计算机科学
COMPUTER SCIENCE

基于汤普森采样的自适应安卓程序测试方法

赵英男, 冷重阳, 韩启龙, 俞程

引用本文

赵英男, 冷重阳, 韩启龙, 俞程. 基于汤普森采样的自适应安卓程序测试方法[J]. 计算机科学, 2025, 52(11): 330-338.

ZHAO Yingnan, LENG Chongyang, HAN Qilong, YU Cheng. Adaptive Android Program Test Method Based on Thompson Sampling [J]. Computer Science, 2025, 52(11): 330-338.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240900150

基于汤普森采样的自适应安卓程序测试方法

赵英男　冷重阳　韩启龙　俞　程
哈尔滨工程大学计算机科学与技术学院　哈尔滨 150001
(zhaoyingnan@hrbeu.edu.cn)

摘　要　近年来,安卓图形化界面测试方法的研究引起了学者的广泛关注。目前,大多数测试方法是基于强化学习开发的,然而,现有方法根据经验选择参数实现对应用程序的探索,无法根据界面变化情况自适应地改变参数设置。为此,提出了一种基于汤普森采样的自适应安卓测试方法,该方法将汤普森采样与 Q-learning 算法相结合,能够根据当前界面控件被探索的情况自适应地调整智能体下一步的探索动作,且能较好地平衡利用与探索的关系,实现更有效的测试。首先,对探索过程中界面的跳转这一事件进行 *Beta* 概率分布建模,得到一个概率分布矩阵,该矩阵与 Q 矩阵加权平均,可以兼顾事件的探索价值与利用价值。同时,对当前界面下的可操作事件的概率分布进行采样,最大采样值即为探索概率值,结合加权后的矩阵可以更全面地指导测试,以此实现对安卓应用界面的自适应探索。在 13 个安卓应用程序上进行了实验,通过与传统强化学习测试工具进行实验对比与分析,验证了所提方法的有效性。

关键词:安卓 GUI 测试;强化学习;Q-learning;汤普森采样
中图分类号　TP311

Adaptive Android Program Test Method Based on Thompson Sampling

ZHAO Yingnan, LENG Chongyang, HAN Qilong and YU Cheng
College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

Abstract　Recently, the research of Android graphical interface test method has attracted wide attention. At present, most test methods are developed based on reinforcement learning. However, the existing methods can explore the application by selecting parameters according to experience, and can not to change the parameter settings according to the interface changes adaptively. This paper proposes an adaptive Android testing method based on Thompson sampling, which combines Thompson sampling with the Q-learning algorithm, enabling the agent to adaptively determine the next exploration action based on the current state of the interface controls being examined so that to balance exploitation and exploration more effectively for superior testing outcomes. Firstly, events of the jump at the interface during exploration are modeled for the *Beta* probability distribution, then a probability distribution matrix is obtained, which is weighted-averaged with the Q-table. It can take into account both the exploration value and the exploition value of the events. At the same time, the probability distribution of operational events under the current interface is sampled, and the maximum sampling value is the exploration probability value, combined with the weighted matrix, it can guide the test more comprehensively, so as to realize the adaptive exploration of the Android application interface. Experiments executed on 13 Android applications confirm the efficacy of the proposed method through experimental comparison and analysis with conventional reinforcement learning testing tools.

Keywords　Android GUI testing, Reinforcement learning, Q-learning, Thompson sampling

1　引言

移动设备和应用程序的普及对人们的日常生活产生了深远的影响,也改变了人们的生活方式。随着智能手机和平板电脑的普及,安卓应用呈现出丰富多彩的态势。从社交媒体、游戏到工具类应用,再到商务和教育工具,安卓应用几乎涵盖了生活的方方面面。开发者不断推陈出新,以满足用户不断增长的需求和期待。与此同时,还需要确保开发的应用在各种不同的设备上能够正常运行,并且具有良好的用户体验。统计数据显示,平均每个用户每天花在移动应用上的时间超过 2h。用户对移动应用功能的多样化的需求越来越大,保证应用的使用质量变成了一个挑战。对于功能逐渐复杂的应用程序,仅靠人工手动测试是远远不够的,这会耗费大量的时间和人力成本。基于此,自动化测试应运而生。

到稿日期:2024-09-25　返修日期:2024-12-09
基金项目:黑龙江省重点研发项目(KY10600230082)
This work was supported by the Key Research and Development Program of Heilongjiang Province(KY10600230082).
通信作者:韩启龙(hanqilong@hrbeu.edu.cn)

目前常见的GUI测试主要分为随机测试、基于模型的测试和基于机器学习的测试3种。Monkey[1]——一个常用的随机测试工具，可以模拟用户的随机操作行为，是这种策略的典型例子。这种策略缺点明显，如操作冗余，无法探索复杂功能等。基于模型的测试[2]，使用基于图的模型来表示用户与应用程序界面的交互，对静态文件手动或自动地构造模型。但是这种建模方式通常是不完整或不准确的，测试时可能会出现状态爆炸的情况，或无法完整地将被测应用状态描述出来。强化学习是一种半监督机器学习，目前已被广泛应用于安卓图形化界面测试[3-5]。强化学习不直接从标记示例或数据集学习，而是让智能体通过不断在环境中试错以及累积奖励来学习。目前，经典的强化学习——Q-learning算法已被广泛应用于自动化测试，而动作探索策略是其重要组成部分，通常采用ε-greedy策略。该策略在每一步利用概率ε来选择随机动作，其中ε会随着时间不断衰减或被设置为常数，用于调控探索概率。这种强化学习算法易于实现和理解，但可能无法在探索和利用之间达到平衡，在探索率过高或过低的情况下都可能导致效果下降。

本文提出的测试方法（Adaptive Thompson Sampling，ATS）被用来评估安卓应用的实用性。该方法将“安卓界面控件触发新状态”这一事件用*Beta*概率分布进行建模，然后构建一个状态-事件概率分布矩阵$\boldsymbol{A}$，该矩阵与$\boldsymbol{Q}$表加权平均，形成新的矩阵$\boldsymbol{M}$。由于$\boldsymbol{Q}$值更多地是体现界面的利用价值，而$\boldsymbol{A}$值体现的是界面的探索价值，因此$\boldsymbol{M}$值可以兼顾动作的利用价值和探索价值，更准确地指导探索方向。同时，对安卓界面中可操作事件的概率分布进行采样，采样结果是事件被选择的概率，即其探索概率。在进行动作决策时，若$\boldsymbol{M}$的最大值与最大采样值对应的事件一致，则选择该事件，否则将按照“1－最大采样值”的概率在$\boldsymbol{M}$表中随机选择一个事件。由于整个采样具有随机性且探索概率是动态变化的，本文方法可以自适应地平衡探索与利用。通过对比先进的测试工具，评估了ATS方法的性能。将ATS与Monkey，Sapienz[6]，Stoat[7]，Droidbot[8]，DroidbotX[9]，Delm[10]在13个安卓应用上进行比较，分析了它们的指令覆盖率、方法覆盖率、活动覆盖率和崩溃检测性能。实验结果显示，ATS实现了更高的覆盖率，并检测到比其他工具更多的崩溃。本工作的贡献可以总结如下：

1）将汤普森采样技术融入安卓图形化界面测试算法中，作为动作决策部分，由此实现了一个自适应的自动化黑盒测试工具；

2）将安卓界面“可操作控件触发新状态”这一事件概率用*Beta*分布建模，实时反映每个可操作控件的探索价值，更好地为测试工具指导探索方向；

3）在安卓应用程序上进行对比实验，通过与传统强化学习测试工具的实验分析，验证了本文方法的有效性。

本文第2章阐述了相关工作；第3章介绍了基础知识；第4章提出了新的方法；第5章呈现了实验结果，将所提方法的准确性与其他自动化测试方法进行了比较；最后讨论了相关的最新研究并进行了总结。

2 相关工作

开发安卓GUI测试工具的目的在于确保安卓应用程序的质量和稳定性，帮助开发人员自动化测试应用的用户界面，从而有效地发现潜在的问题和错误。这些工具可以模拟用户的操作行为，检查应用在不同情况下的响应情况。表1列出了10种安卓应用界面自动化测试方法，其中Humanoid[11]方法采用先进的机器学习技术，尤其是深度学习，基于RICO数据集，利用神经网络模型模拟出更贴近人类实际操作的测试序列。该方法通过理解用户交互痕迹，对众多输入事件进行排序，优先执行与人类最接近的操作序列，从而提升测试的效率和覆盖率。但为了更精确，其可能需要对模型进行大量训练，会消耗一定的时间和资源。Q-testing[12]方法引入了随时间变化的奖励，使用孪生网络来判断界面是否与已经访问过的界面相似，当没有访问过类似的界面时，其会给出更高的奖励。然而，当奖励随时间变化时，会出现方法无法收敛的情况。DroidBotX方法利用上置信界策略平衡测试过程的探索与利用关系，每次决策时都要计算上置信界限，涉及大空间的情况，计算复杂度较高，可能会导致效率低下。由此可见，基于强化学习的自动化测试方法研究还有不足之处。

表1 自动化测试方法

Table 1 Automatic test methods

方法名称	分类	核心思想
Monkey	基于随机的方法	向用户发送随机输入，模拟不规则用户行为，从而测试稳定性
Dynodroid	基于随机的方法	主要通过模拟各种随机用户操作、输入和事件流发现问题，尤其在用户界面交互方面
Intent Fuzzer	基于随机的方法	随机生成多种不同类型的Intent并发送到安卓应用的各个组件中
Stoat	基于模型的方法	采用随机有限状态自动机模型来描述App的行为，根据事件类型和执行频率确定优先级
DroidBot	基于模型的方法	轻量级测试输入生成器，向应用程序发送事件，并生成UI转换图
Sapienz	基于模型和随机的方法	动态分析应用程序的UI结构，使用遗传算法搜索确定测试路径
Humanoid	基于强化学习的方法	采用神经网络模拟人类实际操作序列，集成Droidbot工具，对事件优先级排序，执行与人类最接近的操作序列
Q-testing	基于强化学习的方法	利用好奇心驱动策略，通过内存记录历史访问状态，并在功能场景的粒度上划分不同的状态
DroidBotX	基于强化学习的方法	基于Q-learning的测试方法，采用了UCB探索策略，以最小化事件的冗余执行
Delm	基于强化学习的方法	引入一个深度链接增强的探索方法，集成到Monkey，Delm监督动态探索过程，将工具从无意义的测试循环引导到未探索的GUI页面

表2从3个方面介绍了传统强化学习算法的不足之处。基于价值的算法[13]意味着对动作价值函数$Q^{\pi}(s,a)$的优化，最优策略选取函数$Q^{\pi}(s,a)$最大值所对应的动作，即$\pi^{*}\approx\arg\max Q^{\pi}(s,a)$，这里≈由函数近似误差导致。基于价值的算法

具有采样效率相对较高、值函数估计方差小、不易陷入局部最优等优点。基于策略的算法[14]直接对策略的进行优化，通过对策略的迭代更新，实现累计奖励(回报)最大化。其具有策略参数化简单、收敛速度快的优点，而且适用于连续或者高维动作空间。演员-评论家算法[15]结合了上述基于价值的方法和基于策略的方法，评论家(Critic)利用基于价值的方法学习 Q 值函数或状态价值函数 V 来提高采样效率，演员(Actor)利用基于策略的方法学习策略函数，从而适用于连续或高维动作空间，但其也存在缺点。例如，Critic 存在过估计问题，而 Actor 存在探索不足的问题等。

表 2 传统强化学习算法

Table 2 Traditional algorithms of reinforcement learning

算法类别	代表性算法	算法机制	算法不足
Value-based	Q-learning, SARSA, DQN	计算价值函数，选取最大价值函数对应的动作，隐式获得确定性策略	容易出现过拟合，处理问题复杂度受限，收敛性较差
Policy-based	TRPO, PPO	不依赖价值函数，最大化累积回报选择动作、更新策略。通常获得最优随机策略	样本利用率低。容易陷入局部最优，策略函数易于计算，自带随机离散/连续，评估策略通常效率低、方差大
Actor-critic	A3C, DDPG, SAC, TD3	Actor (Policy-based)根据价值函数更新策略，选取动作；Critic(Value-based)根据动作计算价值函数，单步更新	算法稳定性不足，对超参数敏感

在强化学习[16]中，自适应平衡探索和利用是一个挑战，传统算法通常难以自适应找到合适的平衡点。A3C 和 TD3 这类技术，通过异步或确定性策略更新，尝试解决这个问题，但实现最佳性能仍然是一个挑战。在 DDPG[17]和 SAC[18]等离线算法中，用于训练的样本可能高度相关，导致探索效率低，策略更新次优。在具有非平稳动态或时变奖励结构的环境中，这个问题可能会变得明显。许多强化学习算法对超参数的选择敏感，难以通过调整和优化来获得最佳性能。这种敏感性可能导致次优结果，或者需要进行大量手动调整，特别是在具有非线性动态的复杂环境中。随着安卓应用数量的不断增长和功能的不断丰富，用户对应用的期望也在不断提高。因此，需要确保应用在各种不同的设备上能够正常运行，保持良好的用户体验。

3 基础知识

本文方法主要基于 Q-learning 强化学习算法和汤普森采样技术，下面就相关概念和基本知识予以介绍。

3.1 安卓测试机制

安卓 GUI 测试的作用是确保安卓应用程序的图形用户界面功能的可靠性和可用性。GUI 测试通过多种工具协同进行，而 UIAutomator2 是一种应用广泛的开源自动化测试工具，可以提供一系列的 Python API，方便测试人员用 Python 编写自动化测试脚本，其底层是 UIAutomator 库，在原基础上增加了对 Python 语言的支持功能。该测试框架非常适合编写黑盒式自动化测试，因为此类测试不依赖于目标应用的内部实现细节，可以获取屏幕上任意一个 App 的任意一个控件属性，并对其进行任意操作。

安卓图形化界面测试工作主要分为 3 个部分：Python 客户端、移动设备和 UIAutomator2 工具。在 Python 端运行脚本，向安卓设备发送 HTTP 请求，在移动应用上安装 Automator Test Frameworks，由此启动 UIAutomator2 进行监听，解析接收到的请求，根据元素的 text, hint, contentDescription 等属性进行查找，就可以查找到对应的控件元素。ATS 使用安卓调试桥(Android Debug Bridge, adb)，adb 是一种功能多样的命令行工具，可使脚本与设备进行通信，用于执行各种设备操作，例如安装和调试应用。同时，adb 提供对 Unix shell(可用来在设备上运行各种命令)的访问权限。adb 是一种客户端-服务器程序，当启动某个 adb 客户端时，该客户端会先检查是否有 adb 服务器进程已在运行。如果没有，adb 会启动服务器进程，服务器在启动后会与本地 TCP 端口 5037 绑定，并监听 adb 客户端发出的命令。服务器与所有设备均建立连接后，便可以使用 adb 命令访问这些设备。本文利用 UIAutomator2 来提取 GUI 层次结构，转储当前界面中包含的小部件的信息，将接收到的 XML 格式的屏幕数据分离成一个 UI 元素数组，并将 UI 信息发送给智能体。本文借助 Android SDK 自带的 UIAutomatorViewer 工具获取应用的界面截图并进行分析，提供了一个便利的方式来查看 UI 布局结构，并且可以查看各个控件的相关属性。此外，可以利用这些信息来创建 UI 测试代码。

3.2 强化学习

强化学习属于机器学习的一种，智能体通过与环境不断交互(即在给定状态采取动作)来获得奖励，此时环境从一个状态转移到下一个状态[19]。智能体通过不断优化自身动作策略，期待最大化其长期回报或收益(奖励之和)。环境指一个外部系统，智能体感知环境并执行动作，而智能体则表示强化学习算法，又通过行动改变环境[20]。

图 1 中，环境首先向智能体呈现一个状态，然后智能体采取动作来响应该状态。之后，环境转移到下一个状态，并把奖励返回给智能体。智能体用环境所返回的奖励来进行更新，对上一个动作进行评估。这个循环一直持续，直到环境发送终止状态来结束这个事件。一直以来，Q-learning[21]是比较经典的智能体，不仅可以从过去的经验中学习，还能在有限的状态中收敛到最优策略，且方法易于实现。

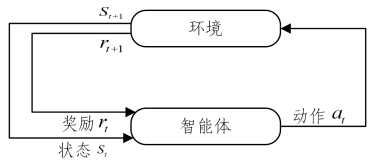


图 1 强化学习的工作流程

Fig. 1 Workflow of reinforcement learning

Q-learning 的核心思想是利用贝尔曼最优方程来更新动作价值函数 $Q(s,a)$。

式(1)描述了最优策略下的动作价值函数与下一状态的动作价值函数之间的关系。

$$Q_{t+1}(s,a)=Q_t(s,a)+\alpha\cdot(R(s,a)+\gamma\cdot\max_{a'}Q_t(s',a')-Q_t(s,a)) \tag{1}$$

其中，α 是学习率，$R(s,a)$表示在状态下采取动作 a 获得的奖励，γ 是折扣因子。式(1)是基于贝尔曼最优方程的形式构建的，并不是直接由其推导的。Q-learning 通过迭代更新 Q 值来逼近最优动作价值函数。在工程应用中，平衡被测试应用的探索价值与利用价值一直是测试人员所研究的问题，目前应用较多的有 ε-greedy 和上置信界算法。

ε-greedy 探索策略在应用时先设置一个 ε 值，用于指导究竟是探索还是利用，如将 ε 设置为 0.1，就表明 10%的概率探索，90%的概率利用。具体操作时，每次生成一个 0-1 的随机数，如果这个数大于 ε，则选择当前认为最好的行为；如果小于 ε，则在所有可能的行为中随机选择。其数学表达式如下：

$$\pi(a|s)=\begin{cases}\dfrac{\varepsilon}{m}+1-\varepsilon, & \text{if } a^*=\arg\max\limits_{a\in\mathcal{A}}Q(s,a)\\ \dfrac{\varepsilon}{m}, & \text{otherwise}\end{cases} \tag{2}$$

其中，m 代表全部可选的行为的数量，$\frac{\varepsilon}{m}$代表以 ε 的概率在 m 个可选行为中选中行为 a 的概率。

最好的行为a^*在两种情况下可能被选到：1)1−ε 的概率下选择当前认为最好的行为；2)随机选择时也恰好选到最好的行为。贪婪策略是选择目前为止的估计结果中最优的行动，但是其他行动有可能更好，只不过探索得不够，尚未发现而已。

上置信界算法探索策略是一种基于置信区间 Q 的方法，根据每个动作的期望奖励和不确定性来计算一个上界，使用置信上限作为行动选择的依据，最终选择具有较高置信上限的行动。上置信界算法行动选择的数学表达式如下：

$$A_t=\arg\max\left[Q_t(a)+c\sqrt{\frac{\ln(t)}{N_t(a)}}\right] \tag{3}$$

其中，$Q_t(a)$表示到目前时刻 t 为止的行动价值估计值，代表成为最优行动的可能性的现实部分；c 表示加权系数，用于控制倾向于探索的程度；$N_t(a)$统计到目前时刻 t 为止，行动 a 被选择的次数；$\sqrt{\frac{\ln(t)}{N_t(a)}}$代表行动 a 的不确定性，或者表示行动 a 的价值估计的方差，或者是置信度(的倒数)。一个行动被选择的次数越多，意味着其行动价值估计的确定性就越小(即成反比关系)，因此$N_t(a)$出现在分母中。如果某个行动被选择次数是 0，就意味着该行动在探索中应该以最高优先度被选择。虽然 UCB 算法在一些方面比贪婪算法表现得好，但是它也是一种确定性算法，得到的奖励相同时，做出的决策也是确定的，而做出的决策只与置信区间的上界有关，上界只与观察值有关。因此当观察值相同时，便会一直做出相同的决策。

3.3 汤普森采样

汤普森采样[22]是一种基于贝叶斯推断的方法，先将事件建模为 *Beta* 概率分布，再根据每个动作的先验分布和观测数据更新后验分布来完成。在每次选择事件时，按照后验概率分布进行采样，每个动作对应的概率就是目前最优的，这一结论是基于到目前为止已经观测到的事实。汤普森采样的核心思想就是通过不断地“试错”来学习，每次选择具有最高探索价值的动作，但是由于抽样是随机的，有可能选择已知较好的动作，也有可能尝试新的动作，在整个过程中无需根据经验设置探索参数，因此通过汤普森采样方法可以自适应指导智能体找到全局最优的动作。实际应用中，若每个时间步抽取一个事件，可能会出现下面 3 种情况。

1)如果一个事件被选中的次数很多，根据 *Beta* 分布特性，其分布形状会很窄，换句话说，这个事件的收益已经非常确定了。按照该分布产生随机数，基本上就在中心位置附近，接近平均收益。

2)如果一个事件的均值很大，接近 1，就确定这是个有价值的事件，平均收益很高，这样就会倾向于利用该事件。反之，平均分布有可能比较接近于 0，那么事件就会具有较高的探索价值。

3)如果一个事件没有被选择太多次，说明其探索与利用倾向不太确定，它的分布就是跳跃的，探索价值没有完全固定，其概率分布的采样会以近似均匀的概率落在整个区间上。

经过上述分析后，根据选择事件后获得的奖励来更新现有的概率分布。通过上述步骤，算法可以自适应地在探索和利用之间实现平衡，并且在实践中表现良好。在强化学习中，汤普森采样可以应用于值迭代、策略迭代等步骤中，通过更新和维护每个状态-动作对的概率模型，其能够根据环境反馈有效地改进探索策略。

4 基于汤普林采样的自适应覆盖探索方法

强化学习适用于解决序贯决策问题，而安卓应用程序界面测试过程可以被建模为序列决策问题，用马尔可夫决策过程(Markov Decision Process，MDP)描述。本章提出基于汤普森采样的自适应探索方法(ATS)来解决安卓 GUI 测试问题。首先，通过使用 UIAutomator2 来描述屏幕的状态，内容不同的界面显示即为不同状态[23]。其次，本文算法可以通过建模自适应生成奖励，使其根据当前搜索状态发生变化[18]。最后，迭代更新，确保函数在学习过程中是动态变化且收敛的。

4.1 问题建模

安卓程序测试可以用马尔可夫过程表示。MDP 表示一个四元组$\langle S,A,P,R\rangle$，关注状态定义、奖励定义和智能体学习过程。其中，S 表示状态，A 表示动作，P 表示转移概率，R 表示奖励。

1)S：为了正确地区分应用程序上不同的 GUI，对每个界面进行基于内容的对比，以此来定义状态。若内容存在差异，则被视为不同的界面，该定义用于更新 Q 表状态。

2)A：本文将安卓应用程序的事件定义为 MDP 的动作。*ATS* 通过分析转储的小部件层次结构和相应的属性(如可点击、可滚动)来推断当前状态下的可执行事件。由于每个事件都与特定的状态相关联，因此也可以使用状态-操作对来表示应用程序状态中的事件可执行文件。为了平衡利用与探索之间的关系，采用汤普森采样策略选择当前状态中的动作。

3)P[24]:状态转移概率矩阵用于描述动作执行后应用程序跳转到其他状态的概率,并且由应用程序确定。ATS使用贝尔曼函数、概率均值矩阵更新Q值,其中状态-动作对$Q(s,a)$的值受S所能到达的所有状态的影响,因此更新值时会考虑所有的转换信息。

4)R:奖励对于安卓测试至关重要,对探索策略具有决定性影响。本文提出了一种奖励函数,ATS在每次触发崩溃或跳转到新界面时,会给予智能体一个较大的奖励。这有助于探索更多界面,更全面彻底地测试整个应用程序。

4.2 方法概述

如图2所示,对GUI进行测试时,脚本与安卓设备进行交互后获取当前界面,将界面转换成脚本可执行的对象,并将其表示为当前状态,每个状态都包含多个可执行的操作事件,这些事件可以表述为"动作"。对"动作触发新状态"这一事件发生的概率进行*Beta*分布建模,*Beta*分布因在[0,1]区间内取值,所以一般被用于建模伯努利实验事件成功的概率分布。本文提出使用$\boldsymbol{A}$矩阵这一新变量来存储安卓应用所有可操作事件*Beta*分布的参数。$\boldsymbol{A}$矩阵是在智能体探索过程中动态构建并不断更新的,可以衡量所有动作的概率分布的探索价值,量化某一状态下的某一动作跳转到新界面的概率。同时,利用Q-learning方法在探索应用程序过程中构建$\boldsymbol{Q}$表,本文中$\boldsymbol{Q}$函数在更新时,设置当动作探索到新界面时奖励值加1。为了更好地平衡探索与利用,将$\boldsymbol{Q}$表与$\boldsymbol{A}$矩阵进行加权平均,得到$\boldsymbol{M}$矩阵。对当前状态下的所有可操作事件的*Beta*分布进行汤普森采样,若最大采样值与最大$\boldsymbol{M}$值对应的事件一致,则选择该"最优动作";否则,按照"1-最大采样值"的概率选择其他动作来指导$\boldsymbol{M}$矩阵与$\boldsymbol{A}$矩阵更新。

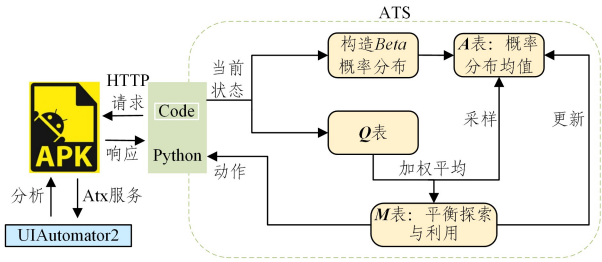


图2 ATS工作的整体流程

Fig. 2 Overall workflow of ATS

可以看出,在测试过程中,智能体在动作决策下更新$\boldsymbol{M}$矩阵并做出动作,将该动作请求反馈到安卓环境中,通过安卓设备中的atx智能体工具封装并启动UIAutomator2服务,解析指令并转换成脚本可识别代码,向智能体响应下一个状态。

4.3 *Beta*概率分布建模

在ATS探索策略中,对奖励函数的设置进行了改进。在测试过程中,当动作触发新状态时奖励值加1,当动作发现一个崩溃时奖励值加100。通常认为,$\boldsymbol{Q}$表中当前状态下最大$\boldsymbol{Q}$值对应的动作即为最优动作,但这种方法缺乏对动作探索价值的考量,因此引入了第二个奖励函数,起到加强指导探索力度的作用。区别于其他启发式奖励函数设置,通过将"事件触发新状态"概率建模为*Beta*分布,当所选择动作触发新状态时,相应的概率分布参数值加1,从而导致$\boldsymbol{A}$矩阵和$\boldsymbol{M}$矩阵数值发生变化,实现自适应探索与利用,同时也省去了繁琐的调参过程,在程序编制过程中减少了人力与物力成本,这也是本文的贡献之一。下面将详细介绍*Beta*概率分布建模。

为每个状态的每个可操作事件都构造*Beta*概率分布。*Beta*概率分布是由α和β两个参数构成的,记作$Beta(\alpha,\beta)$,服从伯努利分布,即若状态更新则结果为1,否则为0。首先,为每个事件都构造成一个先验分布$\theta\sim Beta(\alpha=1,\beta=1)$。在每次动作选择后观察是否找到新的状态,以此更新分布参数。

$$Beta(\alpha_k,\beta_k)\leftarrow Beta(\alpha_t+r_t,\beta_t+1-r_t) \tag{4}$$

$$f(\theta;\alpha,\beta)=\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1},0<\theta<1 \tag{5}$$

式(4)表示若探索到新的状态,则α参数值加1,否则β参数值加1。在实验过程中,当某个事件被选择后,AUT探索到一个新的状态时,均值由0.2857变为0.375,观察到如图3所示的概率分布曲线的变化,该状态被探索后其探索价值随之变大。式(5)表示其概率密度函数,$B(\alpha,\beta)$是一个与α,β相关的常数,用于归一化。*Beta*分布的数学期望可以表示为$\frac{\alpha}{\alpha+\beta}$,其值越大,概率密度分布的中心位置越靠近1,依据此概率分布产生的随机数也多数靠近1,反之则都靠近0。在测试过程中,每个事件对应的数学期望可以形成一个与$\boldsymbol{Q}$表结构相同的矩阵,这也为外部奖励设置提供了基础数据支持。

同时,从Q-learning方法中获得灵感,如4.2节所述,通过$\boldsymbol{Q}$表与$\boldsymbol{A}$表加权平均得到一个$\boldsymbol{M}$矩阵,矩阵中数值表示为$M(s,a)$。后续通过迭代更新$\boldsymbol{M}$函数来指导智能体选择最优动作。

$$M_{t+1}(s,a)=M_t(s,a)+\alpha\cdot(r_t+\gamma*\max_{a'}M_t(s',a')-M_t(s,a)) \tag{6}$$

其中,r_t表示动作的即时奖励。当触发新状态时,奖励值为1;触发一次崩溃时,奖励为100;否则为0。

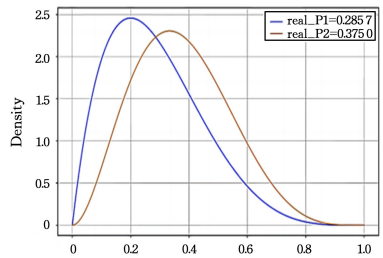


图3 事件概率分布变化

Fig. 3 Changes in the probability distribution of events

*Beta*概率分布建模这一技术,使得智能体可以更精准、更快速地完成应用程序图形化界面测试。

4.4 算法总结

ATS采用汤普森采样策略来测试AUT,可以引导测试工具探索不熟悉的状态,从而高效率地覆盖代码和发现崩溃。

ATS探索策略的具体描述如算法1所示。

算法1 ATS

输入:AUT,被测APP

输出:**S**状态;**Q**表;**A**表;**P**转移函数

1. 初始化Q(s,a)为0

2. 初始化A(s,a)为Beta(1,1)(表示状态-动作Beta概率分布参数)

3. 启动 AUT
4. 定义 getAction(s)：
5. 对当前状态 s 下的所有动作 a 进行 Beta 分布采样；
6. 找到最大采样值对应的动作 a_max
7. 计算 $\mathbf{M}=\mathbf{Q}\times\mathbf{A}$
8. 找到当前状态下 M(s,a)最大值对应的动作 a_Mmax
9. 如果 a_max==a_Mmax：
10. 返回 a_max
11. 否则：
12. 以 P 的概率选择 a_Mmax(P 为最大采样值对应的动作均值)
13. 以(1−P)的概率选择其他动作
14. 返回选择的动作
15. 重复每个回合(episode)：
16. 初始化状态 s
17. 重复每个步骤(step)：
18. 调用 getAction(s)获取动作 a
19. 执行动作 a，观察奖励 r 和下一个状态 s′
20. 更新 **M** 值与 **A** 值：
21. $M(s,a)=M(s,a)+\alpha\cdot(r_t+\gamma\cdot\max_{a'}M(s',a')-M(s,a))$
22. Beta(α+r,β+1−r)
23. 将状态更新为 s′
24. 如果 s 是终止状态：
25. 结束当前回合
26. 直到 s 是终止状态
27. 结束

首先，设定输入为被测 App，初始化 $Q(s,a)$与 $A(s,a)$，初始时将所有 $\boldsymbol{Q}$ 值设为 0，将所有状态-动作对的 *Beta* 分布参数设为 *Beta*(1,1)。第 4−14 行定义 getAction 函数，获取 AUT 当前状态的所有 GUI 操作，对每个可能的动作 a，根据其在 $\boldsymbol{A}$ 表中的 *Beta* 分布进行采样，选择具有最高 *Beta* 分布采样值的动作。第 8 行将得到的 $\boldsymbol{A}$ 表与 $\boldsymbol{Q}$ 表进行加权平均处理，得到 $\boldsymbol{M}$ 表，找到当前状态下 $M(s,a)$最大值对应的动作 a_Mmax。如果最大采样值对应的动作与最大 $\boldsymbol{M}$ 值对应的动作相同，则选择该动作；如果不相同，则以 P 的概率选择 a_Mmax，其中 P 为最大采样值对应的动作的均值；否则以(1−P)的概率选择其他动作，最终返回选择的动作。接下来是 ATS 主循环，对于每个回合的每个步骤，先调用 getAction 函数选择动作，并执行选择的动作，观察其结果，即获得的奖励和下一个状态。根据获得的奖励和下一个状态，更新 $\boldsymbol{M}$ 值和 $\boldsymbol{A}$ 值，在 $\boldsymbol{A}$ 表中存储所有事件的 $B(\alpha,\beta)$分布参数，当状态更新时 r 值为 1，即 α 值加 1，否则 β 值加 1，这一状态下该事件的概率分布形状和均值也会随之变化，从而更新 $A(s,a)$，进而影响采样结果。最终，将当前状态更新为下一个状态。由此可见，ATS 算法具有平衡安卓测试中的探索与利用的能力，并且增加了奖励函数设置，可以更好地了解智能体是否完成测试工作。

5 实验及分析

5.1 实验环境设置

在 6 个测试工具中对 13 个安卓应用程序进行对比实验，表 3 列出了应用程序名称、版本、类别、指令数量、方法数量以及活动数量。其中，指令、方法、活动数量分别是该应用程序所包含的代码、函数、界面数量。在安卓模拟器中进行评估，执行环境如下。

- 操作系统：Windows 11 x64 。
- CPU：13th Gen Intel(R) Core(TM) i7-13620H 2.40GHz 。
- 模拟器：安卓系统版本为 Android 6.0，使用 Android API 26 的安卓 SDK 进行开源应用的编译。

数据集：在 AndroTest[25] 中选取 13 个开源 AUT，AndroTest 已被许多 Android 测试工具用作基准。

为了避免测试过程中随机事件对结果产生影响，让每个测试工具对每一个安卓应用重复测试 5 次。每次测试都记录其结果，最后计算 5 次实验的平均值作为最终结果。

表 3 实验数据集

Table 3 Experimental datasets

应用程序名称	版本	分类	指令数量	方法数量	活动数量
WLAN Scanner	4	Communication	2484	141	1
MunchLife	2.3	Entertainment	551	39	2
Sensors2Pd	2.3	Tools	1346	149	4
Talalarmo	4	Productivity	5122	658	3
BatteryCircle	1.5	Tools	963	79	1
Hot Death	2.1	Game	17679	1203	3
World Clock	2.3	Productivity	5200	315	4
Mirrored	2.3	Magazines	3803	219	4
A2DP Volume	2.3	Maps and Navigation	13452	600	8
Weather notifications	2.3	Weather	8927	667	7
Amaze	3.10	Tools	597241	41475	144
Commons	5.0	Maps	571284	31124	98
K9mail	8.0	Mail	124745	6733	29

5.2 评价指标

使用 4 种指标对实验结果进行评价，分别是指令覆盖率、方法覆盖率、活动覆盖率和触发崩溃。利用 JaCoCo[26] 工具生成代码的覆盖率报告。JaCoCo 包含了多种维度的覆盖率计数器：指令级计数器、分支级计数器、圈复杂度、行覆盖、方法覆盖、类覆盖。

Huang 等[27] 首先提出指令覆盖率的概念。指令覆盖率是所有编程语言和生态系统(包括 Android)的软件开发和质量保证周期的基本要素。它是反映闭源应用测试结果充分性的更为准确有效的测试覆盖率标准[28]，也是评估测试框架有效性最常见的指标之一。

方法覆盖率是被调用的方法数量与总方法数的比值，其值与应用功能被探索数量成正比。

活动覆盖率定义为执行期间探索的活动与 App 中存在的活动总数的比率。本文探索了内容存在差异的界面，通过间歇观察 AUT 上的活动堆栈并记录安卓清单文件中列出的所有活动来测量[29]，这也是比较粗粒度的覆盖率指标。

触发崩溃是指当智能体对应用程序的测试越深入，可能触发的崩溃就会越多。用 LogCat[30] 工具进行崩溃检测查询，LogCat 是通过命令行界面转储所有系统级消息日志的工具，

通过手动分析日志报告，从错误堆栈中识别崩溃[31]。因此，触发崩溃越多，测试工具的探索越彻底，越能发现潜在的崩溃情况。

5.3 实验结果与分析

本文统计了每个应用在5次实验中的平均指令、方法、活动覆盖率和平均崩溃次数[32]。为了评估ATS方法[1)]的有效性，研究了以下两个问题，并对实验过程及结果进行阐述。

1)RQ1：与其他工具相比，覆盖率表现如何？

2)RQ2：与其他工具相比，发现崩溃的表现如何？

6个测试工具获得的结果如表4—表7所列，其中灰色单元格表示测试结果中的最大值，对应着表现最好的测试工具。表中的数值是在每个AUT上执行的测试结果取5次的平均值。

为了验证RQ1的结果，分别进行3种粒度的覆盖率对比实验。

首先是指令覆盖率，将ATS，Monkey，Sapienz，Stoat，Droidbot，DroidbotX方法进行对比，测试工具在10个选定安卓应用程序上获得的指令覆盖率结果如表4所列。ATS的平均表现较好，取得了59.3%的平均指令覆盖率；DroidbotX的表现也较为突出，仅次于本文方法，实现了58%的指令覆盖率，这是所有比较工具中最高的。ATS与其他工具相比，在10个应用程序中的7个中获得最高的数值。Sapienz的平均指令覆盖率为55.7%，其次是Monkey(55.5%)，Stoat(54.2%)和Droidbot(52.5%)。

表4 测试工具指令覆盖率比较

Table 4 Comparison of test tool instruction coverage

(%)

App名称	指令覆盖率					
	Monkey	Sapienz	Stoat	Droidbot	DroidbotX	ATS
WLAN Scanner	58.9	61.3	57.2	59.2	59.4	59.4
MunchLife	73.6	75.0	75.7	72.8	75.1	73.2
Sensors2Pd	74.1	73.6	71.3	71.1	70.4	75.6
Talalarmo	76.0	74.1	69.3	64.8	74.4	78.3
Battery Circle	77.6	78.1	78.1	74.0	81.3	80.0
Hot Death	51.4	54.2	49.6	49.1	53.7	55.4
World Clock	44.4	41.3	42.5	24.8	46.1	46.3
Mirrored	27.5	27.5	25.4	36.1	36.1	36.1
A2DP Volume	25.6	29.5	31.2	35.6	39.1	40.8
Weather notifications	45.8	42.4	41.8	37.6	44.2	48.3
平均值	55.5	55.7	54.2	52.5	58.0	59.3

其次是方法覆盖率，分别在两组被测应用程序中进行实验，增加了实验对象，使实验运行环境更加多样化。表5列出了Sapienz，Monkey，Stoat，Droidbot和DroidbotX的平均方法覆盖率分别为62.8%，63%，61%，59.3%和66.4%，ATS的平均值为66.8%。可以看出，在10个应用程序中，ATS有6个达到了最高值，在方法覆盖率方面优于最先进的工具。经过5轮测试得出结论：ATS在“A2DPVolume”“WLAN Scanner”“Talalarmo”“Weather notifications”“Hot Death”中表现最好。

表5 测试工具方法覆盖率比较(a)

Table 5 Comparison of test tool method coverage(a)

(%)

App名称	指令覆盖率					
	Monkey	Sapienz	Stoat	Droidbot	DroidbotX	ATS
WLAN Scanner	63.5	66.0	59.3	65.4	65.5	66.3
MunchLife	62.1	66.7	66.7	66.2	66.2	66.2
Sensors2Pd	86.6	81.9	83.2	81.6	80.7	81.6
Talalarmo	61.3	56.3	51.8	48.4	57.4	61.3
Battery Circle	83.5	84.1	83.5	76.5	83.5	83.5
Hot Death	71.9	72.8	63.4	66.7	72.5	73.7
World Clock	54.5	43.8	54.9	39.5	57.5	54.9
Mirrored	40.5	44.1	39.7	48.5	47.1	47.1
A2DP Volume	37.2	58.9	59.2	60.7	66.9	66.9
Weather notifications	69.3	53.6	48.3	40.2	66.6	66.6
平均值	63.0	62.8	61.0	59.3	66.4	66.8

表6列出了Monkey，Stoat，Delm和ATS的方法覆盖率，其中Han等提出的Delm使用静态分析方法，难以准确提取界面深层链接的意图信息，且由于Android反编译技术，很难建立准确模型来指导测试路径，因此测试效果不好。本文方法在探索过程中自适应地平衡探索与利用的关系，不依赖静态分析。由于Delm方法在开源应用中只进行方法覆盖率的实验，因此本文方法只在方法覆盖率方面与之进行对比。Delm在“Amaze”“Commons”两个App中表现更好，ATS在“K9mai”“Hot Death”中表现更好，但本文方法的平均方法覆盖率更高，总体表现更好。

表6 测试工具方法覆盖率比较(b)

Table 6 Comparison of test tool method coverage(b)

(%)

App名称	方法覆盖率			
	Monkey	Stoat	Delm	ATS
Amaze	14.00	12.00	14.00	13.000
Commons	12.00	13.00	15.00	13.000
K9mail	45.00	32.00	56.00	58.000
Hot Death	66.00	66.00	72.00	73.700
平均值	34.25	30.75	39.25	39.425

最后是活动覆盖率，如表7所列，ATS的活动覆盖率比指令和方法的覆盖率执行得更好。ATS在所有App中均可以达到最高覆盖率。如图4所示，在活动覆盖率均为100%情况下，与DroidbotX测试工具对比，ATS用时更短，可以在更少的时间资源内达到更好的测试效果。

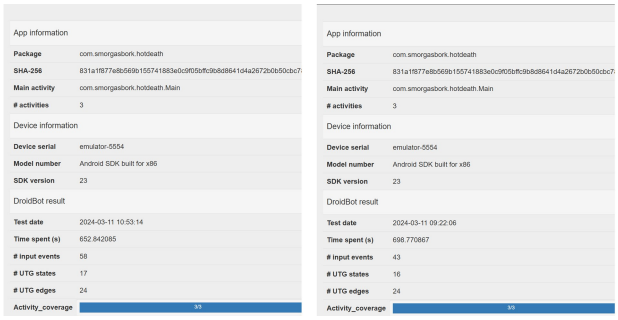
表7 测试工具活动覆盖率比较

Table 7 Comparison of test tool activity coverage

(%)

App名称	活动覆盖率					
	Monkey	Sapienz	Stoat	Droidbot	DroidbotX	ATS
WLAN Scanner	100.0	100.0	100.0	100.0	100.0	100.0
MunchLife	100.0	100.0	100.0	100.0	100.0	100.0
Sensors2Pd	100.0	100.0	100.0	100.0	100.0	100.0
Talalarmo	100.0	100.0	100.0	100.0	100.0	100.0
Battery Circle	100.0	100.0	100.0	100.0	100.0	100.0
Hot Death	73.3	100.0	100.0	100.0	100.0	100.0
World Clock	70.0	95.0	95.0	85.0	100.0	100.0
Mirrored	75.0	75.0	75.0	75.0	75.0	75.0
A2DP Volume	97.5	100.0	100.0	90.0	95.0	100.0
Weather notifications	37.1	45.7	42.9	57.1	57.1	57.1
平均值	85.3	91.6	91.3	90.7	92.7	93.2

1) https://github.com/Deserve-lcy/ATS.git



(a) DroidbotX (b) ATS

图 4 ATS 与 DroidbotX 相同时间内活动覆盖率的对比

Fig. 4 Comparison of activity coverage between ATS and DroidbotX at the same time

为了回答 RQ2，将本文方法与现有测试工具进行比较，不考虑与应用程序执行过程无关的崩溃。应用程序崩溃的次数被用来衡量测试方法的性能。通过两组被测应用程序来评估 ATS 的检测崩溃能力。如表 8 所列，在 10 个应用程序中引发独特崩溃次数依次为 Sapienz(15 次)、Stoat(13 次)、Droidbot(12 次)、DroidbotX (12 次)、Monkey(10 次)和 ATS (16 次)。如表 9 所列，在 4 个应用程序中引发独特崩溃次数依次为 Monkey(6 次)、Stoat(5 次)、Delm(10 次)和 ATS(10 次)。ATS 能识别出更多的崩溃，相对于大多数测试方法有较大优势，更具有指导意义和实践价值。

表 8 测试工具发现崩溃次数比较(a)

Table 8 Number of crashes comparison of testing tools (a)

方法名称	崩溃次数
Monkey	10
Sapienz	15
Stoat	13
Droidbot	12
DroidbotX	12
ATS	16

表 9 测试工具发现崩溃次数比较(b)

Table 9 Number of crashes comparison of testing tools (b)

方法名称	崩溃次数
Monkey	6
Stoat	5
Delm	10
ATS	10

结束语 本文提出一种基于汤普森采样的测试工具 ATS 来完成自动化 GUI 测试。该方法采用汤普森采样探索策略，利用 *Beta* 概率分布采样以及加权平均状态-事件概率分布矩阵，在安卓应用界面测试过程中可以自适应平衡探索与利用，能更好地指导智能体工作，从而提高覆盖率和崩溃检测能力。本研究还对所提出方法的有效性进行实证评估，并使用 13 个安卓应用程序与 6 个 GUI 测试方法进行比较，通过指令覆盖率、方法覆盖率、活动覆盖率和检测到的崩溃数量 4 个标准进行评估。实验结果表明，ATS 方法在覆盖率和崩溃数量检测方面优于目前先进的方法，有较好的发展前景。

在今后的研究中，将进一步结合搜索算法等技术，优化本文方法。此外，安卓设备种类繁多，硬件配置、系统版本、屏幕分辨率等各不相同，导致测试工具在不同设备上的表现可能存在差异。下一步将完善实验环境设置，采用多种测试环境进行实验，使其兼具稳定性与普适性。

参 考 文 献

[1] 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. IEEE, 2017: 23-26.

[2] DONG Z, BÖHME M, COJOCARU L, et al. Time-travel testing of android apps[C]// Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020: 481-492.

[3] 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.

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

[5] VUONG T A T, TAKADA S. A reinforcement learning based approach to automated testing of android applications[C]// Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation. 2018: 31-37.

[6] MAO K, HARMAN M, JIA Y. Sapienz: Multi-objective automated testing for android applications[C]// Proceedings of the 25th International Symposium on Software Testing and Analysis. New York: ACM, 2016: 94-105.

[7] 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. New York: ACM, 2017: 245-256.

[8] 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. IEEE, 2017: 23-26.

[9] YASIN H N, HAMID S H A, YUSOF R J R. DroidbotX: Test Case Generation Tool for Android Applications Using Q-Learning[J]. Symmetry, 2021, 13(2): 310.

[10] HANH, HAN W, RUIQI D, et al. Enhancing GUI Exploration Coverage of Android Apps with Deep Link-Integrated Monkey [J], ACM Transactions on Software Engineering and Methodology, 2024, 33(6): 163.

[11] LI Y, YANG Z, GUO Y, et al. Humanoid: A Deep Learning-Based Approach to Automated Black-box Android App Testing [C]// 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2019: 1070-1073.

[12] PAN M, HUANG A, WANG G, et al. Reinforcement learning based curiosity-driven testing of Android applications[C]// Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM, 2020: 153-164.

[13] MINDOM P S N, NIKANJAM A, KHOMH F. A comparison of reinforcement learning frameworks for software testing tasks [J]. Empirical Software Engineering, 2023, 28(5): article number 111.
[14] FUJIMOTO S, VAN HOOF H, MEGER D. Addressing Function Approximation Error in Actor-Critic Methods[J]. arXiv: 1802.09477, 2018.
[15] GRUSLYS A, AZAR M G, BELLEMARE M G, et al. The Reactor: A Sample-Efficient Actor-Critic Architecture[J]. arXiv: 1704.04651, 2017.
[16] GU S, LILLICRAP T, GHAHRAMANI Z, et al. Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic[J]. arXiv: 1611.02247, 2016.
[17] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning [J]. arXiv: 1509.02971, 2015.
[18] HAARNOJA T, ZHOU A, ABBEEL P, et al. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor[J]. arXiv: 1801.01290, 2018.
[19] PAN M, HUANG A, WANG G, et al. Reinforcement learning based curiosity-driven testing of Android applications[C]// Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2020: 153-164.
[20] MARIANI L, PEZZE M, RIGANELLI O, et al. AutoBlackTest: Automatic BlackBox Testing of Interactive Applications[C]// 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. IEEE, 2012: 81-90.
[21] WATKINS C J C H, DAYAN P. Technical Note: Q-Learning [J]. Machine Learning, 1992, 8(3/4): 279-292.
[22] THOMPSON W R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples [J]. Biometrika, 1933, 25(3/4): 285-294.
[23] FUJIMOTO S, VAN HOOF H, MEGER D. Addressing Function Approximation Error in Actor-Critic Methods[J]. arXiv: 1802.09477, 2018.
[24] MNIHV, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with Deep Reinforcement Learning[J]. arXiv: 1312.5602, 2013.
[25] 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.
[26] JaCoCo. Java code coverage library[DB/OL]. https://www.jacoco.org/jacoco/.
[27] HUANG C Y, CHIU C H, LIN C H, et al. Code Coverage Measurement for Android Dynamic Analysis Tools[C]// 2015 IEEE International Conference on Mobile Services. IEEE, 2015: 209-216.
[28] YANG S, HUANG S, HUI Z. Theoretical Analysis and Empirical Evaluation of Coverage Indictors for Closed Source APP Testing[J]. IEEE Access, 2019, 7: 162323-162332.
[29] 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. New York: ACM, 2018: 2-8.
[30] PILGUN A, GADYATSKAYA O, ZHAUNIAROVICH Y, et al. Fine-grained Code Coverage Measurement in Automated Black-box Android Testing[J]. ACM Transactions on Software Engineering and Methodology, 2020, 29(4): 1-35.
[31] SUT, 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. New York: ACM, 2017: 245-256.
[32] KOROGLUY, SEN A, MUSLU O, et al. QBE: QLearning-Based Exploration of Android Applications[C]// 2018 IEEE 11th International Conference on Software Testing, Verification and Validation. IEEE, 2018.

ZHAO Yingnan, born in 1992, Ph.D, associate professor, master supervisor, is a member of CCF (No. P2652M). His main research interests include reinforcement learning and intelligent software engineering.

HAN Qilong, born in 1974, Ph.D, professor, Ph.D supervisor. His main research interests include industrial data intelligence, big data analysis and mining, data security and privacy computing, knowledge graph theory and applications, and industrial Internet.

(责任编辑:柯颖)