**A Reinforcement Learning Based Approach**

**To Automated Testing Of Android Applications**

**초록**

최근 몇 년 동안 연구자들은 안드로이드 애플리케이션 테스트를 자동화하는 도구를 적극적으로 제안했습니다. 그러나 그들의 기술은 여전히 ​​큰 어려움에 직면 해 있습니다. 첫 번째는 높은 코드 커버리지를 달성하기가 어렵다는 것입니다. 응용 프로그램은 많은 서비스와 상태 전이의 조합을 갖기 때문에 대규모 시스템의 시나리오 테스트를 위해 모든 조합의 시나리오를 테스트하는 것이 많은 시간 소요되고 때로는 그것이 비효율적 일 수 있습니다. 둘째, 특정 기능을 특정 순서로만 실행 시킬 수 있기 때문에 광범위한 응용 프로그램 기능을 실행시켜 보기가 어렵습니다. 따라서 무작위 테스트에서는 테스트 빈도가 낮습니다. 이러한 문제에 직면하여 무작위 및 모델 기반 테스트를 활용하기 위해 Q-Learning 이라는 강화 학습 알고리즘을 적용합니다. Q-Learning 에이전트는 Android 애플리케이션과 상호 작용하여 행동 모델을 작성하고 모델을 기반으로 테스트 케이스를 생성합니다. 에이전트는 최대한 응용 프로그램의 기능을 실행 시킬 수 있는 최선의 방법으로 응용 프로그램을 탐색합니다. Q-Learning 을 통해 탐색하면 코드 적용 범위가 향상되고 랜덤 테스트 및 모델 기반 테스트와 비교하여 더 많은 애플리케이션의 오류를 탐지합니다.

키워드 : Android, 테스트 입력 생성, 강화 학습, Q-Learning

| **관련 정보** |  |
| --- | --- |
| *저자* | Thi Anh Tuyet Vuong,  Shingo Takada |
| *단위* | 게이오 대학 (Keio University) 요코하마 (Yokohama) 일본  과학 기술 대학원 일본 게이오 대학 (Keio University) 요코하마 (Yokohama) 일본 과학 기술 대학원 |
| *출처* | A-TEST '18 |
| *원래 주소* | [Https://dl.acm.org/citation.cfm?id=3278191](https://dl.acm.org/citation.cfm?id=3278191) |
| *출발지 주소* |  |
| *게시 된 시간* | 2018 년 |

**1. 소개**

최근 몇 년 동안 전세계 모바일 소프트웨어의 경제 성장으로 인해 효과적인 테스트 도구에 대한 개발자의 지속적인 요구가있었습니다. 이러한 요구에 부응하여 연구원들은 가능한 한 적은 인원으로 모바일 애플리케이션을 자동으로 테스트 할 수있는 다른 툴을 제안했습니다. 대부분의 연구자들은 안드로이드 폰이 큰 시장 점유율을 가지고 있기 때문에 안드로이드 플랫폼을 선택한다. 안드로이드 테스트 툴은 의심의 여지없이 유용하며 많은 개발자와 관련이있다. 또한 안드로이드 플랫폼의 오픈 소스 특성으로 인해 연구자가 응용 프로그램과 기본 운영 체제에 쉽게 액세스 할 수 있습니다 [11]. Android를 개발하고 Android 애플리케이션 용 테스트 도구를 개발하는 것과 같은 이유가 있습니다.

Android 애플리케이션은 `activities애플리케이션의 UI (사용자 인터페이스)를 담당하는 하나 이상의 구성 요소 로 구성됩니다 . 각 액티비티에는 버튼, 텍스트, 체크 박스 등과 같은 다양한 UI 구성 요소가 포함됩니다. Android 앱은 이벤트 기반이므로 동작은 클릭, 스크롤, 텍스트 입력과 같은 사용자 입력 이벤트 (UI 이벤트)에 대한 응답을 기반으로하며 휴대 전화와 같은 휴대 전화 운영체제의 시스템 이벤트에도 반응합니다. 통화 또는 SMS 신호 등 Android 애플리케이션의 이벤트 기반 특성으로 인해 테스트를 위해 자동으로 입력 이벤트를 생성하는 데 많은 연구가 집중되었습니다. 사용 된 기술과 관계없이 가능한 목표만큼 응용 프로그램의 기능을 표시하는 관련 입력을 생성하는 것이 목표입니다. 그러나이 분야의 연구자들은 여전히 ​​두 가지 주요 문제에 직면 해 있습니다.

* 모바일 응용 프로그램에는 대개 많은 수의 구성 요소와 실행 가능 이벤트가 있습니다. 따라서 구성 요소 및 이벤트의 가능한 모든 조합을 테스트하는 작업은 시간이 많이 걸리고 대규모 시스템으로 확장하기가 어렵습니다. 자동화 된 테스트 도구는 응용 프로그램과 관련된 이벤트 시퀀스, 즉 응용 프로그램의 기능을 테스트하는 시퀀스를 테스트해야합니다.
* 응용 프로그램의 일부 기능은 특정 사건 순서 (도달하기 어려운 기능)로만 구현할 수 있으므로 자동화 된 무작위 테스트에서이를 표시하고 테스트하기가 어렵습니다.

이 백서의 주요 공헌은 안드로이드 애플리케이션을위한 테스트 입력을 자동으로 생성하기 위한 강화 학습, 특히 Q-LEARNING을 위한 애플리케이션을 제안하는 것입니다. 강화 학습을 사용하여 시행 착오적 인 방식으로 응용 프로그램의 기능을 상호 작용하고 탐색하는 테스트 도구를 구축했습니다. 탐색 중에 이 도구는 응용 프로그램의 동작 모델을 동적으로 작성하고 응용 프로그램 기능을 나타낼 가능성이 가장 높은 이벤트 시퀀스를 따라 테스트 사례를 생성합니다. 이 기사에서는 가장 진보 된 자동화 테스트 도구와 비교 한 도구 평가도 보여줍니다.

이 논문의 구조는 다음과 같다 : 2 절에서는 안드로이드 테스트 기술과 관련된 작업을 검토하고, 3 절에서는 Q-LEARNING을 소개하고, 4 절에서는 Q-Learning 도구의 제안 된 방법과 구현 세부 사항을 설명하며, 5 절에서는 제안 된 도구에 대해 설명한다. 평가, 그리고 마지막으로 6 장은 향후 작업을위한 논문과 권고안을 요약합니다.

**2 관련 작품**

이 섹션에서는 Android 애플리케이션 용 자동 테스트 입력 생성의 기존 기법을 살펴 보겠습니다.

탐사 전략 [11]을 기반으로 테스트 입력 생성 기법을 분류 할 수있다. 첫 번째 기술은 무작위 탐색 (Random Exploration)입니다.이 도구에서는 테스트 도구가 무작위 이벤트를 생성하여 응용 프로그램에 전송합니다. 이 기술은 빠르고 간단하지만 생성 된 이벤트는 일반적으로 부적합합니다. 이것은 응용 프로그램이 각 응용 프로그램 상태의 많은 다른 유형의 이벤트에만 응답 할 수 있기 때문입니다. 또한 생성되는 이벤트가 불필요하고 도달하기 어려운 기능을 밝히기가 어렵습니다. Android Monkey [2]는 무작위 테스트 전략을 사용하는 블랙 박스 테스트 도구로 단순성으로 인해 개발자가 널리 사용합니다. 특히 응용 프로그램에 초당 수천 개의 이벤트를 보낼 수있는 이벤트 (UI 이벤트 만 해당)를 생성 할 때 특히 효과적입니다. 원숭이는 대개 테스트 도구에서 가장 높은 코드 커버리지를 얻습니다. 그러나 응용 프로그램과의 인간 상호 작용의 관점에서 볼 때 충돌이 발생하는 것은 재현하기가 어렵고 비실용적입니다. 원숭이 테스트를 사용하는 것이 기능 테스트보다 스트레스 테스트에 더 가깝다고 말하는 것이 더 좋습니다. Dynodroid [18]는 무작위 탐색 전략을 사용하는 블랙 박스 테스트 도구이기도하지만 Monkey보다 약간 개선 된 점이 있습니다. 무작위 이벤트를 선택하는 대신, 애플리케이션의 현재 컨텍스트 (BiasedRandom policy)와 가장 관련이있는 이벤트를 선택합니다. 개발자는 테스트 전에 도구에 대한 인증 정보와 같이 자동으로 생성 할 수 없는 입력을 제공합니다. Dynodroid는 또한 응용 프로그램의 청취자를 분석하여 관련 시스템 이벤트를 생성합니다. 이 카테고리에 속하는 또 다른 도구 세트는 DroidFuzzer [29], Intent Fuzzer [25], Null Intent Fuzzer [5] 등의 퍼지 테스터입니다. 이들은 주로 유효하지 않은 입력을 생성하도록 설계되었으므로 테스트중인 응용 프로그램이 충돌합니다

두 번째 기법은 테스트 도구가 테스트중인 애플리케이션의 GUI 모델을 정적 또는 동적으로 빌드 한 다음 모델을 사용하여 애플리케이션을 탐색하고 이벤트를 생성하는 모델 기반 탐색 전략입니다. A3E [10]은 각 활동을 응용의 상태로 다루는 동적 모델에 기반한 동적 인 심층 탐사를 제안한다. A3E는 각 활동 내부의 행동이 아니라 시험 활동 간 전환에 초점을 맞추기 때문에이 상태 표현을 단순화하면 불완전한 테스트가 발생합니다. PUMA [14]는 동적 분석 프레임 워크입니다. 여기에는 사용자가 처리기를 정의 할 수 있는 고급 이벤트를 표시하는 일반 UI 자동화 기능 (Monkey)이 포함됩니다. 이 핸들러는 Monkey의 탐색을 안내하고 응용 프로그램 탐지를 지정하여 동적 상태 정보를 수집하거나 응용 프로그램 실행 중 환경의 변화를 트리거합니다. AndroidRipper [3] (GUIRipper [8] 또는 MobiGUITAR [7]라고도 함)는 크롤링하여 응용 프로그램 모델을 작성합니다. 각 활동에서 사용 가능한 모든 이벤트를 현재 상태로 등록합니다. 이러한 모든 작업은 깊이 우선 검색 전략을 사용하여 체계적으로 수행됩니다. AndroidRipper는 UI 이벤트 만 생성합니다.

마지막 유형의 기술은 symblic execution 및 genetic algorithm 과 같은 알고리즘을 사용하여 탐색을 안내하는 시스템 탐색 전략입니다. 이 기법은 EvoDroid [19]와 Acteve [9]와 같은 도구에서 사용됩니다. EvoDroid는 Android 응용 프로그램의 시스템 테스트에 대한 진화적인 접근 방식을 제안합니다. 진화 테스트는 각 테스트 케이스를 개인으로 취급합니다. 코드 범위를 극대화하고 테스트 수를 줄이기 위해 일부 휴리스틱을 기반으로 한 개인 그룹을 개발하십시오. EvoDroid는 테스트에서 응용 프로그램을 정적으로 분석하여 동작 모델을 작성합니다. 이 모델은 유전자 구조가 자손에게 어떻게 퍼지게되는지를 진화 적 탐색이 결정할 수있게 해준다 [19]. 한편 액티브 (Acteve)는 설득력있는 테스트를 기반으로 테스트 입력을 체계적으로 생성합니다. 중복 이벤트 실행을 피하기 위해 이벤트 시퀀스 사이에 포함 된 개념을 제시함으로써 유형 테스트의 경로 폭발 문제를 처리합니다.

모델 기반 전략은 동일한 수의 이벤트를 생성하는 확률 적 전략보다 효과적 인 것처럼 보이지만 여전히 많은 어려움에 직면 해 있습니다. 블랙 박스 기반 테스팅은 일반적으로 테스팅 전에 애플리케이션 모델을 정적으로 구축하거나 테스팅 중에 실행중인 애플리케이션의 가능한 모든 이벤트 시퀀스를 동적으로 레코딩함으로써 수행됩니다. 그런 다음 모델의 순서를 포함하는 Test Case 를 생성합니다 [21] [30]. 따라서 생성 된 테스트 케이스의 효과는 레코드 모델의 무결성과 애플리케이션 상태의 표현 방식에 크게 좌우됩니다. 하나 개는 널리 사용되는 기술은 모델을 구축하기 위해 테스트중인 임의 이송 GUI 응용 프로그램입니다, 하지만 복잡한 GUI 를 처리 할 때이 기술은 불필요하고 제한됩니다. 실행되지 않을 수 있습니다 나라에 도달 할 몇 가지 어려운, 그 이유는 일부 주 더 쉽게 이외의 상태가 도착한다는 것입니다, 그래서 그들은 임의의 전략에 더 자주 수행됩니다. 정지 조건을 결정하는 것도 어렵습니다. 무작위 탐색을 통해 완전한 모델을 얻는 데 얼마나 많은 양의 탐사가 충분합니까?

어려운 GUI 테스트 문제에 직면 한 Mariani [20]는 접근하기 어려운 GUI 에 액세스하기 위해 미리 살펴보고 특정 작업을 수행 할 수 있는 일반적인 Java 데스크탑 소프트웨어 용 블랙 박스 GUI 테스트 도구를 제안했습니다. 그들은 특히 이러한 목표를 달성하기 위해 집중적 인 학습 기술인 Q-learning을 사용합니다. 그들의 도구 (AutoBlackTest라고 함)는 동적으로 동작 모델을 작성하고 테스트중인 소프트웨어와 상호 작용할 때 점차적으로 테스트 케이스를 생성합니다. AutoBlackTest는 테스트중인 소프트웨어의 동작 모델 표현을 다 방향 그래프로 작성합니다. node 는 상태 (GUI)를 나타내며 edge 는 실행 가능한 작업을 나타냅니다. 연결 노드는 상태 간 전환에 해당합니다. Q-LEARNING 알고리즘의 각 단계에서 동작 모델을 확장하여 응용 프로그램을 탐색하고 새 상태를 관찰하거나 새 작업을 수행합니다. 도구가 소프트웨어의 가장 관련성이 높은 기능을 테스트하는 방법을 나타내는 초기 테스트 슈트를 제공하면 성능이 향상됩니다.

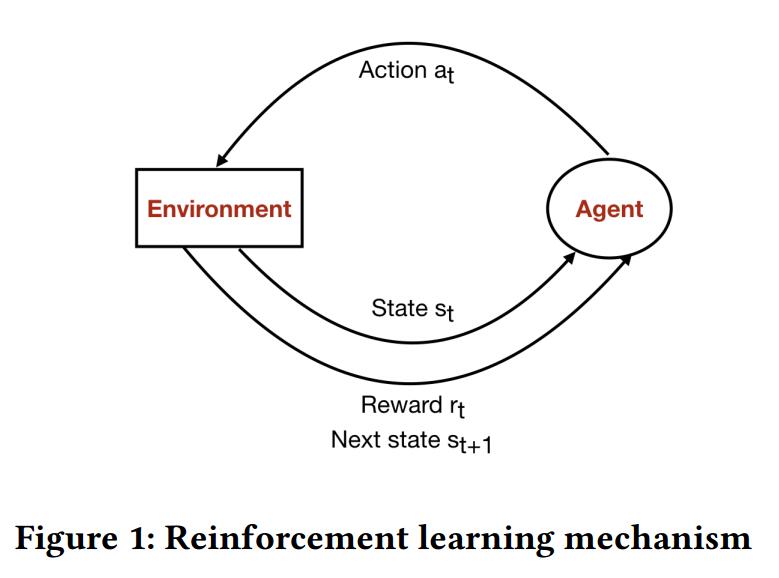
TESTAR [13]은 소스 코드에 액세스하지 않고도 GUI 정보를 기반으로 테스트 시퀀스를 생성하기 위해 Q-learning을 사용하는 테스트 도구입니다. TESTAR 결과로부터의 통계 분석은 Q-LEARNING 행동 선택 전략이 매개 변수가 적절하게 선택 될 때만 효과적이라는 것을 나타낸다. TESTAR는 일반 데스크탑 소프트웨어 및 웹 기반 소프트웨어를 테스트 할 수 있습니다. AutoBlackTest 및 TESTAR를 사용하면 강화 학습 기술이 독립형 Java 응용 프로그램의 GUI 테스트에 매우 유용함이 입증되었습니다. 그러나 Android 플랫폼과 Android 앱 자체의 특수성으로 인해 이러한 도구를 직접 사용하여 Android 앱을 테스트 할 수는 없습니다. 강화 학습은 Android 환경에 적응해야하며 Android 테스트에서 매우 유용하다는 것을 증명해야합니다. 따라서이 연구의 목적은 Android 애플리케이션 테스트에서 강화 학습을 적용하는 것입니다.

다음 섹션에서는 나머지 학습 내용을 이해하는 데 필요한 배경 지식을 제공하는 강화 학습 및 Q-Learning에 대해 설명합니다.

**3 Q- 학습**

Q-Learning [26]은 시행 착오로 환경과 상호 작용하려는 행동 심리학에서 영감을받은 잘 알려진 집중 학습 기술입니다. 각 상호 작용에서 에이전트는 작업을 수행 한 후 보상의 현재 상태 및 환경으로의 최종 전송을 기반으로 현재 상태를 기반으로 작업의 결과를 평가합니다. 전체 목표는 상담원이 누적 보상을 최대화하기 위해 최적으로 행동하는 방법을 배우게하는 것입니다. 이는 누적 보상의 전체 순서가 수행 될 때 수집되는 보상입니다. 학습 문제 강화를위한 수학 공식은 Markov 결정 프로세스입니다. 이것은 [26]에 의해 정의됩니다.

* S : 가능한 상태의 집합
* A : 가능한 조치 모음
* R : 주어진 보상의 분배 (주, 행동)
* P : 전이 확률, 즉 주어진 다음 상태의 분포 (상태, 동작)
* γ : 할인율 그림 1은 강화 학습 메커니즘과 환경과 에이전트 간의 상호 작용을 요약 한 것이다.



Q-LEARNING 에이전트는 일련의 이산 시간 단계 t = 0, 1, 2, 3 ... [26] 각각에서 환경과 상호 작용한다. 시간 t = 0에서 환경은 초기 상태 s0∈S로 시작합니다. 그런 다음 t = 0에서 완료까지 각 시간 단계에서 :

* 에이전트는 ∈A (st)에서 동작을 선택합니다. 여기서 A (st)는 state st에서 사용할 수있는 동작 집합입니다.
* 환경은 보상 R (.st, at)의 분포에 따라 값 보상 rt를 샘플링하여 반환합니다.
* 환경은 샘플링하여 전환 확률 P에 기초하여 다음 상태 st + 1 (. | st, at)로 돌아갑니다.
* 에이전트는 보상 rt를 받고 새로운 상태 st + 1에서 끝난다.

각 단계 t에서 에이전트는 환경의 현재 상태를 관찰하고 정책 π에 기반한 동작을 선택합니다. 이 정책은 에이전트가 환경에 미치는 동작을 나타냅니다. 에이전트가 누적 보상을 최대화하는 최상의 방법으로 행동하기를 원하기 때문에 강화 학습의 목표는 정책 π \* 누적 할인 보상을 최대화하는 것입니다.

$$ \ sum\_ {t> 0} γ ^ tr\_t

$$ Q-LEARNING에서 규칙 π를 감안할 때 (상태, 동작) 쌍에 얼마나 좋은지를 알려주는 Qvalue 함수 (또는 Q 함수)라는 함수를 정의합니다. 상태 s와 액션 a의 모든 조합에 대해 함수는 a 상태에서 시작하여 π 규칙을 따르는 일련의 작업을 수행하여 달성 할 수있는 누적 보상을 반환합니다. 최적의 Q 값 함수 Q \*는 모든 가능한 규칙에 대해 주어진 (상태, 동작) 쌍으로부터 얻을 수있는 예상 최대 누적 보상입니다.

(s, *t) = max* π \ sum\_ {0} (γ ^ tr\_t | s = s\_0, a = a\_t, π)

$$ 다음의 Q *s {t + 1}, a {t + 1}의 최상의 상태 - 동작 값이 알려져 있다면, 최적의 전략은 r + γQ* (s *{t + 1}, a*{t + 1}) 여기서 r은 현재 단계에 대한 직접 보상입니다. Q \*는 Bellman 방정식을 만족시킵니다.

Q ^ \* $$ (S *T, a\_t) = R & LT (S\_T, a\_t) + 감마 맥스* {A *T + Q}. 1 (S* {T} +. 1 A\_ {T} +. 1)

$$ 여기서 γ는 0과 1 사이의 할인율 매개 변수입니다. 즉각적인 보상과 누적 보상의 상관 관계를 조정합니다. 감마가 0에 가까울수록 보상이 빠를수록 감마가 1에 가까울수록 보상이 많이 축적됩니다. 따라서, 최적의 전략 π *는* Q에 *의해* 규정 *된* 가장 높은 Q 값 *을* 갖는 동작 *을 사용하는 것에 대응한다.* Q-LEARNING 알고리즘은 Q 함수의 값을 반복적으로 추정하기 위해 식 (3)을 기초로한다. Q 함수는 기본값으로 초기화됩니다. 에이전트가 상태 st로부터 상태 st + 1에 도달하기위한 동작을 수행하고 보상 rt를 수신 할 때마다 Q- 함수는 다음과 같이 갱신된다 :

Q $$ (S *T, a\_t) ← Q (S\_T, a\_t) + 알파 (γ-max\_a R\_t + Q (S* {T} + 1, A) -. Q (S\_T, a\_t))

$$이 공식에서 α는 학습률 (0 ≤ α ≤ 1)입니다. α는 Q 관측 값에 대한 새로운 관측치의 영향을 나타낸다.

Markov 환경에 적용하면 Q-learning 알고리즘은 제한된 즉각적인 보상과 지속적으로 업데이트되는 상태 - 액션 쌍을 사용하여 실제 Q 기능에 수렴을 보장합니다 [28].

**4 제안 된 방법 : Android 앱 테스트의 Q-Learning**

Q-Learning 알고리즘은 특정 환경 조건을 달성하는 최선의 방법을 결합 할 수 있기 때문에 탐색 기능을 응용 기능이나 도달 할 수없는 상태를 나타내는 상태로 안내하는 데 사용할 수 있습니다. 이 기사에서 Q Learning Agent는 안드로이드 애플리케이션을 탐구한다. 안드로이드 애플리케이션은 GUI 상태의 모음으로 모델링되어 각 반복에서 클릭, 유형, 스 와이프, 스크롤 등과 같은 이벤트를 생성 할 수있다. 에이전트는 점차 응용 프로그램의 기능을 나타내는 방식으로 응용 프로그램을 탐색하는 방법을 단계적으로 학습 한 다음 최대한 많은 코드를 포함하는 테스트 사례를 생성합니다.

이 기사에서 Markov 모델은 다음과 같이 Android 애플리케이션 용 테스트 입력을 생성하도록 설계되었습니다.

* S : 응용 프로그램 상태가 설정되었습니다. 상태는 활동 이름과 해당 화면에서 사용 가능한 UI 이벤트 세트에 의해 결정됩니다. 액세스 상태 집합은 각 Q-LEARNING 반복에서 상태에 액세스 할 때마다 새 상태를 추가하여 업데이트됩니다.
* A : 가능한 GUI 이벤트 세트. 이벤트는 GUI 구성 요소 (예 : 버튼 클릭)에서 수행 할 수있는 작업에 해당합니다. 튜플 (GUI 구성 요소 클래스 이름, 이벤트 유형, GUI 구성 요소 고유 리소스 ID, 화면상의 GUI 구성 요소 위치)로 정의됩니다.
* R : 전환이 주어질 때 값을 반환하는 보상 기능 (상태, 이벤트, 새 상태)
* P : 이벤트가 현재 상태에서 실행되면 새 상태로의 전환은 테스트중인 애플리케이션의 응답에 의해 결정됩니다.
* γ : 할인 계수

우리의 Q-LEARNING 테스트 도구는 두 가지 주요 역할로 구성됩니다 : 환경 및 에이전트. 환경은 테스트중인 Android 기기 및 애플리케이션을 나타냅니다. Observer, Executor 및 Reward Function의 세 가지 보조 모듈이 있습니다. 에이전트는 테스트중인 애플리케이션의 가장 관련있는 동작 모델을 이해하고이 모델을 기반으로 테스트 케이스를 생성합니다. 행동 모델은 Q-LEARNING의 결과를 결합합니다. 상태 공간 S, 각 상태에서 실행될 수있는 이벤트 세트 W 각 (상태, 이벤트) 쌍에 대한 예상 Q 기능을 포함합니다.

테스트 케이스를 생성하기 위해 에이전트는 고정 된 수의 이벤트 (에피소드라고도 함)를 실행합니다. 에피소드를 완료 한 후 상담원은 방문한 사람 중 임의의 상태를 선택하고 거기에서 새 에피소드를 시작합니다. 에피소드의 길이는 각 테스트 케이스에 대한 검색 공간의 크기를 결정할 수 있기 때문에 프록시의 유효성에 영향을 미치는 중요한 매개 변수입니다. 이 매개 변수는 활동의 복잡성 및 여러 활동 간의 상호 작용 빈도와 같이 응용 프로그램 설계에 크게 좌우됩니다. 각 에피소드의 반복 과정은 다음과 같습니다.

t = 0에서 에피소드 끝까지 반복하십시오.

* 1 단계 : 환경 - 관찰자는 테스트중인 응용 프로그램을 관찰하고 현재 추상 상태를 만듭니다.
* 2 단계 : 에이전트는 현재 상태를 확인하고 정책 및 현재 동작 모델을 기반으로 실행할 다음 이벤트를 선택합니다.
* 3 단계 : 환경 - 선택한 이벤트를 실행할 프로그램을 실행합니다.
* 4 단계 : 환경 - 옵저버는 응용 프로그램의 새로운 GUI 상태를 관찰합니다.
* 5 단계 : 환경 - 보상 기능은 후속 보상을 계산합니다.
* 6 단계 : 에이전트가 Q 기능을 갱신합니다.

각 단계의 세부 사항은 다음과 같습니다.

**4.1 환경 관찰 (1 단계 및 4 단계)**

관찰자는 환경의 현재 상태를 관찰하는 기능을 가진 환경의 모듈입니다. Observer의 작업은 상태라고하는 각 반복에서 현재 GUI 트리의 추상 표현을 작성하는 것입니다. GUI 트리는 런타임시 Android UI Automator를 사용하여 테스트중인 애플리케이션에서 추출 할 수있는 GUI 구성 요소의 계층입니다. 추상 GUI 상태는 현재 활성화 된 이름과 현재 GUI 트리에있는 모든 UI 이벤트의 튜플로 구성됩니다. 관찰자는 현재 테스트 된 애플리케이션에 속한 GUI 요소 만 추상적 상태로 유지합니다 (GUI 요소가 속한 패키지를 확인하여). 이 기사에서는 이벤트 클릭, 길게 누르기, 체크 (체크 박스), 텍스트 입력 및 스크롤의 유형 만 고려하십시오. Android 플랫폼은 더 많은 이벤트 유형을 지원하지만, 현재 가장 많이 사용되며 다른 부분을 향후 작업의 일부로 취급하기 때문에 현재이 5 가지 유형 만 지원합니다. 테스트중인 응용 프로그램의 UI 이벤트 외에도 뷰어는 사용 가능한 UI 이벤트로 메뉴 버튼과 장치의 뒤로 버튼을 클릭합니다.

**4.2 현재 상태에서 실행할 이벤트를 선택하십시오 (2 단계)**

각 시간 단계에서 에이전트는 현재 상태에서 실행될 이벤트를 선택합니다. 이벤트는 π 정책에 따라 선택됩니다. 좋은 정책을 선택하는 것은 탐험과 착취 사이의 무역을 조화시키는 것이다 [26]. 막대한 수익을 얻으려면 에이전트는 과거에 시도한 이벤트를 선호하고 보상을 생성하는 데 효과적임을 알아야합니다. 응용 프로그램을 이해하고 가장 높은 Q 값을 갖는 이벤트를 선택하는 데 사용해야합니다. 그러나 이러한 이벤트를 선택하려면 에이전트는 향후 더 나은 선택을하기 위해 새로운 이벤트를 탐색해야합니다. 따라서 확률 방법이 일반적으로 고려됩니다. 이 연구에서는 ε-greedy라는 유명한 정책을 사용했습니다. 에이전트는 현재 상태에서 사용 가능한 임의의 이벤트를 선택하거나, 확률 ε을 가지거나, 1-ε의 확률로 현재 동작 모델에 따라 가장 높은 Q 값을 갖는 이벤트를 선택합니다.

테스트 프로세스가 시작될 때 우리는 에이전트가 애플리케이션에 가장 가까운 완벽한 모델을 구축하기 위해 최대한 많은 주를 탐색 할 수 있기를 원했기 때문에 큰 ε 값을 사용해야합니다. 그러나 일단 모델이 잘 구축되면 에이전트는 Q 값 함수를 따라 응용 프로그램의 기능을 신속하게 가져 와서 테스트하므로 ε 값은 작을 것으로 예상됩니다. 이 논문에서는 최대 탐색을 위해 ε = 1로 시작한 다음 첫 번째 100 회까지의 값을 최종 기간 (평가 기간 동안 경험적 연구 후에 0.5 회)까지 줄임으로써 프록시를 변경합니다 착취의 행동.

**4.3 환경에서 이벤트 실행 (3 단계)**

실행기 모듈 UI 자동화 [15] 안드로이드 디버그 브릿지 [1] 실제 장치 또는 에뮬레이터 안드로이드의 경우에 수행 하였다. 집행자의 관찰자는 모든 유형의 이벤트가 관찰 할 수 수행 할 수 있습니다 클릭, 프레스, 확인, 텍스트 입력, 스크롤. 텍스트 입력의 경우 텍스트는 실행 중에 임의로 생성됩니다.

**4.4 보상 계산 (5 단계)**

보상은 보상 함수 R에 의해 환경에서 계산됩니다. 보상 함수 R은 기본적으로 에이전트에게 어떤 이벤트가 좋고 어떤 것이 나쁜지를 알려줍니다. 보상 기능을 결정할 때 고려해야 할 두 가지 측면이 있습니다 : GUI 변경 및 실행 빈도.

새로운 기능을 트리거하기 위해 휴리스틱 이벤트를 식별하기 위해 보상 기능 추상의 상태에 많은 변화 GUI 리드를 지원하고 그들에게 더 큰 인센티브 이벤트를 제공합니다. 이 추론은 AutoBlackTest [20]에서 사용 된 추론과 유사합니다. 두 가지 상태의 S1과 S2, 및 GUI (S2)의 식별 번호를 비교하여 보상 기능을 감안할 때,하지만 이벤트 S1에서 S2는 다음과 같이 설명에 S1 변화의 정도에서 계산하지 | S2 \ (S1) |. 이어서 비 | 상대 변화를 결정하는 단계는 | | S2 |이고 | GUI의 이벤트의 수 | S2 S2 \ S1 | / | S2. 이 수식은 s2에서 새로 추가되었지만 s1에서 사라지는 구성 요소는 고려하지 않은 구성 요소를 고려합니다. 이렇게하면 응용 프로그램 활동간에 점프를 유발하는 이벤트의 값을 과도하게 프로비저닝하지 않아도됩니다. Q 값을 적당히 증가시켜 각 활동에서 활동과 기능 간의 전환을 테스트 할 수 있습니다.

에이전트가 탐사와 개발의 균형을 유지하는 데 도움이되는 또 다른 요소가 추가되었습니다. 실행 빈도. 우리는 각 이벤트가 실행되는 횟수를 계산하고 실행 빈도가 증가함에 따라 보상을 줄입니다. 보너스 가치는 일정 반복 횟수 후에 GUI 변경 보상에 비해 상대적으로 작아집니다. 그러나 Q-agent가 테스트 초기에 새로운 상태를 더 빠르게 탐색하여 상태 공간의 무결성을 향상시키는 데 도움이됩니다. 요약하면, 보상 함수의 정의는 다음 공식으로 제공됩니다.

R & LT $$ *T = R & LT (S\_T, a\_t, S* .} {1 + T) = FRAC \ {| S *{T} + 1 \ S\_T |.} {| S* {T} + 1 |.}

* \ frac {1} {f (s\_t, a\_t)} $$ 여기서 f (st, at)는 시간 t에서 상태 st에서 이벤트가 실행되는 빈도입니다.

**4.5 Q 값 기능 업데이트 (6 단계)**

에이전트는 현재 반복 중에 발생하는 변환을 기반으로 Q 값 함수, 즉 동작 모델을 작성하고 업데이트합니다.

에이전트는 각 반복 후에 방정식 (4), 학습률 α = 1 및 할인율 γ = 0.9를 사용하여 Q 값 함수를 업데이트합니다. 1에 가까운 학습률 α는 새로운 모델의 새로운 관측에 더 큰 영향을 미칩니다. 에이전트가 응용 프로그램의 작동 방식을 신속하게 이해할 수 있도록 값 α = 1을 선택합니다. 할인 요인은 즉각적인 보상과 미래의 사건 사이의 상관 관계를 균형 잡는데, 직접 보상을 극대화하기보다는 전체 에피소드 동안 수집 된 보상을 최대화하는 0.9의 가치를 지닙니다. 따라서, 방정식 (4)는

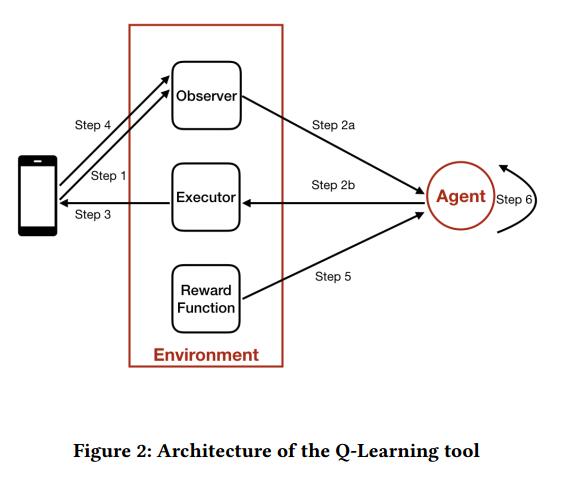
$$ Q (s *t, a\_t) ← r\_t + 0.9 max\_a Q (s* {t + 1}, a)

$$ 활동이 방문하지 않은 상태에 도달하면 공식의 두 번째 항은 우리가주의 Q 값을 추정하지 않았기 때문입니다. 이 경우, Q 값은 즉각적인 보상 일뿐입니다. 그렇지 않은 경우 Q 값은 대상 상태에있는 이벤트의 보너스 값과 Q 값을 기반으로 계산됩니다.

특수 기능은 변환으로 인해 응용 프로그램이 로그 아웃되는 상황을 처리하는 데 사용됩니다. 이 경우 테스트 도구는 테스트중인 애플리케이션의 일부가 아닌 최대 수의 이벤트를 무작위로 실행합니다. 이 최대 수는 두 개의 이벤트입니다. 각 실행 후에 도구가 테스트중인 응용 프로그램으로 되돌아가는 지 확인합니다. 그렇다면 우리는 정상적으로 테스트하고 그렇지 않은 경우 도구가 앱의 실행 프로그램 활동으로 돌아갑니다. 앱 종료를 유발 한 이벤트에는 실행을 피 했으므로 보상이 주어졌습니다. 이는 정상적인 응용 프로그램 종료와 응용 프로그램 충돌 (우리가 잘못 생각하는 것)을 구분할 수 없기 때문에 필요합니다.

**4.6 구현**

이 도구는 Python 2.7로 작성되었습니다. 이 도구의 아키텍처와 워크 플로가 그림 2에 나와 있습니다. Android 장치와 테스트중인 응용 프로그램과 상호 작용하기 위해 Android UI Automator [15] 파이썬 라이브러리와 Android Debug Bridge [1]을 사용합니다.



**5 평가**

**5.1 평가 지표**

도구 유효성 측정은 코드 적용 범위이며 테스트 도구를 평가하는 일반적인 지표입니다. 특히, 행 유효 범위를 사용하여 Q-Learning 도구를 평가하고 가장 작은 적용 범위 단위이기 때문에 다른 테스트 도구와 비교합니다.

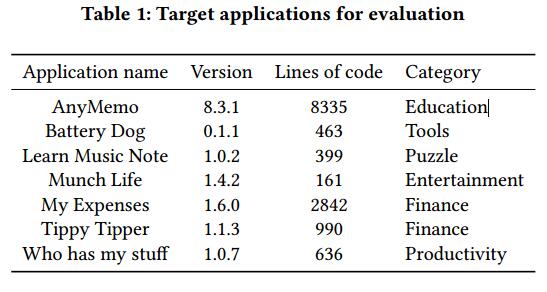
이 도구의 평가는 다음과 같은 연구 질문에 대답하도록 설계되었습니다.

* RQ1 :이 도구는 가장 진보 된 테스트 도구에 비해 코드 커버리지를 개선합니까?
* RQ2 : 공구가 결함을 감지 할 수 있습니까?

이 툴은 Android Monkey, Dynodroid 및 PUMA와 같은 가장 진보 된 테스트 툴과 비교됩니다.

**5.2 대상 응용 프로그램**

AnyMemo [22], Battery Dog [16], Learn Music Notes [24], Munch Life [23], My Expenses [27], Tippy Tipper [12 및 7] 누가 미끼를 가지고 있나 [6]. 그들은 정적 인 내용과 동적 인 내용 모두로 모든 규모와 복잡성을 가지고 있습니다. 표 1은 각 응용 프로그램에 대한 간단한 설명을 제공합니다.



**5.3 평가 설정**

각 도구는 AndroTest [11]에서 제공하는 별도의 가상 시스템에서 실행됩니다. AndroTest는 응용 프로그램 모음에 대해 다양한 테스트 도구를 평가할 수있는 좋은 벤치 마크를 제공합니다. 모든 가상 머신은 6114 MB의 기본 메모리와 2 개의 프로세서가있는 Ubuntu 32 비트를 실행합니다. 각 테스트 세션마다 응용 프로그램은 기본 설정으로 에뮬레이터에만 설치됩니다. 이전 세션의 가짜 데이터를 방지하기 위해 새 테스트를 시작하기 전에 에뮬레이터의 모든 데이터가 제거되고 에뮬레이터가 다시 시작되었습니다. 각 테스트 도구는 각 응용 프로그램에서 네 번 실행 된 다음 평균을 최종 결과로 사용합니다.

**5.4 결과 얻기**

**5.4.1 코드 적용 범위**

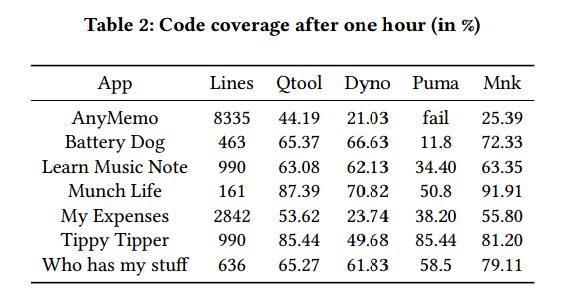
테스트를 실행하기 전에 테스트중인 응용 프로그램에 포함 된 도우미 클래스 인 Emma [4]를 사용하여 코드 커버리지를 얻습니다. 사용자 정의 스크립트는 장치에서 코드 적용 범위를 추출하고이를 5 분마다 사람이 읽을 수있는 보고서로 변환하는 데 사용됩니다. 이 보고서는 패키지, 클래스, 블록 및 행에 대한 자세한 정보를 제공합니다. Emma가 반환하는 범위는 클래스를 검색하는 코드 행을 고려합니다. Emma (응용 프로그램의 코드 줄 수만 계산)의 결과를 기반으로 실제 적용 범위를 계산하고 평가에 사용합니다.

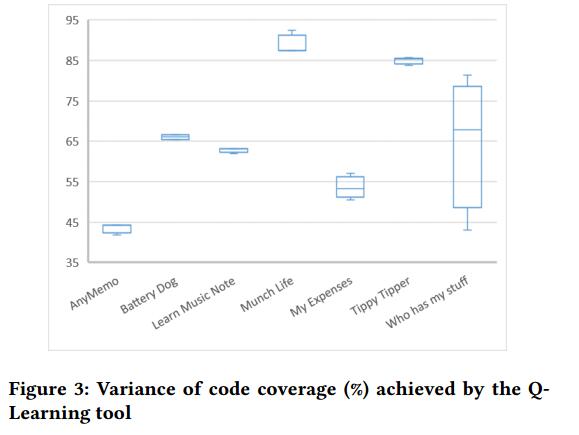
**5.4.2 응용 프로그램 오류**

예기치 않게 응용 프로그램이 충돌하면 도구가 실행될 때 응용 프로그램 오류가 발견됩니다. 잘못된 스택 추적은 시스템 메시지를 덤프하는 명령 행 도구 인 logcat에 의해 기록됩니다. Android Debug Bridge (adb)를 통해 휴대 전화 또는 에뮬레이터에서 오류 추적을 가져옵니다.

**5.5 평가 결과**

Google 도구의 코드 범위는 Android Monkey, Dynodroid 및 Puma의 결과와 비교되었습니다. 각 도구를 한 시간 동안 실행 한 다음 중지합니다. 즉 한 시간 실행 한 후 결과를 비교합니다.





**5.5.1 RQ1 :이 도구는 가장 진보 된 테스트 도구에 비해 코드 커버리지를 개선합니까?**

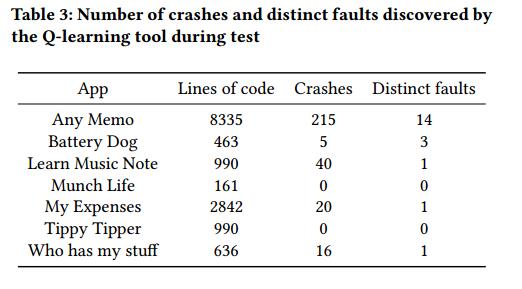
표 2는 7 가지 대상 응용 프로그램의 코드 적용 범위를 비교하고, 그림 3은 Q-LEARNING 도구로 달성 한 코드 적용 범위 변경을 보여줍니다.

Battery Dog 외에도 Dynodroid보다 코드 범위가 약간 낮았으며, 제안 된 방법은 DynoDroid와 Puma보다 7 개 애플리케이션 중 6 개가 더 우수합니다. 이 차이점은 Dynodroid가 시스템 이벤트와 UI 이벤트를 생성 할 수 있다는 사실과 우리의 도구가 UI 이벤트 만 생성 할 수 있다는 사실로 설명 할 수 있습니다. Battery Dog의 경우 앱이 휴대 전화의 배터리 잔량을 모니터링하고 시스템 이벤트에 응답하며 Google 도구가 처리하지 못합니다.

대부분의 애플리케이션에서 Google 툴보다 훨씬 많은 초당 이벤트를 효율적으로 생성 할 수 있기 때문에 Android Monkey의 코드 커버리지가 더 높아졌습니다.

**5.5.2 RQ2 : 도구가 응용 프로그램의 오류를 감지 할 수 있습니까?**

코드 적용 범위는 테스트 도구의 효율성을 나타내는 좋은 지표이지만 테스트의 궁극적 인 목표는 응용 프로그램에서 오류를 찾는 것입니다. 따라서 감지 된 오류를 기반으로 도구를 평가해야합니다. 각 테스트 중에 Android Debug Bridge를 통해 장치의 로그가 기록 된 다음 분석되어 응용 프로그램 관련 오류를 발견합니다. 각 응용 프로그램마다 테스트 중에 응용 프로그램에서 발생한 오류 (또는 충돌) 수와 여러 오류 수를 계산합니다. 표 3은 실험 결과를 요약 한 것이다. 우리의 도구는 대부분의 응용 프로그램에서 충돌을 유발하고 결함을 찾을 수 있습니다. 일부 응용 프로그램에서는 동일한 오류로 인해 여러 번의 충돌이 발생합니다.



**효과성에 대한 6 가지 위협**

효율성에 대한 두 가지 주요 위협은 내부 및 외부 효과입니다. 외부 유효성에 대한 한 가지 위협은 평가에 사용 된 응용 프로그램의 수입니다. 7 개의 응용 프로그램 만 있지만, 우리는 다른 범주와 다양한 크기의 응용 프로그램을 선택하여이 위협을 최소화하려고 노력합니다. 내부 타당성에 대한 위협은 우리의 접근법의 비결정론으로, 각 실행에 대해 다른 코드 적용 범위를 초래할 수 있습니다. 따라서이 위협을 줄이기 위해 여러 번 실행했습니다.

**7 결론**

본 논문에서는 Q-LEARNING 알고리즘을 이용한 안드로이드 애플리케이션 테스트 강화 학습 방법을 제안한다. 제안 된 도구는 Q-LEARNING 에이전트를 기반으로하며 에이전트가 시행 착오적으로 응용 프로그램과 상호 작용할 수 있도록하여 응용 프로그램의 동작 모델을 단계적으로 작성합니다. Q-learning 에이전트는 애플리케이션의 가장 관련성 높은 기능을 구현하여 테스트 케이스를 생성하는 가장 좋은 방법을 탐색합니다. 이 도구를 평가하면 도구가 기존의 테스트 도구에 비해 코드 범위 및 오류를 감지 할 수있는 기능이 크게 향상되었음을 입증했습니다.

향후 개선 우리 툴에는 Q-LEARNING 알고리즘의 보상 기능에 대한 자세한 조사, 시스템 및 컨텍스트 이벤트와 같은 더 많은 유형의 이벤트 지원 및 비 결정적 처리 처리가 포함됩니다.