КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеня бакалавра
За спеціальністю
121 Інженерія програмного забезпечення: програмна інженерія
на тему:

ГЕНЕРАЦІЯ СТРАТЕГІЙ ПЕРЕСЛІДУВАННЯ ЗА ДОПОМОГОЮ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Виконав студент 4-го курсу
Іван РАМИК _____
(підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук
Ярослав ЛІНДЕР _____
(підпис)

Засвідчую, що в цій роботі
немає запозичень з праць інших авторів
без відповідних посилань

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри інтелектуальних
програмних систем

«25» травня 2022 р.

Протокол № 10

Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ — 2022

РЕФЕРАТ

Обсяг роботи 26 сторінок, 6 рисунків, 10 джерел посилань. Ключові слова: ЗАДАЧА ПЕРЕСЛІДУВАННЯ, МАРКОВСЬКІ ІГРИ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, НЕЙРОННІ МЕРЕЖІ.

Об'єктом роботи є процес побудови моделей навчання з підкріпленням для вирішення задачі переслідування. Предметом роботи є моделі навчання з підкріпленням, натреновані за допомогою алгоритмів типу суб'єкт-критик, з використанням нейронних мереж для апроксимації функції цінності.

Метою роботи є ілюстрація ефективності моделей навчання з підкріплення для задачі переслідування.

Інструменти розробки: мова програмування Python, бібліотека для навчання з підкріпленням Rllib, бібліотека глибокого навчання TensorFlow, matplotlib.

Результати роботи: виконано загальний огляд поточного стану моделей навчання з підкріпленням та підходів до їх тренування, проаналізовано переваги та недоліки різних алгоритмів, спроектована, побудована та протестована модель навчання з підкріпленням на базі алгоритму A3C для задачі переслідування, проаналізовані результати її роботи.

Сферами застосування таких моделей може бути, перш за все, військова сфера, де можуть виникати подібні задачі різного рівня складності, та будь які інші види діяльності де існують задачі, які можна представити у вигляді задачі переслідування.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

А3С — Asynchronous Advantage Actor-Critic, асинхронний метод «Суб'єкт-Критик»;

RL — Reinforcement learning, навчання з підкріпл;

ВСТУП

Оцінка сучасного стану об'єкта розробки. Задачі переслідування - достатньо відомий клас задач. Здебільшого вони розглядаються у рамках диференційної теорії ігор, проте можливий підхід за якого їх можна представити як марковський процес прийняття рішень для багатьох агентів. Такі системи називаються марковськими іграми. До марковьких ігор можливо застосувати алгоритми навчання з підкріпленням

Актуальність роботи Задачі переслідування широко зустрічаються в рамках військової сфери. Робота слугує підтвердженням ефективності застосування навчання з підкріпленням до таких задач.

Мета роботи полягає в застосуванні моделей навчання з підкріплення для задачі переслідування та визначенні їх ефективності. Для досягнення цієї мети було поставлено наступні завдання:

- Провести загальний огляд сучасних алгоритмів навчання з підкріпленням, та підходів до тренування.
- Натренувати стратегію декількома алгоритмами для вирішення задачі переслідування.
- Протестувати найкращу стратегію. Порівняти її ефективність зі стратегіями на основі евристик.

Об'єкт, методи і засоби розробки Об'єктом розробки є процес проектування та тренування моделей навчання з підкріпленням. Проектуванню моделі передувало створення математичної моделі, а саме марковської гри, задачі переслідування.

Основою середовища стала операційна система Mac OS, розробка відбувалась на мові загального призначення Python 3. Була використана бібліотека для навчання з підкріпленням Rllib, та бібліотека глибокого навчання TensorFlow. Основним інструментом розробки була обрана середа розробки PyCharm.

Можливі сфери застосування. Сферами застосування таких моделей може бути, перш за все, військова сфера, де можуть виникати подібні задачі різного рівня складності, та будь які інші види діяльності де існують задачі, які можна представити у вигляді задачі переслідування.

РОЗДІЛ 1. НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Марковський процес прийняття рішень - це математична модель прийняття рішень. У марковському процесі прийняття рішень існують стани, доступні дії, ймовірності переходу між станами та винагороди. У кожен момент часу агент знаходиться у деякому стані s. Він вибирає дію a із доступного простору дій, отримуючи відповідну винагороду r, і переходить в наступний стан s'. Процес називається марковським, якщо переходи залежать тільки від поточного стану та вибраної дії. Таким чином, лише останній стан має значення для вибору наступної дії. У процесі вибору дій агенти намагаються знайти стратегію, яка максимізує сумарну винагороду R, яка обчислюється за наступними формулами для скінченних та нескінченних випадків відповідно:

$$R = \sum_{t=0}^{N-1} r_{t+1}, \qquad R = \sum_{t=0}^{\infty} \gamma^t * r_{t+1}$$

Тут $0 < \gamma \leq 1$ - коефіцієнт знецінювання.

1.1 Навчання з підкріпленням

Коли ймовірності переходу між станами невідомі, задача марковського процесу прийняття рішень стає задача навчання з підкріпленням. Агент намагається методом спроб і помилок створити модель навколишнього світу [1].

Таким чином, навчання з підкріпленням - це вид машинного навчання, який полягає в навчанні одного або декількох агентів, ціль якого — це накопичення максимальної сумарної винагороди під час взаємодії з середовищем.

Метод спроб і помилок та немиттєва винагорода є відмінними ознаками навчання з підкріпленням. Один з фундаментальних викликів - це компроміс між використанням та дослідженням [1]. Коли агент виявляє відносно вигідну послідовність дій, він, можливо, захоче продовжувати притримуватись такої стратегії завжди. Однак, можуть існувати інші, ще не випробувані альтернативи, що можуть принести більше вигоди. Агент не буде знати, чи є альтернативи кращими чи ні, якщо не буде систематично їх досліджувати.

1.2 Класифікація алгоритмів навчання з підкріпленням

Класифікацію алгоритмів навчання з підкріпленням можна проводити по багатьом властивостям. Перш за все, алгоритми поділяють на два великі класи: алгоритми, що використовують модель (model-based), та алгоритми, що не використовують модель (model-free) [3]. Алгоритми, що використовують модель, будують, за рахунок досвіду, внутрішню модель переходів середовища. Вибір дії здійснюється за залученням цієї моделі. Алгоритми, що не використовують модель, обробляють досвід для покращення наближення функції цінності стану, дії або параметризованої стратегії.

Якщо розглядати детальніше алгоритми, що не використовують модель, а саме такий алгоритм був використаний у практичній частині роботи, то можна виділити декілька основних напрямків: методи на основі функцій цінності, методи прямого пошуку стратегії, та методи, які поєднують ці дві концепції [7].

У першому випадку агент намагається вивчити функцію цінності, яка визначається на множині станів (функція цінності станів V) або на множині пар дій і станів (функція цінності дій Q). Значення функцій відповідають математичному сподіванню суммарної винагороди, яку отримає агент, якщо почне з цього стану, або обере данну дію, та буде дотримуватися стратегії π [4].

$$Q^{\pi}(s,a) = E[R_t|s_t = s, a], \qquad V^{\pi}(s) = E[R_t|s_t = s]$$

Способом виведення стратегії в таких методах є вибір жадібних дій: вибираються дії, для яких функція цінності вказує, що очікувана винагорода винагорода є найвищою з можливих. Однак такий підхід може вилитися у велику складність обчислень, особливо якщо простір дій неперервний. Тому, такі методи, як правило, дискретизують простір доступних дій.

Методи прямого пошуку стратерій, як правило, працюють з параметризованим сімейством стратегій $\pi(\theta)$, до яких, техніки оптимізації можна застосовувати безпосередньо [2]. Функція втрат буде наступною [10]:

$$J(\theta) = E_{\pi_\theta}[r(\tau)]$$

де $r(\tau)$ траєкторія. Оновлення параметрів через градієнт функції втрат:

$$\theta_{t+1} = \theta_t + \alpha \Delta J(\theta_t)$$

Перевага такого підходу полягає в тому, що дає можливість працювати з неперервним простором дій, але такі методи оптимізації страждають від великої дисперсії в оцінках градієнта, що призводить до повільного навчання.

Методи, які поєднують безпосередній пошук оптимальної стратегії з обчисленням функції цінності дій, класифікуються як Суб'єкт-Критик та поєднують переваги обох підходів [5]. Алгоритм тренує дві параметризованої мережі: Критик апроксимує функцію цінності стану, в той час коли Суб'єкт виконує пошук оптимальної параметризованої стратегії, зокрема на основі наближення, яке забезпечує Критик. За рахунок Суб'єкту-у вдається обичислювати дії з непервного простору. Критик знижує дисперсію, оскільки оцінка критиком очікуваної сумарної винагороди дозволяє актору здійснювати оновлення градієнтом, який має меншу дисперсію, таким чином прискорюючи процес навчання [6].

$$\nabla_\theta J(\theta_t) \approx E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)]$$

$$\Delta\theta = \alpha * [\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)]$$

1.3 Системи з багатьма агентами, ігри Маркова

Матрична гра - це стохастична система, де кожен гравець обирає дію та отримує винагороду на основі неї та дії всіх інших агентів. Назва пояснюється тим, що ігри можна представити у вигляді матриці, можливо багатовимірної. У випадку двох гравців маємо звичайну прямокутну матрицю. Вибір дії пер-

шим гравцем — це фактично вибір рядку, другим — вибір колонки матриці. На відміну від процесів рішення Маркова, ці ігри не мають стану.

Ігри Маркова або стохастичні ігри є узагальненням процесу прийняття рішень Маркова вже з кількома агентами. Крім того, їх ще можна розглядати як розширення матричних ігор зі станами. В іграх Маркова агенти отримують винагороди, в залежності від стану. Наступний стан визначається спільною дією агентів. Гра є грою Маркова, якщо наступний стан залежить лише від поточного стану та поточних дій, що здійснюються усіма агентами.

$$P(a_i^t = a_i | s^t, a_i^t - 1, ..., s^0, a_i^0) = P(a_i^t = a_i | s^t)$$

1.4 Глибоке навчання з підкріпленням

У глибокому навчанні з підкріпленням використовуються нейронні мережі. Зазвичай вони залучаються для апроксимації функцій цінності та параметризованої стратегії.

Алгоритми глибокого навчання з підкріпленням мають кілька переваг у порівнянні з традиційними алгоритмами навчання з підкріпленням. По-перше, вони вже не використовують таблиці станів, оскільки стани узанальнюються. Це дозволяє реалізовувати алгоритми для середовищ з великою кількістю станів або із неперервною множиною станів.

Алгоритм навчання з підкріпленням називається алгоритмом глибокого навчання з підкріпленням, якщо він використовує нейронну мережу.

1.5 Асинхронний Суб'єкт-Критик

Асинхронний Суб'єкт-Критик(A3C) — це алгоритм, заснований на парадигмі Суб'єкт-Крит, асинхронна версія, що підтримує багатопоточність та використовує функцію переваги.

Функція переваги - це метод, що дозволяє значно зменшити дисперсію градієнта стратегії, віднімаючи від сумарної винагороди baseline. Це призводить до зменшення величини градієнту [9]. Таким чином забезпечується краща

збіжність.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{a}_{i,t} \mid \mathbf{s}_{i,t} \right) \left(r \left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t} \right) + \gamma \hat{V}_\phi^\pi \left(\mathbf{s}_{i,t+1} \right) - \hat{V}_\phi^\pi \left(\mathbf{s}_{i,t} \right) \right)$$

Асинхронність дозволяє тренувати модель швидше та з меншим використанням ресурсів [8]. Політика та функція цінності оновлюються один раз в $t_{\max}$ дій або у разі, якщо був досягнутий термінальний стан.

РОЗДІЛ 2. ЗАДАЧА ПЕРЕСЛІДУВАННЯ З ДВОМА ЖЕРТВАМИ

2.1 Постановка задачі

Розглянемо наступний варіант задачі переслідування. На замкненій площині знаходяться три об'єкти: «Хижак» та дві «Жертви».

Задача «Хижака» якомога швидше наздогнати обидві «Жертви». Вважається, що «Хижак» наздогнав «Жертву», якщо відстань між ними стала менше, ніж наперед задана мала величина. Задача «Жертв» не дати «Хижаку» себе наздогнати протягом якомога більшого проміжку часу.

«Хижак» та «Жертви» рухаються на площині зі швидкістю, що не може перевищувати задану максимальну величину. Обмеження для «Хижака» та «Жертви» можуть бути різними. Впливати на свій рух вони можуть змінюючи вектор прискорення, проте модуль вектора прискорення так само обмежений. Немає жодних обмежень на напрям вектора прискорення.

2.2 Задача переслідування з двома жертвами як марковська гра

Задачу з двома жертвами можна представити як марковську гру. Один за агентів буде відповідати за рух «Хижака», а інший - за рух обох «Жертв». Можливий підхід, за якого кожною з «Жертв» керує окремий агент. Але для задачі максимізації загального часу, який потрібен «Хижаку» для того, щоб спіймати обидві жертви, вибраний підхід більш доречний, адже рухи «Жертв» мають узгоджуватись. Визначимо множину станів гри, множину доступних дій, переходи між станами та винагороди як функцію від стану.

2.2.1 Стан гри

Нехай «Хижак» та «Жертви» можуть рухатись у рамках прямокутника розміру $w * h$. Максимальна швидкість «Хижака» $v_{p\max}$. Максимальна швидкість «Жертв» $v_{v\max}$. Максимальне прискорення «Хижака» $a_{p\max}$. Максимальне прискорення «Жертв» $a_{v\max}$. Координати «Хижака» (x_p, y_p). Координати «Жертв» (x_{v_1}, y_{v_1}). Компоненти вектору швидкості «Хижака» (v_{x_p}, v_{y_p}). Ком-

поненти вектору швидкості першої «Жертви» $(v_{x_{v_1}}, v_{y_{v_1}})$. Компоненти вектору швидкості другої «Жертви» $(v_{x_{v_2}}, v_{y_{v_2}})$.

Мінімальна допустима відстань між «Хижаком» та «Жертвою» $d_{\min}$.

Також введемо величини $alive_{v_1}$ та $alive_{v_2}$, які приймають значення 0 або 1 відповідно до того жива відповідна «Жертва», або ні. Тоді станом гри буде наступний кортеж: $(x_p, y_p, v_{x_p}, v_{y_p}, x_{v_1}, y_{v_1}, v_{x_{v_1}}, v_{y_{v_1}}, x_{v_2}, y_{v_2}, v_{x_{v_2}}, v_{y_{v_2}}, alive_{v_1}, alive_{v_2})$,

при чому $\sqrt{v_{x_p}^2 + v_{y_p}^2} \leq v_{p\max}$, $\sqrt{v_{x_v}^2 + v_{y_v}^2} \leq v_{v\max}$, $0 \leq x_p, x_v \leq w$ та $0 \leq y_p, y_v \leq h$. Стан гри, коли обидві жертви були впольовані, тобто коли $alive_{v_1} = alive_{v_2} = 0$, вважається термінальним.

2.2.2 Множина допустимих дій

Дією для агента, що керує «Хижаком» є двійка чисел - компоненти вектору прискорення «Хижака»: (a_{x_p}, a_{y_p}), при чому $\sqrt{a_{x_p}^2 + a_{y_p}^2} \leq a_{p\max}$. Дія для агента, що керує обома «Жертвами» є четвірка - компоненти векторів прискорення для обох жертв $(a_{x_{v_1}}, a_{y_{v_1}}, a_{x_{v_2}}, a_{y_{v_2}})$, при чому $\sqrt{a_{x_v}^2 + a_{y_v}^2} \leq a_{v\max}$.

2.2.3 Перехід між станами

Нехай маємо стан s^t у момент t: $(x_p^t, y_p^t, v_{x_p}^t, v_{y_p}^t, x_{v_1}^t, y_{v_1}^t, v_{x_{v_1}}^t, v_{y_{v_1}}^t, x_{v_2}^t, y_{v_2}^t, v_{x_{v_2}}^t, v_{y_{v_2}}^t, alive_{v_1}^t, alive_{v_2}^t)$, дія агента-«Хижака»: $(a_{x_p}^t, a_{y_p}^t)$, дія аганта-«Жертв»: $(a_{x_{v_1}}^t, a_{y_{v_1}}^t, a_{x_{v_2}}^t, a_{y_{v_2}}^t)$

Порахуємо $(x_p^{t+1}, y_p^{t+1}, v_{x_p}^{t+1}, v_{y_p}^{t+1}, x_{v_1}^{t+1}, y_{v_1}^{t+1}, v_{x_{v_1}}^{t+1}, v_{y_{v_1}}^{t+1}, x_{v_2}^{t+1}, y_{v_2}^{t+1}, v_{x_{v_2}}^{t+1}, v_{y_{v_2}}^{t+1}, alive_{v_1}^{t+1}, alive_{v_2}^{t+1})$.

Використаємо співвідношення

$$\begin{cases} v_{x_p}^{t+1} = v_{x_p}^t + a_{x_p}^t \\ v_{y_p}^{t+1} = v_{y_p}^t + a_{y_p}^t \end{cases}, \tag{1}$$

у разі, якщо модуль отриманої швидкості не більший за максимально допустиме значення. Інакше, результуючим буде вектор співнапрямлений з (1), модуль якого рівний $v_{p\max}$:

$$\begin{cases} v_p = \sqrt{(v_{x_p}^t + a_{x_p}^t)^2 + (v_{y_p}^t + a_{y_p}^t)^2} \\ v_{x_p}^{t+1} = \frac{(v_{xp}^t + a_{xp}^t) v_{p\max}}{v_p} \\ v_{y_p}^{t+1} = \frac{(v_{yp}^t + a_{yp}^t) v_{p\max}}{v_p} \end{cases}, \tag{2}$$

Далі знайдемо положення «Хижака» за формулами

$$\begin{cases} x_p^{t+1} = x_p^t + v_{x_p}^{t+1} \\ y_p^{t+1} = y_p^t + v_{y_p}^{t+1} \end{cases}, \tag{3}$$

В разі, якщо «Хижак» натикнувся на границю, тобто значення однієї з його координат стало меншим за нуль, або досягло масимального значення, то значення цієї координати залишається граничним, а відповідна компонента швидкості змінює знак на протилежний.

Аналогічно рахуємо положення та швидкість живих «Жертв». Якщо «Хижак» вже наздогнав одну з «Жертв», то її положення не змінюється. Якщо відстань від «Хижака» до «Жертви» i менша за мінімальну допустиму ($\sqrt{(x_p^t - x_{v_i}^t)^2 + (y_p^t - y_{v_i}^t)^2} < d_{\min}$), то вважається, що «Хижак» її вполював: $alive_{v_i}^{t+1} = 0$

2.2.4 Винагороди

Для задачі переслідування з двома жертвами можна визначати різні функції винагороди, головним критерієм є те, щоб в рамках одного епізоду кумулятивна винагорода «Хижака» зменшувалась зі збільшенням довжини епізоду, а нагорода «Жертв» навпаки зростала. Найефективніше тренування забезпечила наступна функція винагороди:

Нехай маємо стан $(x_p^{t+1}, y_p^{t+1}, v_{x_p}^{t+1}, v_{y_p}^{t+1}, x_{v_1}^{t+1}, y_{v_1}^{t+1}, v_{x_{v_1}}^{t+1}, v_{y_{v_1}}^{t+1}, x_{v_2}^{t+1}, y_{v_2}^{t+1}, v_{x_{v_2}}^{t+1}, v_{y_{v_2}}^{t+1}, alive_{v_1}^{t+1}, alive_{v_2}^{t+1})$.

Винагорода буде залежати від відстаней d_1, d_2 між «Хижаком» та «Жертвами». $d_1 = \sqrt{(x_p - x_{v_1})^2 + (y_p - y_{v_1})^2}$., $d_2 = \sqrt{(x_p - x_{v_2})^2 + (y_p - y_{v_2})^2}$. Порахуємо винагороду «Хижака» r_p^t та винагороду «Жертв» r_v^t:

$$\begin{cases} r_p^t = -\alpha * (\sqrt{d_1} * alive_{v_1} + \sqrt{d_2} * alive_{v_2}) \\ r_v^t = \alpha * (\sqrt{d_1} * alive_{v_1} + \sqrt{d_2} * alive_{v_2})2 \end{cases}$$

Якщо «Хижак» спіймав «Жертву», то він отримує отримує бонус, який обернено пропорційний номеру кроку t.

$$\begin{cases} r_p^t = -\alpha * (\sqrt{d_1} * alive_{v_1} + \sqrt{d_2} * alive_{v_2}) + \beta/t \\ r_v^t = \alpha * (\sqrt{d_1} * alive_{v_1} + \sqrt{d_2} * alive_{v_2})2 \end{cases}$$

α, β - параметри системи.

Описана функція винагороди дає винагороду пропорційно кореню з відстані між «Хижаком» та кожною з «Жертв», що стимулює «Хижака» зближатися з однією з «Жертв», оскільки за таких умов він буде отримувати менші мінуси на кожному кроці при рівній сумарній відстані до «Жертв».

РОЗДІЛ 3. ЗАСТОСУВАННЯ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДО ЗАДАЧІ ПЕРЕСЛІДУВАННЯ З ДВОМА ЖЕРТВАМИ

У цьому розділі описане застосування алгоритму A3C до задачі переслідування з двома жертвами, яка була формалізована у попередньому розділі цієї роботи.

3.1 Бібліотека Rllib

У роботі була використана бібліотека Rllib. Rllib - це бібліотека з відкритим кодом для навчання з підкріпленням, яка пропонує як високу масштабованість, так і зручний інтерфейс для тренування різних моделей. Rllib підтримує використання TensorFlow, TensorFlow Eager та PyTorch. Також у бібліотеці присутні ефективні реалізації багатьох алгоритмів навчання з підкріпленням, зокрема алгоритму A3C.

3.2 Архітектура апроксимуючих нейронних мереж

Існують два принципово різні підходи до побудови апроксимуючих нейронних мереж для функції цінності та параметризованої стратегії. Перший, очевидний, полягає в тому, щоб тренувати дві окремі нейронні мережі, одна з яких апроксимує функцію цінності $\hat{V_\phi}^\pi(s)$, а інша параметризовану стратегію $\pi_\theta(a|s)$. При застосуванні другого, використовуються одна нейронна мережа, яка одночасно апроксимує обидві функції. Експерименти показують, що використання спільних прихованих шарів нейронної мережі показує більшу ефективність, тож саме такий підхід був використаний при проектуванні моделі.

Вхід моделі кожного з агентів - це спостереження, тобто стан середовища s^t, Вихід моделі агента-«Хижака»: (μ_x, σ_x) - параметри стандартного розподілу компоненти прискорення a_x, (μ_y, σ_y) - параметри стандартного розподілу компоненти прискорення a_y, а також $\hat{V_\phi}^\pi(s)$ - наближення значення функції цінності для даного стану.

Вихід моделі агента-«Жертв»: $(\mu_{x_1}, \sigma_{x_1})$ - параметри стандартного розподілу компоненти прискорення a_{1_x} першої жертви, $(\mu_{y_1}, \sigma_{y_1})$ - параметри стандар-

тного розподілу компоненти прискорення a_{1_y}, аналогічні значення $(\mu_{x_2}, \sigma_{x_2}, \mu_{y_2}, \sigma_{y_2})$ для другої «Жертви» а також $\hat{V_\phi}^\pi(s)$ - наближення значення функції цінності для даного стану.

Кожна модель містить три прихованих шари, в кожному з яких 512 вузлів. В якості функції збудження для прихованих шарів була використана функція ReLU ($f(x) = x^+$).

3.3 Використання власної реалізації розподілу дій

Для запобігання виродження стратегій та створення різноманітних ситуацій, у алгоритмі була використана власна реалізація розподілу дій. Цей підхід полягає в тому, що дія вибирається агентом не завжди згідно розподілу, параметри якого були отримані на виході нейронної мережі. Під час тренування, «Жертви» у восьмидесяти відсотках ситуацій обирали дію відповідно до виходу нейронної мережі, в інших випадках прискорюватися або прямо на «Хижака», або навпаки строго від нього. Моделі «Хижака» та «Жертв», натреновані без використання цієї концепції в багатьох специфічних станах діяли невдало.

РОЗДІЛ 4. ТЕСТУВАННЯ. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Тестування

Для тестування було проведено декілька експериментів, де одна із нетренованих стратегій замінялась на евристичний аналог та порівнювались ефективності натренованої та евристичної стратегії.

4.1.1 Експеримент з натренованими стратегіями «Хижака» та «Жертв»

Перший експеримент був проведений для оцінки довжини епізоду, коли обидва агенти корстуються натренованою стратегією. Після симуляції 1000 епізодів гри був отриманий наступний розподіл епізодів за довжиною:

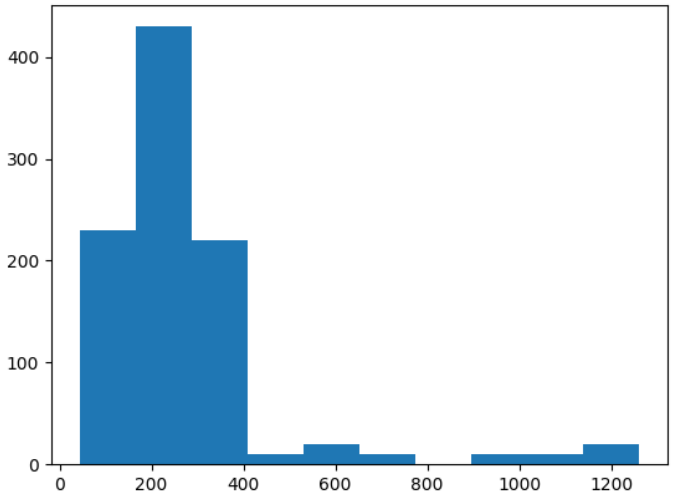

Рисунок 4.1 Кількість епізодів різної довжини для експерименту з натренованими стратегіями «Хижака» та «Жертв»

Були пораховані вибіркове середнє, дисперсія та середнє квадратичне:

$$\hat{x} = \frac{1}{1000} \sum_{i=i}^{1000} x_i = 282.927$$

$$S^2 = \frac{\sum x_i - \hat{x}}{999} = 44639.731$$
$$S = \sqrt{\frac{\sum x_i - \hat{x}}{999}} = 211.281$$

4.1.2 Експеримент з натренованою стратегією «Хижака» та евристичною «Жертв»

Другий експеримент був проведений для оцінки довжини епізоду, коли агент-«Хижак» користувався натренованою стратегією, а агент-«Жертви» - евристичною.

Евристична стратегія полягала в тому, що вектор прискорення кожної з «Жертв» обирався як вектор максимально допустимого розміру співнапрямлений з вектором, що сполучав точку положення «Хижака» та точку положення відповідної «Жертви». Таким чином «Жертва» тікала від «Хижака» по прямій з максимально можливою швидкістю.

Після симуляції 1000 епізодів гри був отриманий наступний розподіл епізодів за довжиною:

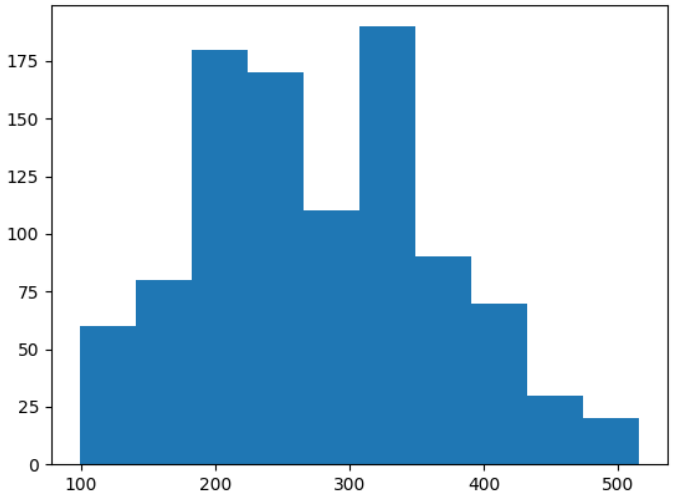

Рисунок 4.2 Кількість епізодів різної довжини для експерименту з натренованою стратегією «Хижака» та евристичною «Жертв»

Були пораховані вибіркове середнє, дисперсія та середнє квадратичне:

$$\hat{x} = \frac{1}{1000}\sum_{i=i}^{1000} x_i = 279.210$$
$$S^2 = \frac{\sum x_i - \hat{x}}{999} = 8461.703$$
$$S = \sqrt{\frac{\sum x_i - \hat{x}}{999}} = 91.988$$

4.1.3 Експеримент з евристичною стратегією «Хижака» та натренованою «Жертв»

Третій експеримент був проведений для оцінки довжини епізоду, коли агент-«Хижак» корстувався евристичною стратегією, а агент-«Жертви» - натренованою.

Евристична стратегія «Хижака»полягала в тому, що вектор прискорення «Хижака» обирався як вектор максимально допустимого розміру співнапрямлений з вектором, що сполучав точку положення «Хижака» та точку положення однієї з «Жертв», поки «Хижак» її не спіймає. Потім вектор прискорення обирався аналогічним чином для того, щоб спіймати другу «Жертву».

Після симуляції 1000 епізодів були пораховані вибіркове середнє, дисперсія та отриманий наступний розподіл епізодів за довжиною:

$$\hat{x} = \frac{1}{1000}\sum_{i=i}^{1000} x_i = 356.188$$
$$S^2 = \frac{\sum x_i - \hat{x}}{999} = 104994.933$$
$$S = \sqrt{\frac{\sum x_i - \hat{x}}{999}} = 324.029$$

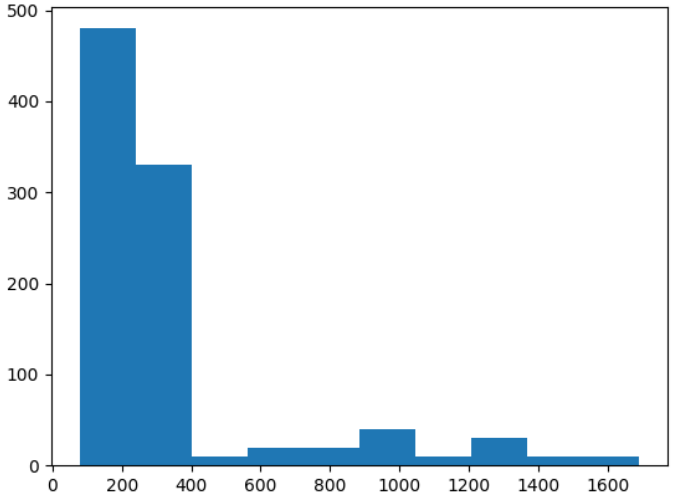

Рисунок 4.3 Кількість епізодів різної довжини для експерименту з евристичною стратегією «Хижака» та натренованою «Жертв»

4.2 Аналіз результатів експериментів

Були отримані вибіркові характеристики для проведених експериментів, а також побудовані гістограми.

№	"Хижак"	"Жертва"	Середня довжина	Ср. кв. відхилення
1	Натренована	Натренована	282.927	211.281
2	Евристична	Натренована	356.188	324.029
3	Натренована	Евристична	279.21	91.988

Табл. 1: Вибіркові значення довжини одного епізоду з різними стратегіями

4.2.1 Порівняння натренованої стратегії «Хижака»

При порівнянні евристичної та натренованої стратегії «Хижака» перше, на що слід звернути увагу - це різниця у середній довжині у другому та третьому експерименті. Евристичному «Хижаку» у середньому потрібно на $\approx 26\%$ більше часу, щоб спіймати натреновані жертви. Це пояснюється тим, що натреновані жертви здійснюють маневри намагаючись пересуватись з одного кута в інший, на які натренований хижак здатен реагувати набагато більш ефективно.

Також, максимальна довжина епізоду в другому експерименті (1691) біль-

ша на $\approx$ 34%, ніж у першому (1262), а середнє квадратичне відхилення на $\approx$ 54%, що показує, що натренований хижак більш стабільно ловить жертв - довжина епізоду не так сильно залежить від початкового положення об'єктів на площині.

4.2.2 Порівняння натренованої стратегії «Жертви»

Якщо порівнювати середню довжину епізоду, то видно, що натренована стратегія трохи більш успішна, але перевага незначна (середня довжина епізоду відрізняється на $\approx$ 1.3%). З цього можна зробити висновок, що евристична стратегія досить ефективна, незважаючи на простоту її визначення. Проте якщо порівняти максимальну довжину епізоду (в натренованої жертви більше у 2.44 рази) та дисперсію (в натренованої жертви більше у 2.44 рази), то можна зробити висновок, що в деяких ситуаціях, які дозволяють ефективно маневрувати, натренована «Жертва» може суттєво збільшити довжину епізоду.

4.2.3 Результати

У роботі були спроектовані та натреновані моделі для задачі переслідування з двома жертвами за допомогою алгоритма A3C. Були описані особливості навчання з підкріпленням для систем з багатьма агентами. Тестування встановило, що отримані стратегії є ефективними, а, отже, навчання з підкріпленням доцільно використовувати для задач переслідування різного рівня складності.

ВИСНОВОК

В даній роботі були описані підходи до побудови моделей навчання з підкріпленням, розглянуті особливості їх тренування. Був застосований алгоритм A3C до задачі переслідування з двома жертвами, формалізованій у термінах марковської гри. Натреновані стратегії «Хижака» та «Жертв» були протестовані. Також було проведено порівняння цих стратегій зі стратегіями на базі евристик.

Таким чином, було показано, що навчання з підкріпленням може розглядатися як один з ефективних способів вирішення подібних, але більш специфікованих задач, які можуть виникати у різних сферах, насамперед у військовій.

Отримані результати виступають підґрунтям для майбутніх досліджень у сфері навчання з підкріпленням, особливо по відношенню до інших задач переслідування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

[1] *Reinforcement Learning: An Introduction* [Текст] / Richard Sutton and Andrew Barto - USA, Westchester Publishing Services / 2018.

[2] *Algorithms for Reinforcement Learning* [Текст] / Csaba Szepesvari / June 9, 2009

[3] *A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients* [Текст] / Ivo Grondman, Lucian Busoniu, Gabriel A.D. Lopes and Robert Babuska

[4] *Human-level control through deep reinforcement learning.* [Текст] / Volodymyr Mnih / 2015

[5] *The Reactor: A fast and sample-efficient Actor-Critic agent for Reinforcement Learning* [Текст] / Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc Bellemare, Remi Munos / 2018

[6] *Actor-Critic Algorithms* [Текст] / Vijay R. Konda John N. Tsitsiklis - Cambridge / 2001

[7] *An Analysis of Temporal-Difference Learning with Function Approximation* [Текст] / John Tsitsiklis and Benjamin Van Roy / May, 1997.

[8] *Asynchronous Methods for Deep Reinforcement Learning* [Текст] / Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P, Lillicrap, David Silver and Koray Kavukcuoglu - University of Montreal / 2016.

[9] *Asynchronous Advantage Actor-Critic: Non-Asymptotic Analysis And Linear Speedup* [Текст] / Han Shen, Mingyi Hong, Kaiqing Zhang and Tianyi Chen - ICLR 2021 Conference Blind Submission / 2021.

[10] *Proximal Policy Optimization Algorithms* [Текст] / John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov - OpenAI, San Francisco / 2017.