Санкт-Петербургский государственный университет

Кафедра системного программирования

Разработка набора инструментов для обучения искусственных нейронных сетей выбору оптимального пути для символьного исполнения

Максим Владиславович Нигматулин, 22.M07-мм

Научный руководитель: к.ф.-м. н. С.В. Григорьев, доцент кафедры системного программирования

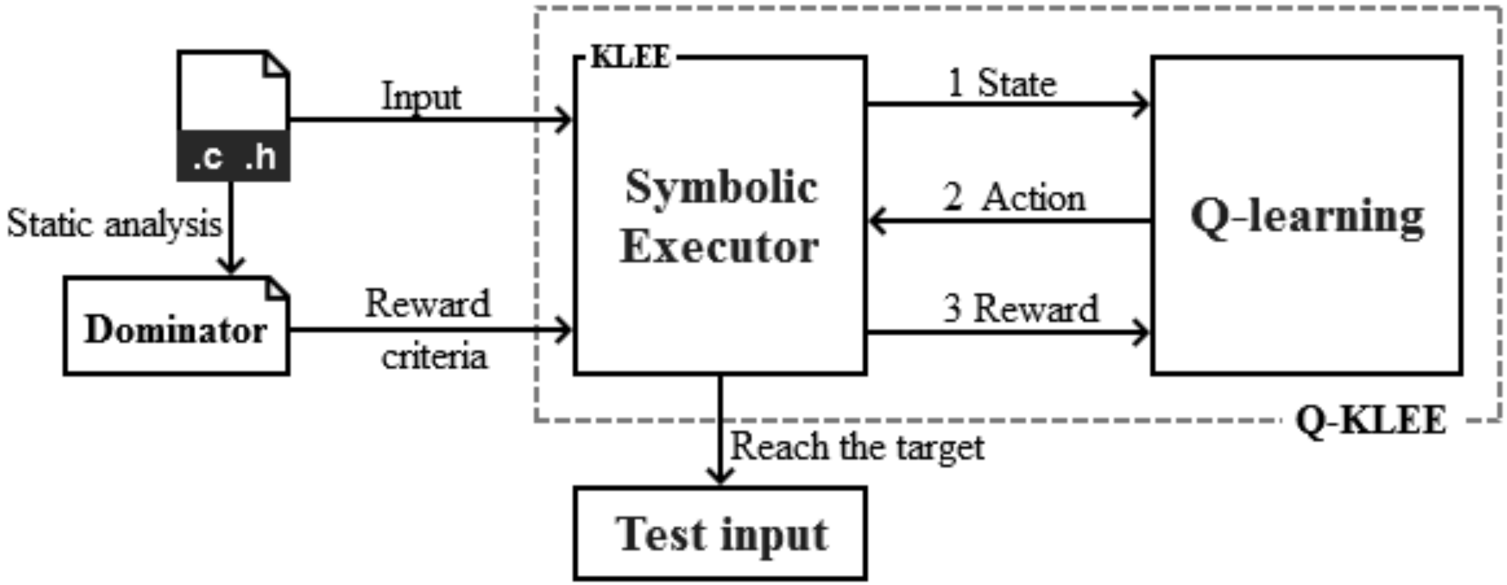
Санкт-Петербург
2023

Введение

- Символьное исполнение — техника анализа ПО, позволяющая понять, какие данные вызывают выполнение каждой части программы
- Одна из проблем — “взрыв” путей, которые нужно исследовать

Существующие решения: Q-KLEE[1]

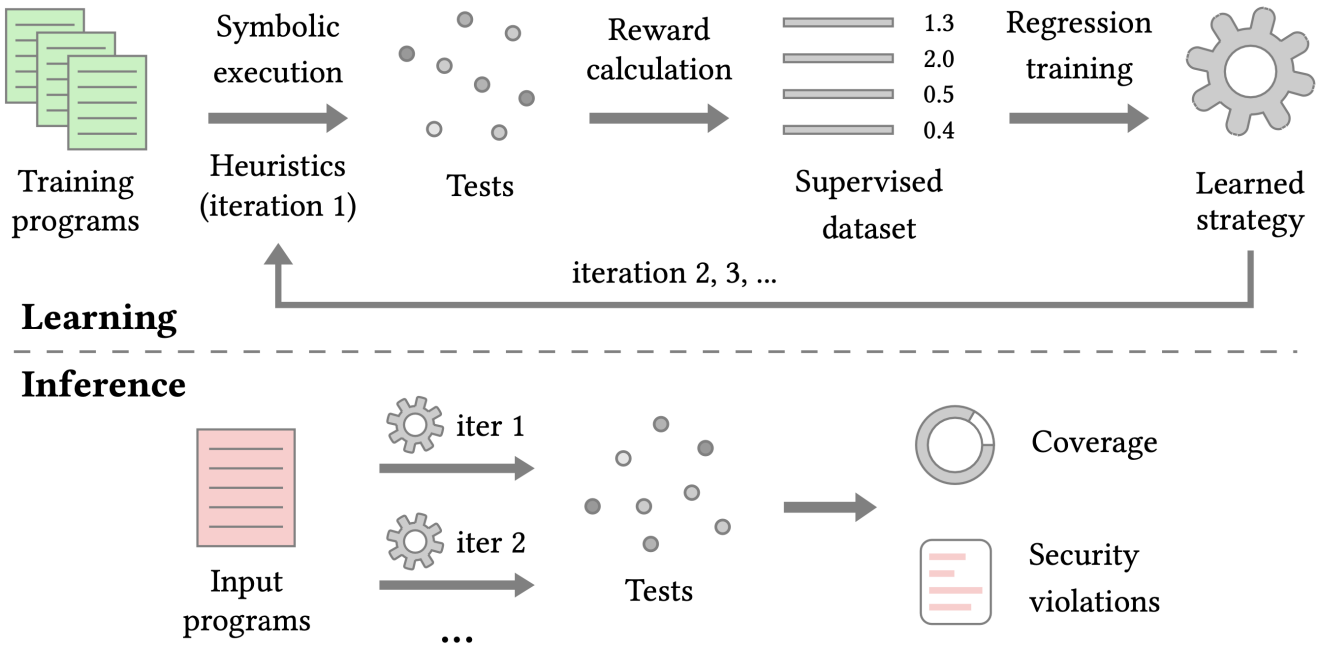
Согласно бенчмаркам, исследует в 10 раз меньше путей, исполняет в 10 раз меньше инструкций за незначительно большее время



[1] J. Wu, C. Zhang and G. Pu, "Reinforcement Learning Guided Symbolic Execution,"2020

Существующие решения: Learch[2]

- KLEE as symbolic execution engine
- Возможность обучать свои модели
- Возможность генерировать датасет на своих данных



[2] Jingxuan He, Gishor Sivanrupan, Petar Tsankov, and Martin Vechev, "Learning to Explore Paths for Symbolic Execution"

Существующие решения: Automatic Heuristics Learning[3]

- Генерация эвристических алгоритмов автоматически
- Результаты выше, чем у алгоритмов, придуманных людьми

[3] Enhancing Dynamic Symbolic Execution by Automatically Learning Search Heuristics, Sooyoung Cha, Seongjoon Hong, Jiseong Bak et al.

Существующие решения

- Q-KLEE — можно улучшить
- Learch — не позвояет работать с GNN
- Automatic Heuristics Learning — подобранные эвристики ограничены правилами, которые придумывают люди

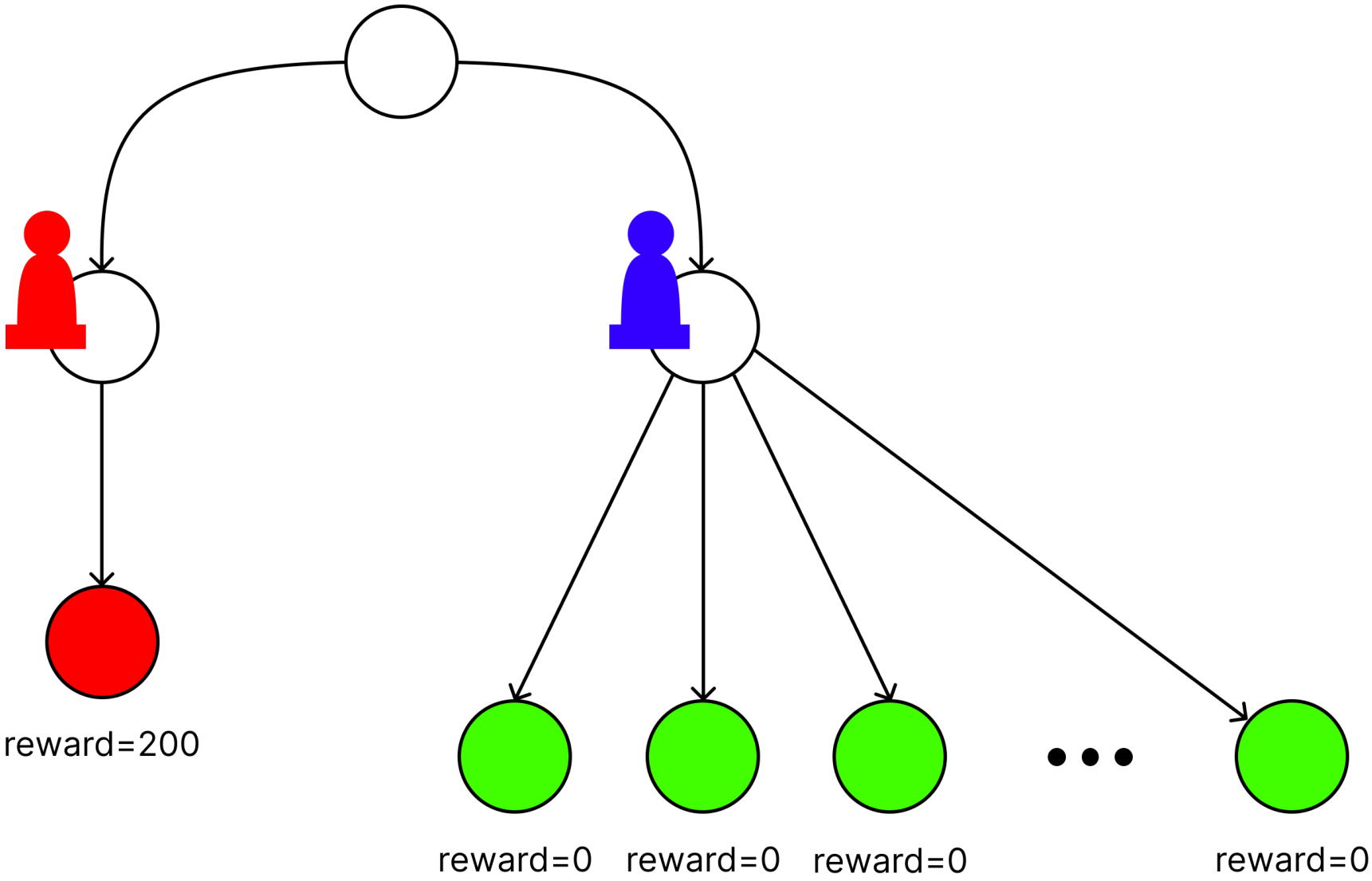
Цель работы: реализовать фреймворк, выполняющий генерацию моделей-учителей в ходе обучения с помощью символьной машины V#.

Поставленные задачи:

1. Создать протокол общения с сервером обучения для получения сигнала об окончании взаимодействия, информации о награде за шаг и состоянии символьного исполнения во время обучения;
2. Создать фреймворк, использующий генетическое обучение для создания и обучения нейронных сетей во время взаимодействия с сервером обучения как с игровой средой;
3. Поддержать возможность одновременного обучения нескольких нейронных сетей;
4. Поддержать возможность использования GPU для ускорения работы нейронных сетей.

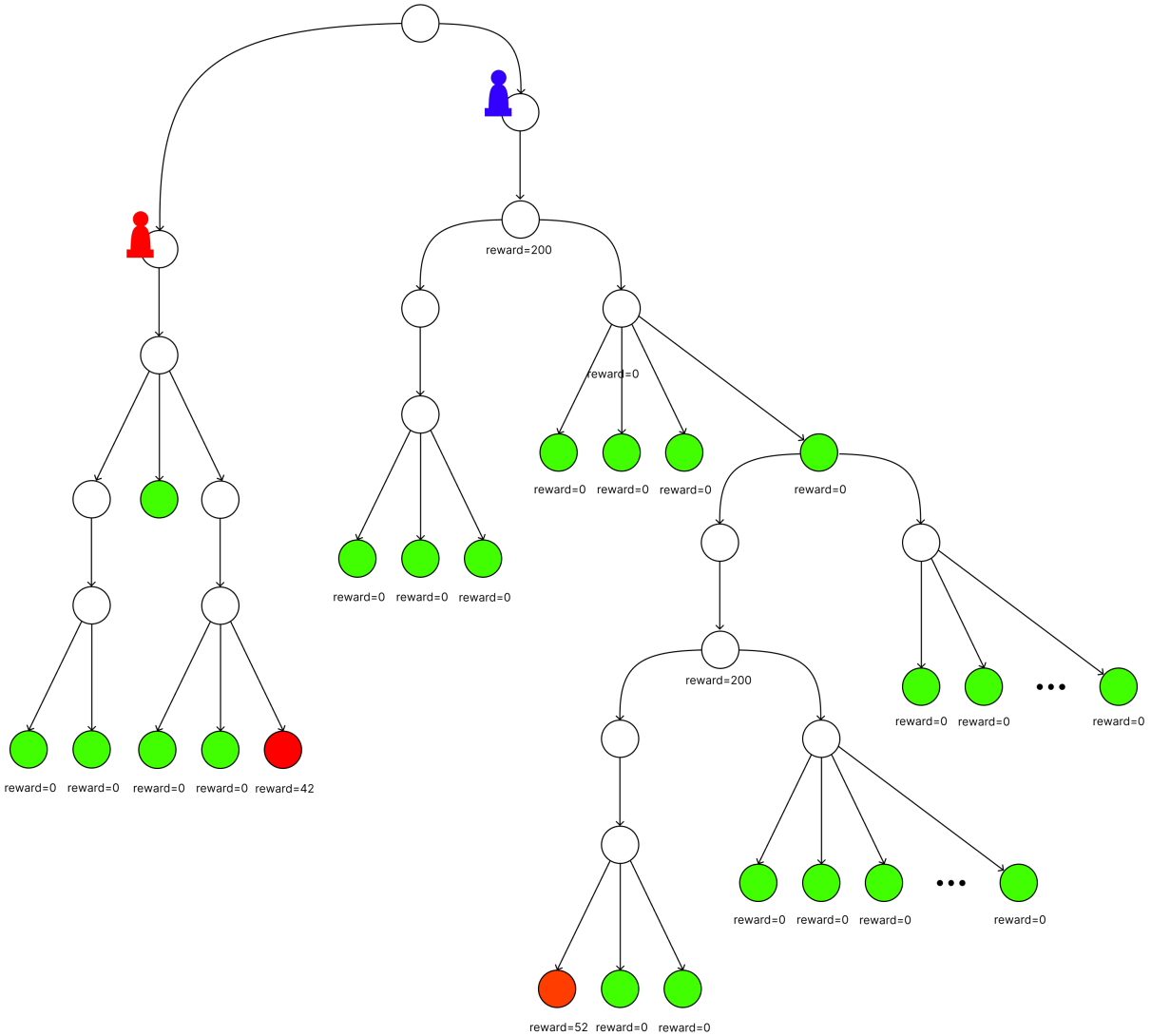
Игровая аналогия

Какую фишку подвинуть?

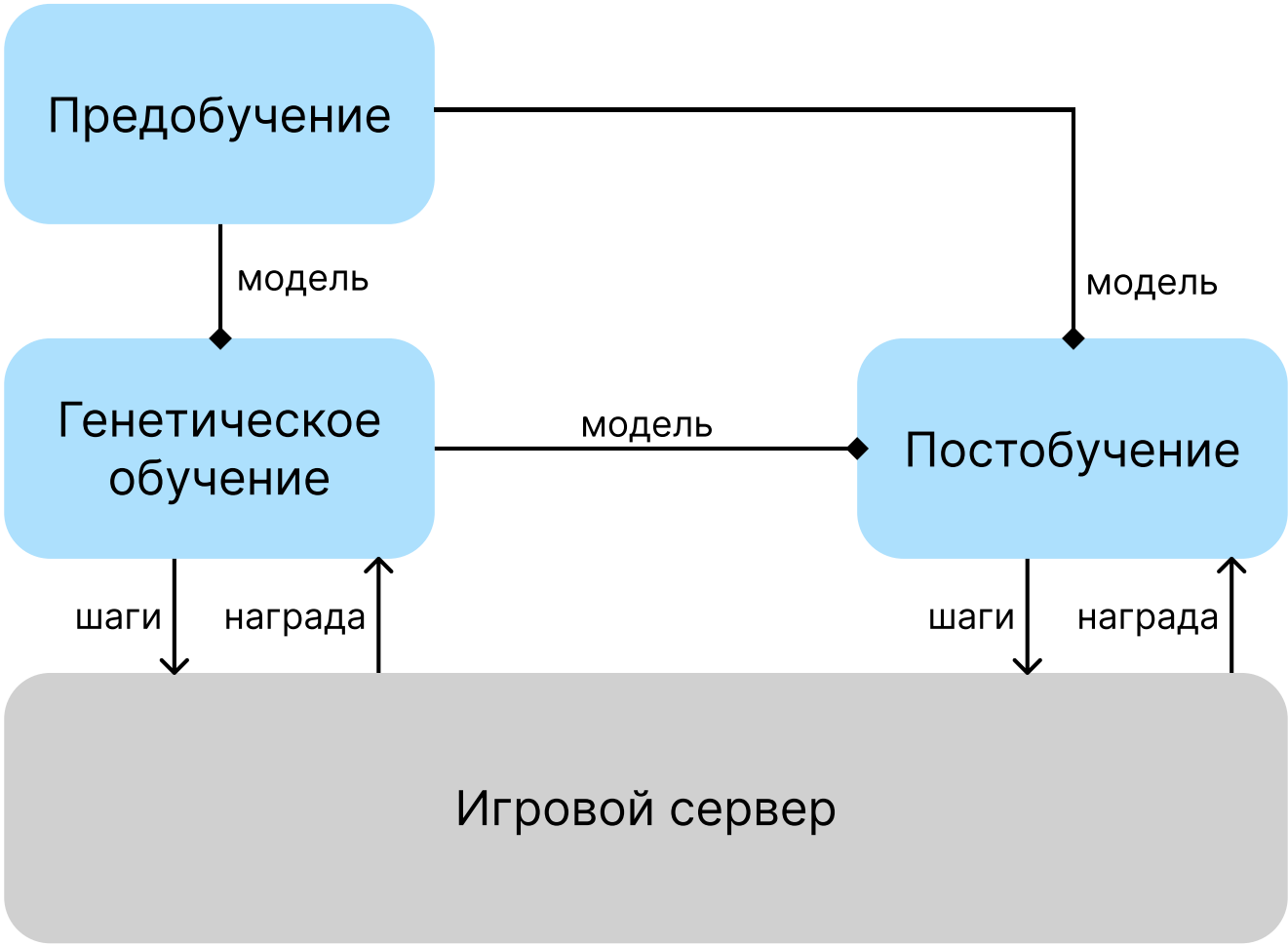


Игровая аналогия

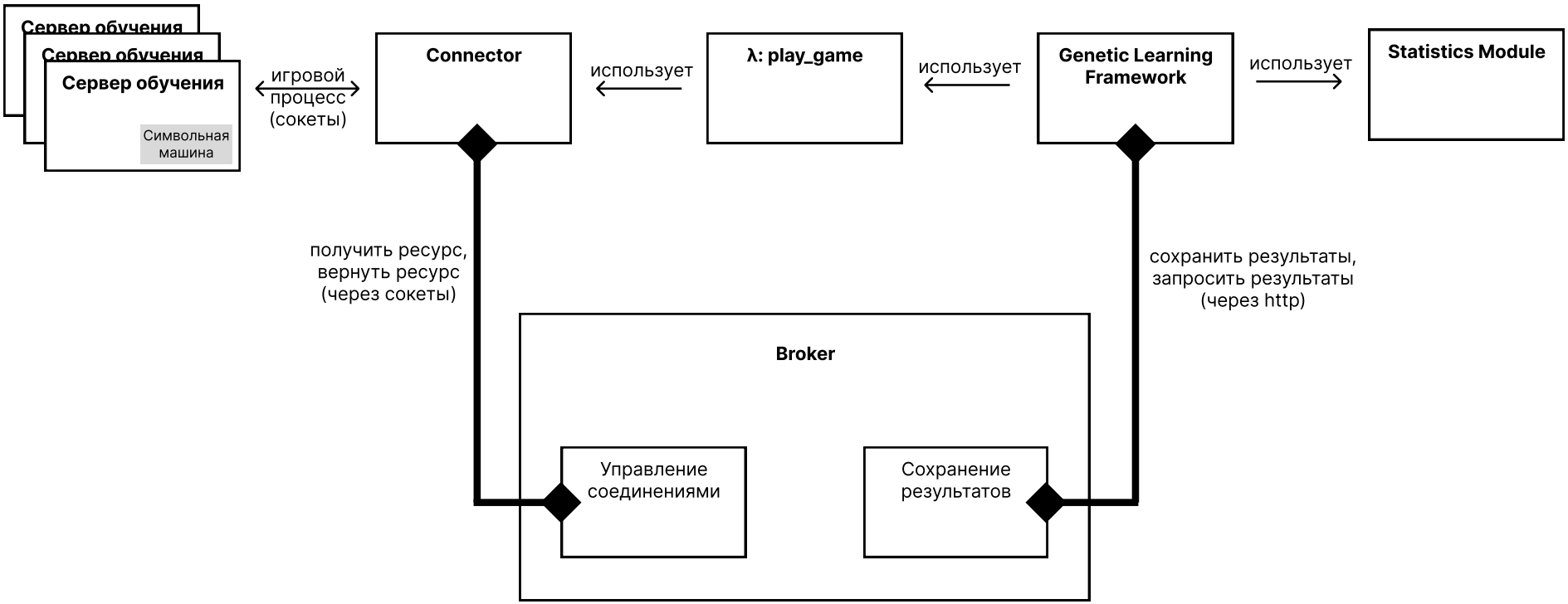
Какую фишку подвинуть?



Структура решения



Генетическое обучение: архитектура



Параллелизм: GPU

- Передача тензоров, инференс на GPU
- Ускорение обучение в 2-3 раза

Benchmark Results

	Среднее покрытие (больше - лучше)	Среднее количество шагов символьной машины (меньше - лучше)	Среднее количество сгенерированных тестов (меньше - лучше)	Среднее количество сгенерированных ошибок (больше - лучше)
V# heuristic[4]	**75.7 ± 33**	2889.18 ± 2193.42	3.64 ± 4.98	**9.95 ± 26.6**
GNN	65.95 ± 44	**1408.75 ± 1982.65**	**2.31 ± 2.59**	2.89 ± 4.11

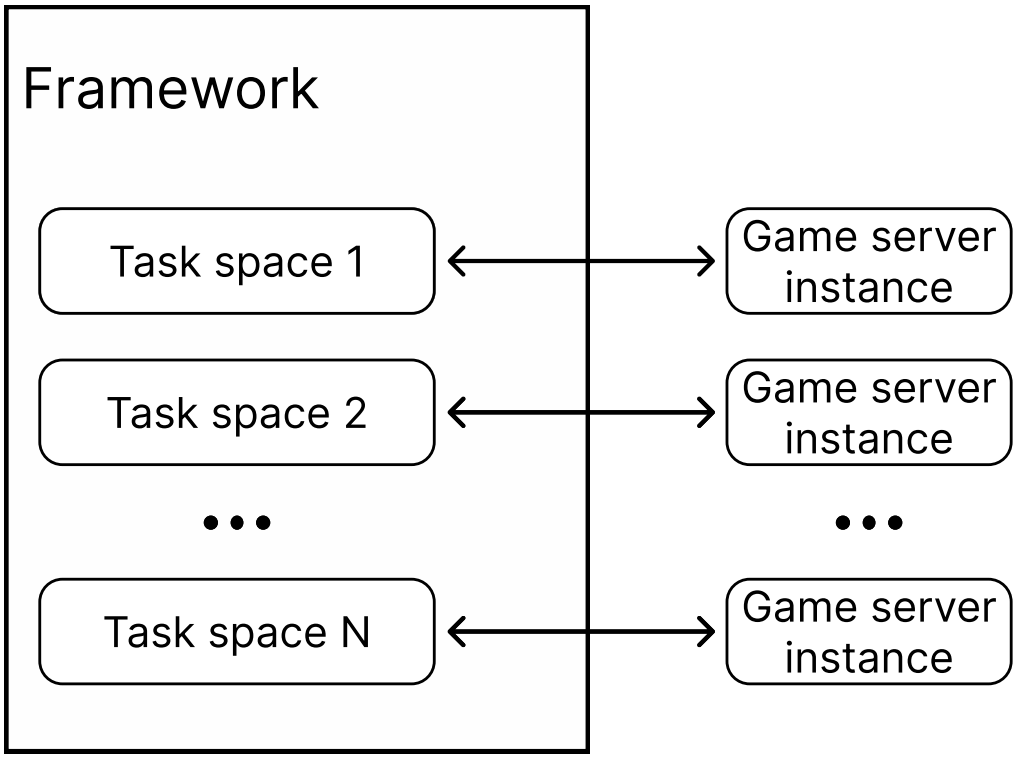
V# integration tests + BizHawk + Powershell + Unity + CosmosOS + JetBrainsLifetimes / 5k шагов / без лимита времени

Результаты

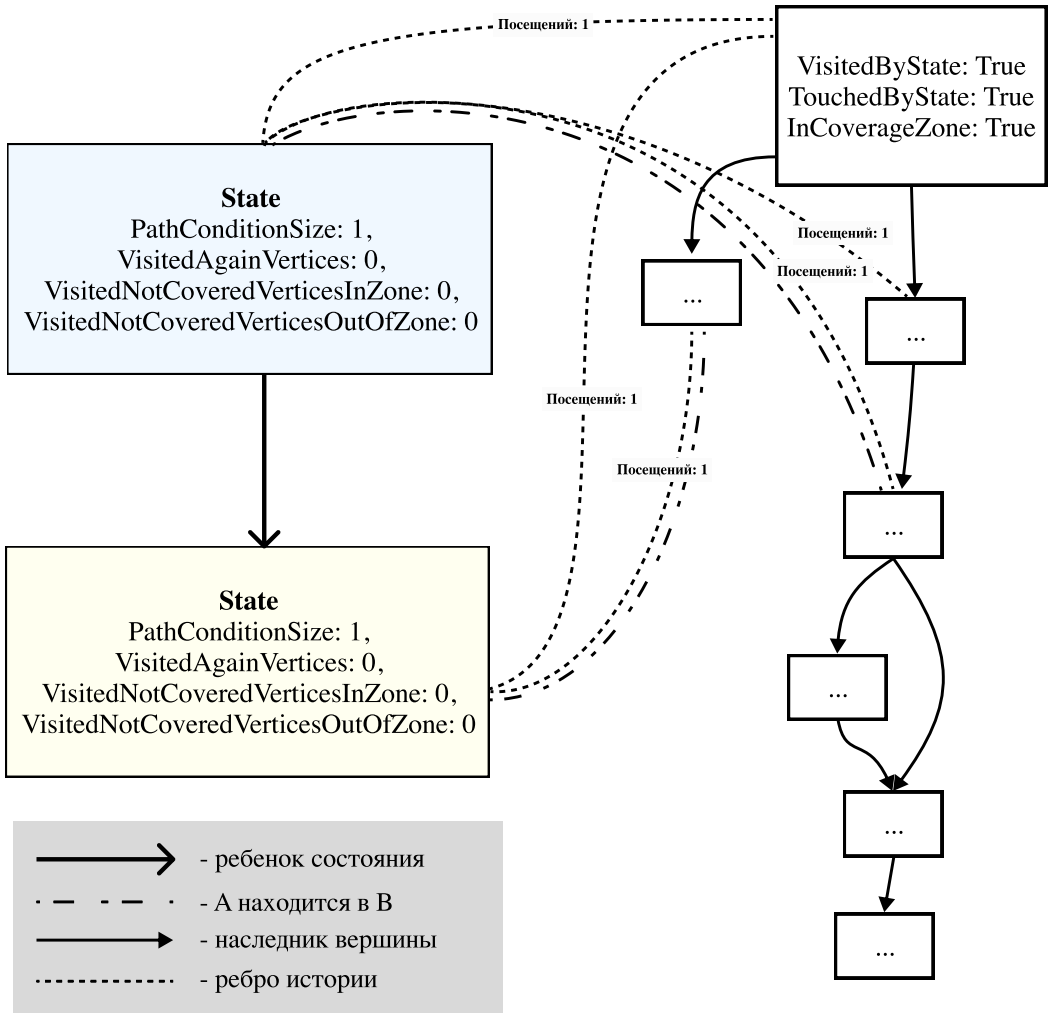
- Создан протокол общения с сервером обучения для получения сигнала об окончании взаимодействия, информации о награде за шаг и состоянии символьного исполнения во время обучения;
- Создан фреймворк, использующий генетическое обучение для создания и обучения нейронных сетей во время взаимодействия с сервером обучения как с игровой средой;
- Поддержана возможность одновременного обучения нескольких нейронных сетей;
- Поддержана возможность использования GPU для ускорения работы нейронных сетей.

Параллелизм: CPU

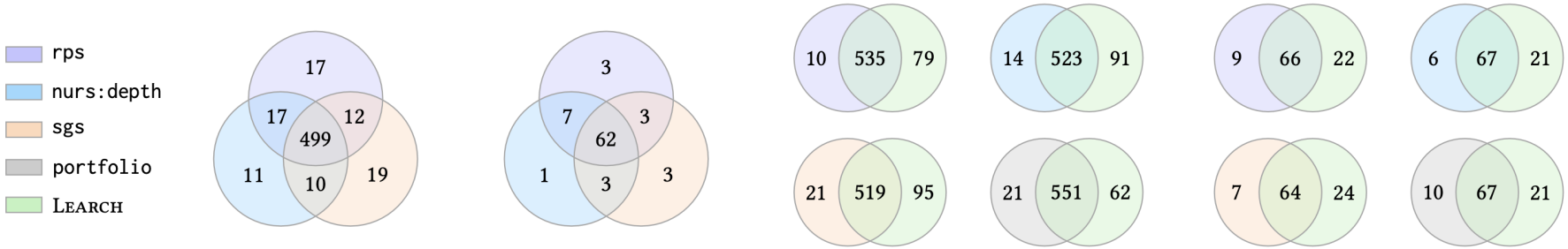
- “Игры” в одном поколении передаются в пул потоков для обработки
- Взаимодействие с несколькими игровыми серверами
- Работает быстрее



Архитектура нейронной сети



ML > эвристики[4]



(a) Average line coverage by heuristics.

(b) UBSan violations found by heuristics.

(c) Average line coverage: Learch v.s. heuristics.

(d) UBSan violations: Learch v.s. heuristics.

Figure 3: Limitations of existing manually designed heuristics and how Learch outperforms them for our `coreutils` test set.

[4] Jingxuan He, Gishor Sivanrupan, Petar Tsankov, and Martin Vechev, "Learning to Explore Paths for Symbolic Execution"