УДК 004.8

Є. Грибенко, В. Таран

ПОКРАЩЕННЯ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДЛЯ СКЛАДНИХ ЗАДАЧ РУХУ РОБОТА

Анотація: Ця стаття присвячена створенню способу, що дозволяє обійти небажані локальні оптимуми в задачах руху робота з використанням *Proximal Policy Optimization (PPO)* та складним формуванням підкріплення (*reward shaping*). Метою статті є вирішення проблеми сходження або застрягання рішення в небажаних локальних оптимумах при використанні *PPO*, що може покращити результат навчання для задач слідування робота заданим точкам або параметрам руху. Запропоноване рішення дозволяє вирішувати складні задачі з конфліктуючими нагородами та локальними оптимумами, демонструючи вирішення проблеми на прикладі задачі руху робота.

Ключові слова: Proximal Policy Optimization, локальні оптимуми, навчання з підкріпленням, машинне навчання, рух робота.

Вступ

Навчання з підкріпленням - сучасний метод машинного навчання, що дозволяє тренувати інтелектуальних агентів без заздалегідь підготовлених даних, збираючи дані безпосередньо з досвіду взаємодії агента з середовищем [1]. Не зважаючи на відсутність потреби прямого збору даних, на розробників покладається задача створення середовища навчання та моделі агента потрібної якості, а також розробка, формування і балансування підкріплення для агента. Вибір алгоритму навчання, підбір гіперпараметрів алгоритму, налаштування середовища та нагород, а також ітерації розробки та спроби покращення результату можуть зайняти дуже багато часу розробника. Тому для зменшення зусиль на розробку рішення, корисними є прийоми, що підвищують ефективність машинного навчання, а саме швидкість процесу та оптимальність рішення.

Не дивлячись на активний розвиток галузі машинного навчання, немає універсального підходу до рішення задач, зокрема задач руху робота, тому ефективність рішення часто залежить від спроб та помилок, вибору та налаштування алгоритмів та середовища, застосування різних прийомів та технік.

При зростанні складності задачі, росте потреба у детальному формуванні нагород (підкріплення), що породжує локальні оптимуми. Метод *PPO* є вразливим до локальних оптимумів [2], тому навчання часто може сходитися до небажаного результату, що зменшує шанси на успіх застосування методу, особливо якщо

підкріплення конфліктують між собою. Втім, ця проблема може бути вирішена різними способами.

Постановка задачі

Складні задачі навчання з підкріпленням потребують детальнішого формування нагород агента, а зі зростанням кількості нагород, може збільшитися їх конфліктування між собою. Конфлікт між нагородами спричиняє локальні оптимуми. Очевидно, гірші локальні оптимуми мають менше сумарне значення підкріплення, а отже при збіганні результату до них, виконання поставленої задачі машинного навчання є менш успішним, що не є бажано.

Одним з найбільш популярних методів навчання з підкріпленням є *Proximal Policy Optimization*, розроблений *OpenAI* [3]. Проблема постає в наступному: метод *PPO* часто є схильним до сходження у локальні оптимуми [2]. Існують модифікації, що допомагають з цією проблемою [2], але вони не вирішують проблему локальних оптимумів повністю, а лише допомагають в дослідженні простору станів, щоб знайти кращі оптимуми. Крім того, не дивлячись на наявність таких модифікацій, *PPO* лишається є одним з найбільш застосовуваних методів через свою простоту та доступність в різних інструментах. Виникає потреба в прийомах, що дозволяють політиці поведінки агента надійніше збігатися до оптимального рішення. Існує велика кількість прийомів та технік для боротьби з локальними оптимумами, проте далеко не всі можуть підходити під поставлену задачу або бажану складність рішення.

Ось простий приклад такої ситуації зі сходженням до локальних оптимумів. Припустимо, що метод руху не містить спрощуючих елементів поведінки таких як *Central Pattern Generator (CPG)*, а навчається поведінці «з нуля», виконуючи випадкові рухи кінцівками у всіх 12 площинах керування (у випадку чотириногого робота). Така велика розмірність дій робить дослідження простору досить складним. Агенту-роботу надається підкріплення за рух вперед, і нараховується значення покарання за ковзання ніг землею, з метою навчити його крокувати правильно піднімаючи ноги без ковзання, щоб не шкодити кінцівки. В залежності налаштування нагороди, агент може або проігнорувати умову ковзання, маючи в поведінці одну ковзаючу ногу-підтримку, щоб стабілізувати свою траєкторію і отримати більшу нагороду за рівний рух, або ж проігнорувати підкріплення за рух вперед, і просто стояти, не отримуючи покарання за ковзання ніг (рис. 1). Спроба задоволення однієї нагороди може призвести до порушення іншої, тому агент застрягає в одному з локальних оптимумів.

Натомість, оптимальним рішенням є задовольнити обидва підкріплення, правильно крокуючи та рухаючись. Ця проблема може бути вирішена інакшим

 ISSN 1560-8956

налаштуванням нагород та покарань, але оскільки це може зайняти багато часу, наявність простішого способу була б дуже корисною.

Рисунок 1. Агент витягується вперед, після чого стоїть або падає, замість того, щоб навчитися крокувати вперед

Таким чином, задачею ставиться знайдення способу, що може зменшити вірогідність сходження результату до небажаних локальних оптимумів, при цьому використання способу повинно бути простішим ніж формування додаткових проміжних нагород та їх тонке налаштування, або вибір та налаштування іншого алгоритму навчання.

Огляд існуючих підходів та прийомів

Одним зі способів для боротьби з небажаними локальними оптимумами, до яких є вразливим *PPO*, є використання модифікацій цього методу, або зовсім іншого методу машинного навчання, з метою покращення здатності алгоритму досліджувати простір станів. Модифікації методу, такі як *PPO-CMA* дозволяють тимчасово збільшити ентропію, щоб краще дослідити простір навколо поточного оптимуму, що дозволяє дослідити сусідні оптимуми, а отже потенційно знайти кращий результат [2]. Методи максимальної ентропії та *off-policy* методи, такі як *Soft Actor-Critic* (*SAC)*, можуть виконати навчання ще краще, з меншою кількістю зразків станів [4]. Проте, вибір іншого методу навчання може мати недоліки, а саме складне налаштування гіперпараметрів або проблеми з сумісністю. Наприклад, *PPO-CMA* не є доступним у багатьох середовищах, таких як *Unity з ML Agents*, тому потребує додаткових складностей інтеграції. Також, переваги *SAC* над *PPO* можуть нівелюватися у випадках роботи з обмеженими обчислювальними ресурсами, або через специфіку задачі, наприклад через те, що *SAC* потребує більшої кількості оновлень моделі [5]. Саме тому, через свою простоту використання, *PPO* досі лишається одним з найбільш використовуваних методів навчання з підкріпленням.

Існують технології, що можна поєднати з методом навчання з підкріпленням, які додатково допомагають з дослідженням простору станів агента. Одним з таких механізмів є *Curiosity driven learning* [5], який змушує агента досліджувати нові стани на основі їх «незвичності». Проте метод призначений для застосування в умовах

розріджених нагород (*sparse rewards*). Те саме стосується методу *Random Network Distillation* [5].

Якщо природа підкріплення в задачі є конфліктна, деякі критерії можуть ігноруватися зовсім. Щоб агент однаково приділяв увагу одразу кільком критеріям, корисним може виявитися застосування добутку між нагородами, замість їх суми. Такий підхід, наприклад, можна побачити в документації *Unity ML Agents*, де для навчання людиноподібного, або павукоподібного агенту ходьбі використовується добуток нагород [6]. Таким чином навіть через нехтування лише одним з критеріїв, нагорода агента може впасти до нуля, а отже повне нехтування критерієм перестає бути локальним оптимумом. З недоліків прийому можна вважати потенційну втрату лінійності навчання.

Іншим способом є збільшення кількості агентів, та, відповідно, досвіду навчання, що спричиняє краще та швидше дослідження простору станів. Наприклад, у 2021 році *NVIDIA* створила високопаралелізоване середовище *Isaac Lab*, що дозволяє виконати швидке навчання роботів за хвилини, замість годин [7]. За допомогою цього засобу був навчений ходити *Anymal C* та інші відомі роботи. В той час як масивне паралізоване середовище дійсно допомагає зі знайденням оптимального результату та пришвидшенням тренування, цей засіб не завжди можна застосувати, через, наприклад, потребу у спеціальному навчальному середовищі, такому як віртуальне середовище гри, або навчання робота під час його реального використання, та використання інших середовищ симуляції роботів, таких як *Gazebo, Webots*. Також, для використання засобу необхідна порівняно велика кількість пам’яті графічного процесора, що у складних задачах може не бути доступним для звичайного користувача.

Корисним прийомом може виявитися *Curriculum Learning* [5, 8], що дозволяє виконати навчання для складної проблеми поетапно. Проблема декомпозується на значно простіші рівні, і передбачається що поступове навчання агента окремим частинам задачі є стабільнішим та іноді швидшим за навчання «всьому і одразу». Простіші рівні навчання мають менший простір корисних станів для дослідження, і при завершенні рівня та переході на наступний, агент має кращу стартову «точку» поведінки, а отже більші шанси виконати наступний рівень, з меншими зусиллями пошуку оптимуму. Отже, за допомогою цього прийому частково вирішується проблема дослідження простору. Разом з цим, в процесі навчання за допомогою прийому можна змінювати логіку нарахування підкріплення, що дозволяє зменшити конфліктність нагород, якщо вони потрібні на різних етапах навчання або в різних частинах задачі. Також підхід може передбачати динамічне керування рандомізації середовища. Можна провести аналогію *Curriculum Learning* з комп’ютерними іграми, де новим

 ISSN 1560-8956

гравцям прийнято починати з першого рівня, і тільки пізніше доходити до найскладнішого. З такою ж ідеєю підхід був застосований і в навчанні роботів *Anymal C* в *NVIDIA Isaac Lab* [7]. Недоліком прийому є потреба в плануванні та ручному налаштуванні рівнів навчання та їх змін.

Оптимальність рішення, отриманого за допомогою *on-policy* навчання з підкріпленням, такого як *PPO*, залежить від корисності зібраних алгоритмом даних. Якщо кількість корисного досвіду (стани якого знаходяться поблизу оптимальної поведінки) значно менша за непотрібний досвід агента, навчання буде виконуватися повільно. У випадку, якщо стартовий стан агента знаходиться серед станів з оптимальною нагородою, простір дослідження можна обмежити, тому корисним може виявитися метод *Termination Curriculum* [9]. Метод має на меті зменшення області досліджуваних станів, щоб мати лише більш корисні стани. Можна провести аналогію з *Curriculum Learning*, тільки тут змінюється не саме середовище, а область досліджуваних станів в ньому.

Розробка способу

В можливій поведінці агента, частина станів знаходиться в околі оптимальної поведінки, решта при цьому є поза нею, і часто є зовсім непотрібним досвідом [9]. З одного боку, детальне дослідження простору є дуже корисним для надійності методу руху та готовності до різних ситуацій, наприклад, якщо робот, навчений машинним навчанням, зустрічає ситуації, коли він перевертається, він має шанс навчитися оговтуватися від падінь, бо вихід з перевернутого стану поверне йому нагороду. З іншого боку, якщо не прийняти міри і не припинити епізод після падіння, велика кількість часу навчання може піти на цей самий перевернутий стан, коли робот рухає кінцівками і не може перевернутися, що може сповільнити або зупинити основний процес навчання ходьбі. Навчання ходьбі та оговтуванню від падінь можна розділити за принципом *Curriculum Learning*, тобто навчити робота поетапно таким діям. В такому випадку агент може швидше навчитися виконувати ці задачі, якщо сконцентрується на одній з них.

Так само, прикладами за межами оптимальної поведінки можна привести випадки, коли робот спотикається, порушує цикл ходьби, вивертає кінцівки, нахиляється. Найбільш суттєвою проблемою є задачі, що передбачають послідовну правильну, точну поведінку та відкладену нагороду, або нагороду, що акумулюється лише з часом. В таких випадках агент може надати перевагу більш моментальним нагородам, бо успішне детальне дослідження згаданих складних нагород не відбулося. Врешті-решт робот застрягає в локальному оптимумі.

Наприклад, задачею робота є слідування кінцівками заданої висоти під час ходьби (рис. 2). На рисунку зелені точки – задана висота ніг. На показаному прикладі робот не здатен слідувати точкам успішно.

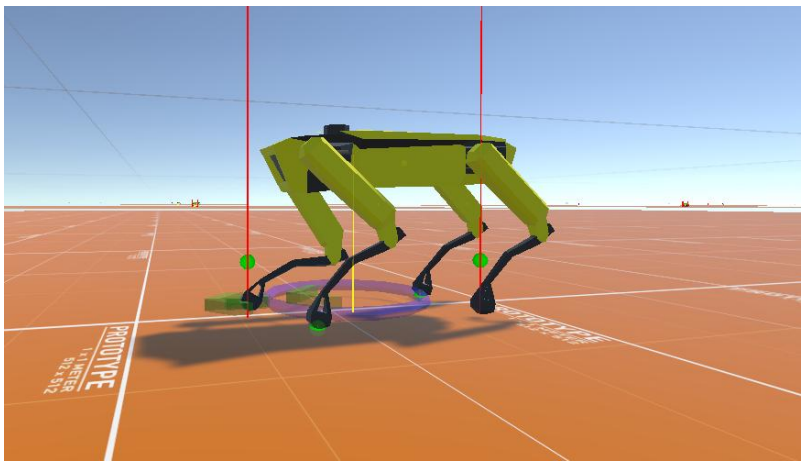

Рисунок 2. Агент не потрапляє в задані точки (в результаті навчання)

Оскільки дослідження заданої траєкторії ніг є досить складним та потребує послідовних правильних дій, які до того ж порушують баланс робота, а отже конфліктують з іншими нагородами такими як стабільність тіла робота, в цьому прикладі агент не знаходить оптимального рішення, і надає перевагу прямій стабільній походці, але за допомогою ковзання, замість крокування. Задачу ускладнює те, що в робота є 12 площин керування, і щоб задовольнити певну нагороду, потрібно виконати багато дій одночасно та злагоджено.

Можна вважати, що коренем проблеми є недосконале формування нагород, тобто задачу можна вирішити, якщо поставити її інакше, спеціально спланувавши простіші для агента нагороди, або ж використати інший метод замість *PPO*. Обидва рішення можуть потребувати часу та складних рішень від розробника. Втім, пропонований в статті спосіб дозволяє виконати задачу такого типу без змін її формулювання, тобто змін та налаштувань підкріплення, та з використанням *PPO*.

Запропоноване рішення передбачає встановлення обмеження на загальну щільну нагороду (*dense reward*) агента, яку він отримує кожен кадр симуляції. При порушенні встановленого порогу епізод навчання зупиняється. З часом навчання поріг зменшується зі встановленого значення ${R_0}^{min}$ до нуля. Щільні нагороди агента при цьому слід привести до проміжку [0; 1] та виконати над ними добуток замість суми, щоб заставити агента приділяти однакову увагу всім параметрам поведінки.

Схожим рішенням до пропонованого є *Termination Curriculum* [9], в якому ставиться динамічний поріг на загальне підкріплення агента, втім рішення фокусується більше на пришвидшенні навчання, а не обходженні поганих локальних оптимумів, тому також використовує такі техніки, як *Reference State Initialization*, щоб перенести початковий стан агента у випадкову позицію на траєкторії. Проте, це рішення в тому ж вигляді, на жаль не підходить для поставленої в цій статті задачі, де

 ISSN 1560-8956

рух робота повинен задовольнити багатьом конфліктним критеріям одночасно (також, *Reference State Initialization* використовуватися не буде через особливості задачі та складність імплементації, що йде всупереч поставленій умові простоти підходу). Проблема застосування підходу, аналогічного до *Termination Curriculum*, до задач зі складним формуванням нагороди полягає в тому, що якщо ставити поріг на всю сумарну нагороду одночасно, агент зможе ігнорувати окремі її частини (краще виконуючи решту складових нагороди, які є простіші), в таких випадках термінація епізоду виконуватися не буде. Це трапляється тому, що в оригінальному рішенні з *Termination Curriculum* нагорода агенту обчислюється сумою, та складається з кількох різних нагород. В рішенні бачимо нагороду як суму нагород за виконання завдання та імітацію зразкових рухів, що в свою чергу складається ще з 4 доданків.

Проблему здатне вирішити приведення нагород до проміжку [0; 1] та виконання добутку між нагородами, які агент отримує кожен кадр. При цьому оптимальне рішення повинно мати всі компоненти нагороди близькі до 1. На рис. 3 можемо побачити що буде, якщо використовувати звичайну суму нагород. Робот крокує на місці, ігноруючи рух (епізод не зупинився через ігнорування лише одного критерію).

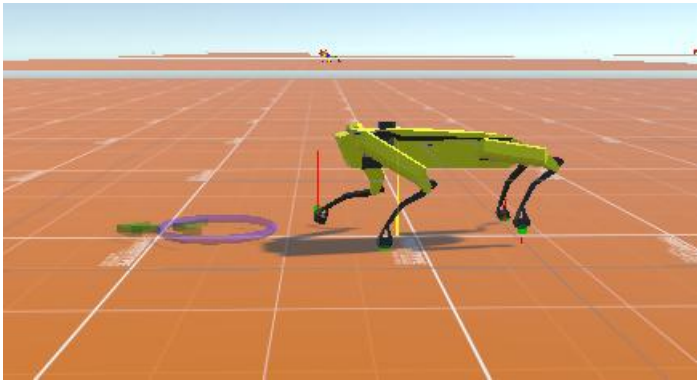

Рисунок 3. Агент ігнорує рух вперед, задовольняючи інші нагороди

Одночасно з цим, якщо використовувати добуток між нагородами (щоб агент більш збалансовано приділяв увагу різним підкріпленням), запропонований в *Termination Curriculum* високий поріг може бути неможливо застосувати до задачі через число множників нагороди (чим більше множників, тим менший добуток, бо всі множники менші за 1), велику кількість конфліктів між підкріпленнями, а також через рандомізацію середовища. Втім використання добутку нагород замість суми є дуже бажаним щоб зменшити кількість поганих локальних оптимумів. Таким чином, в пропонованому способі використовуються нижчі значення порогу.

Також, в умовах рандомізації середовища та багатьох складових-множників нагороди, повне уникнення помилок може бути або неможливе, або надто складне для навчання, тому їх варто дозволити, що не робить оригінальний підхід *Termination*

Curriculum, бо пропонує зменшення порогу з 0.75 до 0.5, а також використовує суму замість добутку. В задачі керування роботом це створює кілька проблем. По-перше, якщо неоптимальні стани до самого кінця навчання будуть термінуватися, робот так і не навчиться оговтуватися від невдач чи помилок, які так чи інакше стаються через рандомізацію середовища. По-друге, невеликою проблемою є те, що пікову продуктивність агента на задачі попросту не буде видно в результатах та на графіку під кінець навчання, бо в статистику входять терміновані епізоди з нижчою нагородою (якби епізод не термінувався, агент би продовжив виконання задачі і так чи інакше отримав більшу нагороду за наступні правильні дії). Тому пропонується до кінця навчання повністю прибирати рамки, щоб звільнити дії агента. Отже, значення порогу R_t^{min} під час епізоду буде набувати наступного значення.

$$R_t^{min} = R_0^{min} * \frac{S_{max} - s_t}{S_{max}}, \quad (1)$$

де

s_t – поточний крок навчання,

R_0^{min} – підібране значення порогу нагороди,

S_{max} – максимальний крок навчання.

Переглянемо застосування запропонованого способу на прикладі, вже проілюстрованому вище. Бачимо, що цього разу робот успішно слідує заданим точкам (рис. 4), що відображені зеленим кольором. Деталі результатів в наступному розділі.

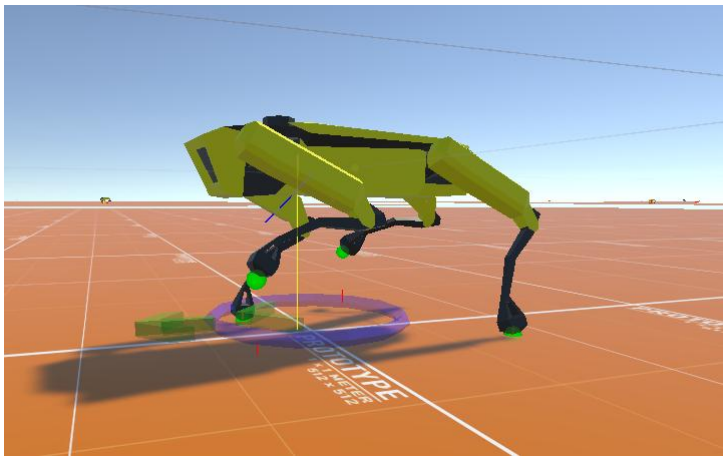

Рисунок 4. Робот успішно слідує заданим точкам

Умовами задачі також може бути поставлено, що на початку епізоду агент має стан, відхилений від оптимального, за значеннями нагороди. В цьому випадку спосіб в попередньому вигляді не можна застосувати з великим порогом, тому необхідні зміни. Для цього пропонується підібрати мале (близьке до мінімального) значення часу t_{max}, за яке агент фізично може досягнути околу оптимальних станів. Поставлений поріг ${R_0}^{min}$, в

 ISSN 1560-8956

такому випадку, можна помножити на значення $\min(\frac{t}{t_{\max}}, 1)$, де t – поточний час епізоду. Таким чином, поріг нагороди буде лінійно збільшений з нуля і до свого звичайного значення, що дасть агенту змогу досягти околу оптимальних станів за час t_{max}.

Отже, в останньому випадку поріг нагороди буде визначено наступним чином:

$$R_t^{min} = \min\left(\frac{t}{t_{\max}}, 1\right) * R_0^{min} * \frac{S_{max} - s_t}{S_{max}}, \quad (2)$$

де

t – поточний час в епізоді,

R_0^{min} – підібране значення порогу нагороди,

$t_{\max}$ – підібраний час досягнення оптимального стану,

s_t – поточний крок навчання,

S_{max} – максимальний крок навчання.

Маємо узагальнену схему застосування способу:

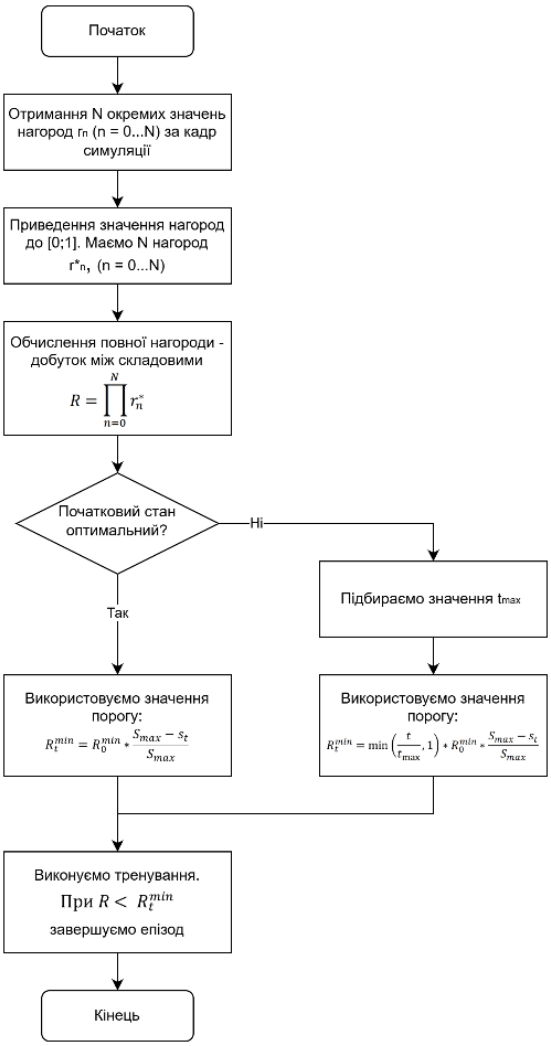

Рисунок 5. Схема застосування для запропонованого способу

Результати та обговорення

Для тестування способу оберемо цю саму задачу, в якій робот повинен слідувати різним точкам та одночасно рухатися вперед. Для даної задачі були поставлені наступні нагороди (табл.1):

Таблиця 1.

Підкріплення агенту для поставленої задачі

Підкріплення	Розрахунок
Слідування кінцівок певній висоті.	Покарання за одну кінцівку дорівнює дистанції до точки, діленій на амплітуду крокування. Обмежується до проміжку [0; 1], а значення обертається. Нагороди з різних кінцівок множимо між собою.
Нагорода за рівне положення тіла агента.	Дорівнює добутку нагород за правильний нахил та поворот тіла за трьома осями, а також за висоту тулуба.
Нагорода за слідування загальній точці руху тіла.	Дорівнює дистанції до точки, діленій на підібране значення (0.35 метра).

Дані категорії нагород приводимо до проміжку [0; 1], множимо між собою і маємо фінальну нагороду, що надається агенту кожен крок симуляції. В цьому конкретному прикладі нагороду було лінійно масштабовано (діленням на час кадру та множенням на 10), але вже після застосування порогу нагороди. Поріг R_t^{min} застосовано до нагороди, що знаходиться в проміжку [0; 1].

В різних тестах спробуємо значення порогу 0.2, 0.3, 0.4, а також спробуємо статичний поріг, для порівняння. Нагорода розраховується добутком у всіх показаних на наступному графіку випадках (при сумарній нагороді, навчання застрягає в локальних оптимумах в будь-якому випадку).

Тестування способу проводиться за допомогою засобів *UnityEngine, Unity ML Agents*. Наступний графік виведено за допомогою tensorboard, під’єднаного до середовища. З тестування з різними значеннями порогу для окремих категорій підкріплення, маємо наступні результати (рис. 6).

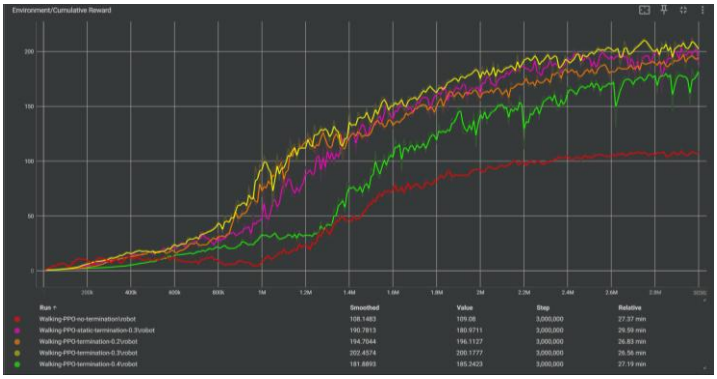

Рисунок 6. Результати навчання з різними значеннями порогу

 ISSN 1560-8956

Отримані фінальні результати також можемо побачити в табл. 2:

Таблиця 2.

Результати тестування

R^{min}_0	Досягнуте підкріплення
no termination	107.8
constant 0.3	181
0.2	196.1
0.3	***200.2***
0.4	178.7

Можна побачити, що застосування способу помітно позитивно впливає на результат навчання для такої задачі, за умови вибору правильного значення порогу термінування епізоду. З графіку видно, що найкращий результат досягається за допомогою порогу 0.3 (позначено жовтим кольором), а при вищих коефіцієнтах максимальна нагорода падає. Найгірший результат на графіку – спроба навчання без використання запропонованої зупинки епізоду, але з тим же принципом розрахунки нагород. Результати можуть варіюватися, в залежності від особливостей задачі, проте в задачах подібного типу схильність до покращення результату зберігається.

Візуально різницю в застосуванні та відсутності використання пропонованого способу, можна побачити між рис. 2 та рис. 4, що були показані раніше. Приведення щільних нагород до проміжку [0;1] та виконання добутку замість звичайної суми, може сприяти знаходженню кращого оптимума, бо обмежує простір станів агента від випадків, коли він ігнорує один або кілька складових підкріплення. Якщо цього не зробити, маємо поганий результат (рис. 3). Іншими словами, метод *Termination Curriculum* в своєму початковому вигляді гірше підходить для даної задачі, якщо не виконати необхідні зміни. Також, *Termination Curriculum* з високими значеннями порогу не може бути застосований у випадках, коли стан агента відхилений від оптимального. Для таких випадків запропонований спосіб пропонує множити поріг нагороди на множник, що збільшуватиметься лінійно з нуля до 1 за підібраний малий час t_{max}, що на початку епізоду дасть можливість агенту досягти околу нормальних станів.

Задачами, до яких гарно підходить запропонований спосіб, можна вважати такі, де агент швидко виходить за межі оптимальної поведінки і більшість процесу навчання проходить впусту через дослідження хибного простору станів. Спосіб дозволяє утримувати агента в околі потрібної поведінки. Необхідною умовою застосування способу є наявність щільних підкріплень (*dense rewards*) в задачі, що будуть нараховуватися агенту кожен кадр симуляції.

Висновки

В ході досліджень, описаних у цій статті, було розглянуто методи, що дозволяють підвищити успішність навчання з підкріпленням, а саме методу *Proximal Policy Optimization*, на прикладі задачі руху робота та слідування заданим точкам руху. Було поєднано різні прийоми та запропоновано спосіб динамічної зупинки епізоду, який здатен вирішувати поставлену задачу з конфліктними нагородами та складними локальними оптимумами, в той час як без його використання, задача не вирішується успішно за допомогою алгоритму *PPO*. Спосіб є простішим у застосуванні, на відміну від вибору іншого алгоритму машинного навчання, додавання чи перебудови нагород, та позбавляє від потреби більш детального дослідження простору більшими обчислювальними ресурсвми чи спеціальними середовищами навчання. Натомість задача виконується успішно саме через зосередження дослідження агентом корисного простору, в околі оптимальної поведінки.

Список використаних джерел

1. MathWorks, Inc. What Is Reinforcement Learning? [Електронний ресурс] // MathWorks.com. – 2020. – Режим доступу: https://www.mathworks.com/discovery/reinforcement-learning.html. – (дата звернення: 25.04.2025).

2. *Hämäläinen P., Babadi A., Ma X., Lehtinen J.* PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation [Електронний ресурс] // arXiv.org. – 2018. – Режим доступу: https://arxiv.org/pdf/1810.02541. – (дата звернення: 16.04.2025).

3. *Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O.* Proximal Policy Optimization Algorithms [Електронний ресурс] // arXiv.org. – 2017. – Режим доступу: https://arxiv.org/pdf/1707.06347. – (дата звернення: 16.04.2025).

4. *Haarnoja T., Zhou A., Abbeel P., Levine S.* Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor [Електронний ресурс] // arXiv.org. – 2018. – Режим доступу: https://arxiv.org/pdf/1801.01290. – (дата звернення: 16.04.2025).

5. Unity Technologies. ML-Agents Toolkit Overview [Електронний ресурс] // Unity-Technologies.github.io. – 2022. – Режим доступу: https://unity-technologies.github.io/ ml-agents/ML-Agents-Overview/. – (дата звернення: 16.04.2025).

6. Unity Technologies. Unity ML-Agents Toolkit: Example Learning Environments [Електронний ресурс] // Unity-Technologies.github.io. – 2022. – Режим доступу: https://unity-technologies.github.io/ml-agents/Learning-Environment-Examples/. – (дата звернення: 25.04.2025).

7. *Rudin N., Hoeller D., Reist P., Hutter M.* Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning [Електронний ресурс] // arXiv.org. – 2021. – Режим доступу: https://arxiv.org/pdf/2109.11978. – (дата звернення: 16.04.2025).

8. *Narvekar S., Peng B., Leonetti M., Sinapov J., Taylor M. E., Stone P.* Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey [Електронний ресурс] // arXiv.org. – 2020. – Режим доступу: https://arxiv.org/pdf/2003.04960. – (дата звернення: 16.04.2025).

9. *Babadi A., Naderi K., Hämäläinen P.* Self-Imitation Learning of Locomotion Movements through Termination Curriculum [Електронний ресурс] // arXiv.org. – 2019. – Режим доступу: https://arxiv.org/pdf/1907.11842. – (дата звернення: 16.04.2025).