验结果，以评估 TRASM。

A. 实验设置

为确保一致性，我们重复使用了 [19] 的数据集[3] 来评估所提出的 TRASM。遵循基线方法的步骤，我们对三个类别中的 15 个应用进行了复用测试，包括浏览器、小费计算器和待办事项列表。这些应用来自 Google Play 和 F-Droid，这两个平台常用于图形用户界面测试领域，以探索应用程序 [5, 24, 25] 的功能。具体而言，表 2 列出了每个应用的类别、名称（版本）和来源。

具体来说，对于每个应用类别，我们选取了两项典型功能，并根据这些功能收集各应用对应的测试。为验证已实现的功能，每个测试用例的最后一个事件被设为预言机。总的来说，三个类别的应用共有六项功能，如表 3 所示。表 3 列出了每项功能的测试用例数量，以及图形用户界面事件和预言机事件的平均数量。

我们的实验在 Nexus 5X 模拟器上进行，该模拟器运行 Android 6.0 (API 23)，安装于一台配备 3.4 GHz Intel Core i7 CPU 和 32 GB RAM 的 Ubuntu 桌面计算机上。

[3]https://sites.google.com/view/craftdroid/

表 2. 应用的具体信息。

类别	应用程序（版本）	来源
a1-浏览器	a11-Lightning（4.5.1）	F-Droid
	a12-安卓浏览器 (6.0)	Google Play
	a13-隐私浏览器 (2.10)	F-Droid
	a14-FOSS 浏览器 (5.8)	F-Droid
	a15-Firefox Focus（6.0）	Google Play
a2-小费计算器	a21-小费计算器 (1.1)	Google Play
	a22-小费计算器 (1.11)	Google Play
	a23-简易小费计算器 (1.2)	Google Play
	a24-小费计算器 Plus (2.0)	Google Play
	a25-免费小费计算器 (1.0.0.9)	Google Play
a3-待办事项列表	a31-Minimal (1.2)	F-Droid
	a32-清单 (1.5.6)	F-Droid
	a33-待办事项列表 (2.1)	F-Droid
	a34-Simply Do（0.9.1）	F-Droid
	a35-购物清单 (0.10.1)	F-Droid

表 3. 典型功能的测试用例。

功能	Test 测试用例	Avg GUIs	Avg 判定器
b11-通过网址访问网站	5	3.4	1
b12-返回按钮	5	7.4	3
b21-计算含小费的账单总额	5	3.8	1
b22-分摊账单	5	4.8	1
b31-添加任务	5	4	1
b32-移除任务	5	6.8	2
总计	30	5.1	1.5

B. 实验结果

本小节在相同的评估指标下展示 TRASM 和基线方法 CRAFTDROID 的实验结果。对于每个类别中的每项功能，我们分别将一个应用程序的测试复用到其余四个应用中，测试复用总次数为 5（测试用例） × 4（目标应用） = 20。本文展示了 20 次不同测试复用的平均结果。为避免随机性，对于每次测试复用，我们对记录的多次结果取平均值。

有效性。 相比之下，TRASM 复用的测试比 CRAFTDROID 具有更高的可用性。以下两个方面，包括对成功复用的评估和对匹配事件的评估，可以证明 TRASM 方法进行测试复用的可用性。

就复用成功率而言，如表4最后一列所示，TRASM 在六项功能中的三项显著提升了测试复用率。对于功能 b21 和 b22，成功复用的测试增加了 10%。对于功能 b32，成功复用率也从 20% 提高到 25%。此外，六项功能中的两项，即 b11 和 b12，均显示出最高的成功复用率，即 100%，无论采用的是 CRAFTDROID 还是 TRASM 方法。

对于匹配事件的评估，表4的第三列和第四列分别列出了图形用户界面事件和预言机事件的精确率与召回率。如表中所示，与 CRAFTDROID 相比，TRASM 在不同功能中的图形用户界面精确率提高了 5% 至 15%。遗憾的是，精确率提升的同时，功能 b22 和 b32 的图形用户界面事件召回率分别略微下降了 2% 和 3%。复用测试的成功与否取决于测试序列中最后一个预言机事件是否匹配正确

因此，成功复用率的提升也意味着预言机事件召回率的提高，如表4所示。其中，功能 b21 和 b22 的提升最为显著，召回率提高了 10%。总体而言，预言机事件精确率和召回率的提升表明，所提出的 TRASM 确实提高了复用测试的可用性。

效率。 图4列出了各项功能的平均测试复用时间。显然，对于每项功能，TRASM 的平均复用测试耗时都少于 CRAFTDROID。最显著的效果体现在功能 b21 上：CRAFTDROID 的平均复用测试耗时为 2581 秒（43 分钟），而我们的 TRASM 只需 890 秒（15 分钟），接近 CRAFTDROID 耗时的 35%。总之，这些结果归因于两个因素：一是存储已探索路径避免了重复耗时，二是自适应策略提高了控件匹配效率。上述结果证明，我们能够突破 CRAFTDROID 在效率方面的限制。

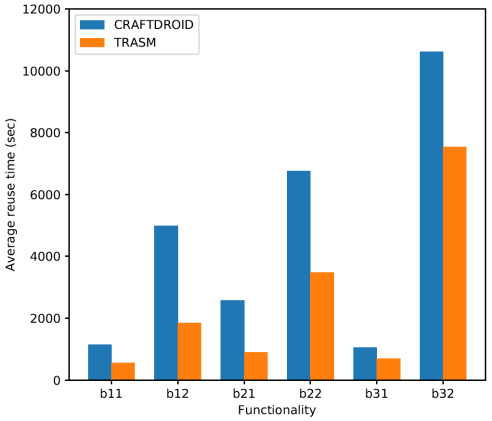


图4. 各项功能的平均测试复用时间 性。

在评估 TRASM 的效率时，一个重要发现是，实施自适应策略可以提高部分复用测试的成功率。不可避免的潜在缺点是，若无法找到更合适的事件匹配，则会增加额外的时间开销。我们需要进一步解决这一关键问题，以平衡效率与有效性。

V. 结论

作为测试生成的替代方法，测试复用可以帮助开发人员验证应用程序的行为。本文提出了一种新颖的测试复用方法，以缓解事件匹配中的语义问题。我们基于初始测试集扩展了图形用户界面事件去重和测试适配，从而构建目标测试。实验结果表明，与基线方法相比，我们提出的方法性能更好，且复用测试的可用性更高。

表4. 有效性和效率评估

功能	方法	图形用户界面事件		预言机事件		成功复用
		精确率	召回率	精确率	召回率	
b11	CRAFTDROID	79%	100%	100%	100%	20/20(100%)
	TRASM	**100%**	100%	100%	100%	20/20(100%)
b12	CRAFTDROID	85%	100%	100%	100%	20/20(100%)
	TRASM	**100%**	100%	100%	100%	20/20(100%)
b21	CRAFTDROID	82%	100%	100%	80%	16/20(80%)
	TRASM	**93%**	100%	100%	**90%**	**18/20(90%)**
b22	CRAFTDROID	80%	**100%**	100%	65%	13/20(65%)
	TRASM	**85%**	98%	100%	**75%**	**15/20(75%)**
b31	CRAFTDROID	78%	100%	85%	100%	17/20(85%)
	TRASM	**87%**	100%	85%	100%	17/20(85%)
b32	CRAFTDROID	69%	**100%**	85%	80%	11/20(55%)
	TRASM	**81%**	97%	**93%**	**81%**	**12/20(60%)**

我们认为，匹配事件是一个很有前景的方向，未来计划进一步研究如何改进匹配策略。此外，我们计划验证该方法的泛化能力，并进一步探索测试复用对更多应用的影响。

致谢

本研究部分获得中国国家自然科学基金（编号：61972197）、江苏省自然科学基金（编号：BK20201292）以及新型软件技术与产业化协同创新中心的资助。T. Chen 部分获得南京大学新型软件技术国家重点实验室海外基金（KFKT2022A03）、Birkbeck BEI 学校项目（EFFECT）以及国家自然科学基金（NSFC）项目（编号：62072309、62272397）的资助。

参考文献

[1] S. Anand、M. Naik、M. J. Harrold 和 H. Yang,《智能手机应用的自动化具体符号执行测试》，*ACM SIGSOFT* 第二十届软件工程基础国际研讨会论文集，第 1–11 页，2012 年 11 月。[2]M. E. Joorabchi、A. Mesbah 和 P. Kruchten,《移动应用开发中的现实挑战》，*2013* 年 *ACM/IEEE* 实证软件工程与度量国际研讨会。IEEE，第 15–24 页，2013 年 10 月。[3] P. S. Kochhar、F. Thung、N. Nagappan、T. Zimmermann 和 D. Lo,《了解应用开发人员的测试自动化文化》，*2015* 年 *IEEE* 第八届软件测试、验证与确认国际会议。IEEE，第 1–10 页，2015 年 4 月。[4] M. Linares-V´asquez、C. Bernal-C´ardenas、K. Moran 和 D. Poshyvanyk,《开发人员如何测试安卓应用程序？》，载于*2017* 年 *IEEE* 软件维护与演化国际会议（*ICSME*）。IEEE，第 613–622 页，2017 年 9 月。[5] K. Mao、M. Harman 和 Y. Jia,《 Sapienz：面向安卓应用程序的多目标自动化测试》，第二十五届软件测试与分析国际研讨会论文集，第 94–105 页，2016 年 7 月。

[6] A. Machiry、R. Tahiliani 和 M. Naik,“ Dynodroid：一种面向安卓应用的输入生成系统”，*2013*年第*9*届软件工程基础联合会议论文集，第224–234页，2013年8月。[7] T. X. Gu、C. N. Sun、X. X. Ma、C. Cao、C. Xu、Y. Yao、Q. R. Zhang、J. Lu 和 Z. D. Su,“通过模型抽象与细化实现安卓应用的实用图形用户界面测试”，*2019*年*IEEE/ACM*第*41*届国际软件工程大会。IEEE计算机学会，第269–280页，2019年8月。[8] N. Mirzaei、H. Bagheri、R. Mahmood 和 S. Malek,“ Sigdroid：面向安卓应用的自动化系统输入生成”，*2015*年*IEEE*第*26*届国际软件可靠性工程研讨会。IEEE计算机学会，第461–471页，2015年11月。[9]M. Ermuth 和 M. Pradel,“猴子看，猴子做：利用推断出的宏事件有效生成图形用户界面测试”，第*25*届国际软件测试与分析研讨会论文集，第82–93页，2016年7月。[10] Y. Zhou、Y. Su、T. Chen、Z. Huang、H. C. Gall 和 S. Panichella,“基于用户评论的移动应用变更文件定位”，*IEEE*软件工程汇刊，第47卷，第12期，第2755–2770页，2020年1月。[11] Z. Dong、M. B¨ohme、L. Cojocaru 和 A. Roychoudhury,“ 安卓应用的时间旅行测试 ”，*2020*年*IEEE/ACM*第*42*届国际软件工程大会，韩国首尔，第481–492页，2020年7月。[12] A. Memon、I. Banerjee 和 A. Nagarajan,“图形用户界面提取：用于测试的图形用户界面逆向工程”，第*10*届逆向工程工作会议，*2003*年。*WCRE 2003*。会议论文集。Citeseer，第260–269页，2003年。[13] T. Su、G. Meng、Y. Chen、K. Wu、W. Yang、Y. Yao、G. Pu、Y. Liu 和 Z. Su,“安卓应用的引导式随机模型驱动图形用户界面测试”，*2017*年第*11*届软件工程基础联合会议论文集，第245–256页，2017年8月。[14]D. Amalfitano、A. R. Fasolino、P. Tramontana、S. De Carmine 和 A. M. Memon,“利用图形用户界面提取对安卓应用进行自动化测试”，*2012*年论文集：第*27*届*IEEE/ACM*国际会议

自动化软件工程。IEEE，第258–261页，2012年9月。[15] J. Wang、Y. Y. Jiang、C. Xu、C. Cao、X. X. Ma 和 J. Lu，“Combodroid：通过用例组合为安卓应用生成高质量测试输入”，*ACM/IEEE*第*42*届国际软件工程大会论文集，第469–480页，2020年6月。[16]F. Behrang 和 A. Orso，“移动应用的自动化测试迁移”，第*40*届国际软件工程大会论文集：伴随会议论文集，第384–385页，2018年5月。[17] F. Behrang 和 A. Orso，“在功能相似的移动应用之间迁移测试”，*2019*年第*34*届*IEEE/ACM*国际自动化软件工程大会。IEEE计算机学会，第54–65页，2019年11月。[18] F. Behrang 和 A. Orso，“Apptestmigrator：用于安卓应用自动化测试迁移的工具”，*2020*年*IEEE/ACM*第*42*届国际软件工程大会：伴随会议论文集。IEEE，第17–20页，2020年10月。[19] J. W. Lin、R. Jabbarvand 和 S. Malek，“通过语义映射在移动应用之间迁移测试”，*2019*年第*34*届*IEEE/ACM*国际自动化软件工程大会，第42–53页，2019年11月。[20] Q. Mao、W. W. Wang、F. You、R. L. Zhao 和 Z. Li，“相似安卓应用之间的用户行为模式挖掘与复用”，《系统与软件杂志》，第183卷，第111085页，2021年9月。[21]L. Mariani、A. Mohebbi、M. Pezz`e 和 V. Terragni，“用于测试复用的图形用户界面事件语义匹配：我们已经做到这一步了吗？”，第*30*届*ACM SIGSOFT*国际软件测试与分析研讨会论文集，第177–190页，2021年7月。[22] X. Qin、H. Zhong 和 X. Y. Wang，“Testmig：将图形用户界面测试用例从iOS迁移至安卓”，第*28*届*ACM SIGSOFT*国际软件测试与分析研讨会论文集，第284–295页，2019年7月。[23] A. Rau、J. Hotzkow 和 A. Zeller，“海报：通过学习其他应用的测试，高效生成图形用户界面测试”，*2018*年*IEEE/ACM*第*40*届国际软件工程大会：伴随会议（*ICSE-Companion*）。IEEE，第370–371页，2018年6月。[24] A. Rau、J. Hotzkow 和 A. Zeller，“在Web应用之间迁移测试”，国际*Web*工程大会。施普林格，第50–64页，2018年5月。[25] S. R. Choudhary、A. Gorla 和 A. Orso，“安卓自动化测试输入生成：我们已经做到这一步了吗？(e)”，*2015*年第*30*届*IEEE/ACM*国际自动化软件工程大会。IEEE，第429–440页，2015年11月。[26] X. Zeng、D. Li、W. Zheng、F. Xia、Y. Deng、W. Lam、W. Yang 和 T. Xie，“安卓自动化测试输入生成：在工业案例中，我们真的已经做到这一步了吗？”，载于*2016*年第*24*届*ACM SIGSOFT*国际软件工程基础研讨会论文集，第987–992页，2016年11月。[27] T. Mikolov、I. Sutskever、K. Chen、G. S. Corrado 和 J. Dean，《词与短语的分布式表示及其组合性》，神经信息处理系统进展，第 26 卷，2013 年。