计算机科学
COMPUTER SCIENCE

基于SA-UCB算法的Android应用程序自动化测试方法

王嬉, 赵春蕾, 步志亮, 杨艺

引用本文

王嬉, 赵春蕾, 步志亮, 杨艺. 基于SA-UCB算法的Android应用程序自动化测试方法[J]. 计算机科学, 2023, 50(11A): 221200145-7.

WANG Xi, ZHAO Chunlei, BU Zhiliang, YANG Yi. Automated Testing Method of Android Applications Based on SA-UCB Algorithm [J]. Computer Science, 2023, 50(11A): 221200145-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221200145

基于 SA-UCB 算法的 Android 应用程序自动化测试方法

王　嬉　　赵春蕾　　步志亮　　杨　艺
天津理工大学计算机科学与工程学院　天津 300384
天津理工大学教育部视觉与系统省部共建重点实验室　天津 300384
天津理工大学天津市智能计算与软件新技术重点实验室　天津 300384
(1376231271@qq.com)

摘　要　针对传统强化学习算法需要行为准则学习导致测试效率偏低这一问题,提出一种基于模型的 Android 应用程序自动化测试方法——SA-UCB。使用 Sarsa 算法对测试过程进行指导,采取 Q 表作为动作策略的选取参照。并针对经典 Sarsa 算法使用的 ε-greedy 策略随机性过强的问题,引入上界置信算法(the Upper Confidence Bound Algorithm,UCB 算法)来平衡测试过程中的"探索-利用窘境",从而对 Sarsa 算法进行改进,使动作决策更加分散化,并将其应用于 Android 自动化测试过程,提高了测试效率。将 SA-UCB 方法与其他 5 种测试方法从测试覆盖率、测试效率、故障检测 3 个方面进行了测试性能的对比实验,结果表明,在相同的实验条件下,SA-UCB 策略在测试覆盖率和测试效率方面具有一定优势。

关键词:Android;自动化测试;强化学习;Sarsa;UCB
中图法分类号　TP391

Automated Testing Method of Android Applications Based on SA-UCB Algorithm

WANG Xi,ZHAO Chunlei,BU Zhiliang and YANG Yi
School of Computer Science and Engineering,Tianjin University of Technology,Tianjin 300384,China
Key Laboratory of Computer Vision and System of Ministry of Education,Tianjin University of Technology,Tianjin 300384,China
Tianjin Key Laboratory of Intelligent Computing and New Software Technology,Tianjin University of Technology,Tianjin 300384,China

Abstract　Aiming at the problem that the traditional reinforcement learning algorithm needs to learn the code of conduct,which leads to low testing efficiency,a model-based automated testing method for Android applications,SA-UCB,is proposed. The Sarsa algorithm is used to guide the test process,and the Q table is used as the reference for action strategy selection. And for the randomness of ε-greedy integrated by the classical Sarsa algorithm is too strong,the upper confidence bound algorithm(UCB algorithm) is introduced to balance the "exploration-exploitation dilemma",which makes action decisions more decentralized. And it is applied to the Android automated testing process,the testing efficiency is improved. The SA-UCB method is compared with other five test methods in terms of test coverage,test efficiency and fault detection. The results show that SA-UCB strategy has certain advantages in test coverage and test efficiency under the same experimental conditions.

Keywords　Android,Automated testing,Reinforcement learning,Sarsa,UCB

1　引言

Android 应用程序在投入市场之前,需要测试人员对其进行充分测试,发现并修复存在的问题,从而给用户带来更好的使用体验。据中国互联网络信息中心的最新数据显示,截至 2022 年 2 月,我国国内市场上监测到的移动应用程序数量为 235 万款[1]。面对如此庞大的市场需求,需要研究人员开发出效果更优、效率更高的 Android 应用程序测试方法。

自动化测试方法大多使用机器学习算法将人为驱动的测试行为转化为机器执行,具有测试成本低、测试效率高的特点,更适用于应用程序的大规模测试,且因为测试过程几乎不需要人工干预,减少了人为偏见。目前大多数机器学习算法都需要预先进行数据训练的操作,再将训练好的数据输入模型中,最后执行相应的任务,这个过程大大增加了工作量。另外,这种方法虽能够保证测试的效率,但由于 Android 完全开源的特点,开发十分灵活,导致先前应用程序的测试经验并不一定适用于当前被测应用程序的测试。

强化学习算法由智能体和环境构成,在和环境交互的过程中,智能体动态地学习策略,无须数据集的训练,大大减少了测试人员手动准备数据的工作量[2]。自 20 世纪 80 年代以来,强化学习已经被应用到许多领域,包括多智能体系统[3]、机器人[4-5]、自动驾驶技术[6]等。强化学习算法可以在测试推进的过程中,为每个应用程序单独训练模型,并将测试用例的生成转换为多摇臂赌博机问题,通过指定算法来决策出对测试

基金项目:科技部"科技助力经济 2020"重点专项(SQ2020YFF0413781)
This work was supported by the Key Special Project of "Science and Technology Helps Economy 2020" of the Ministry of Science and Technology (SQ2020YFF0413781).
通信作者:赵春蕾(zcltjut@126.com)

结果有较大影响的事件。本文的研究工作也围绕基于强化学习的 Android 应用程序自动化测试展开。

本文的研究目标是对 Android 应用程序进行自动化测试，要求能够在有限时间内达到较高的测试覆盖率，同时尽可能多地发现被测应用程序存在的问题。现有的 Android 应用程序测试工具存在生成事件冗余、局部循环等问题，导致测试效率偏低。针对这些问题，本文提出一种 SA-UCB 测试方法，将 Android 自动化测试过程抽象为马尔可夫决策过程，使用 Sarsa 算法对测试过程进行指导，采取 Q 表作为动作策略的选取参照，并在此基础上，使用 UCB 算法来平衡测试过程中的“探索-利用窘境”，来对 Sarsa 算法进行改进，并将其应用于 Android 自动化测试过程。

2 研究背景与相关工作

2.1 基于模型的自动化测试

基于模型的探索策略是目前在 Android 自动化测试中使用最广泛的方法，经常与 ML 算法结合应用，能够根据 AUT 生成以状态为顶点、事件为边的有向图模型。

AndroidRipper[7]，MobiGUITAR[8]，ORBIT[9] 和 AMOLA[10] 使用状态机来表示应用程序模型，但仅使用基础的图搜索算法来实现简单的 UI 探索，因此其性能受到一定限制。Su 等提出了 Stoat[11]，这是一种在应用程序上执行随机模型测试的新型引导方法，利用应用程序的行为模型迭代地优化测试生成过程，以得到高覆盖率和更加多样化的事件序列。Li 等在 2017 年提出一个轻量级的基于模型的测试工具：DroidbotX[12]，该测试工具目前提供 5 种探索策略：dfs_naive，dfs_greedy，bfs_naive，bfs_greedy，memory_guided。其中，memory_guided 策略是 Droidbot 团队于 2021 年引入的新策略，使用 ML 算法来自动识别相似视图并避免冗余探索。字节跳动技术团队所开发的 Fastbot[13] 在服务器端进行模型的构建，结合机器学习和强化学习技术，提出了多套有向有环图的启发式遍历算法，能够避免基于模型的自动化测试过程中容易出现的局部循环问题。该团队在 2022 年开发了 Fastbot2，引入了可重用的事件活动转换知识，并提供概率模型，实现了应用程序功能的快速覆盖[14]。

基于模型的探索策略具有在模型中捕获 AUT 行为信息的优势，能够为 AUT 生成有意义的测试用例。测试人员可以对模型本身进行测试和验证，减少了测试环节所涉及的成本。因此，本文使用基于模型的探索策略，并集成 RL 算法来实现测试过程的有效探索。

2.2 基于强化学习的自动化测试

强化学习算法侧重于从交互中进行目标导向学习，直接将行动与结果联系起来，根据奖赏值判断最佳行动[15]。强化学习任务通常用马尔可夫决策过程来描述，在执行任务的过程中，智能体在探索动作的同时，动态地进行学习，每执行完一个动作，智能体都会从环境中获得奖励。强化学习策略的优劣则取决于长期执行这一策略后得到的累积奖励，而强化学习的目的就是要找到能使长期累计奖励最大化的策略[16]。

2012 年，Mariani 等提出了第一个基于强化学习的 Android GUI 自动化测试工具：AutoBlackTest[17]，它集成了强化学习与启发式学习，将为未知应用程序生成测试用例的问题转变为学习如何在未知环境中有效操作的代理问题。AndroFrame[18] 工具能够学习用于所有应用程序的单个 Q 值矩阵，避免在测试期间为每个被测应用程序进行训练。DroidbotX[19] 使用带有 UCB 策略的 Q-Learning 算法，能够将冗余事件最小化，提高测试覆盖率。2019 年，Li 等基于 DQN 提出了 Humanoid[20]，能够学习人类与应用程序交互的过程，在交互过程中判断事件的重要性，并对其进行优先级排序。Vuong[21] 等首先识别 GUI 元素的语义，再将其作为 DQN 中神经网络的输入，指导测试工具更频繁地探索只能通过特定操作序列访问的功能。

然而，虽然 Q-Learning 拥有良好的最终性能，但相比其他算法，其在测试过程中需要更长的学习时间，导致其测试效率相对较低。DQN 算法能够解决传统的表格型方法在大规模强化学习任务时遇到的执行效率低、存储量低等问题。然而，在对 Q 函数进行估计时，DQN 算法取得的最大化 Q 值会高于真实的最大 Q 值，这种累积的错误会导致坏状态被估计为高值，从而导致次优的策略更新和以及行为发散问题。

Sarsa 是最重要的强化学习算法之一[22]，较为灵活，不需要状态转移矩阵以及完整序列，拥有更高的效率以及更好的收敛特性[23]。AIMDROID[24] 实现了 Sarsa 算法引导的随机方法，通过禁用 Activity 转换来关注单个 Activity，再使用强化学习引导的模糊算法集中对每一个 Activity 进行探索，基于 Sarsa 算法贪婪地选择更有可能触发新状态或崩溃的事件。缺点在于禁用 Activity 转换可能会导致丢弃由 Activity 生命周期引起的一些错误。Khan 等[25] 提出一种基于 Sarsa 的自动化测试方法，使用 ε-greedy 算法作为探索事件和更新 Q 值的策略。然而，ε-greedy 是一种贪心算法，随机性过强，且在探索的过程中仅仅考虑收益回报，导致其健壮性较差，在测试过程中容易受到噪音数据的影响。

因此，为了提高测试效率，本文提出了一种基于 Sarsa 算法的 Android 应用程序自动化测试方法。同时，为了避免 ε-greedy 算法随机性强导致的局部循环问题，引入 UCB 探索——利用策略来指导测试的探索过程。

3 基于强化学习的 Android 应用程序自动化测试方法

3.1 Android 应用程序测试与强化学习任务

本文将 Android 自动化测试过程转化为马尔可夫决策过程，采用强化学习算法中的四元组 (S, A, P, R) 对测试过程中涉及的元素进行定义。

S 表示被测应用程序的状态集合。状态分类标准的粒度越细，最终达到的测试覆盖率越高，测试过程中触发的错误数也就越多。因此，本课题将当前界面的具体内容作为状态的划分标准，使用 UIAutomator 来获取当前界面的控件树，并进行界面信息的比较。

A 表示智能体与被测应用程序交互动作的集合。测试框架生成交互事件来与被测应用程序的控件进行交互，支持的事件类型包括：UI 输入（包括点击、长按等）、Intent 事件、要上传的图像文本文件、传感器数据。

P 用于描述动作执行后的状态跳转情况。该跳转情况是在应用程序的开发阶段由开发人员决定的，测试过程中无法对其进行更改。本节强调的是非确定的状态跳转场景，如今市场上的 Android 应用程序的结构十分复杂，在相同状态下，对同一控件进行相同操作，得到的状态可能是不同的。因此，需要智能体来给状态一个奖赏值，倾向于给予未访问过的

状态更高的奖励，但是访问过的状态依然需要给一定权值，来避免由于两次操作得到的结果不同而影响最终测试结果。

R 表示动作执行后的奖赏给定方式。每当测试框架执行交互事件后，都会获得相应奖赏。SA-UCB 基于被测应用程序当前状态来构建 Q 表，Q 表的每一行表示特定状态的预期 Q 值，行的大小代表该状态的可执行操作数。在信息初始化阶段，需要根据当前界面的信息推断出可执行事件的集合，将其中所有事件赋予初始值为 1，记录在 Q 表中。当执行某一事件时，该操作的值将被降低为 0.99，而不是 0。这样设定的目的是将访问过的控件和未访问过的控件进行区分，未执行过的控件的奖赏值相对较大，从而引导测试过程优先访问未被访问过的控件，同时依然赋予访问过的控件再次被访问的机会，减少先后两次执行同一控件导致不同状态问题对测试结果的影响。

3.2　SA-UCB 算法

本文对经典 Sarsa 算法进行了改进，集成了 UCB 算法来对强化学习中的“探索-利用窘境”进行平衡，同时缓解 ε-greedy 带来的初始化的随机性过强的问题。

该算法在执行动作的过程中进行学习，这意味着 Sarsa 根据当前策略执行的动作而不是决策策略来学习 Q 值，使用式(1)更新状态和动作的 Q 值：

$$Q(s,a)\leftarrow Q(s,a)+\alpha(r+\gamma Q(s',a')-Q(s,a)) \tag{1}$$

其中，α 为学习率，表示在当前决策中要学习的上一次决策误差，根据 Carino[26] 等的工作，本文将其定义为 0.3。衰减因子 γ 用于控制即时奖励与未来奖励的相关性，取值范围通常在 [0,1] 内。γ 值越趋近于 0，代表测试过程越看重当下的即时奖励；γ 值越趋近于 1，代表测试过程越看重未来奖励。先前工作[27] 指出，当 γ 设置为 0.9 时，测试方法能够产生更优的效果。

在测试过程中，测试框架或用户与被测应用程序的交互事件可以看作马尔可夫决策过程中的动作(Action)，在每次进行 Q 表的更新时，Sarsa 都基于确定的事件，$Q(s,a)$ 代表通过执行从状态 s 开始的一系列事件而获得的预期奖励，一个事件对一个新状态的初始 Q 值为 1，Q 值最大的事件更有可能被重复选择。但是，如果相应的事件不能获得积极的奖励，即当它们不能触发新的状态时，最大的 Q 值将会下降。

在测试过程中，UCB 算法通过每个事件的取值区间的上界，来代替奖励期望进行选择，每次决策的目的是找出在下一步能够获得最大期望回报的事件，UCB 算法函数可被描述为：

$$action=\underset{a}{\operatorname{argmax}}\left[Q_t(s_t,a^i)+\sqrt{\frac{\log N_{s_t}}{N(s_t,a^i)}}^{c}\right]$$

其中，Q 值为事件 a 的预期回报，根号中代表对第 t 个事件的价值估计的不确定度，最大值为事件 t 的可能真实值的上限，N 表示事件 a^i 状态 s_t 被选择的次数，对数代表在当前状态中选择事件的频率，c 表示控制探索级别的置信值，置为 1。随着时间的推移，所有事件都有被执行的机会，但是在整个探索过程中具有较低价值估计的事件或者已经被选择了多次的事件被选择的频率较低，从而在保证能够探索到优先级较高的状态的同时，也对较少访问以及价值较低的状态进行访问。

3.3　基于 SA-UCB 的 Android 应用程序自动化测试方法

基于 SA-UCB 算法的 Android 自动化测试方法框架如图 1 所示、主要由 Droidbot 和 Android 端，控制端 3 个模块构成。

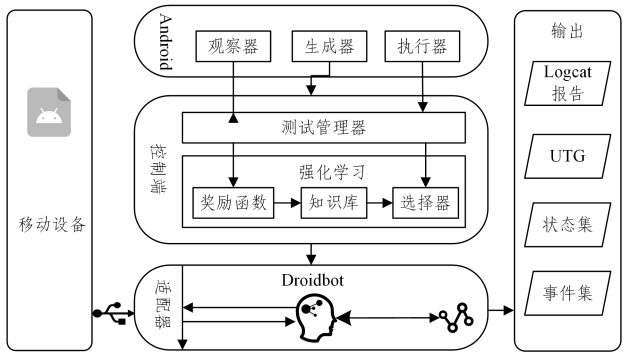


图 1　系统总体结构

Fig. 1　Overall system structure

Android 具有完全开源、开发自由度高的特点，为开发人员带来方便的同时也导致了碎片化问题，给 Android 应用程序的测试带来了新的挑战。DroidBot 是一种轻量级的用户界面(User Interface，UI)引导测试输入生成器，能够做到可以与几乎任何移动设备上的任何被测应用程序进行交互。因此，本文引入了 DroidBot 作为测试环境的一部分，解决了先前测试工具由 Android 碎片化问题导致的兼容性差的问题。

在对应用程序进行测试之前，需要将测试工具通过 Android 调试桥(Android Debug Bridge，ADB)与测试设备进行连接，该设备可以是真机、模拟器或定制的沙箱。系统将被测应用程序的 Android 应用程序包(Android Application Package，APK)文件作为输入，适配器充当测试环境和测试生成算法之间的桥梁，提供测试环境和被测应用程序的抽象。它能够实时监听设备和被测应用程序的状态，将当前的状态信息转换为结构化数据，同时接受算法生成的测试输入，并将其转换为命令。

Android 端模块包含观察器、动作生成器以及执行器 3 个部分。其中，观察器用于对被测应用程序当前状态的布局控件等信息进行观察分析，动作生成器可以根据当前图形用户界面(Graphical User Interface，GUI)的状态来实时构建状态和动作，包括点击、长按、滑动、输入文本等，并将其转换为测试输入，再将该输入反馈到强化学习模块，执行器用于在 Android 设备上执行控制端发送的指令。

控制端可分为测试管理器和强化学习两部分。测试管理器是整个测试过程的中枢，相当于在整个测试过程中产生的动作、状态及结果的中转站。强化学习模块包含奖励函数、知识库和动作选择器。其中，奖励函数用于给状态-动作对赋值，知识库用于储存各动作以及状态的相关信息，动作选择器基于 SA-UCB 算法给出的知识来选择 Android 端要执行的动作。

基于 SA-UCB 的 Android 自动化测试方法将整个测试过程抽象为若干个迭代过程，如图 2 所示。

1)获取被测应用程序当前状态。在每次迭代过程中，首先使用 UIAutomator 来获取被测应用程序的当前状态。

2)给定奖赏值。被测应用程序的当前状态会与部分已探索到的状态进行比较，这些状态被记录在知识库中进行保存。如果状态与知识库中已有的某个状态内容相同，则就给其一个较小的奖赏值；否则将其定义为新状态，并赋予其一个较大的奖赏值，并记录到知识库中。

3)更新价值。计算得到奖赏值后，使用 Sarsa 函数对上一轮迭代过程中的状态-动作对的价值进行更新，并将更新后的值存储在 Q 表中。

4)推断可执行事件。完成状态-动作对的更新后,需要对当前状态下可执行的事件进行推断。首先,需要先查询 Q 表中是否有当前状态对应的信息,如果有,则直接获取可执行事件集合;如果没有,则需要对当前状态的 Q 表进行初始化。在信息初始化阶段,需要根据当前界面的信息推断出可执行事件的集合,并将其赋予初始值为 1,记录在 Q 表中。首次探索的状态-动作对会被赋予较大的值,以引导测试工具优先执行未被触发的操作。

5)选择并执行事件。最后根据 UCB 函数对当前状态所有可执行时间进行评估,选出事件交由 Droidbot 框架执行。执行后,被测应用程序会跳转到新的状态,整个测试过程进入到一次新的迭代过程中。

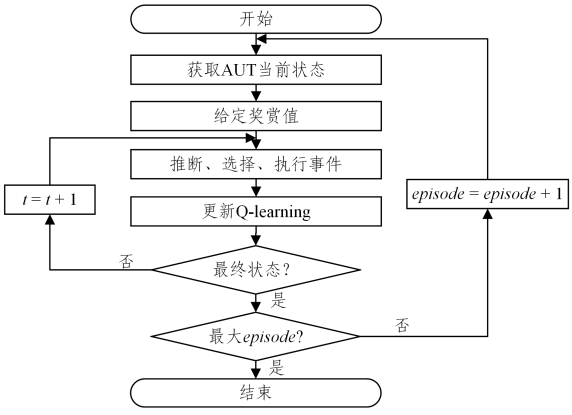


图 2 系统总体结构

Fig. 2 SA-UCB flowchart

在整个测试过程中,会动态地对状态转换图进行构建。可以对测试过程结束的标准进行设定,当测试时间或生成事件数量达到预设标准时,测试过程会自动停止,并输出如图 3 所示的 UI 状态转换图(UI Transition Graph,UTG)、Logcat 报告、状态集、事件集等文件。

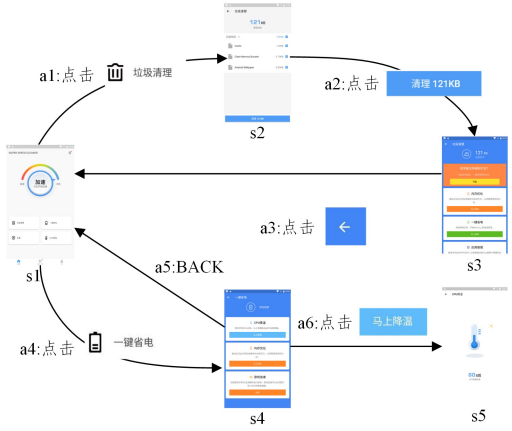


图 3 UTG 示例

Fig. 3 UTG example

4 评估

本文选取出当前最先进的 5 种开源自动化测试方法,在 51 个开源应用程序以及 102 个商用应用程序上和本文提出的 SA-UCB 方法进行了对比实验,并针对以下 3 个问题进行测试性能的对比。

RQ1:与其他测试方法相比,SA-UCB 算法是否达到了更高的测试覆盖率?

RQ2:与其他测试方法相比,SA-UCB 算法是否具有更高的测试效率?

RQ3:与其他测试方法相比,SA-UCB 算法是否能发掘更多的故障?

4.1 实验配置

本次实验的测试环境为:Windows10 操作系统;CPU:Intel Core i7-6700 处理器;16 GB 内存;GPU:GTX950M;Pycharm 2020;Python3. 9 脚本语言;Android 7. 0;Java;Gym;Stable-Baselines;Tensorflow;夜神模拟器。

行覆盖率能够反映出测试过程中被测应用程序执行过的代码量,是衡量测试是否完全的重要指标,与 Activity 覆盖率相比,其能够更精确地体现测试过程的完整性。然而,由于在检查被测应用程序行覆盖率情况时,需要对源代码进行插桩操作,行覆盖率只适用于开源应用程序。

F-Droid 是一个提供开源应用程序的免费下载平台,先前的很多研究[28-29]都使用这一平台提供的应用程序作为测试对象。为了避免主观偏见,同时保证数据集中应用程序类别的多样性,本文从 F-Droid 提供的 17 个类别的应用程序中每类随机选取 3 个,最终将得到的 51 个应用程序作为第一组数据集进行实验。然而,图 4 表明,F-Droid 平台提供的应用程序大多结构简单、功能单一,仅仅使用该数据集得到的测试结果代表性较差。

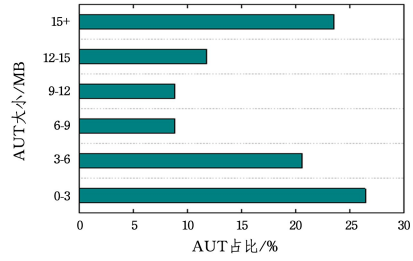


图 4 开源数据集

Fig. 4 Open source datasets

因此,第二组数据集来自于应用市场,使用爬虫技术,对国外 Apkfab[30]和国内腾讯软件中心[31]两个应用市场的应用程序进行爬取操作。相似地,本文分别在 17 个类别排名前 50 的应用程序中随机选取 6 个,将获取到的 102 个应用程序作为第二组数据集,得到的应用程序大小分布情况如图 5 所示。

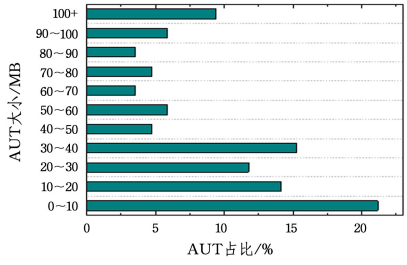


图 5 应用市场数据集

Fig. 5 Market application dataset

将本文提出的测试方法与近几年市场上较有代表性的自动化测试方法进行测试性能的对比,包括 Droidbot 集成的 dfs_naive 和 memory_guided 策略、Fastbot、DroidbotX 以及传统 Sarsa 算法。本文中用来设置对比实验的测试工具均属于黑盒测试,使用基于模型的测试策略,提供错误报告,支持回放,能够在真机和虚拟环境下进行测试,能够在测试过程中动态地分析被测应用程序的 GUI 界面,并提取出所有可能的

事件序列的过程。其中，Fastbot 仅支持生成 GUI 事件，不能生成系统以及文本事件作为测试输入，而其余 3 个测试工具均能够生成 GUI、系统以及文本作为输入事件。Fastbot 工具基于 Monkey[32] 开发，因此生成事件速度非常快，平均每小时生成 57 444 个事件，其他测试工具平均每小时生成约 230 个事件。

4.2　实验结果

RQ1（测试覆盖率）：本文使用测试覆盖率来对应用程序在所选测试工具上的表现进行性能评估。针对开源应用程序，使用行覆盖率作为评判标准；针对应用市场获取到的应用程序，使用 Activity 覆盖率作为评判标准。本文对上述应用程序在各个测试工具运行 1 h 的测试覆盖率的表现来进行对比分析。为了避免实验的偶然性，实验共重复 3 次，取平均值作为最终的实验结果，各个工具的测试覆盖率表现情况如图 6 和图 7 所示。其中，dn，Fb，SU，DX，mg，SA 分别代表 dfs_naive、Fastbot、SA-UCB、DroidbotX、memory_guided 以及经典 Sarsa 测试方法。

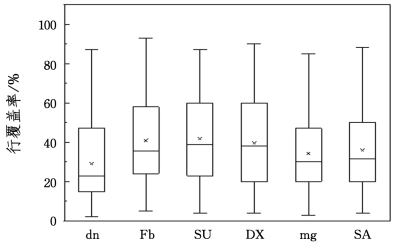


图 6　开源应用程序的行覆盖率对比

Fig. 6　Line coverage comparison of open source applications

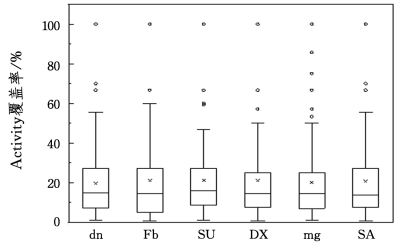


图 7　商用应用程序的 Activity 覆盖率对比

Fig. 7　Comparison of Activity coverage of commercial applications

对于开源应用程序，上述测试方法分别实现了平均 29.09%，40.91%，41.71%，39.71%，34.24%，35.93% 的行覆盖率；对于应用市场获取到的应用程序，分别实现了平均 19.45%，21.07%，21.24%，21.17%，20.14%，20.76% 的 Activity 覆盖率。特别地，本文提出的 SA-UCB 测试方法在两个数据集上的平均测试覆盖率均达到最高。dfs_naive 算法的行覆盖率相对较低，通过观察使用该算法生成的 UTG 发现，测试过程中存在着大量的环路，这表示其测试过程容易陷入局部循环当中，导致了过多的冗余操作，只覆盖到了有限的状态而无法退出，因此其测试性能相对较差。

值得注意的是，测试应用市场中收集到的应用程序得到的 Activity 覆盖率的异常值相对较多，平均值相对较低，且下限较低，分布在 0.68%～1.18% 之间。这是由于从应用市场获取到的应用程序结构更为多样，如图 5 所示。对于结构复杂的应用程序，1 h 的测试时间不足以使测试工具对其进行充分探索，因此其 Activity 覆盖率也会相对较低。

与其他测试方法相比，本文提出的 SA-UCB 算法、传统 Sarsa 算法以及 DroidbotX 提出测试方法在测试商用应用程序时的 Activity 覆盖率下限更低。通过观察数据集中大小超过 100 Mb 的 11 个应用程序测试 1 h 的平均 Activity 覆盖率发现，对于结构复杂的应用程序，强化学习算法的 Activity 覆盖率相对更低，如图 8 所示。这是由于强化学习算法在测试过程中均需要进行行为准则的学习，这个过程需要消耗一定的时间，而结构复杂的应用程序会消耗更多的学习时间。因此，在短时间内，Sarsa 算法和 Q-Learning 算法在面对结构复杂的应用程序的表现相对较差。

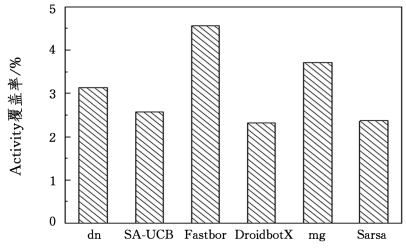


图 8　100Mb+应用程序 Activity 覆盖率

Fig. 8　100Mb+application Activity coverage

RQ2（测试效率）：图 9 给出了开源应用程序数据集运行在各测试工具上的渐进式覆盖率情况。在测试开始的前 15 min，被测应用程序的任何状态均为首次探索，因此，各个测试工具的行覆盖率均迅速增加。在测试开始的前约 28 min，Fastbot 表现出了最快的增长速度，行覆盖率也达到最高，原因在于其生成事件速度较其他测试工具更快，如表 1 所列，能够在短时间内利用 Q-Learning 快速捕捉到大量新状态。本文提出的 SA-UCB 算法的行覆盖率在约 28 min 后开始领先，这是由于经过前 30 min 行为准则学习的铺垫，SARSA 算法已经能够捕获更容易触发新状态的动作，并针对其特点大量执行，因此其行覆盖率开始稳步增长，并在测试时间到达 60 min 后，依然保持着最快的行覆盖率增长速度。

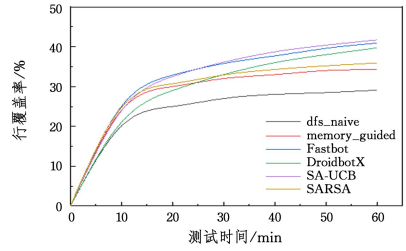


图 9　开源应用程序的行覆盖率增长趋势

Fig. 9　Line coverage growth trend of open source applications

图 10 给出了 102 个应用程序市场获取到应用程序的 Activity 覆盖率随测试时间增加的变化趋势对比。与在开源应用程序上的测试结果类似，不同点在于，在测试时间到达 60 min 后，除 dfs_naive 算法以外，其他测试策略均有明显的继续上涨的趋势，这是由于应用市场上的应用程序结构复杂，60 min 的测试时间不足以对其进行充分探索。

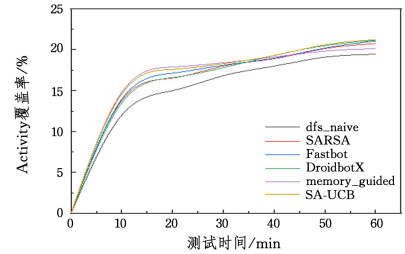


图 10　商用应用程序的 Activity 覆盖率增长趋势

Fig. 10　Activity coverage growth trend of commercial applications

在测试的开始阶段，集成了 UCB 与 Q-Learning 算法的 DroidbotX 的行覆盖率增长速度相对较慢。另外，在实验中发现，DroidbotX 工具平均每小时能够生成 159 个输入事件，而本文提出的测试方法平均每小时能够生成 212 个输入事件，生成事件的速度提高了 33.33%，且达到同一测试覆盖率的平均时间消耗减少了约 12%。因此与 Q-Learning 算法相比，SARSA 在测试的过程中表现出了更快的收敛特性，测试效率更高。

在测试时间到达 30 min 左右时，SARSA 算法的测试覆盖率虽继续增长，但增长速度与其他强化学习测试方法相比最为缓慢。这是由于 SARSA 算法集成的 ε-greedy 策略虽然在理论上能够尽可能保证对所有动作都能有至少一次访问，但这种贪心策略在选择上过于随机，导致测试过程中因初始化而没能选择到全局最优的动作，因此最终测试覆盖率相对较低。

此外，与其他策略相比，DroidbotX 在测试时间到达 60 min时的 Activity 覆盖率增长曲线有着最大的斜率，这表示在本次对比实验结束时，DroidbotX 工具的 Activity 覆盖率有着最快的增长速度，因此 DroidbotX 可能会在未来更长的测试时间达到最高的 Activity 覆盖率。

RQ3(故障检测)：一般地，自动化测试框架的故障检测性能可以从两个方面来衡量：1)运行中应用程序 BUG(闪退、黑屏等)检测能力；2)运行过程中，应用程序抛出异常数量。因测试时间定义为 1 h，本次实验过程中未能发现应用程序触发到 BUG。因此，本文将从各测试方法在运行过程中抛出的异常数量作为测试方法故障检测能力的评判标准。

在测试开始前，每个应用程序的故障数量未知。首先对各测试工具测试 1 h 产生的日志信息进行处理分析，其中，Fastbot 的测试记录从模拟器的日志文件中获取，其余工具的记录信息从输出文件 logcat. txt 中获取。使用 pycharm 工具对获取到的日志文件进行分析处理，提取出测试过程中产生的错误信息，最终结果如表 1 所列。

表 1 运行中各测试方法抛出异常数量

Table 1 Number of faults found by each test method during operation

故障名	SA-UCB	dfs_naive	Memory_guided	DroidbotX	Fastbot	Sarsa
NullpointerException	935	294	63	819	642	336
RuntimeException	5	8	5	5	6	5
ActivityNotFoundException	18	54	66	21	37	46
InterruptedException	3	0	0	3	0	0
UncaughtException	3	2	2	14	5	10
IllegalStateException	6	35	24	98	86	62
IllegalArgumentException	27	20	0	23	37	26
ClassNotFoundException	15	90	72	21	69	36
NoSuchMethodException	3	39	35	50	68	43
CertPathValidatorException	2	0	0	2	0	2
ClasscastException	5	0	0	0	0	0
其他	378	216	310	284	361	247
共计	1 400	758	577	1 340	1 311	813

通过观察检测到的错误信息发现，与其他 5 种工具(方法)相比，本文提出测试方法检测出的故障数量最多，且检测到一类独特的崩溃 ClasscastException，并触发 5 次。值得注意的是，不同的测试工具在对相同的被测应用程序进行测试时，尽管检测出的崩溃数量相同，但具体触发的故障可能并不相同。例如对于 air. com. syriastocks. SyrianExchangePrices. apk，Fastbot，Droidbot，DroidbotX 以及 SA-UCB 均检测到 3 个故障，然而各工具触发的故障均不同。

4.3 结果评估

4.3.1 有效性威胁

选择偏见。实验中所选取的被测应用程序会影响到测试结果的代表性。为了尽量缓解这一威胁，本文从不同的来源对实验对象进行选择，并将其分为开源应用程序和商用应用程序两组。所选取出的应用程序具有以下特点：1)大小不同；2)下载量大；3)种类繁多，避免了数据集选取造成的实验误差。

随机性。本文所提出的测试方法带有一定随机性，导致测试同一个被测应用程序可能会得到不同的测试结果。因此，为了避免实验的偶然性，对于每个被测应用程序，对比实验均重复 3 次，并取其平均值作为最终的实验结果。

4.3.2 局限性和未来工作

观察表 1 可以发现，本文提出的 SA-UCB 算法在故障检测方面，仅在 NullPointerException 以及 ClassCastException 两种异常上的表现较优。此外，本文提出的方法没能检测出应用程序在测试过程中出现的 Bug。这是由于，在设置实验参数时，本文把发现新状态设置为每个 episode 的目标，目的在于在更短的时间内检测到更多的被测应用程序状态。因此在未来工作中，将尝试将发现故障作为目标来设置实验，并增加实验数据量，进一步评估 SA-UCB 在故障检测等方面的表现。

结束语 本文介绍了一种基于模型的 Android 应用程序自动化测试方法 SA-UCB，用于改进传统强化学习算法测试效率偏低的问题。本文分别将其应用到开源应用程序和应用市场应用程序两个数据集上，通过多组实验数据验证了本文提出测试方法的有效性。然而，观察数据集中的应用程序可以发现，55.29%的应用程序都需要经过账号登录的操作，才能探索到应用程序的特定功能，导致其测试覆盖率相对较低。因此在未来工作中，将使用 Xposed 技术绕过登陆界面，直接对应用程序登录后的功能进行探索，从而进一步提高测试覆盖率。

参 考 文 献

[1] China Internet Network Information Center. The 49th China Statistical Report on Internet Development[R]. Beijing: China Internet Network Information Center, 2022.

[2] LIU I J, JAIN U, YEH R A, et al. Cooperative exploration for multi-agent deep reinforcement learning[C]// International Conference on Machine Learning. PMLR, 2021: 6826-6836.

[3] BELHADI A, DJENOURI Y, SRIVASTAVA G, et al. Rein-

forcement learning multi-agent system for faults diagnosis of mircoservices in industrial settings[J]. Computer Communications，2021，177：213-219.

[4] IBARZ J，TAN J，FINN C，et al. How to train your robot with deep reinforcement learning：lessons we have learned[J]. The International Journal of Robotics Research，2021，40(4/5)：698-721.

[5] ZHU K，ZHANG T. Deep reinforcement learning based mobile robot navigation：A review[J]. Tsinghua Science and Technology，2021，26(5)：674-691.

[6] ZHANG Z，LINIGER A，DAI D，et al. End-to-end urban driving by imitating a reinforcement learning coach[C]// Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021：15222-15232.

[7] AMALFITANO D，FASOLINO A R，TRAMONTANA P，et al. Using gui ripping for automated testing of android applications[C]// 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. IEEE，2012：258-261.

[8] AMALFITANO D，FASOLINO A R，TRAMONTANA P，et al. Mobiguitar：Automated model-based testing of mobile apps[J]. IEEE Software，2014，32(5)：53-59.

[9] YANG W，PRASAD M R，XIE T. A grey-box approach for automated gui-model generation of mobile applications[C]// International Conference on Fundamental Approaches to Software Engineering. Springer，2013：250-265.

[10] BAEK Y M，BAE D H. Automated model-based android gui testing using multi-level gui comparison criteriaC]// Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. 2016：238-249.

[11] SU T，MENG G，CHEN Y，et al. Guided，stochastic model-based gui testing of android apps[C]// Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017：245-256.

[12] LI Y，YANG Z，GUO Y，et al. Droidbot：a lightweight ui-guided test input generator for android[C]// 2017 IEEE/ACM 39th International Conference on Software Engineering Companion(ICSEC). IEEE，2017：23-26.

[13] CAI T，ZHANG Z，YANG P. Fastbot：A multi-agent model-based test generation system[C]// Proceedings of the IEEE/ACM 1st International Conference on Automation of Software Test. 2020：93-96.

[14] LV Z，PENG C，ZHANG Z，et al. Fastbot2：Reusable Automated Model-based GUI Testing for Android Enhanced by Reinforcement Learning[C]// 37th IEEE/ACM International Conference on Automated Software Engineering. 2022：1-5.

[15] ADAMO D，KHAN M K，KOPPULA S，et al. Reinforcement learning for android gui testing[C]// Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design，Selection，and Evaluation. 2018：2-8.

[16] ZHOU Z. Machine Learning[M]. Tsinghua University Press，2016.

[17] MARIANI L，PEZZE M，RIGANELLI O. Autoblacktest：Automatic black-box testing of interactive applications[C]// 2012 IEEE Fifth International Conference on Software Testing，Verification and Validation. IEEE，2012：81-90.

[18] KOROGLU Y，SEN A，MUSLU O. Qbe：Qlearning-based exploration of android applications[C]// 2018 IEEE 11th International Conference on Software Testing，Verification and Validation(ICST). IEEE，2018：105-115.

[19] YASIN H N，HAMID S H A，RAJA YUSOF R J. Droidbotx：Test case generation tool for android applications using q-learning[J]. Symmetry，2021，13(2)：310.

[20] LI Y，YANG Z，GUO Y，et al. A deep learning based approach to automated android app testing[J]. arXiv：1901. 02633，2019.

[21] VUONG T A T，TAKADA S. Semantic analysis for deep q-network in android gui testing[C]// SEKE. 2019：123-170.

[22] SUTTON R S，BARTO A G. Reinforcement learning：An introduction[M]. MIT Press，2018.

[23] WANG Y H，LI T H S，LIN C J. Backward q-learning：The combination of sarsa algorithm and q-learning[J]. Engineering Applications of Artificial Intelligence，2013，26(9)：2184-2193.

[24] GU T，CAO C，LIU T. Aimdroid：Activity-insulated multi-level automated testing for android applications[C]// 2017 IEEE International Conference on Software Maintenance and Evolution(ICSME). IEEE，2017：103-114.

[25] KHAN M K，BRYCE R. Android gui test generation with sarsa[C]// 2022 IEEE 12th Annual Computing and Communication Workshop and Conference(CCWC). IEEE，2022：487-493.

[26] CARINO S，ANDREWS J H. Dynamically testing GUIs using ant colony optimization(T)[C]// 2015 30th IEEE/ACM International Conference on Automated Software Engineering(ASE). IEEE，2015：138-148.

[27] ABUL O，POLAT F，ALHAJJ R. Multiagent reinforcement learning using function approximation[J]. IEEE Transactions on Systems，Man，and Cybernetics，Part C(Applications and Reviews)，2000，30(4)：485-497，56.

[28] CHOUDHARY S R，GORLA A，ORSO A. Automated test input generation for android：Are we there yet? [C]// 2015 30th IEEE/ACM International Conference on Automated Software Engineering(ASE). IEEE，2015：429-440.

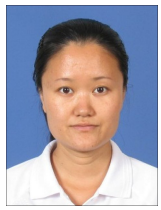
[29] LAM W，WU Z，LI D. Record and replay for android：Are we there yet in industrial cases? [C]// Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017：854-859.

[30] APKFab. [OL]. https://apkfab. com/.

[31] Tencent Software Center[OL]. https://pc. qq. com/.

[32] Google. Ui/application exerciser monkey[EB/OL]. [2022-09-26]. https://developer. android. com/studio/test/monkey. html.

WANG Xi，born in 1998，postgraduate. Her main research interests include Android automated testing and so on.



ZHAO Chunlei，born in 1979，Ph.D，associate professor，is a member of China Computer Federation. Her main research interests include cybersecurity and so on.