МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра систем штучного інтелекту

КУРСОВА РОБОТА

з дисципліни «Штучний інтелект в ігрових застосуваннях»

на тему:

“ Безпека інформаційних технологій”

Виконав:

ст. групи КНСШ-14

Підбурачинський Р. А.

Перевірив:

процесор каф. СШІ, д.т.н.,

Камінський Р.М.

ЛЬВІВ – 2022

Зміст

ВСТУП

Сфера штучного інтелекту (ШІ) в іграх швидко розвивається завдяки прогресу в обчислювальній потужності та алгоритмах. Штучний інтелект став незамінним компонентом у різних жанрах ігор, починаючи від стратегій і симуляторів і закінчуючи екшенами та рольовими іграми. Оскільки попит на більш складні ігрові середовища та інтелектуальні агенти зростає, оптимізація та вдосконалення алгоритмів штучного інтелекту є життєво важливими для задоволення цих очікувань.

Вивчення паралельних і розподілених обчислень відіграє важливу роль у підвищенні ефективності та продуктивності алгоритмів ШІ в іграх. Обмежені ресурси та зростаюча обчислювальна складність вимагають інноваційних підходів для скорочення трудомістких завдань. Існуючі модифікації алгоритмів включають розпаралелювання та розподіл обчислень для прискорення обробки та мінімізації витрат на обчислення.

Ця курсова робота спрямована на створення, дослідження та вдосконалення агента штучного інтелекту з використанням алгоритмів Q-навчання, методів паралельних обчислень і Python для вирішення ігрової проблеми. Гіпотеза полягає в тому, що розпаралелювання алгоритму Q-навчання може призвести до швидшого та ефективнішого пошуку оптимального шляху, таким чином перевершуючи традиційні послідовні методи ШІ.

РОЗДІЛ 1. АНАЛІЗ НАЯВНИХ ДОСЛІДЖЕНЬ

Кілька недавніх досліджень показали, що паралельні обчислення можуть значно скоротити час, необхідний для навчання агентів штучного інтелекту, включаючи алгоритми навчання з глибоким підкріпленням [4]. Ці методи застосовуються до різних областей, включаючи робототехніку, обробку природної мови, комп'ютерне бачення та ігровий штучний інтелект[5].

Алгоритми паралельного та розподіленого Q-навчання були досліджені для вирішення проблем обчислювальної складності та швидкості навчання в навчанні з підкріпленням. Цей підхід використовує переваги багатоядерних процесорів і обчислювальних кластерів для розпаралелювання процесу навчання, скорочення часу навчання та підвищення ефективності [6].

1.1. Опис проблеми

Проблема, яка розглядається, полягає в тому, щоб знайти оптимальний шлях для досягнення агентом штучного інтелекту цільової позиції в ігровому середовищі на основі двовимірної сітки, збираючи якомога більше бонусних клітинок. Середовище містить перешкоди, яких агент ШІ повинен уникати.

Ігрове середовище представлено сіткою, у якій кожна клітинка може мати один із наступних станів:

- Початкова позиція: спочатку агент ШІ розміщується у початковій позиції сітки.
- Цільова позиція: агент ШІ має досягти цільової позиції, максимізуючи винагороду.
- Перешкоди: комірки сітки, яких агент штучного інтелекту повинен уникати, оскільки вони створюють проблеми або блокують агента.
- Бонусні комірки: комірки сітки, які надають винагороду агенту ШІ за відвідування.

1.2. Наявні підходи до вирішення проблеми

Q-learning — це техніка навчання з підкріпленням без моделі, що не

відповідає правилам[1]. У Q-навчанні агент вивчає Q-функцію, яка допомагає йому вибрати найкращу дію в кожному стані для максимізації сукупної винагороди. Постійно оновлюючи Q-таблицю шляхом дослідження та експлуатації, агент покращує свою оцінку Q-функції та в кінцевому підсумку наближається до оптимальної політики вибору дій.

Плюси:

- Не потребує попередньої моделі чи знання середовища, що робить його універсальним для вирішення різних проблем.
- Легко реалізувати, для навчання потрібні обмежені обчислення, особливо при використанні навчання на основі таблиць.
- З належними налаштуваннями навчання гарантовано наближається до оптимальної політики з часом.
- Вивчає оптимальну політику незалежно від стратегії дослідження агента, дозволяючи досліджувати з використанням різних політик, не впливаючи на конвергенцію.

Мінуси:

- Проблеми з масштабованістю - Q-таблиця може стати дуже великою для середовищ із численними станами та діями, що призводить до проблем із пам’яттю та обчисленнями.
- Для Q-навчання може знадобитися велика кількість епізодів або ітерацій, щоб точно оцінити оптимальну політику вибору дій, особливо в складних середовищах.
- Збалансувати розвідку та розробку є складним і сильно залежить від конкретної проблеми. Неадекватний баланс може призвести до неоптимального вивчення політики.
- Розроблено для дискретних просторів станів і дій. Його адаптація до безперервних просторів потребує модифікацій або наближень, таких як використання апроксиматорів функцій, таких як нейронні мережі.

Deep Q-Network — це комбінація Q-learning та глибоких нейронних

мереж[3]. Вони можуть обробляти простори станів великої розмірності та знаходити оптимальні політики, не вимагаючи спеціального представлення вхідних даних.

Плюси:

- Можуть добре узагальнюватися для великих просторів станів, оскільки використовують глибокі нейронні мережі для апроксимації функцій.
- Можуть вивчати широкий спектр функцій із необроблених вхідних даних, таких як значення пікселів у зображеннях, без необхідності ручної розробки функцій.

Мінуси:

- Потребують великої кількості навчальних даних, і може знадобитися багато часу, щоб досягти стабільної політики.
- Є складнішими, ніж звичайне Q-навчання, і їх навчання може бути дорогим з обчислювальної точки зору.

A* search: — це обґрунтований алгоритм пошуку, який використовує комбінацію фактичної вартості досягнення поточного вузла та евристичної оцінки вартості досягнення цільового вузла[5]. Він широко використовується для пошуку шляху в сіткових іграх.

Плюси:

- Є детермінованим і зазвичай знаходить глобально оптимальний шлях.
- Є ефективним, коли обрана евристика прийнятна та послідовна, оскільки алгоритм зосереджується на розширенні вузлів, які з більшою ймовірністю приведуть до найкоротшого шляху.

Мінуси:

- Вимагає відповідної евристичної функції, яку не завжди легко розробити.
- Може мати потреби в пам’яті, які значно збільшуються, особливо

для великих просторів пошуку та нетривіальних задач.

Пошук по дереву Монте-Карло — це алгоритм пошуку по дереву, який особливо ефективний у проблемах прийняття рішень, коли простір станів великий або заздалегідь невідомий[7]. MCTS будує дерево пошуку, повторюючи серію з чотирьох кроків: вибір, розширення, моделювання та зворотне поширення. Він балансує між дослідженням нових вузлів і використанням інформації, зібраної з раніше відвіданих вузлів.

Плюси:

- Може надати розумні рішення за відносно короткий час, навіть для великих просторів пошуку.
- Особливо ефективна в середовищах з невизначеністю або прихованою інформацією.
- Не вимагає евристичної функції або моделі середовища.

Мінуси:

- Може бути обчислювально дорогим, залежно від кількості ітерацій і коефіцієнта розгалуження.
- Ефективність MCTS може бути не такою послідовною, як інші методи, такі як A*, особливо коли баланс розвідки/видобутку налаштований неправильно.

1.3. Аналіз літературних джерел

В останні роки штучний інтелект в ігровій індустрії стрімко розвивається, і дослідники зосереджуються на створенні більш реалістичних і привабливих вражень для гравців. У цьому розділі представлено аналіз сучасного стану застосування штучного інтелекту для ігор шляхом вивчення останніх досліджень і відповідних методологій.

Зростання популярності методів навчання з підкріпленням в іграх спонукало дослідників досліджувати нові підходи глибокого навчання з підкріпленням для вирішення більш складних середовищ запропонували [2] метод на основі DRL для інтелектуальних агентів у відеоіграх, підкреслюючи

його потенціал у вирішенні різноманітних проблем в іграх.

Паралельні обчислення відіграють вирішальну роль у зниженні обчислювальної складності та прискоренні алгоритмів ШІ в ігрових програмах. Багато було описано[4] переваг паралельних обчислень у глибокому навчанні та нейронних мережах, підкресливши, як паралелізм можна ефективно використовувати в ігрових середовищах, керованих ШІ.

Протягом останніх кількох років дослідники досліджували вплив штучного інтелекту на дизайн ігор, відкриваючи нові можливості для покращення ігрового процесу. Балансування ігор за допомогою штучного інтелекту, створення процедурного контенту та адаптивний геймплей показали багатообіцяючі результати[6].

У контексті ігрових персонажів досліджували[8] вплив керованої штучним інтелектом поведінки персонажів, наприклад навігації та прийняття рішень, на занурення та залучення гравців. Ці висновки підкреслюють важливість систем ШІ для покращення ігрового досвіду.

Також було досліджено поєднання DRL і паралельних обчислювальних технік, прикладом успішного застосування яких є такі ігри, як Dota 2[9] запропонували структуру на основі DRL, яка використовує паралельні обчислення в навчанні агентів для масштабних стратегічних ігор у реальному часі, демонструючи значні покращення продуктивності та часу навчання.

Дослідники також продовжують досліджувати інноваційні алгоритми та методи, які розширюють роль ШІ в ігровій індустрії. Нещодавно запропонували[10] алгоритм навігації на основі Q-навчання з покращеною продуктивністю, який має потенційне застосування в складних ігрових середовищах.

Враховуючи ці останні досягнення, галузь ШІ в іграх пропонує багато можливостей для подальшого дослідження. Дослідники повинні зосередитися на вдосконаленні алгоритмів, використовуючи паралельну обробку для зменшення обчислювальної складності та досліджуючи більш просунуті методи штучного

інтелекту, щоб стимулювати розробку та еволюцію все більш складних ігор.

1.4 Мета дослідження

Метою цього дослідження є вивчення та підвищення ефективності алгоритмів штучного інтелекту в іграх, особливо зосереджуючись на методології пошуку шляху. Дослідження аналізуватиме застосовність алгоритмів Q-learning та досліджуватиме потенційні переваги підходів до паралельних і розподілених обчислень для підвищення продуктивності алгоритму. Основною метою є розробка та оцінка оптимізованого паралелізованого агента Q-learning, здатного ефективно знаходити найкращий шлях для досягнення мети в складному ігровому середовищі, одночасно збираючи бонуси та уникаючи перешкод. Очікується, що результати цього дослідження покращать наше розуміння використання методів паралельних обчислень для штучного інтелекту в іграх і потенційно покращать ігровий досвід за рахунок скорочення часу обчислення та підвищення продуктивності ШІ.

РОЗДІЛ 2. ЗАСТОСУВАННЯ МЕТОДІВ ТА ЇХ РЕЗУЛЬТАТИ

У цьому дослідженні пропонується алгоритм, який використовує Q-навчання з паралельними обчисленнями, щоб знайти оптимальний шлях для агента в грі на основі сітки. Алгоритм спрямований на те, щоб допомогти агенту долати перешкоди, збирати бонуси та досягати цільової позиції за найкоротший шлях.

Q-Learning — це широко використовуваний алгоритм Reinforcement Learning, який створює Q-таблицю, яка зберігає очікувані винагороди для кожної комбінації станів і дій. За допомогою ітераційних оновлень Q-таблиця наближається до оптимальної функції дії-значення, що дозволяє агенту вибрати найкращу дію для будь-якого стану.

Алгоритм складається з наступних основних компонентів:

- Ініціалізація початкового стан, що містить позицію та кількість зібраних бонусів.
- Вибір дії на основі поточного стану та значень Q-таблиці, дотримуючись ε-жадібного підходу.
- Розрахунок наступного стан після виконання дії, враховуючи просторові обмеження та перешкоди.
- Визначення винагороди на основі наступного стану та оновлення Q-таблиці за допомогою рівняння оновлення Q-навчання.
- Повторювання процесу, доки не буде досягнуто цілі та не буде зібрано всі бонуси.

Щоб скористатися паралелізмом, ми навчаємо кількох агентів Q-навчання одночасно, кожен із власним набором епізодів. Для цього використовується багатопроцесорна бібліотека Python. Паралельні обчислення допомагають ефективніше досліджувати простір рішень і потенційно скорочують необхідний час навчання.

Вибір паралельної обробки в цьому дослідженні мотивований необхідністю прискорити процес навчання та ефективно досліджувати різні

шляхи в середовищі.

Обчислювальна складність запропонованого алгоритму залежить від часу, необхідного для оновлення Q-таблиці, і кількості епізодів, необхідних для конвергенції. Оскільки операція Q-навчання має часову складність O(1)[11], загальна складність для одного агента дорівнює O(n_episodes * n_states * n_actions), де n_episodes — кількість епізодів, n_states — кількість станів для кожного епізоду. , а n_actions — кількість дій, доступних для агента.

Паралельні обчислення допомагають розподілити робоче навантаження між кількома процесами, ефективно зменшуючи складність часу на коефіцієнт, що дорівнює кількості агентів, що використовуються паралельно. Таким чином, для k агентів часова складність зменшується до O(n_episodes * n_states * n_actions / k).

На завершення, запропонований алгоритм Q-навчання з паралельними обчисленнями використовує переваги одночасної роботи кількох агентів, зменшуючи обчислювальну складність і час навчання.

Створена програма це - агент, який використовує Q-learning для навігації в ігровому середовищі на основі сітки. Мета агента — знайти оптимальний шлях із максимальною винагородою, уникаючи перешкод і відвідуючи всі бонусні клітини. Сітка містить динамічно генеровані перешкоди, бонусні комірки, початкову позицію та цільову позицію, додаючи складності та варіації середовищу, в якому агент повинен вчитися.

2.1. Ігрова середовище

Ігрове середовище складається зі світу, заснованого на сітці, де розумний агент повинен орієнтуватися, щоб знайти оптимальний шлях від початкової позиції до цільової позиції, уникаючи перешкод і збираючи бонусні клітини. Сітка має заздалегідь визначений розмір, кожна клітинка представляє порожній простір, перешкоду або бонусну клітинку. Агент починає зі стартової позиції та прагне досягти та зупинитися на цільовій позиції, максимізуючи свою загальну винагороду.

Основними елементами ігрового середовища є:

- Початкова позиція: це початкова позиція, з якої агент починає навігацію сіткою – позначається синім кольором (див. рис. 1).
- Цільова позиція: кінцевий пункт призначення, куди агент повинен досягти та зупинитися, максимізуючи винагороду в процесі – позначається зеленим кольором (див. рис. 1).
- Перешкоди: це клітини в сітці, яких агент повинен уникати, оскільки вони блокують доступ і негативно впливають на загальну винагороду – позначається червоним кольором (див. рис. 1).
- Бонусні комірки: ці комірки надають додаткові винагороди агенту, коли він їх відвідує – позначаються золотим кольором (див. рис. 1).

Щоб створити різноманітне та складне навчальне середовище для агента, ігрові елементи динамічно генеруються. Створюючи нові конфігурації елементів у кожному навчальному епізоді, агент стикається з різними сценаріями проблем і вчиться відповідно адаптувати свою стратегію. Ця динамічна генерація елементів забезпечує універсальність ігрового середовища та гарантує, що агент може узагальнювати свої вивчені стратегії для різних сценаріїв, а не просто запам’ятовувати один ідеальний шлях. У результаті агент стає більш стійким і адаптивним, що можна побачити в його продуктивності в різних випадках середовища.

2.2. Алгоритм та емуляція поведінки

Алгоритм Q-навчання — це форма безмодельного навчання з підкріпленням, яка дозволяє агенту вивчати оптимальну поведінку в дискретному середовищі, спостерігаючи за результатами виконання різних дій у різних станах. Q-навчання базується на концепції Q-таблиці, матриці, яка зберігає очікувані значення для кожної комбінації дії-стану. Коли агент досліджує середовище, взаємодіючи з перешкодами, бонусними комірками та досягаючи цільової позиції, він навчається оцінювати ці значення та використовує їх для прийняття рішень.

Під час процесу навчання агент постійно оновлює свою Q-таблицю за допомогою рівняння оновлення Q-значення, яке визначається як:

- $Q(s, a) = Q(s, a) + \alpha * (R(s, a) + \gamma * \max(Q(s', a')) - Q(s, a))$

Змінні в цій формулі:

- s: поточний стан;
- a: дія, виконана в стані s;
- s': наступний стан після виконання дії a зі стану s;
- α (альфа): швидкість навчання;
- R(s, a): негайна винагорода, отримана за виконання дії a зі стану s;
- γ (гамма): дисконтний коефіцієнт;
- max(Q(s', a')): максимальне очікуване значення серед усіх дій у стані s'.

Алгоритм Q-навчання вимагає тонкого налаштування кількох параметрів для досягнення оптимального навчання.

- Коефіцієнт дослідження - цей параметр визначає баланс між дослідженням і використанням[12]. Це важливо, оскільки випадкове дослідження дозволяє агенту досліджувати незвідану територію та потенційно відкривати нові винагороди, тоді як експлуатація дозволяє агенту найкращим чином використовувати зібрану інформацію. Під час навчання коефіцієнт дослідження зазвичай починається з високого значення і експоненціально спадає з кожним епізодом. Ця стратегія спонукає агента досліджувати більше на початку та більше покладатися на свої набуті знання в міру накопичення досвіду.
- Швидкість навчання є критичною для визначення швидкості, з якою агент оновлює свою Q-таблицю. Високий рівень навчання (наприклад, 0,05) швидко включає нову інформацію, потенційно призводячи до покращення швидкості конвергенції. Однак це може спричинити надмірну чутливість агента до нещодавно отриманої інформації, що призводить до нестабільної Q-таблиці,

тоді як низька швидкість навчання призводить до більш стабільної Q-таблиці, але збільшує час, необхідний для конвергенції. Вибір оптимальної швидкості навчання залежить від конкретної проблеми та середовища, і зазвичай вимагає тонкого налаштування або використання передових методів, таких як адаптивна швидкість навчання[12].

- Коефіцієнт дисконту визначає, наскільки агент оцінює майбутні винагороди порівняно з негайними. Високий дисконтний коефіцієнт спонукає агента шукати довгострокові прибутки та знаходити оптимальний шлях, який накопичує винагороди з часом. Навпаки, низький дисконтний коефіцієнт змушує агента віддавати пріоритет негайним винагородам, що може бути неоптимальним у довгостроковій перспективі. Вибір відповідного коефіцієнта дисконтування залежить від часових характеристик проблеми та необхідного балансу між короткостроковим і довгостроковим плануванням[12].

Точне налаштування цих параметрів має важливе значення для забезпечення ефективності алгоритму Q-навчання та оптимізації продуктивності агента. Експериментування з різними налаштуваннями параметрів і вивчення адаптивних стратегій може призвести до покращення навчання та прийняття рішень у динамічному середовищі.

2.3 Процес навчання

Процес навчання є критично важливим компонентом у розвитку інтелектуального агента, забезпечуючи його здатність навчатися та адаптуватися до різноманітних ситуацій у динамічному ігровому середовищі. За допомогою алгоритму Q-навчання агент визначає найбільш ефективні рішення та дії під час навігації сіткою. У цьому розділі ми детально розглянемо різні аспекти процесу навчання агента та обговоримо фактори, які впливають на його ефективність протягом навчання.

Кількість епізодів і мета агента пройти задану кількість епізодів під час навчального процесу, причому кожен епізод є екземпляром навігації агента ігровим середовищем на основі сітки. Ці кілька епізодів дозволяють агенту

досліджувати та розуміти різні конфігурації та виклики, які представляє гра. Збільшуючи кількість епізодів, агент має більше можливостей оцінювати різні ситуації та поступово вивчати оптимальну політику для ефективного досягнення своїх цілей.

Коефіцієнт дослідження ініціалізується зазвичай числом близьким до одиниці. Він представляє ймовірність того, що агент вибере випадкову дію, а не покладається на свої поточні значення Q. Високий початковий коефіцієнт дослідження гарантує, що агент має вищу схильність досліджувати середовище, що особливо важливо на початкових етапах навчання, оскільки це дозволяє агенту збирати цінну інформацію про оточення. Ретельно досліджуючи оточення, агент може дізнатися про наслідки дій, що веде до процесу прийняття обґрунтованого рішення.

У міру того, як агент накопичує більше знань і вдосконалює свій процес прийняття рішень з часом, він повинен поступово переключати свою увагу з розвідки на розробку, використовуючи вивчені значення Q, щоб зробити кращий вибір. Для плавного керування цим переходом коефіцієнт дослідження зменшується за допомогою стратегії зменшення. Коли агент повторює епізоди, фактор дослідження зменшується, і агент більше покладається на свої вивчені значення Q для прийняття рішень. Цей поступовий зсув забезпечує збалансований підхід до навчання та покращує ефективність навігації агента.

Щоб оцінити продуктивність агента під час процесу навчання, відстежується загальна винагорода, отримана в кожному епізоді. Кілька факторів впливають на продуктивність агента, і розуміння цих факторів може допомогти оптимізувати процес навчання:

Швидкість навчання контролює швидкість, з якою агент оновлює свої Q-значення на основі спостережуваних винагород. Вищий рівень навчання прискорює здатність агента оновлювати Q-значення та адаптуватися до середовища. Однак, якщо рівень навчання надто високий, процес навчання агента може стати нестабільним, суттєво коливатися та навіть стагнувати з точки

зору покращення.

Дисконтний коефіцієнт визначає вагу майбутніх винагород у рівнянні оновлення Q-learning. Високе значення гамми підкреслює важливість довгострокових винагород, заохочуючи агента здійснювати стратегічне планування. І навпаки, низьке значення гами зосереджується на негайних винагородах, потенційно змушуючи агента втрачати оптимальний шлях.

Досягнення правильного балансу між розвідкою та розробкою має вирішальне значення для ефективного навчання. Надмірний акцент на розвідці може призвести до того, що агент не зможе ефективно застосувати свої набуті знання, тоді як надто значний пріоритет використання може призвести до того, що агент погодиться на неоптимальні рішення.

Під час навчання агент стикається з середовищем, яке постійно змінюється, з перешкодами, бонусними осередками, початковими позиціями та цільовими позиціями, які створюються заново для кожного епізоду. Агент повинен впоратися з цими динамічними елементами, навчаючись ефективно орієнтуватися в сітці. Параметри повинні бути точно налаштовані, щоб адаптуватися до цих змін, і агент повинен поступово покращувати свою продуктивність, коли він стикається з новими та змінними конфігураціями.

Враховуючи та оптимізуючи ці аспекти навчального процесу, агент може ефективно навчатися та опановувати свою стратегію навігації в динамічному ігровому середовищі. Отримані знання та досвід агента дозволять йому впевнено та ефективно протистояти ситуаціям, які раніше не зустрічалися.

2.4 Процес навчання

Покращення продуктивності завдяки додаванню кількох ядер можна віднести до властивих характеристик паралельних обчислень і алгоритму Q-навчання. Паралельні обчислення дозволяють одночасне виконання кількох процесів, що призводить до кількох переваг, як пояснюється нижче.

З більшою кількістю ядер, що працюють паралельно, агент може виконувати кілька епізодів навчання одночасно, що допомагає скоротити

загальний час, необхідний для вивчення оптимального шляху. Оскільки для виконання кожного епізоду потрібен певний час, одночасне виконання кількох епізодів на окремих ядрах значно скорочує загальний час навчання. Агент може вивчати та вдосконалювати свою Q-таблицю ефективніше, оскільки він обробляє більшу кількість епізодів за коротший період.

При використанні кількох ядер доступні обчислювальні ресурси збільшуються, що дозволяє краще справлятися зі складними завданнями. Агент може ефективніше аналізувати, оновлювати та коригувати свої Q-значення, прискорюючи процес оптимізації. Розширена обчислювальна потужність дозволяє агенту одночасно обробляти різні можливі пари стан-дія, таким чином покращуючи процес прийняття рішень і планування шляху в середовищі.

У Q-навчанні агенту потрібно збалансувати дослідження невідомих шляхів і використання відомого найкращого шляху, щоб приймати найбільш обґрунтовані рішення. Завдяки паралельній роботі декількох ядер агент може ефективніше досліджувати кілька дій і станів одночасно. Це посилене дослідження дозволяє агенту дізнатися більше про середовище, зменшуючи ймовірність застрягти в локальних оптимумах і, зрештою, сприяючи підвищенню продуктивності.

Якщо використовується кілька ядер, агент може ділитися своїм досвідом і знаннями в різних епізодах, що працюють на окремих ядрах. Роблячи це, агент може збирати колективну мудрість і ідеї з різних випадків, які сприяють уточненню Q-таблиці та покращенню загальної продуктивності агента.

Для перевірки покращення виконання використовувалась наступна конфігурація:

- Розмір сітки: 50 х 50 комірок (див рисунок 1).
- Кількість перешкод: 500
- Кількість бонусних комірок: 500
- Кількість епізодів: 5000

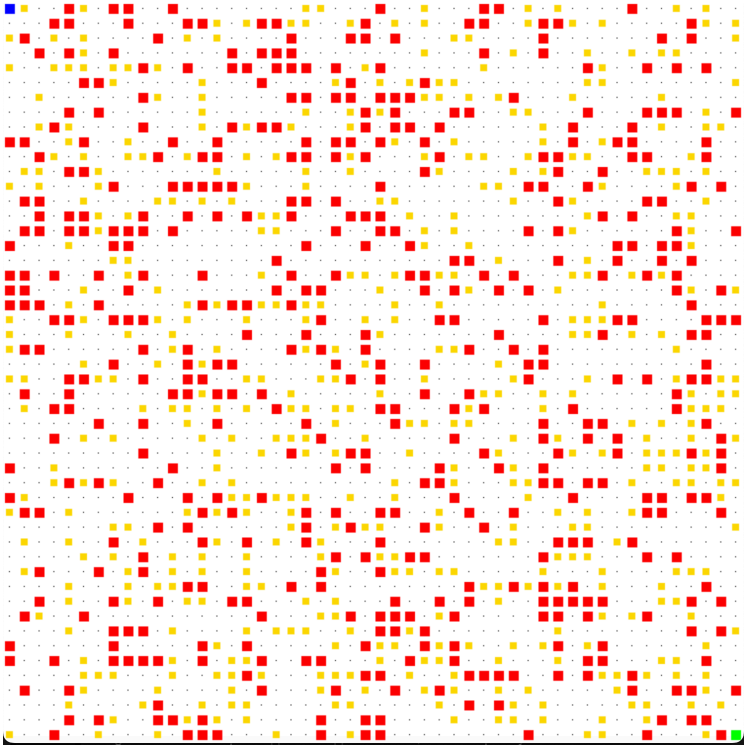

Рисунок 1 – Сітка на якій відбувається тестування продуктовності

Вплив паралельного розподілення навчання агента наведено нижче:

- Використовуючи лише одне ядро, процес навчання нашого агента розтягнувся на більш тривалий період навчання через обмежену обчислювальну потужність. Час навчання агента був 10 хвилин та 36 секунд, і хоча він завершив процес, продуктивність була обмежена доступними ресурсами.
- Збільшивши кількість доступних ядер до 3, можна спостерігати помітне зниження коефіцієнта часу навчання. Час навчання агента скоротився приблизно до 3 хвилин та 50 секунд, а загальна продуктивність покращилася в результаті розширених можливостей паралельної обробки. Ця конфігурація дозволила агенту ефективно розподіляти обчислення між кількома ресурсами, сприяючи швидшому прийняттю рішень.
- Збільшення кількості використовуваних ядер до 5 призвело до подальших покращень. Час навчання тепер скорочено приблизно до 2 хвилин та 24 секунд при збереженні аналогічної ефективності порівняно з попередніми конфігураціями. Маючи більше доступних ресурсів, агент міг би плавно

розподілити обчислення між ядрам і значно прискорити процес навчання.

- Використання 10 ядер призвело до найсуттєвішого скорочення часу навчання. Агент скористався перевагами додаткових ресурсів і отримав час навчання приблизно 1 хвилина та 15 секунд. Обчислення ефективно розподілялися між ядрами, що прискорювало процес навчання агента та досягнення оптимального шляху.

Протягом різних основних конфігурацій процес навчання агента зміг продемонструвати суттєві покращення продуктивності, що призвело до швидшого виявлення оптимального шляху. Це було досягнуто завдяки використанню переваг паралельних обчислень, оскільки вони ефективно мінімізують час навчання та покращують здатність агента адаптуватися та орієнтуватися в динамічному середовищі. Завдяки більшій кількості ядер агент міг розподіляти обчислення між декількома ресурсами, що сприяло точнішому прийняттю рішень і більшій загальній адаптивності в складних ігрових сценаріях.

ВИСНОВКИ

Це дослідження підкреслює переваги інтеграції методів паралельних обчислень з алгоритмами Q-навчання для оптимізації пошуку шляху в ігрових середовищах, заснованих на сітці. Використовуючи потужність паралельних обчислень, можна значно прискорити процес навчання, що призведе до більш швидкого та ефективного виявлення оптимального шляху, водночас перевершуючи традиційні послідовні методи ШІ.

Ключовий висновок цього дослідження полягає в тому, що включення паралелізму дозволяє більш ефективно досліджувати та використовувати процес навчання. Це досягається за рахунок того, що кілька агентів можуть працювати паралельно, досліджуючи різні дії та стани одночасно, що веде до більш повного розуміння ігрового середовища. Крім того, паралелізм дозволяє розподіляти обчислювальне навантаження між декількома ядрам, що дозволяє обробляти складні завдання зі скороченим часом навчання та покращеною продуктивністю.

Це дослідження є цінним внеском у існуючу сукупність досліджень штучного інтелекту в іграх, демонструючи ефективність методів паралельних обчислень у підвищенні ефективності та продуктивності алгоритмів ШІ. Запропонований алгоритм розпаралеленого Q-навчання продемонстрував свій потенціал у вирішенні складних ігрових проблем і пропонує багатообіцяючі напрямки для майбутніх досліджень в ігровій індустрії, керовані ШІ.

У майбутніх дослідженнях можна досліджувати інтеграцію більш просунутих паралельних обчислювальних технік, таких як алгоритми, прискорені графічним процесором, для подальшого підвищення продуктивності та ефективності штучного інтелекту в ще складніших ігрових середовищах. Крім того, дослідники могли б досліджувати поєднання інших алгоритмів навчання з підкріпленням або підходів на основі нейронної мережі з методами паралельних обчислень, щоб розширити діапазон потенційних застосувань і переваг як в ігровому, так і в неігровому контекстах.

Таким чином, інтеграція паралельних обчислень зі штучним інтелектом

може призвести до помітних успіхів в ігровій індустрії, покращуючи ігровий досвід для гравців і відкриваючи нові шляхи для більш захоплюючих і реалістичних ігрових сценаріїв. Застосування таких інноваційних підходів відіграватиме вирішальну роль у поточній еволюції ігрового штучного інтелекту, сприяючи зростанню та розвитку у сфері, що постійно розширюється.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Q-Learning: Everything you Need to Know | Simplilearn. .
2. Shao K., Tang Z., Zhu Y., Li N., Zhao D. A Survey of Deep Reinforcement Learning in Video Games. arXiv, 2019.
3. Deep Q-Network (DQN) Agents - MATLAB & Simulink. .
4. Torrado R. R., Bontrager P., Togelius J., Liu J., Perez-Liebana D. Deep Reinforcement Learning for General Video Game AI. Maastricht:IEEE, 2018. ISBN 978-1-5386-4359-4.
5. A* Algorithm in Artificial Intelligence You Must Know in 2023 | Simplilearn. 2021.
6. A State-of-the-Art Survey on Deep Learning Theory and Architectures - ProQuest. .
7. Monte-Carlo Tree Search (MCTS) — Introduction to Reinforcement Learning. .
8. Guan C., Mou J., Jiang Z. Artificial intelligence innovation in education: A twenty-year data-driven historical analysis. *International Journal of Innovation Studies*. 2020. Вип. 4, № 4. С. 134–147.
9. Stuart K. Think, fight, feel: how video game artificial intelligence is evolving. 2021.
10. Liu T., Tian B., Ai Y., Li L., Cao D. Parallel Reinforcement Learning: A Framework and Case Study. *IEEE/CAA Journal of Automatica Sinica*. 2017. Вип. 5.
11. arvindpdmn dineshpathak Algorithmic Complexity. 2017.
12. Katahira K. The relation between reinforcement learning parameters and the influence of reinforcement history on choice behavior. *Journal of Mathematical Psychology*. 2015. Вип. 66. С. 59–69.