Разработка инфраструктуры для обучения искусственных нейронных сетей выбору оптимального пути для символьного исполнения

Нигматулин М.В., СПбГУ, Санкт-Петербург, mvnigma@gmail.com,
Чистякова А.А., СПбГУ, Санкт-Петербург, chi.vinny0702@gmail.com,
Григорьев С.В., СПбГУ, Санкт-Петербург, rsdpisuy@gmail.com

Аннотация

В данной работе представлено описание инфраструктуры для обучения графовых нейронных сетей выбору оптимального пути для символьного исполнения. Использование графовых нейронных сетей позволяет учитывать структуру программы, что в перспективе может улучшить процесс генерации тестов.

Введение

Множество инструментов для генерации тестов [11, 7, 2, 3] используют технику, позволяющую исследовать все возможные пути (ветви) исполнения программы путем использования символьных данных вместо реальных — символьное исполнение [9]. Однако у этой техники есть проблема экспоненциального увеличения числа путей (взрыва путей) во время исследования программы [4], что значительно увеличивает время генерации тестов.

Одним из способов решения проблемы взрыва путей является оптимизация выбора путей исполнения. Выбирая более оптимальные пути, можно сократить количество исследуемых состояний исполнения.

Одним из подходов к разработке стратегий выбора оптимального пути является применение машинного обучения для автоматического вывода правил выбора путей. Этот подход основывается на анализе данных об особенностях состояний исполнения [8, 5].

Так как множество состояний и путей между ними представляют собой граф, называемый также графом исполнения [6], новые данные для анализа могут быть предоставлены графовыми нейронными сетями [10] (англ. *Graph Neural Networks, GNN*). Графовые нейронные сети могут улавливать зависимости между вершинами в графах, что можно использовать для разработки стратегии выбора оптимальных путей в графе исполнения, учитывающей не только особенности состояний, но и взаимосвязи между ними.

В данной работе описывается инфраструктура для обучения графовых нейронных сетей выбору оптимального пути исполнения.

Цель работы

Цель работы — реализация инфраструктуры для обучения графовых нейронных сетей выбору оптимальных путей на основе структуры графа программы.

Для достижения этой цели необходимо создать модуль для обучения графовых нейронных сетей выбору оптимального пути для символьного исполнения, разработать протокол взаимодействия с различными символьными машинами. Дополнительно необходимо разработать модули для обеспечения возможности интеграции обученных нейронных сетей с символьными машинами и сравнения стратегий выбора пути.

Обзор

Инфраструктурные решения для машинного обучения уже существуют и успешно применяются в области создания стратегий выбора оптимального пути. Рассмотрим их с точки зрения используемых символьных машин и алгоритмов машинного обучения.

ParaDySE. В данном подходе авторы представляют решение для автоматической генерации эвристических стратегий выбора путей для символьного исполнения. В качестве символьной машины ParaDySE использует KLEE [3] и CREST [2]. В процессе символьного исполнения программы ParaDySE оптимизирует функцию выбора состояния путем подбора весов вектора признаков состояний исполнения (например, находится ли ветвь в цикле), составляющих оптимальный эвристический алгоритм для конкретной программы [5].

Q-KLEE. В данном подходе авторы используют Q-learning [12] — метод машинного обучения с подкреплением для обучения выбору оптимального пути, статический анализ программы для определения инструкций, которые Q-learning будет оценивать наградой, и KLEE в качестве символьной машины [13].

LEARCH. Еще один подход с использованием машинного обучения — LEARCH. Основной компонент LEARCH — модель машинного обучения, оценивающая награду за выбор состояния. Для каждого состояния вычисляется вектор признаков, который затем подается в модель машинного обучения для оценки [8]. Для создания набора данных для обучения и оценки моделей, LEARCH использует символьную машину KLEE.

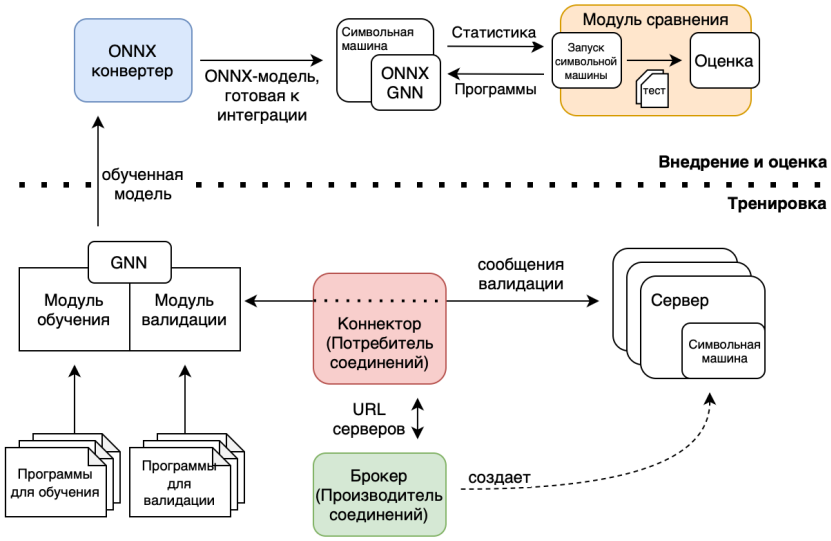

Рис. 1: Схема компонентов

Выводы

Хотя существующие подходы используют машинное обучение для создания оптимального алгоритма выбора пути для символьного исполнения, рассмотренные подходы не предполагают использование графовых нейронных сетей и реализованы на базе конкретных символьных машин. Таким образом, возникает потребность в разработке нового набора инструментов.

Описание инфраструктуры

Разработанное решение состоит из нескольких компонентов, отношения между которыми изображены на рисунке 1. На рисунке цветами обозначены разработанные модули.

- Серверы с символьными машинами — служат в качестве механизма исследования графа исполнения;
- Коннектор — реализует протокол взаимодействия и логику управления символьным исполнением на серверах со стороны модуля валидации;

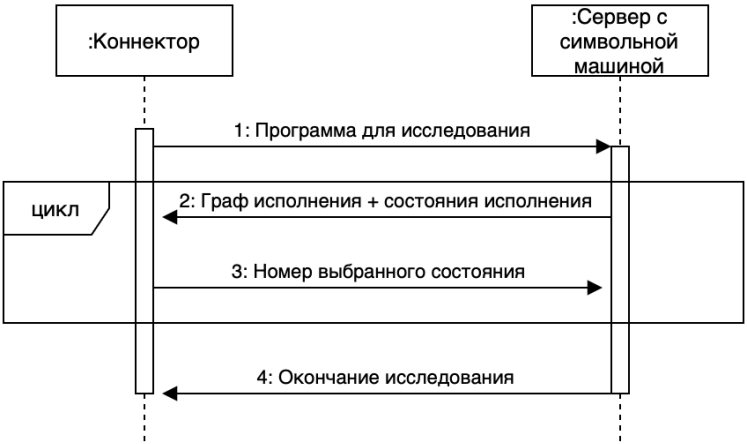

Рис. 2: Диаграмма последовательности протокола взаимодействия

- Брокер — создает серверы с символьными машинами и управляет соединениями с ними;
- Модуль обучения — осуществляет обучение нейронной сети;
- Модуль валидации — осуществляет валидацию, взаимодействует с сервером через Коннектор;
- ONNX конвертер — конвертирует обученные графовые нейронные сети (GNN) в формат ONNX [1].

Во время обучения нейронная сеть проходит через две фазы: тренировочную и валидационную. В тренировочной фазе сеть обучается выбирать оптимальный путь, а в валидационной фазе оценивается ее качество с помощью взаимодействия с сервером символьной машины. Так как символьная машина работает сравнительно медленно, валидационная фаза использует несколько серверов. Для создания и распределения серверов по задачам валидационной фазы был создан медиатор, обозначенный на рисунке как Брокер. Брокер создает сервера с символьными машинами и выдает их адреса Коннектору.

Универсальный протокол взаимодействия

Коннектор инкапсулирует логику взаимодействия с сервером. Он реализует протокол взаимодействия со стороны модуля валидации и отвечает за

обработку ошибок в логике взаимодействия. Коннектор может получать и обрабатывать сообщения, содержащие актуальное состояние графа исполнения и информацию о завершении работы символьной машины. Коннектор отправляет серверу сообщения о начале работы символьной машины и выбранном для исследования состоянии исполнения. Диаграмма последовательности процесса взаимодействия представлена на рисунке 2.

Обеспечение внедряемости результатов

Для внедряемости результатов обучения был разработан модуль экспорта графовых нейронных сетей в формат ONNX, предназначенный для представления моделей машинного обучения. ONNX определяет общий формат файлов, чтобы разработчики могли использовать модели в различных инструментах, средах выполнения и компиляторах [1].

Модуль сравнения результатов

Для получения данных и визуализации результатов экспериментов был разработан модуль сравнения результатов работы символьной машины с использованием различных стратегий выбора пути. Модуль запускает процесс символьной машины с запросом на генерацию тестов для выбранной программы и считывает результаты. После получения результатов производится сравнение стратегий с помощью визуализатора.

На рисунке 3 изображен график, созданный визуализатором. На нем представлено сравнение результатов работы символьной машины с использованием модели, обученной с помощью разработанной инфраструктуры, с существующими стратегиями.

Заключение

В результате работы была реализована инфраструктура для обучения графовых нейронных сетей выбору оптимального пути для символьного исполнения, позволяющая обучать нейронные сети во время взаимодействия с символьными машинами, внедрять обученные модели и сравнивать стратегии выбора пути.

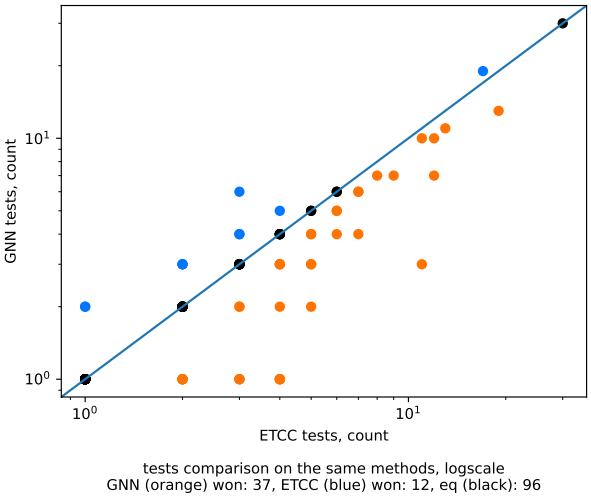

Рис. 3: Сравнение по количеству сгенерированных тестов с существующей стратегией на методах с одинаковым покрытием для двух стратегий. Оранжевым отмечены точки, где предложенный подход выигрывает сравнение.

Список литературы

[1] Junjie Bai, Fang Lu, Ke Zhang и др. *ONNX: Open Neural Network Exchange*. `https://github.com/onnx/onnx`. Accessed: 2024-05-13. 2019.

[2] J. Burnim и K. Sen. «Heuristics for Scalable Dynamic Test Generation». В: *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering*. ASE '08. USA: IEEE Computer Society, 2008, с. 443–446. isbn: 9781424421879. doi: `10.1109/ASE.2008.69`. url: `https://doi.org/10.1109/ASE.2008.69`.

[3] Cristian Cadar, Daniel Dunbar и Dawson Engler. «KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs». В: *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*. OSDI'08. San Diego, California: USENIX Association, 2008, с. 209–224.

[4] Cristian Cadar и Koushik Sen. «Symbolic execution for software testing: three decades later». В: *Commun. ACM* 56.2 (2013), с. 82–90. issn: 0001-0782. doi: `10.1145/2408776.2408795`. url: `https://doi.org/10.1145/2408776.2408795`.

[5] Sooyoung Cha и др. «Enhancing Dynamic Symbolic Execution by Automatically Learning Search Heuristics». В: *IEEE Transactions on Software Engineering* 48.9 (2022), с. 3640—3663. doi: 10.1109/TSE.2021.3101870.

[6] Keith D. Cooper и Linda Torczon. «Chapter 4 - Intermediate Representations». В: *Engineering a Compiler (Third Edition)*. Под ред. Keith D. Cooper и Linda Torczon. Third Edition. Philadelphia: Morgan Kaufmann, 2023, с. 159—207. isbn: 978-0-12-815412-0. doi: https://doi.org/10.1016/B978-0-12-815412-0.00010-3. url: https://www.sciencedirect.com/science/article/pii/B9780128154120000103.

[7] Patrice Godefroid, Michael Y. Levin и David Molnar. «SAGE: Whitebox Fuzzing for Security Testing: SAGE Has Had a Remarkable Impact at Microsoft.» В: *Queue* 10.1 (2012), с. 20–27. issn: 1542-7730. doi: 10.1145/2090147.2094081. url: https://doi.org/10.1145/2090147.2094081.

[8] Jingxuan He и др. «Learning to Explore Paths for Symbolic Execution». В: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, с. 2526–2540. isbn: 9781450384544. doi: 10.1145/3460120.3484813. url: https://doi.org/10.1145/3460120.3484813.

[9] James C. King. «Symbolic Execution and Program Testing». В: *Commun. ACM* 19.7 (1976), с. 385–394. issn: 0001-0782. doi: 10.1145/360248.360252. url: https://doi.org/10.1145/360248.360252.

[10] Franco Scarselli и др. «The Graph Neural Network Model». В: *IEEE Transactions on Neural Networks* 20.1 (2009), с. 61—80. doi: 10.1109/TNN.2008.2005605.

[11] Symflower. *Symflower - Automated Unit Testing for Software Developers*. https://symflower.com/en/. Accessed: 2024-05-20.

[12] Christopher J. C. H. Watkins и Peter Dayan. «Q-learning». В: *Machine Learning* 8.3 (май 1992), с. 279—292. issn: 1573-0565. doi: 10.1007/BF00992698. url: https://doi.org/10.1007/BF00992698.

[13] Jie Wu, Chengyu Zhang и Geguang Pu. «Reinforcement Learning Guided Symbolic Execution». В: *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2020, с. 662—663. doi: 10.1109/SANER48275.2020.9054815.