Санкт-Петербургский государственный университет

Кафедра системного программирования

Создание графовой нейронной сети, предсказывающей пути при символьном исполнении

Анна Артуровна Чистякова, группа 21.Б10-мм

Научный руководитель: к.ф.-м.н. С. В. Григорьев, доцент кафедры системного программирования
Рецензент: ведущий разработчик ООО «ИИнтелКод» М. П. Костицын

Санкт-Петербург
2025

Введение

- Растет необходимость тестировать сложные системы
- Символьное исполнение помогает не пропустить входы, на которых программа работает неправильно
- Проблема — большое количество возможных путей исполнения программы
- Символьное исполнение решает разные задачи
  - В этом проекте — максимизация процента тестового покрытия с учетом минимизации времени и количества сгенерированных тестов

Существующие решения проблемы большого числа путей исполнения

- Алгоритмы на графах (DFS, BFS)
- Различные эвристики (например, присваивание вероятности выбора пути на основе текущих свойств состояния)
- Подходы, основанные на машинном обучении:
  - Learch[1] — линейная модель
  - Q-learning[2]
  - FeatMaker[3]
  - ! Не учитывают графовую структуру кода

[1] "Learning to explore paths for symbolic execution", Jingxuan He, Gishor Sivanrupan, Petar Tsankov, Martin Vechev, 2021

[2] "Reinforcement learning guided symbolic execution", Wu Jie, Zhang Chengyu, Pu Geguang, 2020

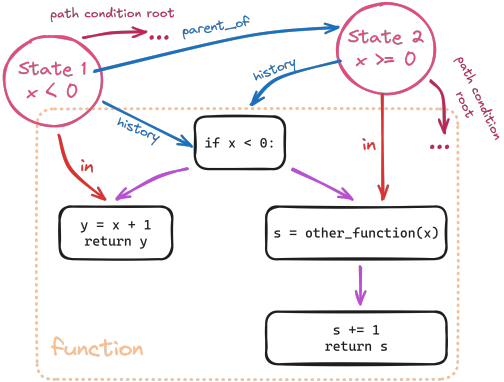
[3] "FeatMaker: Automated Feature Engineering for Search Strategy of Symbolic Execution", Jaehan Yoon and Sooyoung Cha, 2024

PySymGym

- Инфраструктура для обучения графовых нейронных сетей выбору путей при символьном исполнении
- Два этапа обучения
  - Обучение с учителем на имеющемся наборе данных
  - Обновление набора данных
- Входные данные — графовое представление состояния символьной машины

Символьное исполнение

```
def function(x: int):
    if x < 0:
        y = x + 1
        return y
    else:
        s = other_function(x)
        s += 1
        return s
```



Постановка задачи

Целью работы является создание графовой нейронной сети, предсказывающей пути при символьном исполнении

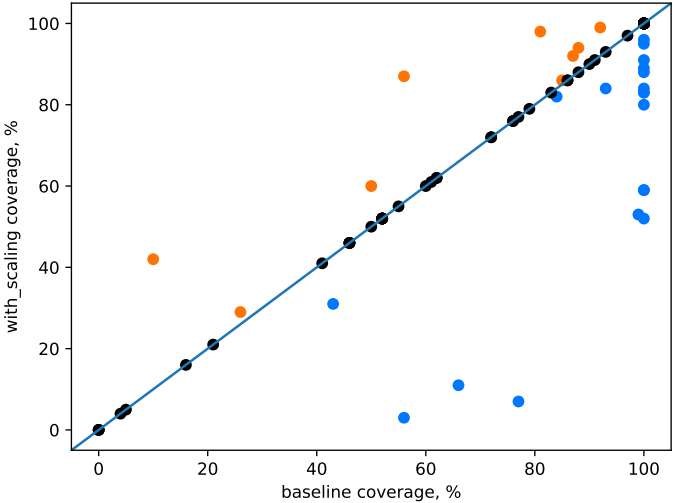
Задачи:

- Интегрировать MLFlow в проект
- Провести эксперименты с нормализацией атрибутов вершин
- Провести эксперименты с обучением с учителем в рамках решения задачи классификации состояний
- Реализовать поддержку path condition
- Провести эксперименты с path condition

Интеграция MLFlow

- MLFlow — это инструмент для управления экспериментами
- В рамках его интеграции добавлена поддержка логирования
  - моделей
  - гипер-параметров запуска
  - метрик для оценки качества модели

Нормализация атрибутов вершин



with_scaling (orange) won: 9, baseline (blue) won: 19, eq (black): 138

Атрибуты с неопределенным диапазоном значений: размер базового блока

- На некоторых методах с нормализацией покрытие больше
- Без нормализации лучше на большем числе методов, чем с ней

Классификация состояний

Без классификации состояний

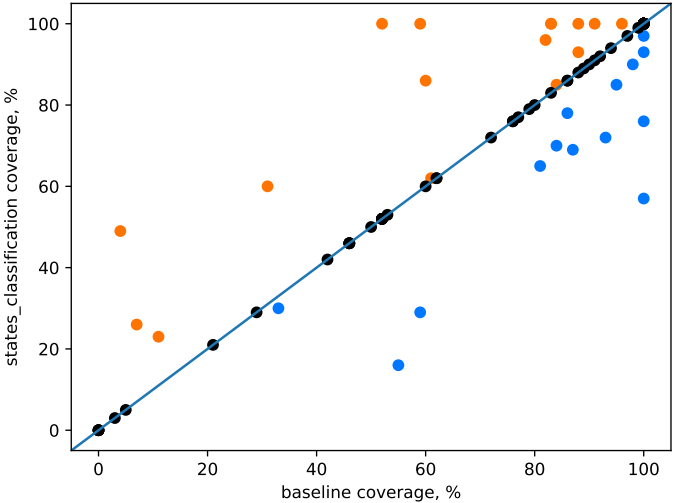
- У каждого состояния есть некоторая вероятность быть выбранным

С классификацией состояний

- Каждое состояние принадлежит одному из двух классов

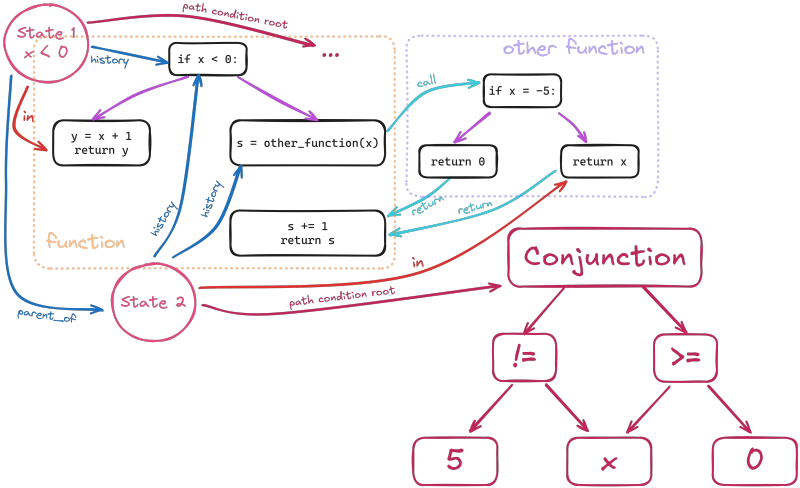
Вывод

- Результаты почти одинаковые

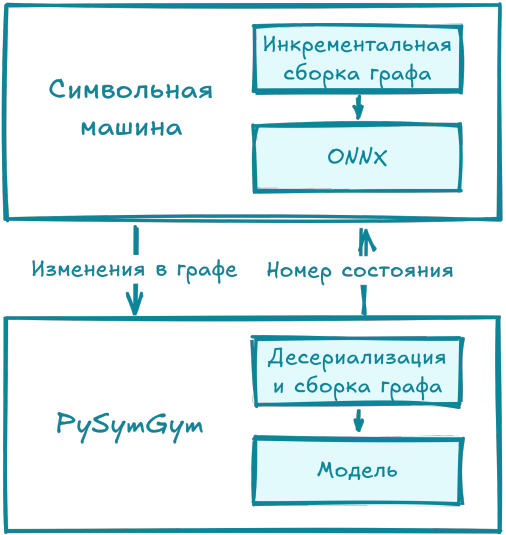


states_classification (orange) won: 16, baseline (blue) won: 14, eq (black): 139

Path Condition

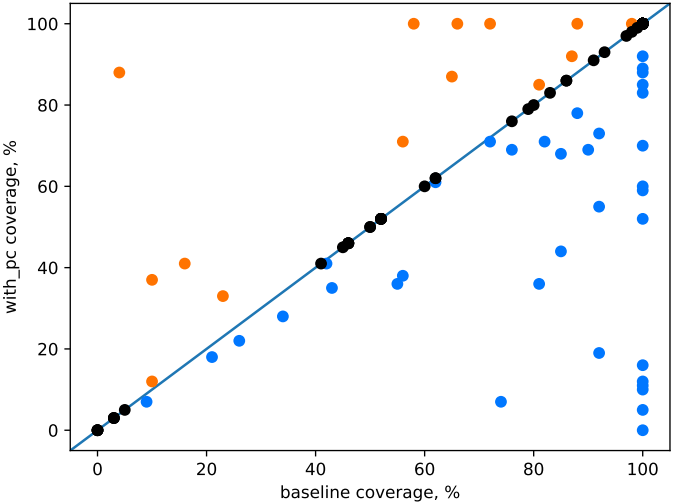


Поддержка Path Condition



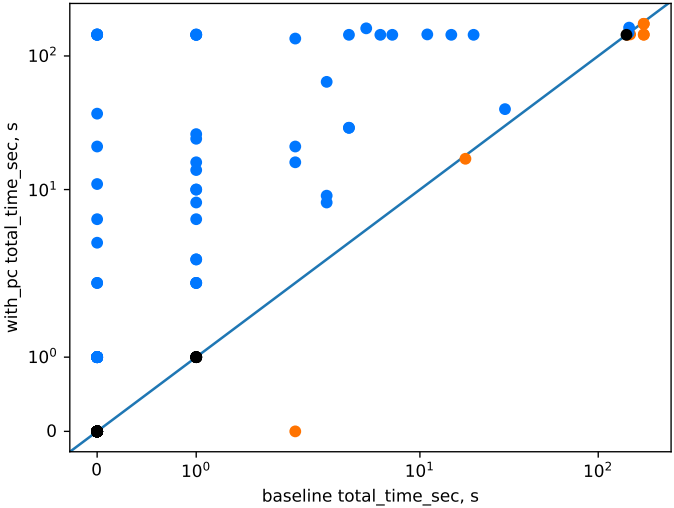
Эксперименты с Path Condition

- Path Condition не улучшил работу модели



with_pc (orange) won: 14, baseline (blue) won: 36, eq (black): 133

Анализ результатов с Path Condition



with_pc (orange) won: 11, baseline (blue) won: 67, eq (black): 55

- Модель дольше работает (в 98% случаев path condition занимает бОльшую часть графа)
- Среднее покрытие в наборе данных с path condition 82%, а без — 86,2%
- Нужен детальный подбор архитектуры

Результаты

- Интегрирован MLFlow[4]
- Проведены эксперименты с нормализацией атрибутов вершин
  - Результат с нормализацией хуже, чем без нее
  - Функция потерь не согласуется с оптимизацией тестового покрытия
- Проведены эксперименты с обучением с учителем в рамках решения задачи классификации состояний
  - Результаты сравнимы с предыдущим подходом
- Реализована поддержка path condition[5]
- Проведены эксперименты с path condition
  - Результаты оказались не лучше
  - Требуются дальнейшие исследования архитектуры и улучшение набора данных

[4] `https://github.com/PySymGym/PySymGym` — ссылка на репозиторий проекта (дата доступа: 2025-04-04). Имя пользователя Anya497

[5] `https://github.com/PySymGym/VSharp/tree/mlSearcher` — ссылка на репозиторий проекта (дата доступа: 2025-04-04). Имя пользователя Anya497

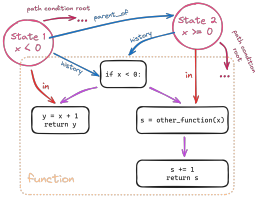
Результаты

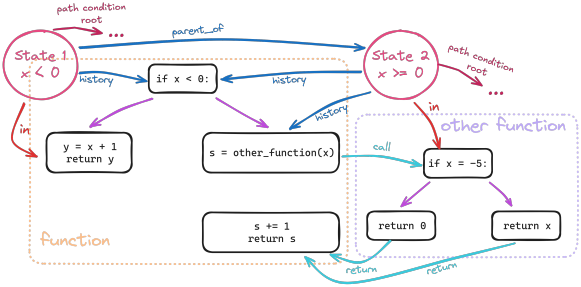
- Проект представлен на постерной сессии на летней школе по оптимизации в Новосибирске. Одержана победа на конкурсе постеров
- Работа была представлена на воркшопе STATIC 2025 при конференции ICSE

Символьное исполнение

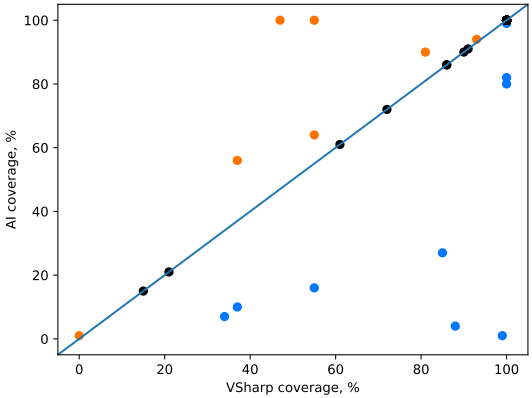
```
def function(x: int):
    if x < 0:
        y = x + 1
        return y
    else:
        s = other_function(x)
        s += 1
        return s

def other_function(x: int):
    if x == -5:
        return 0
    else:
        return x
```

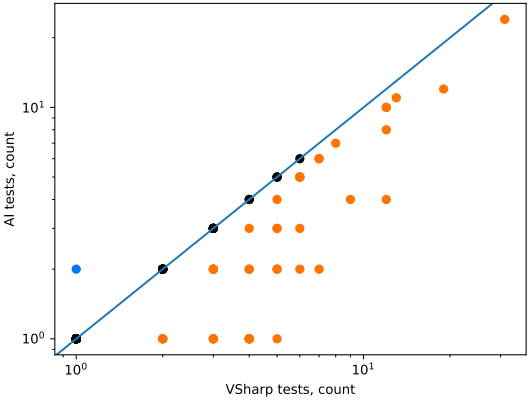


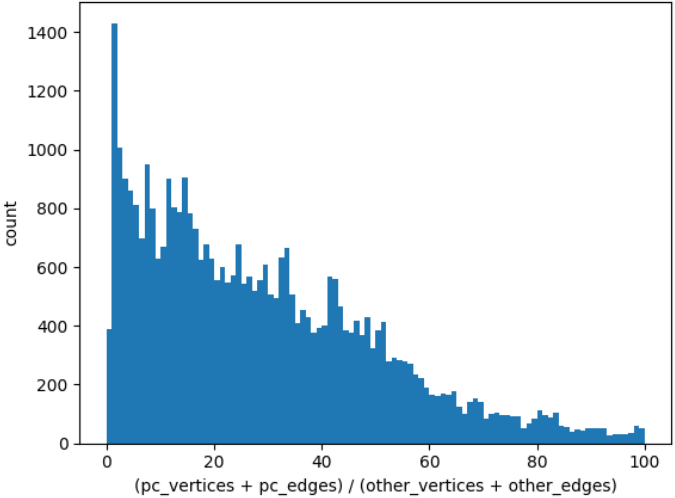


Лучшая модель



AI (оранжевый) лучше: 7, VSharp (синий) лучше: 9,
одинаковый результат (черный): 113



AI (оранжевый) лучше: 43, VSharp (синий) лучше: 1,
одинаковый результат (черный): 69

Соотношение размеров Path Condition и других подграфов