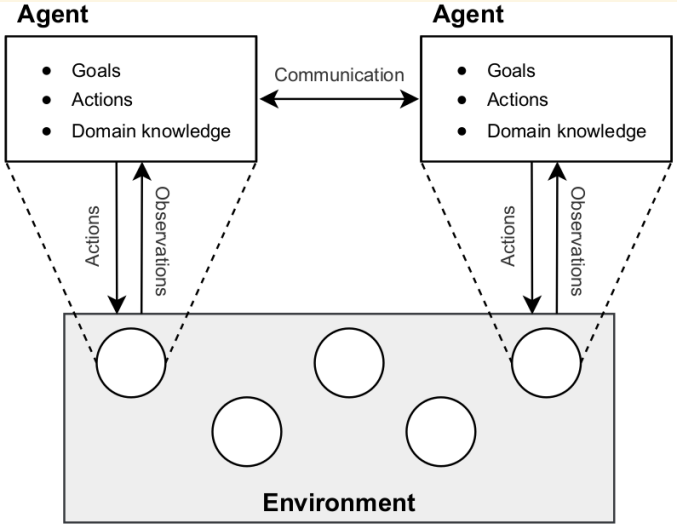
Применение техник обучения с подкреплением по учебной программе и самостоятельной игры для конкурентных сред в казахских национальных играх

June 9, 2024

Динара Жусупова
Руководитель:Елена Кантонистова
Машинное обучение и высоконагруженные системы, НИУ Высшая школа экономики

Мультиагентная среда



Алгоритмы обучения с подкреплением могут обучать агентов, которые решают проблемы в сложных, интересных средах.

RL

RL – это эффективный инструмент, который помогает системам искусственного интеллекта (ИИ) достигать оптимальных результатов в полно/частично наблюдаемых средах.

MARL

Мультиагентная среда или система состоит из среды и множества агентов, принимающих решения, которые взаимодействуют в среде для достижения определенных целей.

Self-play (самостоятельная игра)

В конкурентной многоагентной среде агенты обучаются с помощью техники self-play.

Обучение агентов в мультиагентных средах

В мультиагентном обучении с подкреплением существуют различные проблемы:

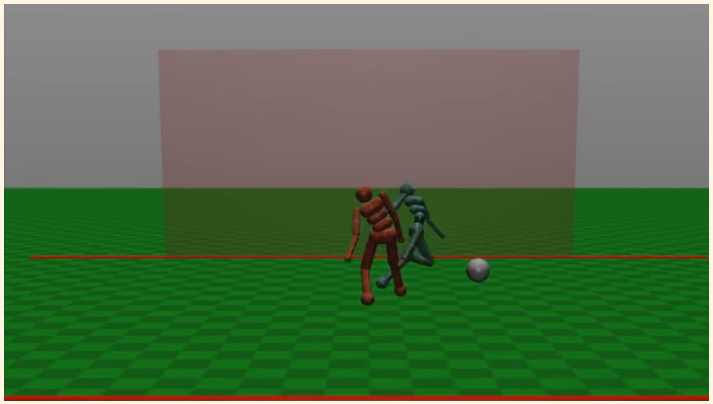
- агенты владеют различными частичными знаниями о состоянии среды
- перед агентами в среде могут стоять различные задачи
- агенты параллельно исследуют среду и обучаются

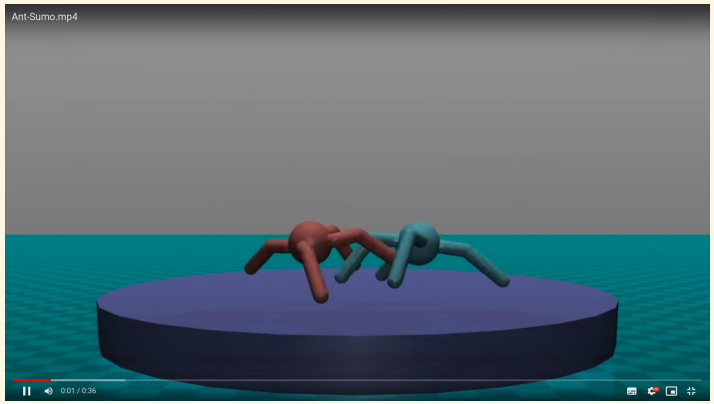
Важной особенностью MARL является нестационарность, вызванная постоянно меняющейся политикой агентов в процессе их обучения

Обучение агентов в мультиагентных средах

> T. Bansal, UM. Amherst, J. Pachocki, S. Sidor, I. Sutskever, I. Mordatch. EMERGENT COMPLEXITY VIA MULTI-AGENT COMPETITION https://arxiv.org/abs/1710.03748

В этой статье представлены несколько конкурентных мультиагентных сред, в которых агенты соревнуются в трехмерном мире с симулированной физикой. Обученные агенты изучают широкий спектр сложных и интересных навыков, хотя сама среда относительно проста. Авторы рассматривают два трехмерных тела агента: муравья и гуманоида. Муравей представляет собой четвероногое тело с 12 степенями свободы и 8 приводящими в движение суставами. Гуманоид имеет 23 степени свободы и 17 приводимых в действие суставов.

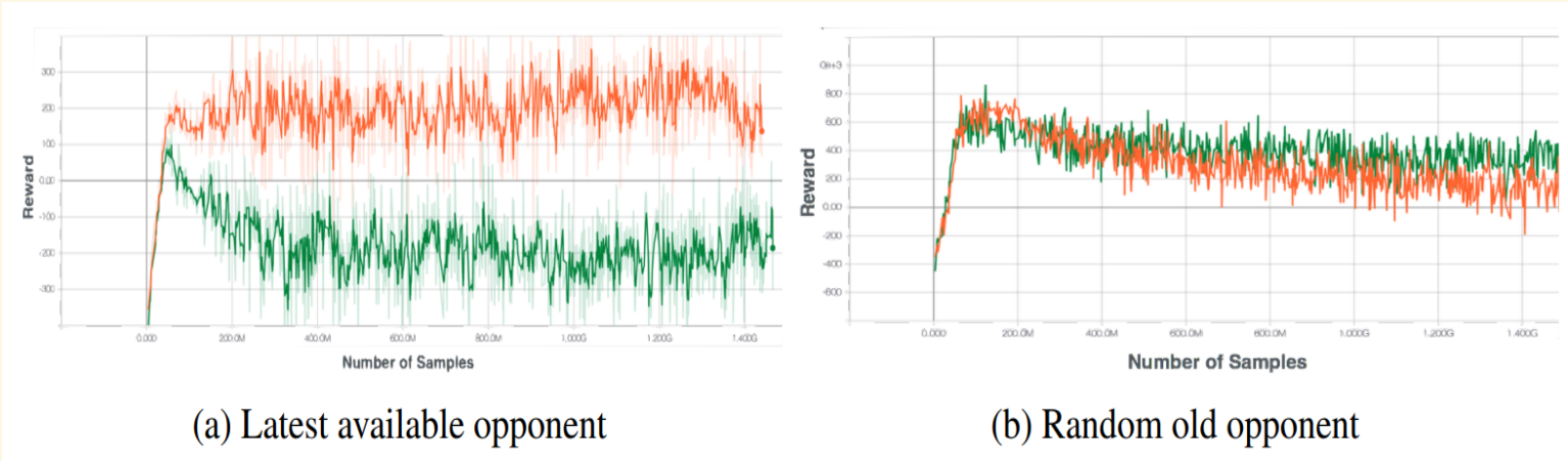




Обучение конкурентных агентов. Сэмплирование оппонентов

Навыки противников, с которыми сталкиваются во время обучения, могут оказать существенное влияние на обучение агентов.

Обучение агентов против самого последнего противника приводит к дисбалансу в обучении, когда один агент становится более опытным, чем другой агент, на ранних этапах обучения, а другой агент не может восстановиться.



(a) Latest available opponent (b) Random old opponent

Метод Q-Learning

$$\mathrm{Q}_\pi(\mathrm{s},\mathrm{a}) = \mathbb{E}_\pi[\mathrm{G_t}|\mathrm{S_t}=\mathrm{s},\mathrm{A_t}=\mathrm{a}] = \mathbb{E}_\pi\left[\sum_{\mathrm{k}=0}^{\infty}\gamma^{\mathrm{k}}\mathrm{R_{t+k+1}}|\mathrm{S_t}=\mathrm{s},\mathrm{A_t}=\mathrm{a}\right].$$

$$\mathrm{Q}(\mathrm{S_t},\mathrm{A_t}) \longleftarrow \mathrm{Q}(\mathrm{S_t},\mathrm{A_t}) + \alpha\left[\mathrm{R_{t+1}} + \gamma\max_{\mathrm{a}}\mathrm{Q}(\mathrm{S_{t+1}},\mathrm{a}) - \mathrm{Q}(\mathrm{S_t},\mathrm{A_t})\right].$$

Алгоритм 1 Q-Learning

Параметры алгоритма $\alpha \in [0,1]$ и $\varepsilon > 0$ малое число
Инициализировать $Q(s,a)$ для всех a и s произвольно, за исключением того, что $Q(terminal,\cdot) = 0$
for episode **do**
Выбрать A из S, используя политику, полученную из Q (например, «ε-жадно»)
Выполнить выбранное на предыдущем шаге действие A и получить наблюдение S' и вознаграждение R
Обновить Q-значение:

$$Q(S,A) = Q(S,A) + \alpha\left[R + \gamma\max_a Q(S',A) - Q(S,A)\right]$$

Повторить, пока не будет достигнуто терминальное состояние
end for

Метод DQN Deep Q-network

Deep Q-Learning использует глубокую нейронную сеть для аппроксимации различных значений Q(S, A) для каждого возможного действия A в состоянии S (оценка функции значения)

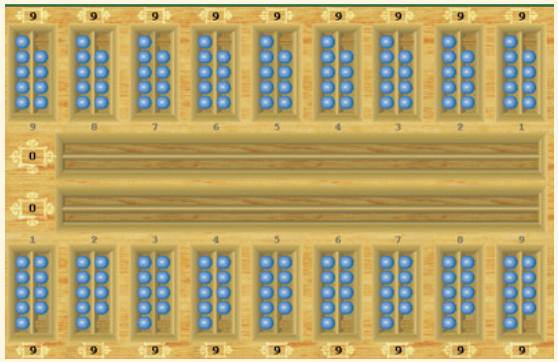
Алгоритм 2 DQN

Инициализировать размер Replay Buffer
Инициализировать функцию значения действия Q рандомными весами θ
Инициализировать целевую функцию $\hat{Q}$ с весами $\theta^- = \theta$
for episode **do** Инициализировать последовательность $s_1 = x_1$ и последовательность препроцессинга $\phi_1 = \phi(s_1)$
  for t **do** Выбрать действие a_t с вероятностью ε или с помощью формулы $a_t = argmax_a Q(\phi(s_t), a; \theta)$
    Выполнить действие a_t и получить награду и следующее состояние
    Присвоить $s_{t+1} = s_t$, a_t и подготовить препроцессинг $\phi_{t+1} = \phi(s_{t+1})$
    Сохранить в Replay Buffer D кортеж $(\phi_t, a_t, r_t, \phi_{t+1})$
    Сэмплировать минибатч с кортежами $(\phi_t, a_t, r_t, \phi_{t+1})$ из D
    Присвоить

$$y_j = \begin{cases} r_j & \text{если эпизод заканчивается на шаге } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a_j; \theta^-\right) & \text{иначе} \end{cases}$$

    Реализовать шаг градиентного спуска для $\left(y_j - Q\left(\phi_{j+1}, a_j; \theta\right)\right)^2$ квадратичного отклонения относительно параметра θ
    Через каждые C шагов сбрасывать $\hat{Q} = Q$
  end for
end for

Об игре «Тогызкумалак» (каз. Тоғызқұмалақ)



Описание игры

Казахская национальная логическая настольная игра на доске, в которой 18 игровых и две накопительные лунки. В 2020 году ЮНЕСКО включило эту игру в репрезентативный список нематериального наследия. Доска для игры в тогыз кумалак имеет по 9 игровых лунок для каждого игрока. Кроме этого, в середине доски располагаются две большие лунки для сбора выигранных камней. Количество камней - 162 штуки. Для обозначения лунки-туздыка используются специальные знаки, при их отсутствии могут использоваться два камня отличающихся по форме или цвету от игровых камней.

- **Количество игроков:** 2
- **Исходная позиция:** Перед началом игры в каждую игровую лунку раскладывают по 9 камней. Накопительная лунка пуста. Каждому игроку принадлежит ближний к нему ряд из 9-ти лунок (уй – дом) и одна накопительная лунка (казан – котёл), располагающаяся ближе к игроку или по правую руку.

Основные правила игры

Ходы

Ходы делают по очереди. Право первого хода взаимно оговаривается или разыгрывается жребием, начинающий игру садится с белой стороны. Во время своего хода игрок берёт все камни из любой своей лунки «дом» и, начиная с этой же лунки, раскладывает их по одному против часовой стрелки в свои и чужие дома. Если в исходной лунке только один камень, то он перекладывается в следующую лунку.

Выигрыш камней

Если последний кумалак попадает в дом соперника и количество кумалаков в нём становится чётным, то кумалаки из этого дома переходят в казан игрока, совершившего ход.

Туздык

Туздык — выигранная лунка на стороне соперника. Если при ходе игрока А последний кумалак попадает в дом игрока Б и в нём после этого оказывается три кумалака, то этот дом объявляется туздыком игрока А (каз. туздык уй). Эти три кумалака попадают в казан игрока А. В последующем каждый кумалак, попавший в туздык, переходит в казан игрока А. Существует 3 основных правила взятия туздыка:

1. игрок не может завести себе туздык в самом последнем (девятом) доме соперника,
2. игрок не может завести себе туздык в доме с таким же порядковым номером, который имеет лунка-туздык соперника,
3. каждый игрок в течение игры может завести себе только один туздык.

Отказаться от туздыка или изменить его положение нельзя, он действует до конца игры.

Стек

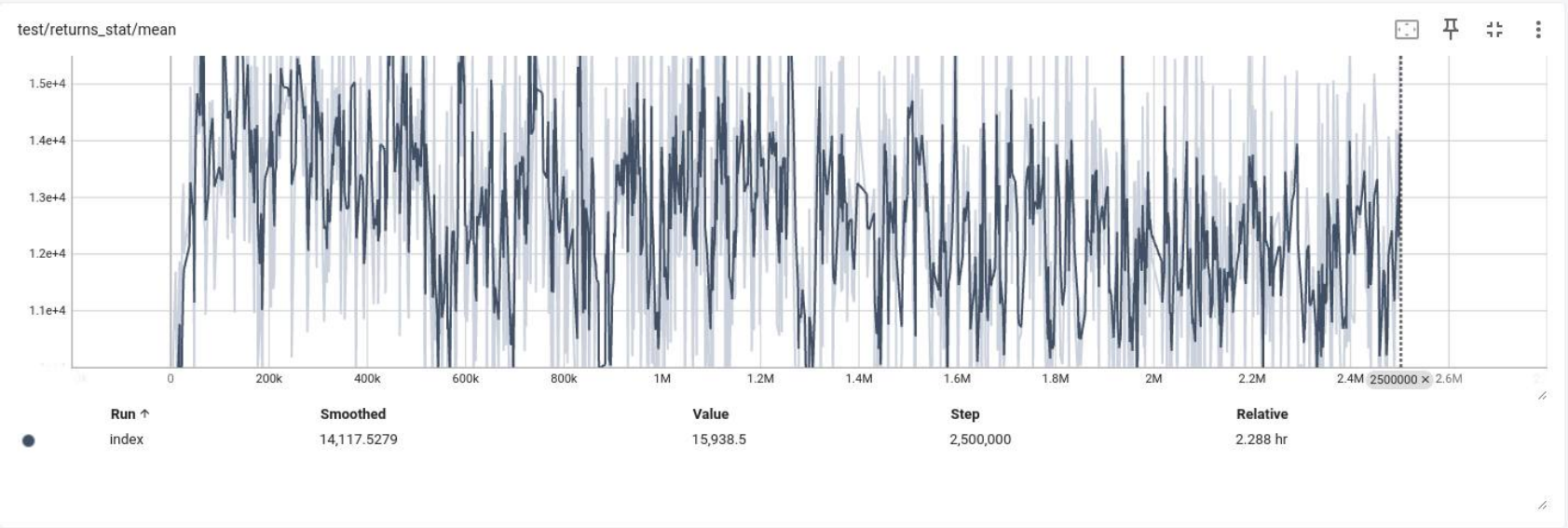
- **Для создания мультиагентной среды:** PettingZoo

- **Для обучения конкурентных агентов:** Tianshou

- **Для создания веб приложения игры:** Dash

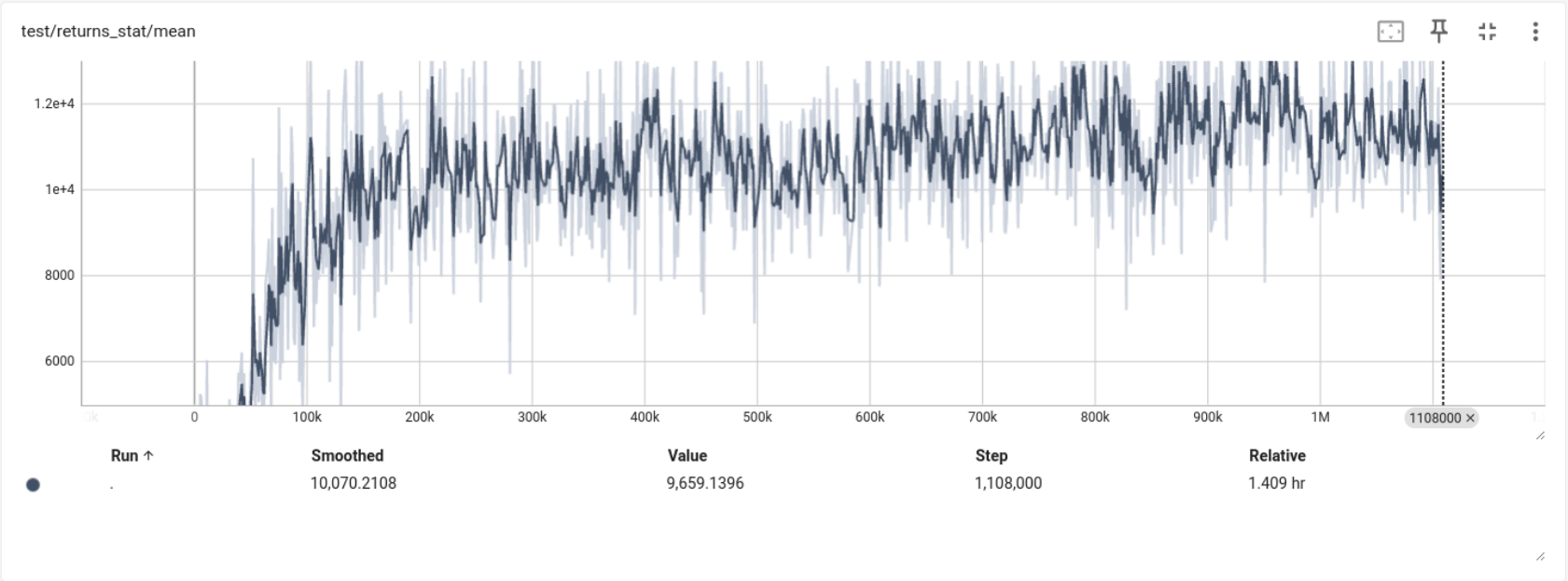
Начальные эксперименты с обычным вознаграждением

Обучение игрока Бастаушы (неудачная попытка)



Эксперименты: подход zero-sum (игра Тогызкумалак как антагонистическая)

Обучение игрока Бастаушы



Детали обучения:

- алгоритм - Double Deep Q-Network
- количество эпох - 3 000
- размер батча: 256
- обновление таргета - 352
- количество шагов в 1 эпохе - 1000
- количество эпизодов при тестировании - 50
- lr = 3e-04
- размер ReplayBuffer - 100 000

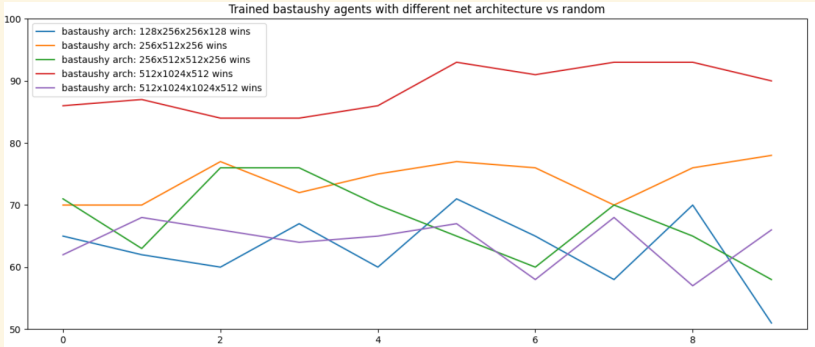
После обучения игрок побеждает рандомную политику в 95–98% случаях.

Выбор архитектуры DQN

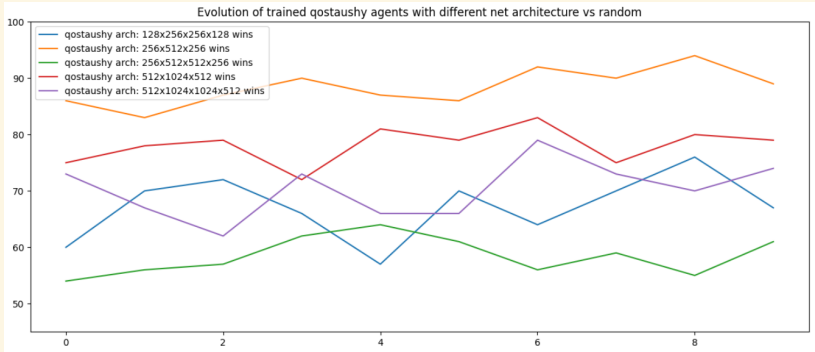
Потенциальные MLP архитектуры NET_ARCHS = [[128, 256, 256, 128], [256, 512, 256], [256, 512, 512, 256], [512, 1024, 512], [512, 1024, 1024, 512]]. Исследование и последующий выбор осуществлялся при помощи обучения нейронных сетей с рандомными политиками в роли противника.
Детали обучения:

- алгоритм - Double Deep Q-Network
- количество эпох - 800
- размер батча - 256
- обновление таргета - 352
- количество шагов в 1 эпохе - 1000
- количество эпизодов при тестировании - 50
- lr = 3e-04
- размер ReplayBuffer - 100 000

Результаты обучения игрока Бастаушы



Результаты обучения игрока Костаушы



Дизайн вознаграждения

Для улучшения результатов в системе вознаграждения были введены следующие изменения:

zero-sum

согласно правилам антагонистической игры, при выигрыше камней одним игроком, у противника соответственно уменьшалась награда на такое же количество камней

wins reward

награда за победу присваивалась в размере 500 камней

tuzdyq

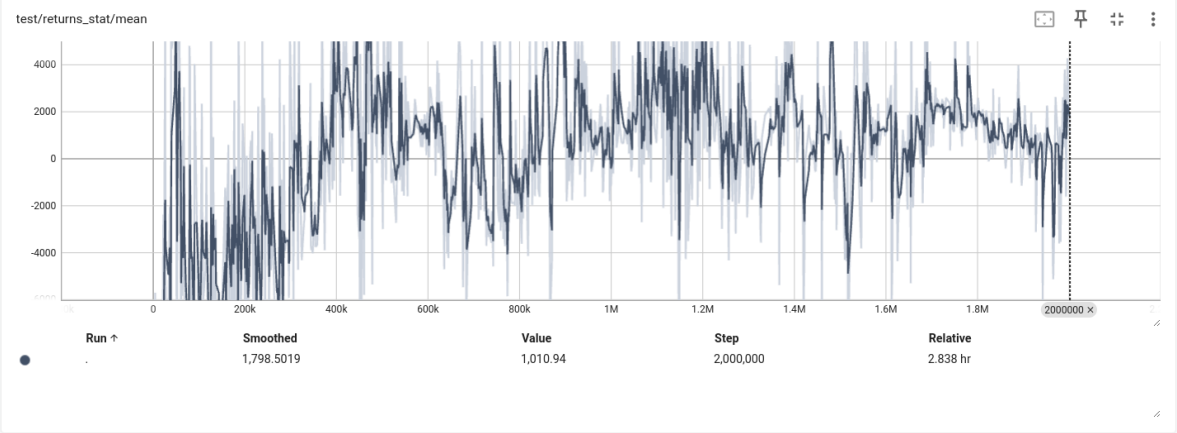
дополнительная награда за завоевание Туздыка была равна 50 камням.

Выигрыши партий при смене ролей

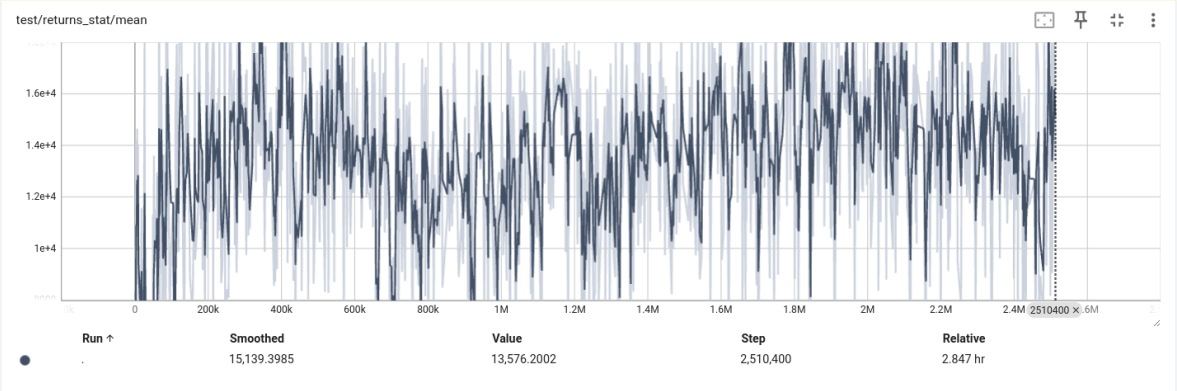
Агент	128x256x256x128	256x512x256	256x512x512x256	512x1024x512
Бастаушы	58	55	22	22
Костаушы	50	44	17	42

Self-play

Результаты обучения с обученным оппонентом и $\varepsilon = 0.0$-жадной политики



Результаты обучения с обученным оппонентом и $\varepsilon = 0.5$-жадной политики



Результаты партий с человеком

Агенты	1	2	3	4
Бастаушы/Человек	0:1	0:1	0:1	1:0
Человек/Костаушы	1:0	1:0	0:1	1:0

Веб приложение игры

Проект: https://github.com/zhus-dika/togyz-qumalaq-agent.git

Запуск приложения: python app/app.py