

Automatic Web Testing Using Curiosity-Driven Reinforcement Learning

Curiosity駆動の強化学習を用いた 自動Webテスト

Yan Zheng^{†,‡,*}, Yi Liu^{‡,‡,*}, Xiaofei Xie^{‡,✉}, Yepang Liu[‡], Lei Ma^{*}, Jianye Hao[†], and Yang Liu[‡]

[†]Tianjin University, Tianjin, China; [‡]Nanyang Technological University, Singapore

[‡]Dept. of Comp. Sci. and Engr., Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Southern University of Science and Technology, Shenzhen, China; ^{*}Kyushu University, Fukuoka, Japan.

In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, ES, 2021 pp. 423-435.

- Webアプリケーションは、デスクトップアプリケーションと同じくらい広く使われている
- クライアント側とサーバ側で異なる言語(HTML, JavaScript, Go, PHP)で実装されることが多い
 - ⇒ テストが困難

一般的には、より多くWebページを探索し、より多くのWebアプリの状態をカバーするほど、不具合が発見されやすくなる

⇒ 十分な探索を行い、多様なテストケースを生成することが望まれる

ランダムベースのアプローチ^[13, 14]（実用的な開発に広く採用）

Webアプリ上で可能なランダムな操作を実行

- ボタンへのテキスト入力を行うなどの無効なテストケースの生成
- 到達しにくいページのテストケースが生成されない可能性

⇒ 探索効率が悪い

[13] Google, “Monkey,” <https://developer.android.com/>, 2018.

[14] “Crawljax,” <https://github.com/crawljax/crawljax>, 2017.

一般的には、より多くWebページを探索し、より多くのWebアプリの状態をカバーするほど、不具合が発見されやすくなる

⇒ 十分な探索を行い、多様なテストケースを生成することが望まれる

モデルベースのアプローチ[15, 16, 17, 18, 19]

テスターがナビゲーションモデル(ユーザの操作と、それに伴うページ遷移の関係)を構築し、それに応じたテストケースを生成

- 探索範囲がテスターの構築するモデルに依存
- 事前に与えるモデルでは、動的なコンテンツへの対応が困難

⇒ 多様なテストケースの生成が困難

[15] M. Biagiola, A. Stocco, F. Ricca, and P. Tonella, “Diversity-based web test generation,” in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 142–153.

[16] A. Mesbah, A. Van Deursen, and S. Lenseslink, “Crawling ajax-based web applications through dynamic analysis of user interface state changes,” ACM Transactions on the Web (TWEB), vol. 6, no. 1, pp. 1–30, 2012.

[17] A. Marchetto, P. Tonella, and F. Ricca, “State-based testing of ajax web applications,” in 2008 1st International Conference on Software Testing, Verification, and Validation. IEEE, 2008, pp. 121–130.

[18] S. Athaiya and R. Komondoor, “Testing and analysis of web applications using page models,” in Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, July 10 - 14, 2017, T. Bultan and K. Sen, Eds. ACM, 2017, pp. 181–191.

[19] M. Schur, A. Roth, and A. Zeller, “Procrawl: mining test models from multi-user web applications,” in International Symposium on Software Testing and Analysis, ISSTA '14, San Jose, CA, USA - July 21 - 26, 2014, C. S. Pasareanu and D. Marinov, Eds. ACM, 2014, pp. 413–416.

目的とアプローチ

目的

先行研究の課題を解決するため、**事前知識(モデル)不要で到達しにくいページも探索可能**なWebテストフレームワークWebExplorを構築

アプローチ

- 多様なテストケースの生成を促す報酬関数を定義した**強化学習**を用いて、オンザフライ※での最適化を行い、効率的な探索
- 到達しにくいページを探索できるようにするため、過去に探索した状態を参照し**探索回数が少ない状態を探索するように導く**ガイドを設ける

※オンザフライ(on-the-fly)：事前に学習を行うのではなく、テスト中に学習を行う

方策関数

強化学習の目的は、方策関数 π を学習すること

$a_t = \pi(s_t)$: ある状態 s_t において、最善の行動 a_t を返す

- 状態：WebページのHTMLタグ構造
- 行動：ボタンのクリック、テキスト入力など
- タイムスタンプを進めて得られる一連の行動 → テストケース

Q関数

- 方策関数の学習には、Q関数を利用

$$Q(s', a) = \underbrace{R(s', a)}_{\text{状態} s' \text{ で行動 } a \text{ を取った時の報酬 (後ほど定義)}} + \lambda \underbrace{\max_{a'} Q(s, a')}_{\text{状態 } s' \text{ で行動 } a \text{ を取り次の状態 } s \text{ に遷移した際に、最善の行動 } a' \text{ をとると得られる長期的報酬}} : \text{ある状態 } s' \text{ で行動 } a \text{ を取り、以降得られる報酬が最大となる行動をとり続けた時に得られる長期的報酬を返す}$$

状態 s' で行動 a を取った時の報酬(後ほど定義)

状態 s' で行動 a を取り次の状態 s に遷移した際に、最善の行動 a' をとると得られる長期的報酬

- Q関数は再帰関数で、次にとれる行動がないとき第2項が0となり、Q関数の値が確定して逆伝搬していく仕組み
- 方策関数では、Q関数の値(=長期的報酬)が最大となる行動を返す

報酬関数

多様なテストケースの生成を促すため、**多くの状態を探索することで報酬が高くなる必要**
そこで、curiosity(好奇心)^[42, 43]を使った報酬関数を定義

$$curiosity(s', a, s) = \frac{1}{\sqrt{N(s', a, s)}}$$

$N(s', a, s)$: 初期値1, 状態 s' から s に遷移する度に1加算

- N が増える = **同じ状態遷移を繰り返す → 報酬が減る**

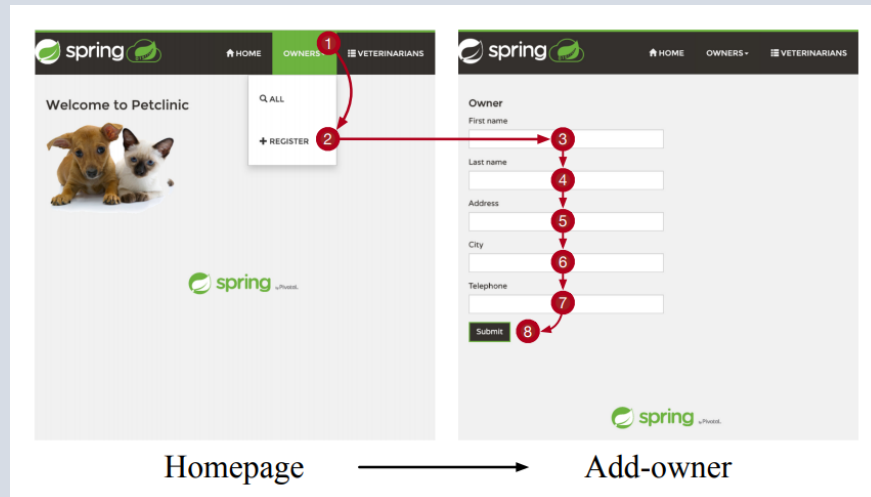
[42] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 16–17.

[43] N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T. P. Lillicrap, and S. Gelly, "Episodic Curiosity through Reachability." ICLR, 2019.

提案手法 | 到達しにくいページを探索する方法

到達しにくいページとは

到達するのに、長い一連の行動が必要なページ



複数のフォームへの適切な入力 + Submitボタンを押す
→ 新たなページへ遷移が可能

仮に各遷移確率が0.9と高かったとしても、
新たなページへ到達する確率は、 $0.9^7 \simeq 0.48$

解決策

先ほど報酬関数で利用した*curiosity*を使う

方策関数による探索で新しい状態を発見できなくなったら、*curiosity*が高い状態へ遷移させる

⇒ 探索していない状態の*curiosity*は高い値なので、未探索の状態をより深く探索できるようになる

RQ1：ブランチカバレッジは，既存手法と比べてどの程度か？

RQ2：不具合検出の性能は，既存手法と比べてどの程度か？

RQ3：到達しにくい状態への遷移をガイドすることは，カバレッジと不具合検出にどの程度効果的か？

RQ4：実世界のWebアプリへのテストにどの程度効果的か？

提案手法の評価 | RQ1, 2の評価方法

テスト対象

先行研究で対象とされた, GitHub上に公開されているOSSのWebアプリ6つ^[15]

比較対象の既存手法

• モデルフリーの手法

探索アルゴリズム自体はどちらもランダム

Random^[13], Crawljax^[14] (提案手法はモデルフリーに該当)

• モデルベースの手法

- DIG^[15]: 多様性を重視した探索

- SUBWEB^[5]: カバレッジを重視した探索

モデル構築方法に自動(APO), 手動(MPO)の2種類がありそれぞれ評価

[15] M. Biagiola, A. Stocco, F. Ricca, and P. Tonella, "Diversity-based web test generation," in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 142–153.

[13] Google, "Monkey," <https://developer.android.com/>, 2018.

[14] "Crawljax," <https://github.com/crawljax/crawljax>, 2017.

[5] M. Biagiola, F. Ricca, and P. Tonella, "Search based path and input data generation for web application testing," in International Symposium on Search Based Software Engineering. Springer, 2017, pp. 18–32.

提案手法の評価 | RQ1：ブランチカバレッジ※

モデルフリー

提案手法

モデルベース

Subjects	Average Branch Coverage (%)						
	WebExplor	Crawljax	Random	DIG		SUBWEB	
	Navigation model-free			APO	MPO	APO	MPO
Dimeshift	51.0 (2.7)	11.8 (0.0)	24.1 (12.8)	38.5 (1.6)	40.1 (1.6)	36.7 (2.3)	38.9 (1.8)
Pagekit	36.0 (1.3)	0.5 (0.0)	0.5 (0.0)	24.3 (2.6)	31.7 (4.9)	24.7 (2.8)	27.6 (3.8)
Splittypie	43.4 (0.3)	37.7 (3.9)	18.9 (1.1)	46.9 (1.8)	45.3 (2.5)	44.1 (3.5)	44.7 (2.0)
Phoenix	81.7 (1.3)	30.2 (9.2)	42.1 (0.0)	60.1 (5.7)	65.1 (9.6)	61.3 (2.3)	63.1 (8.7)
Retroboard	61.4 (0.3)	22.2 (0.0)	56.1 (0.0)	68.7 (2.1)	70.9 (4.3)	68.1 (2.7)	71.9 (3.8)
Petclinic	85.0 (0.0)	18.0 (2.5)	35.0 (26.3)	80.5 (7.3)	49.5 (7.9)	83.0 (2.9)	49.5 (7.9)
Average	59.8	20.1	29.45	53.2	50.4	53.0	49.3

- 各手法ランダム性があるため、15回繰り返した平均
- 太字が各プロジェクトで一番高い値

- 全体として、モデルフリー < モデルベース
- 提案手法はモデルフリーにもかかわらず、多くのテスト対象で最も高いスコアを達成

※ブランチカバレッジ：プログラムの分岐のうち、テストを実施した割合

提案手法の評価 | RQ2：不具合検出の性能

モデルフリー

提案手法

モデルベース

Subjects	Average Unique Failures (#)						
	WebExplor	Crawljax	Random	DIG		SUBWEB	
	Navigation model-free			APO	MPO	APO	MPO
Dimeshift	9.0 (1.7)	1.0(0.0)	1.4 (0.5)	2.6 (0.8)	2.9 (0.5)	2.4 (0.9)	2.9 (0.5)
Pagekit	4.2 (0.9)	0.0 (0.0)	0.0 (0.0)	1.0 (0.0)	2.9 (0.4)	1.0 (0.0)	0.7 (1.0)
Splittypie	8.0 (1.4)	5.6 (0.5)	3.0 (0.0)	4.0 (0.0)	4.0 (0.0)	4.0 (0.0)	6.0 (0.0)
Phoenix	5.3 (0.5)	0.0 (0.0)	1.2 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.1 (0.5)
Retroboard	1.0 (0.0)	0.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
Petclinic	11.7 (1.0)	0.7 (0.0)	2.0 (0.0)	1.5 (0.0)	3.0 (0.7)	1.5 (2.1)	3.3 (0.7)
Average	6.5	1.2	1.4	2.0	2.6	2.0	2.7

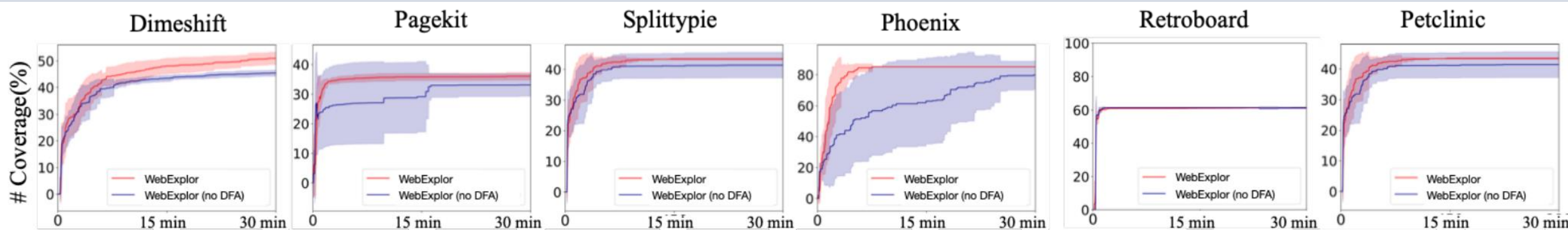
- 各手法ランダム性があるため、15回繰り返した平均
- 太字が各プロジェクトで一番高い値

- 不具合：JavaScriptランタイムエラー，クライアントエラー，サーバーエラーなど
- 表は，ユニークな不具合数をカウント
- 提案手法がすべてのテスト対象で最も高いスコアを達成

提案手法の評価 | RQ3：到達しにくい状態への遷移をガイドする有効性（1/2）

ブランチカバレッジへの有効性

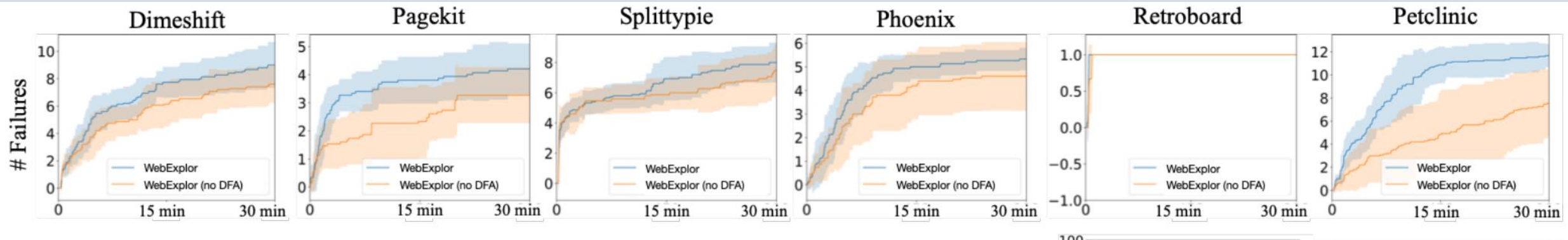
- 提案手法のガイドあり，ガイドなしで比較
- 横軸はテスト時間，縦軸はブランチカバレッジ
- ランダム性があるので，15回繰り返して，線は平均値，影の部分は標準偏差



すべてのプロジェクトにおいて，
ガイドありの方が，平均値の立ち上がりが早く，標準偏差も小さく安定している

不具合検出への有効性

- 提案手法のガイドあり，ガイドなしで比較
- 横軸はテスト時間，縦軸は検出されたユニークな不具合数
- ランダム性があるので，15回繰り返して，線は平均値，影の部分は標準偏差



すべてのプロジェクトにおいて，
ガイドありの方が，平均値の立ち上がりが早く最終的な値も高い上，
標準偏差も小さく安定している

商用のSaaSシステム(匿名)に対して，テストを実施

⇒ 12個の不具合を検出

不具合の例

- モジュール読み込み失敗によるサーバ側のディレクトリ構造漏洩
- 非同期レンダリングにおいて，レンダリングが完了する前の要素にアクセスしたことによるNull Pointerエラー

⇒ 提案手法は，実世界のWebアプリにおいても，人間による手作業や事前知識なしに不具合検出可能

目的

先行研究の課題を解決するため，**事前知識不要**で**到達しにくいページも探索可能**なWebテストフレームワークWebExplorを構築

アプローチ

- 多様なテストケース生成を促す報酬関数を定義した**強化学習**を用いて，効率的な探索
- 到達しにくいページを探索できるようにするため，過去に探索した状態を参照し**探索回数が少ない状態を探索するように導く**ガイドを設ける

提案手法の評価

- 既存の手法よりも，**カバレッジと不具合検出の性能が向上**
- 到達しにくいページへの探索を導く**ガイドの有効性を確認**
- **実世界のWebアプリでも不具合検出可能**