

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

БОЧОК В'ЯЧЕСЛАВ ОЛЕКСАНДРОВИЧ

УДК 004.94

ДИСЕРТАЦІЯ
МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ДЕЦЕНТРАЛІЗОВАНОГО
ОБМІНУ ЗНАННЯМИ В СИСТЕМАХ БАГАТОАГЕНТНОГО НАВЧАННЯ З
ПІДКРІПЛЕННЯМ

121 – Інженерія програмного забезпечення

12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Бочок В.О.

Науковий керівник: Федорова Наталія Володимирівна, доктор технічних наук,
професор

Київ – 2026

АНОТАЦІЯ

Бочок В.О. Методи та програмні засоби децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії з галузі знань 12 Інформаційні технології за спеціальністю 121 Інженерія програмного забезпечення. – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2026.

Дисертаційна робота присвячена розробці науково-методичного апарату щодо підвищення продуктивності багатоагентної системи слабо пов'язаних агентів у стаціонарному середовищі за допомогою механізму поширення знань між агентами на базі децентралізованого підходу.

Багатоагентні системи є важливим напрямом сучасних інформаційних технологій та штучного інтелекту, оскільки дозволяють моделювати складні розподілені процеси у вигляді сукупності автономних агентів, що взаємодіють між собою та з середовищем. Такі системи знаходять широке застосування у промисловості, робототехніці, транспортних системах, фінансовому аналізі, системах підтримки прийняття рішень та комп'ютерних симуляціях.

Одним з ключових механізмів адаптації агентів у таких системах є навчання з підкріпленням, яке дозволяє агентам формувати ефективні стратегії поведінки на основі взаємодії із середовищем. Однак сучасні алгоритми навчання з підкріпленням характеризуються високою складністю навчання, значною потребою у взаємодіях із середовищем (sample complexity), чутливістю до гіперпараметрів та нестабільністю процесу збіжності. Ці проблеми посилюються при переході до багатоагентних систем, де декілька агентів навчаються одночасно.

Розвитком навчання з підкріпленням є багатоагентне навчання з підкріпленням (Multi-Agent Reinforcement Learning, MARL), у якому декілька агентів взаємодіють із середовищем паралельно. У таких системах навіть у випадку слабкої взаємодії між агентами виникають проблеми неефективного

використання досвіду, оскільки кожен агент накопичує знання незалежно, що призводить до дублювання процесу навчання. Це, у свою чергу, збільшує обчислювальні витрати та уповільнює досягнення оптимальних стратегій. Додатковою проблемою є відсутність ефективних механізмів узагальнення знань між агентами, що працюють у подібних або ідентичних умовах. У результаті кожен агент змушений самотійно проходити всі етапи навчання, навіть якщо інші агенти вже отримали релевантний досвід. Це призводить до неефективного використання накопиченої інформації та знижує загальну швидкість збіжності системи. Це обмеження стає особливо критичним у великомасштабних системах, де кількість агентів зростає, а витрати на незалежне навчання збільшуються пропорційно.

Перспективним напрямом підвищення ефективності навчання є використання механізмів передачі знань між агентами. Обмін знаннями дозволяє повторно використовувати досвід, накопичений різними агентами, що потенційно скорочує час навчання та підвищує загальну продуктивність системи. Однак застосування таких підходів пов'язане з низкою проблем, серед яких ризик негативної передачі знань (*negative transfer*), коли менш ефективні агенти передають некоректні або застарілі знання, відсутність механізмів контролю якості переданих знань, а також складність адаптивного регулювання інтенсивності обміну. Крім того, значна частина існуючих підходів базується на централізованих архітектурах, що обмежує їх застосування у розподілених системах зі слабо пов'язаними агентами. Таким чином, виникає необхідність у розробленні підходів, що поєднують переваги обміну знаннями з можливістю децентралізованого прийняття рішень, забезпечуючи при цьому контроль якості переданих знань та адаптивне регулювання інтенсивності їх використання кожним агентом. Відтак, проектування децентралізованих підходів, що дозволяють агентам самотійно визначати момент та спосіб використання зовнішніх знань, є ключовою вимогою для підвищення ефективності навчання без додаткових витрат на координацію.

Питання розробки та оптимізації багатоагентних систем, а також методів їх навчання досліджували такі вчені, як Richard S. Sutton, Andrew G. Barto, Michael Wooldridge, Nicholas R. Jennings, Yoav Shoham, Katia Sycara, Victor Lesser, Michael L. Littman, Peter Stone, Коваль О.В., Кочура Ю. П., Аврунін О. Г. та інші. Запропоновані ними підходи заклали фундамент сучасних досліджень у галузі багатоагентних систем та навчання з підкріпленням. Водночас більшість існуючих методів або орієнтовані на централізоване управління агентами, або не враховують необхідність контролю якості переданих знань. Крім того, значна частина підходів потребує значних обчислювальних ресурсів або не забезпечує достатньої адаптивності при зміні умов середовища, що обмежує їх практичне застосування у децентралізованих системах.

Окреслене сформувало протиріччя між потребою підвищення ефективності навчання агентів у багатоагентних системах та можливістю використання передачі знань між агентами для прискорення навчання, що ускладнюється ризиком негативної передачі знань і відсутністю механізмів контролю інтенсивності такого обміну. Це призводить до збільшення часу навчання, зростання обчислювальних витрат та зниження загальної ефективності системи. Відповідне протиріччя може бути вирішене шляхом розроблення методів і програмних засобів децентралізованого обміну знаннями між агентами, що забезпечують адаптивний контроль якості та інтенсивності передачі знань.

Об'єктом дослідження є процеси аналізу, розроблення, забезпечення якості та впровадження методів та програмних засобів децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами.

Предметом дослідження є методи та програмне забезпечення децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами.

Метою дисертаційної роботи є підвищення продуктивності багатоагентної системи слабо пов'язаних агентів у стаціонарному середовищі за допомогою механізму поширення знань між агентами на базі децентралізованого підходу.

Наукова новизна результатів дослідження полягає в наступному.

Набув подальшого розвитку метод децентралізованого вибору вчителя для передачі знань, що дозволяє контролювати якість поширюваних знань уникаючи негативного переносу.

Вперше розроблено метод динамічного децентралізованого контролю інтенсивності передачі знань між агентами, що дозволяє ефективно використовувати набуті системою знання та протидіяти перенавчанню.

Вперше розроблено архітектуру програмного забезпечення для реалізації механізму обміну знаннями в системі багатоагентного навчання, характерною особливістю якої є можливість децентралізованого обміну знаннями із застосуванням механізмів контролю якості та інтенсивності передачі.

Практичне використання результатів роботи, розроблених методів та програмних засобів дозволило підвищити продуктивність багатоагентної системи за метрикою нормалізованої площі під кривою навчання (normalized AUC) сумарної винагороди за епізод на 4–20% для середньої інтегральної продуктивності системи та на 3,4–18,3% для максимальної продуктивності окремого агента залежно від конфігурації механізму обміну знаннями та режиму автономного навчання.

Експериментальні дослідження проведено у тестових середовищах CartPole-v1 та LunarLander-v3 бібліотеки Gym/Gymnasium. Запропонований механізм обміну знаннями продемонстрував результативність як для алгоритмів Deep Q-Network (DQN), так і Double Deep Q-Network (DDQN).

Результати дослідження прийнято до впровадження в ТОВ «Квалітек» (акт впровадження від 20.03.2026); в навчальному процесі Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (акт впровадження від 20.03.2026 р.) при викладанні дисциплін «Проектування кібер-фізичних систем».

Наукові результати дослідження є внеском у розвиток теоретичних і прикладних основ підвищення ефективності навчання у багатоагентних системах навчання з підкріпленням шляхом розроблення методів контролю якості та

інтенсивності передачі знань і програмних засобів децентралізованого обміну знаннями між слабо пов'язаними агентами у стаціонарних середовищах.

Отримані результати розширюють підходи до організації навчання у децентралізованих багатоагентних системах та створюють передумови для подальшого розвитку методів масштабованого навчання агентів.

В якості можливих напрямків продовження дослідження можна відмітити розширення запропонованих методів децентралізованого обміну знаннями на інші алгоритми навчання з підкріпленням та різні типи багатоагентних середовищ (зокрема кооперативні та змішані), дослідження альтернативних підходів до оцінювання продуктивності агентів, а також розроблення механізмів оцінки різноманітності досвіду агентів для підвищення ефективності передачі знань.

Ключові слова: багатоагентні системи, навчання з підкріпленням, підвищення продуктивності багатоагентної системи, передача знань між агентами, контроль інтенсивності передачі знань, глибоке Q-навчання, дистиляція політики, дистиляція знань, штучний інтелект, машинне навчання, нейронні мережі.

ANNOTATION

Bochok V.O. Methods and Software Tools for Decentralized Knowledge Sharing in Multi-Agent Reinforcement Learning Systems. – Qualification scientific work as a manuscript.

Dissertation for the degree of Doctor of Philosophy in the field of knowledge 12 Information Technology, specialty 121 Software Engineering. – National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, 2026.

The dissertation is devoted to the development of a scientific and methodological framework for improving the performance of a multi-agent system of weakly coupled agents in a stationary environment through a decentralized knowledge sharing mechanism between agents.

Multi-agent systems are an important area of modern computer science and artificial intelligence, as they enable modeling complex distributed processes as a set of autonomous agents interacting with each other and with the environment. Such systems are widely applied in industry, robotics, transportation systems, financial analysis, decision support systems, and computer simulations.

One of the key mechanisms for agent adaptation in such systems is reinforcement learning, which allows agents to develop effective behavioral strategies through interaction with the environment. However, modern reinforcement learning algorithms are characterized by high training complexity, significant sample complexity, sensitivity to hyperparameters, and instability of the convergence process. These issues become even more pronounced in multi-agent systems, where multiple agents are trained simultaneously. In addition, reinforcement learning algorithms often require extensive tuning of hyperparameters and careful design of reward functions, which complicates their practical application. In multi-agent settings, these challenges are further amplified due to the simultaneous learning processes of multiple agents, even in stationary environments with weak inter-agent dependencies.

An extension of reinforcement learning is Multi-Agent Reinforcement Learning (MARL), in which multiple agents interact with the environment in parallel. Even in

scenarios with weak interaction between agents, such systems suffer from inefficient use of experience, since each agent accumulates knowledge independently, leading to duplication of the learning process. This, in turn, increases computational costs and slows down convergence to optimal strategies. An additional challenge is the lack of effective mechanisms for generalizing knowledge between agents operating in similar or identical conditions. As a result, each agent must independently go through all stages of learning, even when other agents have already acquired relevant experience, which reduces overall system efficiency. This limitation becomes particularly critical in large-scale systems, where the number of agents increases, and the cost of independent learning grows proportionally. Without mechanisms for sharing and reusing knowledge, the scalability of such systems remains limited.

A promising direction for improving learning efficiency is the use of knowledge sharing mechanisms between agents. Knowledge exchange enables reuse of experience accumulated by different agents, potentially reducing training time and improving overall system performance. However, such approaches are associated with several challenges, including the risk of negative transfer, when less effective agents propagate incorrect or outdated knowledge, the lack of mechanisms for controlling the quality of transferred knowledge, and the difficulty of adaptively regulating the intensity of knowledge exchange. Furthermore, many existing approaches rely on centralized architectures, which limits their applicability in distributed systems with weakly coupled agents. Therefore, there is a need to develop approaches that combine the advantages of knowledge sharing with decentralized decision-making, while ensuring quality control of transferred knowledge and adaptive regulation of its usage by each agent. Therefore, designing decentralized approaches that allow agents to independently decide when and how to use external knowledge becomes a key requirement for improving learning efficiency without introducing additional coordination overhead.

The problems of development and optimization of multi-agent systems and their learning methods have been studied by such researchers as Richard S. Sutton, Andrew G. Barto, Michael Wooldridge, Nicholas R. Jennings, Yoav Shoham, Katia Sycara, Victor Lesser, Michael L. Littman, Peter Stone, O.V. Koval,

Yu.P. Kochura, O.H. Avrunin, and others. Their work has laid the foundation for modern research in multi-agent systems and reinforcement learning. However, most existing methods are either focused on centralized control or do not consider the need for controlling the quality of transferred knowledge. In addition, many approaches require significant computational resources or lack adaptability under changing environmental conditions, which limits their practical applicability in decentralized systems.

The above considerations reveal a contradiction between the need to improve learning efficiency in multi-agent systems and the possibility of using knowledge sharing to accelerate learning, which is complicated by the risk of negative transfer and the lack of mechanisms for controlling the intensity of such exchange. This leads to increased training time, higher computational costs, and reduced overall system efficiency. This contradiction can be resolved by developing methods and software tools for decentralized knowledge sharing that provide adaptive control of both the quality and intensity of knowledge transfer.

The object of research is the processes of analysis, design, quality assurance, and implementation of methods and software tools for decentralized knowledge exchange in multi-agent reinforcement learning systems operating in stationary environments with weakly coupled agents.

The subject of research is methods and software for decentralized knowledge sharing in multi-agent reinforcement learning systems in stationary environments with weakly coupled agents.

The aim of the dissertation is to improve the performance of a multi-agent system of weakly coupled agents in a stationary environment through a decentralized knowledge sharing mechanism.

The proposed solutions address key limitations of existing approaches by introducing adaptive and decentralized mechanisms for knowledge exchange between agents.

The scientific novelty of the obtained results is as follows.

The method of decentralized teacher selection for knowledge transfer has been further developed, which allows controlling the quality of propagated knowledge and avoiding negative transfer.

For the first time, a method of dynamic decentralized control of knowledge sharing intensity between agents has been developed, which enables efficient use of acquired knowledge and prevents overfitting.

For the first time, a software architecture for implementing knowledge sharing mechanisms in multi-agent reinforcement learning systems has been developed. Its distinctive feature is the support for decentralized knowledge exchange combined with mechanisms for controlling the quality and intensity of knowledge transfer.

The practical application of the developed methods and software tools has made it possible to improve the performance of a multi-agent system in terms of normalized area under the learning curve (normalized AUC) of cumulative reward per episode by 4–20% for average system performance and by 3,4–18,3% for the maximum performance of an individual agent, depending on the configuration of the knowledge sharing mechanism and the autonomous learning regime.

Experimental studies were conducted in the CartPole-v1 and LunarLander-v3 environments of the Gym/Gymnasium library. The proposed knowledge sharing mechanism demonstrated effectiveness for both Deep Q-Network (DQN) and Double Deep Q-Network (DDQN) algorithms, confirming its general applicability. The results demonstrated consistent improvements across environments of varying complexity, confirming the robustness of the proposed approach under different experimental conditions.

The research results have been implemented at TOB «Квалітек» (implementation act dated 20.03.2026) and in the educational process of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute” (implementation act dated 20.03.2026) in teaching the course “Design of Cyber-Physical Systems”.

The obtained scientific results contribute to the development of theoretical and applied foundations for improving learning efficiency in multi-agent reinforcement learning systems by introducing methods for controlling the quality and intensity of

knowledge transfer and software tools for decentralized knowledge exchange between weakly coupled agents in stationary environments.

The results expand existing approaches to learning organization in decentralized multi-agent systems and create prerequisites for further development of scalable agent learning methods. The proposed framework establishes a foundation for applying decentralized knowledge sharing techniques in practical scenarios where agents operate independently, such as distributed monitoring and autonomous resource management.

Future research directions include extending the proposed decentralized knowledge sharing methods to other reinforcement learning algorithms and different types of multi-agent environments (including cooperative and mixed settings), investigating alternative approaches to agent performance evaluation, and developing mechanisms for assessing experience diversity to improve knowledge transfer efficiency.

Keywords: multi-agent systems, reinforcement learning, improving multi-agent system performance, knowledge transfer between agents, control of knowledge transfer intensity, deep Q-learning, policy distillation, knowledge distillation, artificial intelligence, machine learning, neural networks.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Бочок В., Федорова Н. Багатоагентні системи та проблеми їх оптимізації. Вчені Записки Таврійського національного університету імені В.І. Вернадського. Серія. Технічні науки. 2023. Том 34 (73) № 2. С.131-137. <https://doi.org/10.32782/2663-5941/2023.2.1/21>

Внесок авторів публікації: **Бочок В.:** аналіз літературних джерел, систематизація підходів до оптимізації багатоагентних систем, написання статті. **Федорова Н.:** наукове керівництво, формулювання концепцій дослідження та редагування рукопису.

2. Бочок В., Федорова Н. Централізоване навчання для deep Q-learning моделей. Інформаційні технології та суспільство. 2024. Вип. 2 (13). С. 6-11. <https://doi.org/10.32689/maup.it.2024.2.1>

Внесок авторів публікації: **Бочок В.:** розробка та реалізація методів децентралізованого вибору вчителя і динамічного контролю інтенсивності передачі знань, проведення експериментів, написання статті. **Федорова Н.:** наукове керівництво, аналіз результатів, перевірка та редагування рукопису.

3. Бочок В., Федорова Н. Обмін знаннями в Independent DQN багатоагентній системі та особливості його реалізації. Записки Таврійського національного університету імені В.І. Вернадського. Серія. Технічні науки. 2025. Том 36 (75) № 3. С. 113-119. <https://doi.org/10.32782/2663-5941/2025.3.2/15>

Внесок авторів публікації: **Бочок В.:** розробка архітектури програмного забезпечення для децентралізованого обміну знаннями з механізмами контролю якості та інтенсивності, написання статті. **Федорова Н.:** наукове керівництво, валідація архітектури, аналіз результатів, перевірка та редагування рукопису.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. Бочок В., Федорова Н. Обмін досвідом між агентами в багатоагентній системі. Матеріали І-ї міжнародної науково – практичної конференції «Сучасні аспекти інженерії програмного забезпечення». – м. Київ, 14 грудня 2023 р. С.92
Внесок авторів публікації: **Бочок В.:** аналіз літературних джерел з оптимізації багатоагентних систем, написання тез. **Федорова Н.:** наукове керівництво, редагування рукопису.

5. Бочок В., Федорова Н. Обмін та збереження інформації між агентами, здатними до навчання. Збірник матеріалів III Міжнародної науково-технічної конференції “Системи і технології зв’язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку”, 30 листопада 2023 року, м. Київ. Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, 2023. С. 89.

Внесок авторів публікації: **Бочок В.:** аналіз методів обміну та збереження інформації в багатоагентних системах, написання тез. **Федорова Н.:** наукове керівництво, редагування рукопису.

6. Бочок В., Федорова Н. Оптимізація багатоагентних систем. XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики» / XXI International scientific and practical conference of young scientists and students "Modern problems of scientific support of power engineering". 2024

Внесок авторів публікації: **Бочок В.:** аналіз літературних джерел з оптимізації багатоагентних систем, написання тез. **Федорова Н.:** наукове керівництво, редагування рукопису.

7. Бочок В., Федорова Н. Визначення якісної та кількісної різниці в досвіді агентів для його передачі. XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики». 2024.

Внесок авторів публікації: **Бочок В.:** теоретичне обґрунтування та розробка підходів до контролю якості й інтенсивності передачі досвіду між агентами, написання тез. **Федорова Н.:** наукове керівництво, редагування рукопису.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.....	18
ВСТУП.....	19
РОЗДІЛ 1. АНАЛІЗ ПІДХОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ НАВЧАННЯ АГЕНТІВ	26
1.1 Підходи до підвищення ефективності навчання агентів у системах навчання з підкріпленням	26
1.2 Багатоагентне навчання з підкріпленням.....	29
1.3 Методи передачі знань у системах навчання з підкріпленням	32
1.3.1 Передача досвіду та повторне використання знань	32
1.3.2 Дистиляція політики та передача моделей	34
1.3.3 Передача знань через поради та навчальні стратегії	35
1.3.4 Методи колективного поширення знань	35
1.3.5 Обмеження існуючих підходів передачі знань	36
1.4 Аналіз вимог до програмного забезпечення та постановка наукового завдання.....	37
1.5 Висновки до розділу	41
РОЗДІЛ 2. ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ СИСТЕМИ СЛАБКО ПОВ'ЯЗАНИХ АГЕНТІВ ШЛЯХОМ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ЗНАННЯМИ	43
2.1 Формалізація задачі навчання з підкріпленням	43
2.2 DQN як модель незалежного агента.....	48
2.2.1 Буфер досвіду як механізм декореляції.....	51
2.2.2 Онлайн та цільова мережі	52
2.2.3 Independent DQN у слабо пов'язаній системі	54

2.2.4 Модифікації DQN	56
2.2.6 Архітектура нейромережі	60
2.3 Поширення знань за допомогою дистиляції знань	60
2.4 Метод децентралізованого вибору вчителя для контролю якості передачі знань.....	65
2.5 Метод динамічного децентралізованого вибору кількості епох дистиляції політики для контролю сили передачі знань	71
2.6 Поєднання методів контролю якості та інтенсивності передачі знань	76
2.7 Висновки до розділу	77
РОЗДІЛ 3. РОЗРОБКА МЕТОДУ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ЗНАННЯМИ З МЕХАНІЗМАМИ КОНТРОЛЮ ЯКОСТІ ТА ІНТЕНСИВНОСТІ ПЕРЕДАЧІ	79
3.1 Методологія експериментального дослідження	79
3.1.1 Формулювання дослідницьких гіпотез	80
3.1.2 Опис середовищ та постановка задачі	81
3.1.3 Метрики оцінювання	83
3.2 Архітектура та реалізація експериментальної системи	86
3.2.1 Архітектура IDQN-агента.....	86
3.2.2 Реалізація навчання з механізмом обміну знаннями	89
3.3 Налаштування гіперпараметрів та стратегія підбору	90
3.4 Результати експериментального дослідження.....	94
3.4.1 Середовище CartPoleV1	95
3.4.2 Середовище LunarLanderV3	101
3.4.3 Аналіз та інтерпретація результатів	105
3.5 Висновки до розділу	109

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОЇ АРХІТЕКТУРИ СИСТЕМИ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ЗНАННЯМИ У БАГАТОАГЕНТНОМУ НАВЧАННІ З ПІДКРІПЛЕННЯМ.....	111
4.1 Вимоги до системи реалізації методів	111
4.1.1 Функціональні вимоги	112
4.1.2 Нефункціональні вимоги	114
4.1.3 Архітектурні обмеження задачі	116
4.2 Архітектурна модель системи реалізації методів	116
4.3 Модель інтеграції у зовнішні системи	122
4.3.1 Інтеграція із середовищами навчання	124
4.3.2 Інтеграція з фреймворками навчання з підкріпленням	125
4.3.3 Інтеграція з бібліотеками глибокого навчання	126
4.4 Аналіз застосовності	127
4.4.1 Переваги архітектури	127
4.4.2 Класи задач, для яких доцільне використання запропонованого підходу	129
4.4.3 Особливості використання запропонованого підходу	130
4.4.4 Обмеження застосування.....	131
4.4.5 Потенціал подальшого розвитку.....	132
4.5 Платформа для проведення експериментів	133
4.5.1 Архітектура системи запуску експериментів	134
4.5.2 Реалізація системи запуску експериментів.....	136
4.6 Висновок до розділу.....	137
ВИСНОВКИ.....	140
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	144

ДОДАТОК А СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА	156
ДОДАТОК Б АКТИ ВПРОВАДЖЕННЯ.....	158
ДОДАТОК В ЛІСТИНГ КОДУ ПРОГРАМНИХ РЕАЛІЗАЦІЙ	161

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AUC – Area Under Curve (площа під кривою).

DDQN – Double Deep Q-Network (подвійна глибока Q-мережа).

DQN – Deep Q-Network (глибока Q-мережа).

DL – Deep Learning (глибоке навчання).

RL – Reinforcement Learning (навчання з підкріпленням).

DRL – Deep Reinforcement Learning (глибоке навчання з підкріпленням).

Gym / Gymnasium – бібліотека тестових середовищ для задач навчання з підкріпленням.

MARL – Multi-Agent Reinforcement Learning (багатоагентне навчання з підкріпленням).

MDP – Markov Decision Process (марковський процес прийняття рішень).

ML – Machine Learning (машинне навчання).

NN – Neural Network (нейронна мережа).

ReLU – Rectified Linear Unit (випрямлена лінійна функція активації).

ПЗ – програмне забезпечення.

ВСТУП

Актуальність теми дослідження. Багатоагентні системи є одним з важливих напрямів сучасної інформатики та штучного інтелекту, оскільки дозволяють моделювати складні розподілені процеси у вигляді сукупності автономних інтелектуальних агентів, що взаємодіють між собою та з середовищем [1, 2]. Такі системи застосовуються у широкому спектрі практичних задач, зокрема в промислових системах, робототехніці, транспорті, соціально-економічних моделях, системах підтримки прийняття рішень та комп'ютерних іграх [3–10].

Однією з ключових характеристик багатоагентних систем є можливість розподіленого прийняття рішень без єдиної точки відмови, що підвищує їхню надійність та стійкість до збоїв окремих компонентів [11]. Крім того, агентний підхід дозволяє розглядати складну систему як набір взаємодіючих інтелектуальних компонентів, здатних до адаптації та вдосконалення власних стратегій поведінки. Одним з найбільш ефективних механізмів такої адаптації є навчання з підкріпленням (Reinforcement Learning, RL), у межах якого агенти формують політику поведінки на основі сигналів винагороди, отриманих під час взаємодії з середовищем [6]. Подальшим розвитком цього підходу є багатоагентне навчання з підкріпленням (Multi-Agent Reinforcement Learning, MARL), у якому декілька агентів навчаються одночасно, взаємодіючи з одним середовищем або з кількома подібними середовищами [12].

Попри значний розвиток алгоритмів навчання з підкріпленням, підвищення ефективності навчання агентів залишається актуальною проблемою. У складних задачах агенти потребують великої кількості взаємодій з середовищем для досягнення прийняттого рівня продуктивності, що призводить до значних витрат часу та обчислювальних ресурсів [6, 12]. Додатково у багатоагентних системах виникає проблема дублювання досвіду, коли кілька агентів незалежно досліджують подібні стани або повторно вивчають однакові закономірності, не використовуючи результати навчання один одного.

Одним з перспективних напрямів підвищення ефективності навчання є використання передачі знань між агентами. Завдяки обміну досвідом агенти можуть використовувати результати навчання інших агентів, що дозволяє прискорити процес формування ефективних стратегій та покращити загальну ефективність функціонування системи [12]. Передача знань може здійснюватися через різні механізми, зокрема обмін досвідом, копіювання параметрів моделей або дистиляцію політик.

Разом із тим застосування передачі знань між агентами пов'язане з низкою суттєвих проблем. Однією з основних є негативна передача знань, коли передача інформації від менш ефективного агента може призводити до погіршення результатів навчання інших агентів. Іншою проблемою є відсутність контролю інтенсивності передачі знань, унаслідок чого вплив зовнішнього досвіду може бути або надмірним, або недостатнім. Крім того, значна частина існуючих підходів орієнтована на централізовані архітектури навчання, що обмежує їх застосування у системах зі слабко пов'язаними агентами.

Питання розробки та оптимізації багатоагентних систем, а також методів їх навчання досліджували такі вчені, Richard S. Sutton, Andrew G. Barto, Michael Wooldridge, Nicholas R. Jennings, Yoav Shoham, Katia Sycara, Victor Lesser, Michael L. Littman, Peter Stone, Коваль О.В., Кочура Ю. П., Аврунін О. Г. та інші. Запропоновані ними підходи заклали фундамент сучасних досліджень у галузі багатоагентних систем та навчання з підкріпленням. Водночас значна частина існуючих методів має переважно теоретичний характер або потребує значних обчислювальних ресурсів, що ускладнює їх практичне застосування, особливо в умовах розподілених або децентралізованих систем [12].

Таким чином, виникає протиріччя між потребою підвищення ефективності навчання агентів у багатоагентних системах та можливістю використання передачі знань між агентами для прискорення навчання, що ускладнюється ризиком негативного передавання знань та відсутністю механізмів контролю його інтенсивності.

Актуальним є вирішення наукового завдання з розробки науково-методичного апарату щодо підвищення продуктивності багатоагентної системи слабо пов'язаних агентів у стаціонарному середовищі за допомогою механізму поширення знань між агентами на базі децентралізованого підходу.

Зв'язок роботи з науковими програмами, планами, темами.

Дисертаційне дослідження відповідає вимогам статті 5 Закону України «Про пріоритетні напрями розвитку науки і техніки» від 11 липня 2001 року № 2623-III (зі змінами та доповненнями від 12.01.2023 р.), пункту першого розділу другого «Переліку пріоритетних тематичних напрямів наукових досліджень і науково-технічних розробок на період до 2023 року», затвердженого Постановою КМУ від 7 вересня 2011 р. № 942. Дисертаційна робота виконана відповідно з поточними та перспективними планами наукової та науково-технічної діяльності Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» для подальшого розвитку інженерії програмного забезпечення.

Результати дисертаційної роботи є складовими науково-дослідних робіт, а саме: “Програмне забезпечення системи підтримки прийняття рішень забезпечення техногенно-екологічної безпеки” (РК № 0121U109761) та «Розробка алгоритмів і методів збору та обробки великих даних для оцінки параметрів діяльності організації» (РК № 0121U110722), в яких автор приймав особисту участь.

Метою дисертаційної роботи є підвищення продуктивності багатоагентної системи слабо пов'язаних агентів у стаціонарному середовищі за допомогою механізму поширення знань між агентами на базі децентралізованого підходу.

Для досягнення поставленої мети необхідно виконати наступні часткові завдання:

- провести аналіз існуючих методів навчання з підкріпленням для індивідуальних агентів та багатоагентних систем, а також відповідних алгоритмічних реалізацій;

- провести аналіз існуючих підходів до застосування механізмів обміну знаннями між агентами у багатоагентних системах;
- дослідити вплив обміну знаннями між слабо пов'язаними агентами на середній показник ефективності роботи багатоагентної системи у різних конфігураціях;
- розробити метод контролю якості обміну знаннями між агентами з метою запобігання або зменшення негативної передачі знань;
- розробити метод контролю інтенсивності обміну знаннями між агентами для прискорення поширення знань зі збереженням індивідуальних стратегій навчання агентів;
- розробити архітектуру програмного забезпечення для реалізації механізму децентралізованого обміну знаннями з метою підвищення ефективності навчання та функціонування системи слабо пов'язаних агентів;
- розробити програмний метод реалізації децентралізованого обміну знаннями між агентами із застосуванням механізмів контролю якості та інтенсивності передачі знань.

Об'єктом дослідження є процеси аналізу, розроблення, забезпечення якості та впровадження методів та програмних засобів децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами.

Предметом дослідження є методи та програмне забезпечення децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами.

Методи дослідження. Для досягнення поставленої мети використано методи системного, алгоритмічного та порівняльного аналізу — для дослідження існуючих підходів і постановки наукового завдання. Для побудови моделей багатоагентного навчання застосовано методи математичного моделювання та навчання з підкріпленням, зокрема підхід глибокого Q-навчання. Для розроблення механізмів обміну знаннями та відповідної програмної архітектури використано методи проєктування алгоритмів і програмних систем. Оцінювання ефективності

запропонованих методів здійснювалося з використанням експериментальних досліджень і методів математичної статистики на основі показників сумарної винагороди та нормалізованої площі під кривою навчання. Для перевірки результатів використано порівняльний аналіз із базовими підходами.

Наукова новизна одержаних результатів:

1. Набув подальшого розвитку метод децентралізованого вибору вчителя для передачі знань, що дозволяє контролювати якість поширюваних знань уникаючи негативного переносу.

2. Вперше розроблено метод динамічного децентралізованого контролю інтенсивності передачі знань між агентами, що дозволяє ефективно використовувати набуті системою знання та протидіяти перенавчанню.

3. Вперше розроблено архітектуру програмного забезпечення для реалізації механізму обміну знаннями в системі багатоагентного навчання, характерною особливістю якої є можливість децентралізованого обміну знань з застосуванням механізмів контролю якості та інтенсивності передачі.

Практичне використання результатів роботи, розроблених методів та програмних засобів дозволило підвищити продуктивність багатоагентної системи за метрикою нормалізованої площі під кривою навчання (normalized AUC) сумарної винагороди за епізод на 4–20% для середньої інтегральної продуктивності системи та на 3,4–18,3% для максимальної продуктивності окремого агента залежно від конфігурації механізму обміну знаннями та режиму автономного навчання. Експериментальні дослідження проведено у тестових середовищах CartPole-v1 та LunarLander-v3 бібліотеки Gym/Gymnasium. Запропонований механізм обміну знаннями продемонстрував результативність як для алгоритмів Deep Q-Network (DQN), так і Double Deep Q-Network (DDQN).

Результати дослідження прийнято до впровадження в ТОВ «Квалітек» (від 20.03.2026); в навчальному процесі Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (акт реалізації від 20.03.2026) при викладанні дисциплін «Проектування кібер-фізичних систем».

Особистий внесок здобувача. Дисертаційна робота є самостійно виконаною науковою працею. Всі представлені наукові результати, приклади та експериментальні розрахунки, що викладені у дисертації, одержані здобувачем одноосібно. У наукових роботах, що опубліковані у співавторстві, в дисертаційній роботі використані лише ті результати, які становлять індивідуальний внесок автора. У друкованих працях, опублікованих у співавторстві, здобувачеві належать: [13] – розроблений метод контролю якості передачі знань між агентами у системах багатоагентного навчання з підкріпленням, що дозволяє зменшити негативну передачу знань під час децентралізованого обміну знаннями; [14] – розроблений метод контролю інтенсивності передачі знань між агентами, який забезпечує адаптивне регулювання сили навчання залежно від різниці продуктивності між агентами; [15] – запропоновану архітектуру програмного забезпечення для реалізації механізмів децентралізованого обміну знаннями між агентами у багатоагентних системах навчання з підкріпленням.

Апробація матеріалів дисертації. Результати дисертаційного дослідження апробовано на міжнародних науково-практичних конференціях та семінарах: I Міжнародна науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення». – м. Київ, 14 грудня 2023 р. – С.92\$ III Міжнародна науково-технічна конференція “Системи і технології зв’язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку”, 30 листопада 2023 року, м. Київ. Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, 2023. С. 89; XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики» / XXI International scientific and practical conference of young scientists and students "Modern problems of scientific support of power engineering". 2024; XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики». 2024.

Публікації. За результатами досліджень опубліковано 3 наукові праці. Основні наукові положення викладено в 3 наукових статтях [13-15] у

спеціалізованих фахових виданнях України. За матеріалами виступів на науково-технічних конференціях опубліковано 4 тез доповідей [16-19].

Структура і обсяг роботи. Дисертація складається зі вступу, 4 розділів, висновків, 3 додатків та списку використаних джерел із 130 найменувань на 12 сторінках. Повний обсяг дисертації складає 175 сторінок, з них 155 сторінок основного тексту.

РОЗДІЛ 1. АНАЛІЗ ПІДХОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ НАВЧАННЯ АГЕНТІВ

Ефективність навчання у мультиагентних системах значною мірою залежить від здатності агентів використовувати накопичений досвід та обмінюватися знаннями. Незважаючи на розвиток методів глибинного навчання з підкріпленням, питання організації ефективної взаємодії агентів залишається відкритим.

У цьому розділі розглядаються теоретичні та прикладні аспекти навчання з підкріпленням у мультиагентному середовищі. Основну увагу приділено аналізу алгоритмів DQN та їх модифікацій, а також підходів до knowledge sharing, включаючи policy distillation та інші механізми передачі знань між агентами.

1.1 Підходи до підвищення ефективності навчання агентів у системах навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning, RL) є одним із базових підходів машинного навчання, у межах якого агент формує свою поведінку шляхом послідовної взаємодії зі середовищем. На відміну від навчання з учителем, де модель отримує приклади правильних відповідей, у RL агент отримує лише сигнали винагороди, які відображають якість виконаних дій [20]. На основі цих сигналів агент поступово формує політику поведінки, спрямовану на максимізацію сумарної очікуваної винагороди в довгостроковій перспективі [21, 22]. Формалізація задачі навчання з підкріпленням зазвичай здійснюється через модель марковського процесу прийняття рішень MDP. Така модель описується кортежем $\langle S, A, P, R, \gamma \rangle$, де S — множина можливих станів середовища, A — множина доступних агенту дій, P — функція переходів між станами, R — функція винагороди, а γ — коефіцієнт дисконтування майбутніх винагород. У межах цієї моделі передбачається, що наступний стан середовища залежить лише

від поточного стану та виконаної дії, що відповідає марковській властивості процесу [21, 23].

Основними компонентами системи навчання з підкріпленням є стан середовища, дія агента, політика, функція винагороди та функція цінності. Політика визначає правило вибору дій у кожному стані та може бути детермінованою або стохастичною. Функція цінності оцінює очікувану суму майбутніх винагород, які агент може отримати, починаючи з певного стану або виконуючи конкретну дію. Саме оцінка цінності використовується для оновлення політики у процесі навчання та покращення поведінки агента [22, 24].

Попри чіткість математичної постановки задачі, практичне застосування RL пов'язане з низкою суттєвих труднощів. Однією з головних проблем є низька вибіркова ефективність (sample inefficiency). Для досягнення прийнятної продуктивності агенту часто необхідно виконати велику кількість взаємодій зі середовищем, що робить застосування RL дорогим у задачах, де отримання даних є складним або ресурсомістким [25, 26].

Ця проблема стала ще більш актуальною з розвитком методів глибокого навчання з підкріпленням (Deep Reinforcement Learning, DRL), у яких функції політики або цінності апроксимуються нейронними мережами. Такі методи дозволяють працювати з високорозмірними просторами станів, зокрема із зображеннями або складними сенсорними даними, однак водночас значно збільшують потребу в обчислювальних ресурсах та обсягах навчальних даних [25, 27, 28].

Іншим важливим обмеженням є нестабільність процесу навчання. У RL агент сам генерує дані для навчання, а зміна політики призводить до зміни розподілу станів, які він відвідує. Унаслідок цього виникає взаємозалежність між політикою, функцією цінності та навчальними даними, що може спричиняти коливання продуктивності або навіть втрату раніше набутих знань [29, 30].

Це явище частково пояснюється так званою «deadly triad», яка виникає при поєднанні трьох факторів: використання апроксимації функцій, бутстрепінгу та

навчання поза політикою (off-policy learning). Комбінація цих механізмів створює ризик розходження алгоритмів або нестабільної збіжності [21].

Крім того, алгоритми RL характеризуються високою чутливістю до вибору гіперпараметрів. Параметри, такі як швидкість навчання, коефіцієнт дисконтування, стратегія дослідження середовища або розмір буфера досвіду, можуть суттєво впливати на результати навчання. У багатьох випадках налаштування цих параметрів виконується емпірично, що ускладнює перенесення алгоритмів у нові середовища або прикладні задачі [29, 38].

Зі зростанням складності середовища також збільшується розмір простору станів і дій, що ускладнює процес пошуку оптимальної політики. У складних задачах агенту необхідно дослідити велику кількість можливих ситуацій, що значно уповільнює процес навчання. Тому одним із напрямів сучасних досліджень є розробка методів, які дозволяють ефективніше використовувати накопичений досвід або переносити знання між різними агентами чи задачами [25, 31].

У сучасних дослідженнях пропонується низка підходів до підвищення ефективності RL-систем. Серед них можна виділити використання буферів досвіду, пріоритетного повторного програвання досвіду, навчання з демонстрацій, методів перенесення знань, а також колективного навчання в багатоагентних системах [32-34].

Зокрема, у багатьох задачах декілька агентів можуть паралельно взаємодіяти зі середовищем, накопичуючи різний досвід дослідження. У такому випадку виникає можливість обміну знаннями між агентами, що дозволяє скоротити час навчання та підвищити ефективність використання зібраних даних.

Таким чином, попри значний розвиток алгоритмів навчання з підкріпленням, проблема підвищення ефективності навчання агентів залишається актуальною. Основними напрямками досліджень є покращення вибіркової ефективності, підвищення стабільності навчання та зменшення залежності від ручного налаштування параметрів. Одним із перспективних напрямів у цьому контексті є використання багатоагентних систем, у яких знання, отримані одним агентом, можуть бути використані іншими. Саме цей підхід створює передумови для

розробки механізмів поширення знань між агентами, що розглядаються у наступних підрозділах.

1.2 Багатоагентне навчання з підкріпленням

Багатоагентне навчання з підкріпленням (Multi-Agent Reinforcement Learning, MARL) є розширенням класичної парадигми RL на випадок, коли у взаємодії з середовищем одночасно беруть участь декілька агентів [35]. У такій системі кожен агент отримує власні спостереження, виконує дії та отримує винагороди, які можуть залежати як від його власної поведінки, так і від дій інших агентів. Це призводить до більш складної динаміки навчання порівняно з одноагентними системами, оскільки поведінка одного агента може впливати на результати інших [13, 36, 37].

У загальному випадку MARL-системи можуть описуватися через узагальнення марковського процесу прийняття рішень, яке часто називають марковською грою або стохастичною грою. У такій постановці стан середовища є спільним для всіх агентів, а перехід до нового стану визначається сукупністю дій усіх учасників. Кожен агент має власну функцію винагороди, яка може бути частково або повністю узгодженою з винагородами інших агентів [36].

Залежно від характеру взаємодії між агентами, MARL-середовища зазвичай поділяють на кооперативні, конкурентні та змішані. У кооперативних системах агенти мають спільну мету і прагнуть максимізувати загальну винагороду. У конкурентних середовищах інтереси агентів суперечать один одному, і виграш одного агента може означати втрату для іншого. Змішані сценарії поєднують обидва типи взаємодії, коли агенти можуть одночасно співпрацювати та конкурувати залежно від контексту задачі [37].

Організація процесу навчання у MARL може здійснюватися за різними схемами. Одним із підходів є централізоване навчання, при якому використовується спільна функція оптимізації або глобальна інформація про стан системи. Такий підхід дозволяє враховувати взаємозалежність між агентами та

потенційно покращує стабільність навчання. Проте централізовані алгоритми мають обмежену масштабованість і вимагають наявності повної інформації про систему, що не завжди можливо у практичних застосуваннях [37].

Іншою альтернативою є децентралізоване навчання, у якому кожен агент навчається незалежно, використовуючи лише власні спостереження та локальні винагороди. Такий підхід добре підходить для розподілених або автономних систем, однак може призводити до нестабільності, оскільки кожен агент розглядає поведінку інших агентів як частину середовища, що постійно змінюється [32].

Компромісним варіантом є схема централізованого тренування з децентралізованим виконанням (centralized training with decentralized execution, CTDE) [36, 37, 39]. У цьому випадку під час навчання використовується глобальна інформація про стан системи або дії інших агентів, тоді як під час виконання кожен агент діє автономно, використовуючи лише локальні спостереження [40]. Така схема широко застосовується в сучасних алгоритмах MARL, оскільки дозволяє поєднати переваги централізованого аналізу та децентралізованої поведінки агентів [14, 37].

Однією з ключових проблем MARL є нестационарність середовища. У класичному RL передбачається, що динаміка середовища залишається незмінною протягом усього процесу навчання. Проте у багатоагентних системах поведінка інших агентів змінюється у процесі їх навчання, що призводить до зміни розподілу станів і винагород. Для кожного окремого агента середовище виглядає таким, що постійно змінюється, що ускладнює збіжність алгоритмів навчання [32, 41].

Іншою важливою проблемою є масштабування системи. Із зростанням кількості агентів різко збільшується розмір об'єднаного простору станів і дій, що ускладнює як представлення функцій цінності, так і пошук оптимальних стратегій. У багатьох випадках повний перебір можливих комбінацій дій стає практично неможливим, що вимагає використання наближених методів або спеціалізованих алгоритмів координації [36, 41].

Додатковою проблемою є дублювання досвіду під час навчання. Якщо декілька агентів одночасно досліджують середовище, вони можуть незалежно вивчати однакові закономірності або повторно досліджувати одні й ті самі області простору станів. У результаті частина обчислювальних ресурсів використовується неефективно, оскільки агенти виконують однакові експерименти, не використовуючи досвід один одного [32].

Водночас наявність декількох агентів створює нові можливості для підвищення ефективності навчання. Якщо агенти накопичують різний досвід взаємодії із середовищем, цей досвід може бути використаний іншими агентами для прискорення їхнього навчання. Таким чином, MARL відкриває перспективу використання механізмів колективного навчання, у межах яких інформація або знання, отримані одним агентом, можуть бути передані іншим [42, 43].

Під знаннями у контексті MARL зазвичай розуміють узагальнену інформацію про поведінку агента в середовищі. Це можуть бути параметри функції цінності, політики прийняття рішень, латентні представлення станів або інші внутрішні характеристики моделі, які відображають досвід агента. Передача таких знань між агентами може відбуватися у різних формах, зокрема шляхом обміну досвідом, копіювання параметрів моделей або дистиляції політик [42, 43].

У системах із великою кількістю агентів ефективний механізм обміну знаннями може суттєво зменшити кількість необхідних взаємодій із середовищем, прискорити збіжність алгоритмів і покращити узгодженість поведінки агентів. Водночас неконтрольована передача знань може призводити до негативного перенесення, коли неякісні або нерелевантні знання погіршують результати навчання інших агентів.

Таким чином, багатоагентне навчання з підкріпленням поєднує як додаткові труднощі, пов'язані з нестаціонарністю середовища та масштабуванням системи, так і нові можливості для підвищення ефективності навчання через колективне використання досвіду. У зв'язку з цим важливим напрямом сучасних досліджень є розробка методів передачі знань і досвіду між агентами, які дозволяють

ефективніше використовувати накопичену інформацію та прискорювати процес навчання. Ці підходи розглядаються у наступному підрозділі.

1.3 Методи передачі знань у системах навчання з підкріпленням

Одним із перспективних напрямів підвищення ефективності навчання агентів є використання механізмів передачі знань (knowledge transfer). У класичних алгоритмах навчання з підкріпленням кожен агент навчається незалежно, поступово накопичуючи досвід взаємодії із середовищем. Однак у багатьох практичних задачах різні агенти можуть досліджувати подібні області простору станів або отримувати подібний досвід. У таких випадках використання знань, отриманих іншими агентами або під час попередніх етапів навчання, може суттєво прискорити процес навчання та підвищити ефективність використання даних [42, 44].

Передача знань у системах RL може здійснюватися різними способами, зокрема через обмін досвідом, копіювання параметрів моделей, дистиляцію політик або використання механізмів порад між агентами. У багатоагентних системах такі механізми дозволяють використовувати колективний досвід групи агентів, що відкриває можливості для значного прискорення процесу навчання [43, 45, 46].

1.3.1 Передача досвіду та повторне використання знань

Одним із найпростіших підходів до передачі знань є обмін досвідом між агентами. У цьому випадку агенти можуть використовувати спільні буфери досвіду або передавати один одному зібрані траєкторії взаємодії із середовищем. Такий підхід дозволяє ефективніше використовувати накопичені дані та зменшувати дублювання дослідження однакових станів середовища [33].

Подібні ідеї розвиваються у роботах, присвячених повторному використанню знань [47] у багатоагентному навчанні. Наприклад, у дослідженні

Knowledge Reuse of Multi-Agent Reinforcement Learning in Cooperative Tasks запропоновано підхід, у якому агенти використовують досвід інших агентів як додаткове джерело навчальних даних. Це дозволяє прискорити процес навчання та підвищити стабільність алгоритмів у кооперативних задачах.

Однак простий обмін досвідом має низку обмежень. Зокрема, не всі отримані переходи можуть бути однаково корисними для інших агентів. У разі значних відмінностей між стратегіями агентів або між умовами середовища переданий досвід може виявитися малокорисним або навіть негативно впливати на процес навчання.

Окремим напрямом досліджень у навчанні з підкріпленням є перенесення знань між різними задачами (transfer learning) [48]. У цьому підході знання, отримані під час навчання агента в одному середовищі або при розв'язанні однієї задачі, використовуються для прискорення навчання в іншій задачі. Такий підхід дозволяє повторно використовувати вже сформовані представлення станів, функції цінності або політики поведінки [31, 44].

Передача знань між задачами може здійснюватися різними способами. Найпростішим варіантом є перенесення параметрів нейронної мережі, навченої в одній задачі, як початкового наближення для навчання у новому середовищі. Це дозволяє зменшити час навчання, оскільки модель уже містить узагальнену інформацію про структуру задачі.

Іншим підходом є використання спільних представлень станів або латентних ознак, які можуть бути корисними у кількох задачах. У таких методах нейронна мережа навчається формувати універсальні представлення середовища, які можуть використовуватися різними агентами або в різних сценаріях [31].

Перенесення знань між задачами особливо ефективно у випадках, коли задачі мають подібну структуру або спільні характеристики середовища. У таких ситуаціях частина отриманого досвіду може бути використана повторно, що дозволяє суттєво скоротити кількість необхідних взаємодій із середовищем [49-51].

Водночас застосування transfer learning у системах RL також пов'язане з ризиком негативного перенесення знань. Якщо задачі мають суттєві відмінності, використання знань, отриманих у попередньому середовищі, може погіршити процес навчання або призвести до формування неефективних стратегій поведінки.

1.3.2 Дистиляція політики та передача моделей

Більш структурованим підходом до передачі знань є дистиляція політики (policy distillation). У цьому випадку знання одного агента передаються іншому шляхом навчання моделі-учня відтворювати поведінку моделі-вчителя. Передача знань здійснюється шляхом мінімізації відмінності між розподілами дій двох моделей, що дозволяє ефективно переносити інформацію про ефективні стратегії поведінки [52].

Однією з перших робіт у цьому напрямі є дослідження Policy Distillation [53], у якому запропоновано метод перенесення політик між різними агентами або між різними задачами. Автори показали, що дистиляція може використовуватися для об'єднання знань кількох агентів або для створення компактних моделей, які зберігають ефективні стратегії поведінки.

Подальшим розвитком цієї ідеї стала робота Policy Distillation with Value Matching, у якій запропоновано розширення методу дистиляції для багатоагентних систем. У цьому підході передача знань відбувається не лише через політику, але й через узгодження функцій цінності агентів. Це дозволяє враховувати кооперативну структуру задачі та покращувати узгодженість поведінки агентів [34, 54, 55].

Методи дистиляції політик мають значний потенціал для використання у багатоагентних системах, однак вони також мають певні обмеження. Зокрема, ефективність передачі знань значною мірою залежить від вибору агента-вчителя та якості його політики. А існуючі методи зазвичай передбачають статичний вибір агента-вчителя та не враховують динаміку якості політик під час навчання. Крім того, недостатньо дослідженим залишається питання адаптивного регулювання

інтенсивності передачі знань між агентами, що може призводити як до надлишкового копіювання, так і до недостатнього використання корисного досвіду.

1.3.3 Передача знань через поради та навчальні стратегії

Іншим напрямом досліджень є використання механізмів порад (advice) між агентами. У таких підходах більш досвідчений агент може рекомендувати дії менш досвідченому агенту у певних станах. Подібні механізми часто реалізуються у вигляді взаємодії «вчитель – учень», де агент-вчитель може надавати поради лише обмежену кількість разів.

У деяких роботах пропонується використовувати спеціальні стратегії планування навчання (learning schedule) [56, 57], які визначають моменти та частоту передачі знань між агентами. Наприклад, агент може отримувати поради лише на ранніх етапах навчання або у складних станах середовища. Такий підхід дозволяє поєднати самостійне дослідження середовища з використанням досвіду інших агентів [58-60].

Механізми порад можуть суттєво зменшити кількість помилкових дій на початкових етапах навчання, однак вони також потребують ефективного механізму визначення моментів, коли передача знань є найбільш корисною [61, 62].

1.3.4 Методи колективного поширення знань

Окремий напрям досліджень пов'язаний із розробкою механізмів колективного поширення знань між агентами. Одним із прикладів таких підходів є метод KnowSR, у якому агенти обмінюються знаннями про політики поведінки з метою прискорення процесу навчання.

У таких системах агенти можуть передавати один одному інформацію про свої моделі або результати навчання, що дозволяє швидше поширювати ефективні

стратегії поведінки в межах всієї системи. Подібні механізми можуть бути особливо корисними у випадках, коли агенти діють у схожих середовищах або виконують подібні задачі.

Водночас неконтрольований обмін знаннями може призводити до поширення неякісних або нерелевантних стратегій, що може негативно впливати на результати навчання.

1.3.5 Обмеження існуючих підходів передачі знань

Попри значну кількість запропонованих методів передачі знань, більшість із них мають певні обмеження. По-перше, у багатьох підходах відсутній механізм оцінки якості знань, які передаються між агентами. У результаті передача знань може здійснюватися від агентів із низькою продуктивністю, що призводить до негативного перенесення знань.

По-друге, існуючі методи зазвичай не враховують оптимальну інтенсивність передачі знань. Надмірна кількість передач може призводити до втрати різноманітності стратегій та до зниження здатності агентів до самостійного дослідження середовища. З іншого боку, недостатня кількість передач може не дати помітного ефекту прискорення навчання.

До того ж, більшість методів сфокусовано на архітектурах, що мають певною мірою централізовані компоненти, як Actor-critic [63]. А також працюють в умовах нестационарного середовища, та агентів, що взаємодіють та впливають один на одного, та здатні на кооперацію чи інші види взаємодії [64].

Таким чином, незважаючи на значний прогрес у розвитку методів передачі знань у системах навчання з підкріпленням, питання ефективного контролю якості джерел знань та інтенсивності їх передачі між агентами залишається відкритим. Розробка підходів, які дозволяють адаптивно керувати цими процесами, є важливим напрямом досліджень у галузі багатоагентного навчання з підкріпленням і становить предмет подальших досліджень у межах цієї роботи.

1.4 Аналіз вимог до програмного забезпечення та постановка наукового завдання

Попри значний прогрес у дослідженнях щодо поширення знань і перевикористання досвіду між агентами у багатоагентному навчанні з підкріпленням MARL, залишається низка критичних обмежень, які обмежують застосування таких підходів у децентралізованих та стаціонарних середовищах слабо пов'язаних агентів [65]. Зокрема, більшість сучасних рішень не передбачають механізмів контролю якості поширюваних знань. Уявлення про те, що будь-який агент є потенційно корисним джерелом досвіду, ігнорує реальну різницю в продуктивності окремих агентів. Це створює ризик негативної передачі знань, коли менш ефективні або хибно навчені агенти можуть вплинути на інших, знижуючи загальну продуктивність системи. Крім того, сучасні підходи не розрізняють кількісні та якісні аспекти знань — обсяг досвіду не завжди корелює з його цінністю, що особливо актуально в умовах часткового охоплення середовища.

Ще одним обмеженням є відсутність адаптивного контролю інтенсивності обміну знаннями. Як правило, передача реалізується з фіксованою частотою або в однаковому обсязі незалежно від того, наскільки релевантним є джерело інформації чи потреби конкретного агента. Ігнорування асиметрії між «сильними» і «слабкими» агентами не дозволяє ефективно використовувати потенціал високопродуктивних учасників. Це особливо критично в сценаріях, де агенти перебувають на різних стадіях навчання або вивчають різні області простору станів.

Більшість описаних у літературі рішень передбачають централізовану або частково централізовану архітектуру — як-от спільні буфери, дистильовані політики, або глобальні контролери. Така залежність від координації знижує надійність системи, особливо в умовах відмов окремих агентів або обмеженої комунікації. Крім того, більшість робіт фокусуються на кооперативних або нестаціонарних середовищах, де агенти мають спільну мету або адаптуються до

змінної динаміки. Натомість стаціонарні середовища з незалежними агентами, які не координують дії та не мають спільної мети, залишаються слабо дослідженими, попри їхню прикладну важливість (наприклад, в автономному спостереженні, дистрибутивному моніторингу або ресурсному плануванні без прямої взаємодії) [15].

З огляду на зазначене, формується задача дослідження: розробка науково-методичних підходів до підвищення ефективності навчання у системах слабо пов'язаних агентів у стаціонарному середовищі шляхом децентралізованого обміну знаннями, що не потребує централізованого керування або синхронізації дій. Такий обмін має бути адаптивним, враховувати якість джерела знань, а також потенційний вплив на продуктивність кожного агента.

У якості базової архітектури для реалізації та оцінки запропонованого підходу обрано IDQN — модифіковану версію класичного DQN, орієнтовану на незалежне навчання кожного агента без урахування дій інших. Цей вибір обумовлений кількома міркуваннями: (1) простота та масштабованість архітектури, (2) відсутність централізованих компонентів у навчальному процесі, (3) придатність до стаціонарного середовища, де динаміка не змінюється з часом, (4) мінімальні припущення щодо взаємодії агентів, що дозволяє створити ізольовані експериментальні сценарії.

Таким чином, постає задача розробки методів та програмних засобів, що дозволяють децентралізовано контролювати якість та силу передачі знань між слабо пов'язаними (незалежними) агентами в стаціонарному середовищі, та на основі цього розробити програмне забезпечення, яке дозволить покращити продуктивність системи.

У більшості класичних систем MARL динаміка взаємодії між агентами моделюється у вигляді марковської гри, що передбачає необхідність синхронізації агентів на кожному кроці ітерації. У такій постановці середовище визначає наступний глобальний стан лише після отримання дій від усіх агентів, тобто один крок навчання виконується лише після колективного оновлення дій усіх

учасників. Така структура є виправданою в умовах, де агенти взаємодіють тісно, мають спільне середовище чи цілі, і де дії одного впливають на динаміку інших.

Однак у досліджуваному випадку система складається зі слабо пов'язаних агентів, кожен з яких діє у власному стаціонарному середовищі, дії в якому не впливають на інших. За таких умов жорстка синхронізація на кожному кроці не лише недоцільна, а й може бути обмеженням для масштабування і гнучкості системи. Навпаки, виникає потреба у підтримці асинхронної моделі взаємодії, де кожен агент може самостійно здійснювати дії, накопичувати досвід і оновлювати свою політику незалежно від прогресу інших.

Проте, потреба координації може виникати на більшому часовому проміжку, наприклад, після фіксованого числа ітерацій (взаємодій з середовищем), у кінці епізоду або за виконання певної умови.

Така координація передбачає не постійну, а періодичну синхронізацію: для обміну знаннями, узгодження статистики або регулювання стратегії обміну, тощо. У цьому випадку синхронізація означає блокування агента, який виконав дію або завершив епізод раніше за інших — тобто, агент повинен призупинити виконання і дочекатися завершення інших, що призводить до програмного блокування.

Ці аспекти суттєво впливають на вимоги до архітектури програмного забезпечення. Система повинна підтримувати асинхронне виконання агентів із можливістю періодичного або умовного узгодження, уникати повної зупинки процесу через відставання одного з учасників і забезпечувати гнучке керування механізмами обміну знаннями без втрати продуктивності. У свою чергу, це накладає вимоги і на постановку задачі — дослідження має враховувати децентралізоване, асинхронне середовище, де агенти діють незалежно, але мають потенціал до взаємного покращення через адаптивний обмін знаннями в контрольовані моменти часу.

Багатоагентні системи в цілому, а також механізми обміну знаннями між автономними агентами, що функціонують у їх складі, висувають низку специфічних вимог до програмного забезпечення.

Виходячи з поставленої наукової проблеми, сформульовано функціональні вимоги (ФВ), які представлено у вигляді таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги до системи

Код вимоги	Вимога
ФВ1	Забезпечення можливості децентралізовано контролювати якість передачі знань між агентами за допомогою механізму дистиляції політики.
ФВ2	Забезпечення можливості децентралізовано контролювати силу передачі знань між агентами за допомогою механізму дистиляції політики.
ФВ3	Забезпечення можливості синхронізувати агентів з можливістю блокування виконання роботи для поширення знань. А також / або можливість працювати за умови не ідеальності / відсутності синхронізації.

Перелік основних нефункційних вимог (НФВ), які рекомендовано врахувати при розробці ПЗ даного типу, наведено в таблиці 1.2.

Таблиця 1.2 – Нефункціональні вимоги до системи

Код вимоги	Вимога та її опис
НФВ1	Надійність: запропоновані методи повинні враховувати динамічні природу системи, де агенти можуть приєднуватися, від'єднуватися чи тимчасово ставати недоступними.
НФВ2	Сумісність та інтеграція: при розробці ПЗ потрібно врахувати можливість інтеграції методів у вже існуючі системи.

Метою дисертаційної роботи є підвищення продуктивності багатоагентної системи слабко пов'язаних агентів у стаціонарному середовищі за допомогою механізму поширення знань між агентами на базі децентралізованого підходу.

Об'єктом досліджень є процес підвищення продуктивності в системах багатоагентного навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами.

Предметом дослідження є методи та програмне забезпечення децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами.

Для досягнення мети в дисертації вирішено наступні наукові завдання:

- провести аналіз існуючих методів навчання з підкріпленням для індивідуальних агентів та багатоагентних систем, конкретних алгоритмів реалізацій;
- провести аналіз існуючих методів застосування механізму обміну знаннями між слабо пов'язаними агентами системи;
- дослідити вплив обміну знаннями між слабо пов'язаними агентами на середній показник ефективності роботи багатоагентної системи у різних конфігураціях;
- розробити метод контролю якості обміну знаннями між агентами для запобігання або зменшенню негативної передечі знань;
- розробити метод контролю інтенсивності обміну знаннями між агентами для пришвидшеного поширення знань зі збереженням індивідуальності;
- розробити архітектуру програмного забезпечення для реалізації механізму децентралізованого обміну знаннями з метою підвищення якості навчання та роботи системи слабо пов'язаних агентів;
- розробити програмний метод децентралізованого обміну знаннями між агентами.

1.5 Висновки до розділу

У цьому розділі проведено всебічний аналіз існуючих підходів до підвищення ефективності навчання агентів у парадигмі навчання з підкріпленням (RL), з особливим акцентом на багатоагентні системи (MARL). Встановлено, що

класичні RL-методи мають низку обмежень, таких як низька вибіркова ефективність, нестабільність навчання, чутливість до гіперпараметрів та погана масштабованість, що обмежує їх застосовність у складних або реальних середовищах.

Проаналізовано особливості MARL, де взаємодія між агентами створює як додаткові труднощі (локальна нестационарність, масштабування, дублювання досвіду), так і нові можливості — зокрема, завдяки обміну знаннями між агентами. Визначено, що механізми поширення інформації, такі як обмін досвідом, дистиляція політик, та узгодження функцій цінності (value matching), можуть суттєво покращити ефективність навчання. Селективне поширення корисного досвіду або політик дозволяє уникати дублювання зусиль, прискорює збіжність і зменшує вимоги до кількості зразків.

Розглянуті підходи (KnowSR, DVM, MAPTF, офлайн-дистиляція політик) продемонстрували ефективність у кооперативних середовищах, однак мають спільні обмеження: централізованість архітектури, відсутність контролю якості переданих знань, фіксована інтенсивність обміну, та слабка адаптивність до гетерогенності або асинхронності середовища.

На основі виявлених обмежень сформульовано наукову проблему: підвищення ефективності навчання у багатоагентних системах зі слабо пов'язаними агентами в стаціонарному середовищі шляхом реалізації децентралізованого, адаптивного обміну знаннями без централізованого управління. Обґрунтовано вибір базової архітектури Independent Deep Q-Network (IDQN) для реалізації дослідження та постановки задачі, що дозволяє уникнути централізованої координації та забезпечити масштабованість у незалежних середовищах.

РОЗДІЛ 2. ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ СИСТЕМИ СЛАБКО ПОВ'ЯЗАНИХ АГЕНТІВ ШЛЯХОМ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ЗНАННЯМИ

Результати аналізу, проведеного у попередньому розділі, свідчать про наявність ряду обмежень існуючих підходів до навчання мультиагентних систем, зокрема у контексті організації ефективного обміну знаннями між агентами. Це зумовлює необхідність розробки нових методів та моделей, орієнтованих на підвищення ефективності навчання.

У цьому розділі розглядаються методологічні основи побудови мультиагентної системи на базі глибинного навчання з підкріпленням. Основну увагу приділено формалізації задачі, розробці моделі взаємодії агентів, а також визначенню механізмів передачі знань у рамках teacher-student підходу.

2.1 Формалізація задачі навчання з підкріпленням

Формалізація задачі навчання з підкріпленням ґрунтується на моделі Марковського процесу прийняття рішень (Markov Decision Process, MDP), яка є стандартним математичним апаратом для опису послідовного прийняття рішень у стохастичних середовищах [21, 23]. MDP задається кортежем $M = \langle S, A, P, R, \gamma \rangle$, де S — множина станів середовища, A — множина допустимих дій, $P(s' | s, a)$ — імовірність переходу до стану s' при виконанні дії a у стані s , $R(s, a, s')$ — функція миттєвої винагороди, а $\gamma \in [0, 1]$ — коефіцієнт дисконтування [21].

Марковська властивість означає, що розподіл наступного стану залежить лише від поточного стану та виконаної дії, але не від попередньої історії взаємодії. Це припущення забезпечує можливість рекурсивного представлення функцій цінності та лежить в основі алгоритмів динамічного програмування і методів навчання з підкріпленням [21, 66].

Метою агента є знаходження політики — відображення зі станів у дії — яка максимізує очікувану дисконтовану суму винагород. Повернення (return) визначається як:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \gamma^{k-1} r_{t+k}, \quad (2.1)$$

де r_{t+k} — винагорода, отримана через k кроків після моменту t .

Для кількісної оцінки якості станів і дій вводяться функції цінності. Зокрема, функція цінності стану $V^\pi(s)$ визначається як математичне сподівання повернення при старті зі стану s та подальшому дотриманні політики:

$$V(s) = E_\pi[R_0 \mid s_0 = s]. \quad (2.2)$$

Аналогічно визначається функція цінності дії (або Q -функція) як очікувана віддача від пари “стан–дія” при подальшому дотриманні політики π :

$$Q(s, a) = E_\pi[R_1 \mid s_0 = s, a_0 = a], \quad (2.3)$$

де R_1 — віддача, починаючи з наступного кроку після виконання дії a в стані s .

Функції цінності задовольняють рекурсивні співвідношення, відомі як рівняння Беллмана [21, 23]. В загальному вигляді рівняння $Q(s, a)$ має вигляд:

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a) Q(s', a'), \quad (2.4)$$

де a' — дія, яку задає політика π в новому стані s' (при стохастичній політиці — середнє по всіх можливих діях).

Це рекурентне співвідношення відображає той факт, що цінність поточної дії дорівнює безпосередній винагороді $R(s, a)$ плюс дисконтована цінність наступного стану. Для оптимальної політики замість математичного сподівання за діями використовується оператор максимуму. Тоді оптимальне рівняння Беллмана для функції стан–дія набуває вигляду:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a) \max_{b \in A} Q^*(s', b). \quad (2.5)$$

Ці співвідношення лежать в основі алгоритмів Q -learning (Q -навчання), SARSA та інших методів глибокого навчання з підкріпленням, зокрема DQN [67]. Застосування бутстрапінгу — використання поточних оцінок функції цінності для формування цільових значень — дозволяє здійснювати ітеративне наближення до оптимального розв’язку навіть за відсутності повної моделі середовища.

На практиці аналітичне розв’язання рівняння Беллмана для $Q^*(s, a)$ є складним для нетривіальних середовищ, оскільки розмірності S і A можуть бути дуже великими, а перехідні ймовірності невідомі. Натомість застосовують чисельні методи, які поступово покращують наближення до оптимальних значень функцій цінності, ґрунтуючись на досвіді, отриманому агентом. Якщо модель середовища (функції P і R) відома, можна використати алгоритми динамічного програмування (наприклад, ітерацію цінності або політики). У більш загальному випадку, коли модель невідома, застосовують модель-незалежні (model-free) методи навчання з підкріпленням, які оцінюють і покращують стратегію лише на основі вибірових траєкторій взаємодії агента із середовищем. Такі методи відрізняються способом формування цільового сигналу та характером використання доступних даних, однак усі вони можуть бути інтерпретовані як наближення до розв’язку рівнянь Беллмана.

Значний виклик становить ситуація, коли винагорода надається із затримкою, і агенту потрібно приписати заслуги або провину окремим діям у довгому ланцюжку.

Нехай $V(s)$ — поточна оцінка функції цінності стану для фіксованої політики. Узагальнене правило стохастичного оновлення можна записати у вигляді

$$V(s_t) \leftarrow V(s_t) + \alpha * \delta_t, \quad (2.6)$$

де $\alpha \in (0,1]$ — крок навчання,

δ_t — похибка оцінювання. Конкретна форма δ_t визначає тип алгоритму.

Методи Монте-Карло (МС) вирішують цю проблему, оцінивши якість дій постфактум: агент генерує повний епізод до кінця, після чого отримана сумарна віддача (2.1) використовується для оновлення оцінок цінності всіх пройдених станів і дій. Такі методи не вимагають знання моделі середовища і зрештою можуть обчислити точні значення функцій цінності при достатній кількості епізодів.

У цьому випадку цільове значення формується виключно на основі фактичних винагород траєкторії, без використання поточних оцінок майбутніх

станів. Таким чином, метод Монте-Карло не використовує бутстрапінг. За відповідних умов та достатньої кількості вибірок оцінка збігається до істинної функції цінності заданої політики однак характеризується відносно високою дисперсією. Недоліком МС-підходів є неможливість оновлення оцінок до завершення епізоду: якщо епізоди довгі або рідко закінчуються успіхом, агент отримує дуже відстрочений сигнал помилки.

На противагу цьому, методи тимчасових різниць (Temporal-Difference, TD) виконують оновлення оцінок протягом епізоду, ще до отримання кінцевого результату. Ідея TD-методів полягає у використанні поточної наближеної оцінки майбутньої віддачі як “підказки” для оновлення. Наприклад, після виконання дії a в стані s і переходу в s' з миттєвою нагородою r , можна оновити наближення $Q(s, a)$ на основі локальної помилки: $\delta = r + \gamma[\hat{Q}(s', b) - \hat{Q}(s, a)]$ – поточна оцінка, а b – деяка наступна дія. Тут оцінка майбутнього стану використовується для формування цільового сигналу. Такий механізм називається бутстрапінгом і безпосередньо впливає з рекурсивної структури рівнянь Беллмана. Бутстрапінг дозволяє виконувати оновлення після кожного переходу, зменшуючи дисперсію оцінки порівняно з методом Монте-Карло, однак вводить потенційне зміщення через використання наближених значень.

У найпростішому випадку беруть $b = \arg \max_{a'} \hat{Q}(s', a')$, тобто використовують жадібну оцінку наступного кроку; тоді формула спрощується до:

$$\delta = r + \gamma [\max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a)]. \quad (2.7)$$

Отриману TD-помилку δ використовують для коригування значення:

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha \delta. \quad (2.8)$$

Такий підхід називається однокроковим TD (або TD(0)) і фактично реалізує рівняння Беллмана як локальне рекурентне правило оновлення. Він лежить в основі алгоритмів Q-навчання та SARSA.

Більш загальним підходом є використання середньої n -крокової віддачі. В n -step TD методах для оцінки проміжної цінності стану або дії беруть суму фактичних нагород за n кроків вперед плюс наближення цінності після n кроків.

Граничним випадком є $n \rightarrow \infty$, що еквівалентно методу Монте-Карло (вся віддача до кінця епізоду). Для об'єднання цих підходів вводиться механізм слотів успадкування (eligibility traces). Алгоритм $TD(\lambda)$ акумулює внесок багатьох n -крокових віддач одночасно, використовуючи параметр $0 \leq \lambda \leq 1$ для поступового затухання ваг внеску далеких кроків. При $\lambda = 0$ $TD(\lambda)$ відповідає однокроковому $TD(0)$, а при $\lambda = 1$ – методам Монте-Карло; проміжні значення дають компроміс між швидкістю та точністю оцінок.

Існують два принципово різні підходи до того, як агент використовує отриманий досвід для навчання стратегій. On-policy алгоритми оцінюють і покращують ту саму політику, яку агент наразі застосовує для взаємодії з середовищем. Натомість off-policy алгоритми дозволяють відокремити політику, з якої збираються дані, від політики, яка оптимізується. Іншими словами, off-policy методи можуть навчатися на основі досвіду, згенерованого іншою, можливо субоптимальною або випадковою, стратегією. Класичний приклад on-policy алгоритму – метод SARSA, де оцінка $Q(s, a)$ оновлюється на основі дії s' , що фактично була виконана агентом у наступному стані s' відповідно до поточної політики. Таким чином, SARSA завжди оцінює ту політику, якою користується агент (включно з його поточною схемою дослідження). На противагу, алгоритм Q-learning є off-policy: оновлення $Q(s, a)$ здійснюється з використанням максимальної потенційної нагороди в наступному стані s' (тобто припускається, що в s' агент діятиме оптимально надалі), незалежно від того, яку дію агент насправді виконав у s' . Отже, Q-learning прямо наближає оптимальну Q-функцію, тоді як SARSA схильний оцінювати більш безпечну поведінку з урахуванням поточного рівня дослідження. Off-policy алгоритми є привабливими тим, що дозволяють навчатися на заздалегідь зібраних даних (включно з історичними траєкторіями або демонстраціями) та більш ефективно використовувати досвід за рахунок повторного програвання минулих епізодів. З іншого боку, on-policy методи часто більш стабільні, оскільки навчання завжди узгоджене з тією політикою, що випробовується, і не потребує додаткових заходів для стабілізації [67, 68].

2.2 DQN як модель незалежного агента

Одним із центральних алгоритмів навчання з підкріпленням є алгоритм Q-навчання (Q-learning), запропонований Воткінсом (Watkins) [69]. Це модель-незалежний off-policy метод, який базується на описаному вище рівнянні Беллмана для оптимальної Q-функції. Ідея Q-навчання полягає в тому, щоб поступово наближатися до $Q^*(s, a)$ через перегляд оцінок на основі спостережених переходів. У найпростішому табличному випадку алгоритм підтримує таблицю значень $Q(s, a)$ для всіх пар стан-дія. Ітеративно, взаємодіючи з середовищем, агент оновлює оцінки за правилом:

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a') - Q(s_t, a_t) \right], \quad (2.9)$$

де α – коефіцієнт навчання.

Ця формула оновлення реалізує однокроковий TD-метод, описаний вище: значення $Q(s_t, a_t)$ коригується в напрямку збільшення наближення до цільового значення $y = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$.

Доведено, що при достатньому дослідженні простору станів і дій (зокрема, якщо кожна дія в кожному стані випробовується нескінченно часто), стаціонарності середовища та належного вибору кроку навчання та при зменшенні α з часом такий процес збігається до оптимальних значень $Q^*(s, a)$ у випадку дискретного кінцевого MDP. Q-навчання не потребує знання моделі середовища і гарантовано знаходить оптимальну стратегію навіть при взаємодії з довільною (достатньо часто дослідницькою) поведінковою політикою – завдяки off-policy характеру, описаному вище [21, 23].

Проте табличне представлення є непридатним для задач із великим або неперервним простором станів, оскільки потребує зберігання окремого параметра для кожної пари (s, a) . У високорозмірних задачах, зокрема при роботі з сенсорними спостереженнями, кількість можливих станів стає практично нескінченною.

У таких випадках здійснюється перехід до функціональної апроксимації, коли Q-функція подається у вигляді параметризованої моделі $Q(s, a; \theta)$ де θ — вектор параметрів. У загальному випадку параметри оптимізуються шляхом мінімізації темпорально-різницевої похибки [70].

У 2015 році дослідники DeepMind запропонували алгоритм DQN, що поєднав Q-навчання з глибокими нейронними мережами як універсальними функціональними апроксиматорами. Такий підхід дозволяє ефективно працювати з високорозмірними спостереженнями та складними нелінійними залежностями між станом і діями. Він наблизив навчання з підкріпленням до рівня, порівнянного з людським, у низці складних задач. Зокрема, агент DQN продемонстрував здатність навчитися грати в класичні відеоігри Atari на рівні професійного тестера, отримуючи на вході лише необроблені пікселі екрану та поточний рахунок гри. Цей результат став першим випадком, коли штучний агент досяг людського рівня в широкому класі задач, не використовуючи ніяких апріорних знань про правила цих ігор [23].

По суті, DQN (рис. 2.1) реалізує той самий алгоритм Q-навчання, але замість таблиці значень використовується нейронна мережа для наближення функції $Q(s, a, \theta)$, де θ – ваги глибокої нейромережі [71]. Вхід мережі складається з подання стану s (наприклад, послідовності кадрів гри), а вихід – оцінки Q для всіх можливих дій a . Параметри мережі θ навчаються градієнтними методами шляхом мінімізації помилки передбачення на вибірці досвіду. Тобто політика має вигляд $\pi(s) = \arg \max_{a \in A} Q(a, s, \theta)$.

Цільове значення визначається як

$$y^{\text{target}} = r + \gamma * \max_{a' \in A} Q(s', a', \theta), \quad (2.10)$$

де величина y^{target} відповідає наближенню до правої частини рівняння Беллмана для оптимуму, $Q(s, a, \theta)$ – поточній оцінці лівої частини; різниця між ними є TD-помилкою.

Функція втрат визначається як (у випадку використання середньоквадратичної похибки MSE)

$$L(\theta) = E_{(s,a,r,s') \sim D} \left[(y_t - Q(s_t, a_t; \theta))^2 \right], \quad (2.11)$$

де D — розподіл переходів, сформований з буфера досвіду.

На практиці математичне сподівання замінюється середнім по міні-батчу. Хоча також зустрічається Huber-loss (smooth L1) замість MSE, так як він більш стійкий до викидів.

Таким чином, DQN реалізує стохастичну мінімізацію відхилення від оптимального рівняння Беллмана. Алгоритм залишається модель-вільним, оскільки не використовує явного знання функції переходів P або функції винагород R .

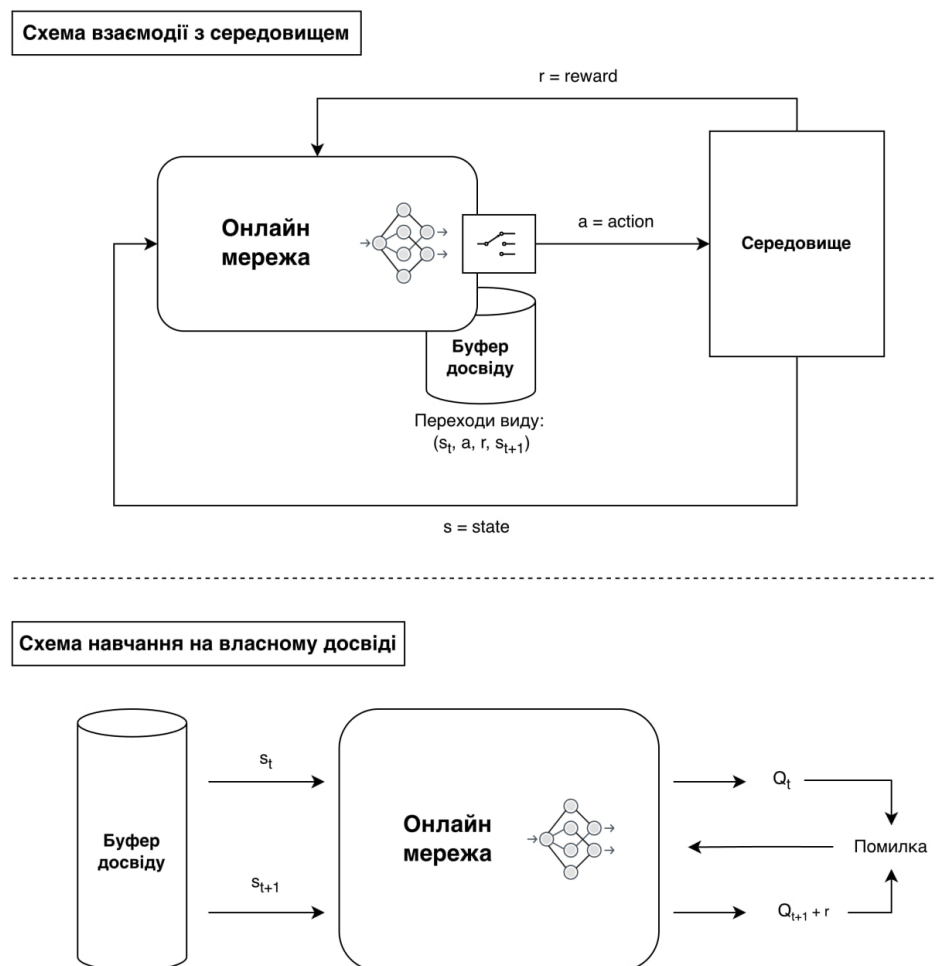


Рисунок 2.1 – Схема DQN з однією мережею

Поєднання функціональної апроксимації, бутстрапінгу та off-policy оновлення може призводити до нестабільності або розбіжності оцінок — явища,

відомого як поєднання «deadly triad» [21]. У базовій архітектурі DQN ця проблема пом'якшується двома механізмами: використанням буфера досвіду (experience replay), який декорелює послідовні спостереження, та застосуванням окремої цільової мережі [27].

2.2.1 Буфер досвіду як механізм декореляції

Однією з ключових проблем при використанні функціональної апроксимації в алгоритмах типу DQN є кореляція між послідовними спостереженнями. У стандартному процесі взаємодії з середовищем переходи $(s_t, a_t, r_{t+1}, s_{t+1})$ утворюють часово залежну послідовність, що порушує припущення незалежності вибірок, яке лежить в основі більшості методів стохастичної оптимізації [21]. Безпосереднє використання таких корельованих даних у градієнтному навчанні нейронної мережі може призводити до нестабільності оновлень, осциляцій параметрів та навіть розбіжності алгоритму [27].

Для зменшення цього ефекту в DQN використовується механізм буфера досвіду (experience replay), запропонований у роботі [27]. Буфер досвіду є обмеженою пам'яттю D фіксованого розміру N , у якій зберігаються переходи агента у вигляді кортежів

$$D = \langle s, a, r, s' \rangle. \quad (2.12)$$

На кожному кроці взаємодії новий перехід додається до буфера, а при перевищенні місткості найстаріші елементи видаляються. Навчання мережі відбувається не на останньому переході, а на випадковій міні-вибірці B , сформованій шляхом рівномірного семплювання з D . Таким чином, розподіл переходів, що використовується для оцінювання градієнта, наближається до емпіричного розподілу, сформованого з історії взаємодії агента із середовищем. Це дозволяє частково усунути короткострокову кореляцію між сусідніми станами та стабілізувати оцінку градієнта

Опціонально агент може збирати і додаткові параметри, такі як флаг завершення епізоду, а також флаг, що вказує, чи закінчився епізод в провальному

термінальному стані. Це дає можливість змінювати функцію винагороди, накладаючи пенальті чи додаткову винагороду, якщо це не передбачено середовищем.

З точки зору стохастичної оптимізації використання буфера досвіду виконує дві функції. По-перше, декорелює послідовні зразки, що наближає умови навчання до припущення незалежних та однаково розподілених вибірок. По-друге, підвищує ефективність використання даних, оскільки кожен перехід може бути використаний у декількох ітераціях оновлення параметрів, що зменшує дисперсію оцінки градієнта.

Важливо підкреслити, що буфер досвіду змінює характер розподілу даних, на яких навчається мережа. Оскільки вибірки формуються з історичних переходів, алгоритм набуває властивостей off-policy навчання: оновлення параметрів Q-функції відбувається незалежно від того, якою саме політикою було згенеровано конкретний перехід. Це узгоджується з природою Q-learning як off-policy алгоритму.

У контексті незалежного агента в системі слабо пов'язаних агентів кожен агент і підтримує власний буфер D_i та власні параметри θ_i . За відсутності обміну сирими переходами між агентами процес навчання залишається повністю децентралізованим [72]. Водночас наявність буфера створює потенційну основу для майбутніх механізмів передачі знань, оскільки накопичений досвід може використовуватись не лише для локального оновлення, але й для формування узагальнених представлень політики або функції цінності.

2.2.2 Онлайн та цільова мережі

Поєднання нелінійної функціональної апроксимації, бутстрапінгу та off-policy оновлення може призводити до нестабільності навчання, що було показано як теоретично, так і експериментально [21]. Основна причина полягає в тому, що цільові значення у рівнянні Беллмана формуються на основі поточних параметрів тієї ж самої мережі, яка оптимізується. У такому випадку мінімізується функція

втрата із так званою “рухаючою ціллю”, оскільки зміна параметрів θ змінює як прогноз $Q(s, a, \theta)$, так і цільове значення. Зміна параметрів θ впливає на обидва члени виразу (формула 2.11), що може спричинити осциляції та розбіжність алгоритму

Для стабілізації навчання в алгоритмі DQN запропоновано використовувати окрему цільову мережу (target network) (рис. 2.2). Таким чином, у DQN підтримуються дві нейронні мережі однакової архітектури:

- онлайн мережа з параметрами θ ,
- цільова мережа з параметрами θ^- .

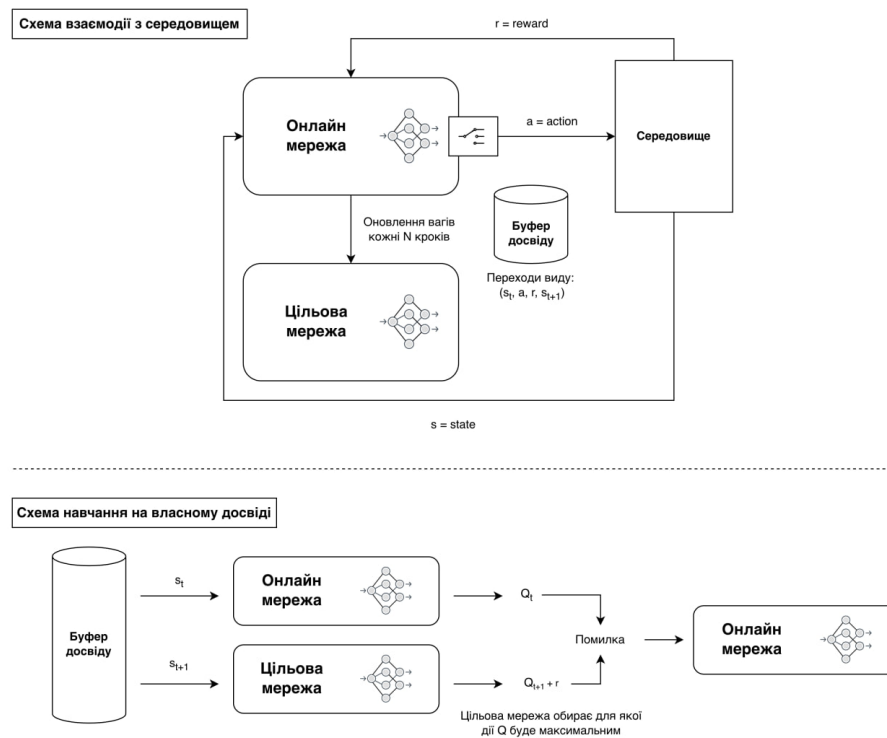


Рисунок 2.2 – Схема DQN з двома мережами (online та target)

Онлайн мережа використовується для вибору дій та оптимізації функції втрат. Цільова мережа використовується виключно для формування цільових значень:

$$y^{\text{target}} = r + \gamma * \max_{a' \in A} Q(s', a', \theta^-). \quad (2.13)$$

Параметри θ^- не оновлюються на кожному кроці. Замість цього використовується періодичне копіювання параметрів онлайн мережі. Таке

оновлення виконується кожні N кроків навчання. У період між копіюваннями цільова мережа залишається фіксованою, що забезпечує відносну сталість цільового сигналу [73]. Альтернативою повному копіюванню параметрів є так зване “м’яке” оновлення, яке частіше застосовується в похідних алгоритмах (наприклад, у DDPG). З формальної точки зору введення цільової мережі означає, що під час мінімізації функції втрат градієнт обчислюється тільки за параметрами онлайн мережі.

У контексті незалежних агентів у системі слабо пов’язаних агентів кожний агент i підтримує власні параметри θ_i та θ_i^- . Таким чином, навіть за повної децентралізації процесу навчання механізм цільової мережі забезпечує локальну стабільність оцінювання Q -функції кожного агента.

2.2.3 Independent DQN у слабо пов’язаній системі

У багатоагентних системах з навчанням з підкріпленням загальна динаміка середовища визначається спільною поведінкою множини агентів. У загальному випадку така система формалізується як стохастична гра (Markov game), у якій функція переходів має вигляд

$$P(s'|s, a_1, \dots, a_N), \quad (2.14)$$

де $a_i \in A_i$ — дія агента i .

Винагорода кожного агента визначається індивідуальною функцією

$$R_i(s, a_1, \dots, a_N). \quad (2.15)$$

У повністю загальному випадку середовище для кожного окремого агента є нестационарним, оскільки зміна політик інших агентів змінює розподіл переходів і винагород у часі. Це порушує базові припущення класичного Q -learning щодо стаціонарності. Independent DQN (IDQN) є практичним підходом, у якому кожний агент розглядає інших агентів як частину середовища та навчається незалежно, використовуючи власну Q -функцію. Таким чином, кожний агент мінімізує власну функцію втрат, не використовуючи явну інформацію про політики інших агентів.

У випадку слабого впливу одного агента на іншого (шанс зміни стану або винагороди агента і через різні дії агента j достатньо малий, або ж зміни достатньо незначні) середовище, що спостерігається агентом i , є квазі-стаціонарним. Це дозволяє наближено застосовувати результати теорії збіжності Q-learning для стаціонарних MDP.

З практичної точки зору слабкий зв'язок може виникати в системах, де:

- агенти впливають лише на локальні підмножини станів;
- винагороди є переважно індивідуальними;
- частота взаємного впливу є низькою порівняно з частотою локальних оновлень.

Важливо зазначити, що формальних гарантій збіжності IDQN у загальних стохастичних іграх не існує. Проте емпіричні дослідження демонструють, що у слабо пов'язаних або частково декомпованих задачах незалежне навчання забезпечує задовільну стабільність та масштабованість [37, 75].

У запропонованій моделі розглядається система з агентів, кожен з яких взаємодіє з власним середовищем. Для агента i середовище формалізується як марковський процес прийняття рішень. Глобальний стан системи може бути представлений як декартовий добуток локальних станів. При цьому перехідна ймовірність повністю факторизується, а функція винагороди декомонується по агентам. Таким чином, динаміка кожного агента є повністю незалежною від дій інших агентів. Глобальний процес є прямим добутком незалежних MDP, а оператор Беллмана факторизується по агентам. За цих умов навчання агентів методом Independent DQN є еквівалентним незалежним одноагентним алгоритмам DQN, які виконуються паралельно.

У такій постановці відсутня нестационарність, характерна для загальних стохастичних ігор, оскільки політика одного агента не впливає на перехідні ймовірності або винагороди іншого. Відповідно, теоретичні припущення щодо стаціонарності середовища, необхідні для коректності Q-learning [21, 76], залишаються виконаними.

З формальної точки зору така система є не слабко пов'язаною, а повністю декомпозованою. Це забезпечує стабільність навчання та дозволяє інтерпретувати подальші механізми обміну знаннями не як необхідність для компенсації нестаціонарності, а як засіб прискорення збіжності.

Особливістю реалізації є незалежність термінальних станів. Для кожного агента визначено власну множину термінальних станів $T_i \subset S_i$. Досягнення стану $s_i \in T_i$ не впливає на динаміку інших агентів. Глобальний процес переходить у термінальний стан лише тоді, коли всі агенти завершили відповідні епізоди.

У випадку асинхронного завершення епізодів можливі два режими:

- незалежний перезапуск середовища агента з початкового стану;
- перебування в абсорбуючому стані до завершення епізодів інших агентів.

Незалежний перезапуск може призводити до різної швидкості накопичення досвіду між агентами. Це, у свою чергу, формує асиметрію в якості навчених політик, що є важливим фактором для подальших механізмів передачі знань. Зокрема, агент, який швидше досягає термінального стану, може накопичувати більше епізодів за той самий проміжок часу, що впливає на стабільність та ефективність дистиляції.

Таким чином, хоча система формально декомпозована, асинхронність епізодів створює додаткові аспекти динаміки навчання, які мають значення для аналізу поширення знань між агентами.

2.2.4 Модифікації DQN

Базовий алгоритм DQN продемонстрував високу ефективність у задачах із великим простором станів, однак подальші дослідження виявили низку систематичних обмежень, зокрема переоцінювання Q-значень, чутливість до вибірки досвіду та обмеження архітектурної виразності моделі [27]. У відповідь на ці обмеження було запропоновано декілька модифікацій DQN, серед яких найбільш впливовими є Double DQN, Prioritized Experience Replay (PER), Dueling DQN та інтегрована модель Rainbow.

Однією з фундаментальних проблем класичного DQN є систематичне переоцінювання значень $Q(s,a)$, що виникає через використання оператора максимуму в цільовому значенні. Оскільки одна й та сама апроксимація використовується і для вибору дії, і для її оцінювання, статистичний шум у значеннях Q призводить до позитивного зміщення оцінки максимуму [73]. Double DQN (рис. 2.3) усуває це зміщення шляхом розділення операцій вибору та оцінювання дії. Онлайн мережа з параметрами θ використовується для визначення оптимальної дії. DDQN використовує ті самі два типи мереж, що й DQN:

- онлайн-мережа $Q(s, a; \theta)$ — навчається безпосередньо за даними з досвіду;
- цільова мережа $Q(s, a; \theta^-)$ — періодично копіює ваги з онлайн-мережі (через певну кількість ітерацій, або ж з прив'язкою до епізодів).

На відміну від DQN, у DDQN (рис. 2.3) цільове значення обчислюється за формулою:

$$y^{\text{target}} = r + \gamma Q_{\text{target}}(s', a^*, \theta^-), \quad (2.16)$$

де:

$$a^* = \arg \max_{a' \in A} Q_{\text{online}}(s, a', \theta). \quad (2.17)$$

PER прискорює навчання за рахунок фокусування на інформативних переходах, однак ускладнює статистичні властивості алгоритму та вводить додаткові гіперпараметри [33].

В Dueling DQN нейронна мережа має дві вихідні гілки [77]:

- value stream — обчислює оцінку цінності поточного стану $V(s)$,
- advantage stream — оцінює, наскільки кожна дія ає кращою за середню в цьому стані: $A(s, a)$.

Ці два компоненти комбінуються для отримання підсумкової оцінки $Q(s, a)$:

$$Q(s, a) = V(s) + (A(s, a) - \text{mean}_{a'} A(s, a')). \quad (2.18)$$

Такий підхід дозволяє ефективніше оцінювати стани, у яких вибір конкретної дії має незначний вплив на результат, покращуючи вибіркову ефективність навчання.

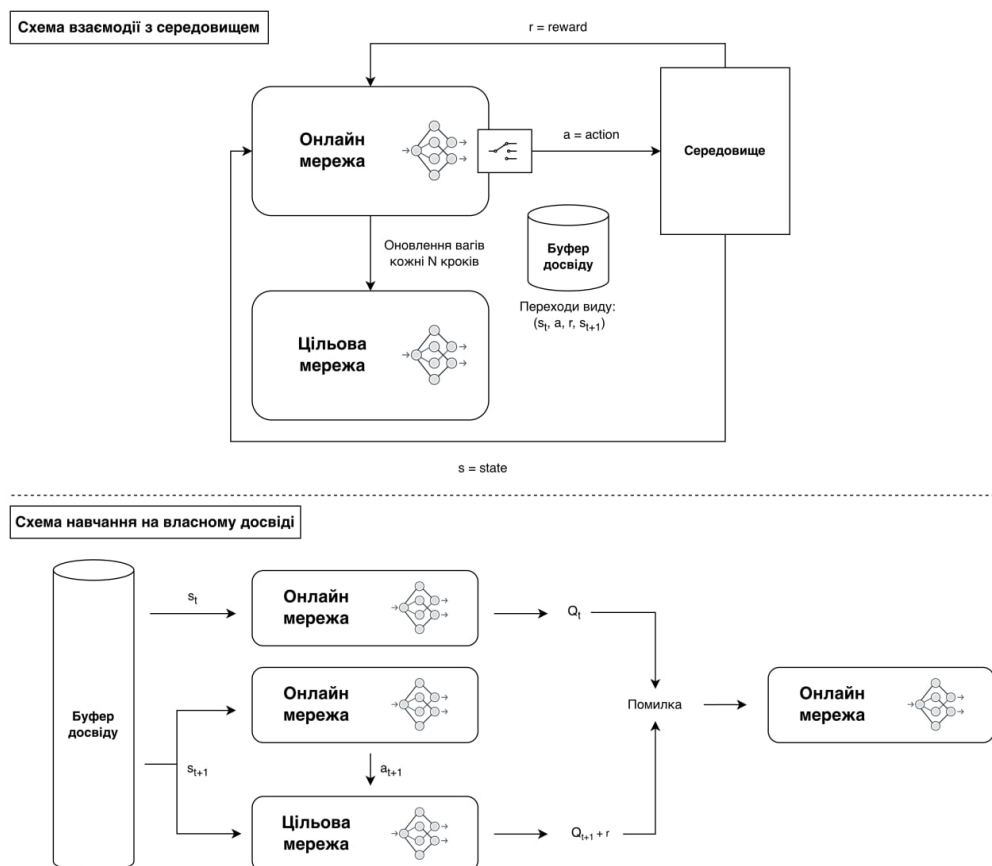


Рисунок 2.3 – Спрощена схема DDQN

Подальший розвиток ідей призвів до створення моделі Rainbow DQN, яка інтегрує декілька покращень одночасно, зокрема Double DQN, PER, Dueling архітектуру, багатокрокові повернення та розподільне представлення Q-функції. Rainbow демонструє високі результати на стандартних бенчмарках, проте значно ускладнює інтерпретацію результатів та збільшує кількість гіперпараметрів.

2.2.5 Баланс дослідження та використання

Однією з фундаментальних проблем навчання з підкріпленням є баланс між дослідженням (exploration) та використанням (exploitation). Агент повинен

одночасно використовувати вже набуті знання для максимізації поточної винагороди та досліджувати нові дії з метою покращення майбутньої політики. Надмірне використання призводить до передчасної збіжності до субоптимальної стратегії, тоді як надмірне дослідження уповільнює навчання та збільшує дисперсію результатів [21].

У контексті алгоритмів на основі Q-функції вибір дії найчастіше здійснюється відповідно до жадної політики $a_t = \arg \max_{a \in A} Q(s_t, a)$, що відповідає повному використанню поточної оцінки. Для забезпечення дослідження зазвичай застосовується ϵ -жадна стратегія, у якій з імовірністю ϵ вибирається випадкова дія, а з імовірністю $(1 - \epsilon)$ — жадна дія:

$$\pi(s) = \begin{cases} \text{uniform}(A), & \text{з ймовірністю } \epsilon \\ \arg \max_{a \in A} Q(a, s), & \text{з ймовірністю } 1 - \epsilon \end{cases} \quad (2.19)$$

Параметр $\epsilon \in [0, 1]$ може бути сталим або змінюватися у часі. Найпоширенішою є схема поступового зменшення ϵ , щоб поступово переходити від фази пошуку до все більш жадібної експлуатації знайденої стратегії. Наприклад як:

$$\epsilon^{t+1} = \epsilon^t * \epsilon_{\text{decay}}, \quad (2.20)$$

де $\epsilon_{\text{decay}} \in (0, 1)$ — швидкість падіння ϵ за епізод (коефіцієнт затухання).

У багатоагентному контексті баланс дослідження та використання набуває додаткової складності. Навіть у слабко пов'язаній системі незалежних агентів надмірне дослідження одним агентом може впливати на глобальні метрики продуктивності системи, наприклад середній час навчання. У повністю незалежній декомпованій постановці, що використовується в даному дослідженні, кожен агент підтримує власний параметр ϵ_i та здійснює дослідження незалежно від інших. Таким чином, стратегія дослідження не порушує стаціонарність локального середовища кожного агента.

2.2.6 Архітектура нейромережі

Архітектура мережі визначає здатність моделі узагальнювати в просторі станів та істотно впливає на стабільність і швидкість навчання. У випадку дискретного простору дій типовою є архітектура, у якій на вхід мережі подається вектор стану s , а на виході формується вектор Q -значень для всіх можливих дій:

$$Q(s; \theta) = [Q(s, a_1), Q(s, a_2), \dots, Q(s, a_k)]. \quad (2.21)$$

У класичній реалізації DQN для Atari-ігор нейронна мережа мала кілька згорткових (convolutional) шарів для виділення ознак із зображень екрана, після чого – кілька повнозв'язних шарів, що видавали оцінки Q для всіх дій. Одним і тим самим мережевим виходом агент користувався як для вибору дії (під час гри), так і для обчислення цільових значень при навчанні.

У задачах із компактним векторним представленням стану (наприклад, положення, швидкість, кутові параметри) доцільним є використання повнозв'язної багат шарової перцептронної архітектури. У більшості реалізацій DQN використовується нелінійність ReLU, що забезпечує стабільність градієнтів і ефективність обчислень [27]. Вихідний шар є лінійним, оскільки Q -значення можуть набувати довільних дійсних значень.

2.3 Поширення знань за допомогою дистиляції знань

У багатоагентних системах кожен агент взаємодіє із середовищем, накопичуючи власний досвід. Якщо агенти працюють незалежно, то їхній досвід може суттєво відрізнятися, і кожен агент може вивчити різні аспекти середовища. Логічно припустити, що обмін інформацією між агентами може підвищити продуктивність навчання, оскільки дозволить об'єднати їхні знання про середовище. Ранні підходи до багатоагентного навчання пропонували обмін досвідом між агентами – наприклад, спільне використання зібраних переходів (стан, дія, винагорода, наступний стан) або параметрів моделей. Обмін досвідом забезпечує швидше дослідження середовища і може прискорити збіжність

навчання [78]. Зокрема, показано, що навіть просте спільне використання траєкторій досвіду між агентами дозволяє досягати вищих винагород та прискорювати навчання порівняно з незалежним навчанням кожного агента. Проте суттєвим недоліком обміну необробленим досвідом є те, що кожен агент мусить повторно опрацювати чужі дані – фактично навчатися на досвіді інших, що створює додаткове навантаження та може бути неефективним [79]. До того ж, за відсутності належної вибіркової, масовий обмін досвідом може призвести до інформаційного перенавантаження або завадити стабільності навчання. Сучасні дослідження вказують, що для максимальної ефективності варто вибірково ділитися досвідом між агентами, відбираючи корисну інформацію [43], однак навіть у такому разі залишається проблема повторного навчання на чужих даних.

Альтернативним підходом є обмін знаннями замість сирого досвіду. Замість того щоб передавати усі зібрані переходи, агенти можуть передавати один одному узагальнені висновки зі свого досвіду – тобто навчені політики або оцінки. У цьому підході кожен агент ділиться не самими даними, а своєю інтерпретацією даних відповідно до власної моделі. Іншими словами, агент виступає як вчитель, що передає іншому агенту – учню – вже здобуті знання про те, які дії є перспективними у певних ситуаціях [80]. Такий підхід усуває потребу повторної обробки сирих даних і потенційно дозволяє швидше поширювати успішні стратегії між агентами. Наприклад, якщо один з агентів дослідив певну ділянку простору станів і вивчив ефективну послідовність дій, інші агенти можуть засвоїти цю стратегію без прямого відвідування тих самих станів.

Наприклад, дослідники [44] дослідники пропонують розглядати багатоагентну проблему як набір підзадач: кожен агент досліджує свій підпростір станів, а потім “ділиться та об’єднує” отримані рішення з іншими. Це дозволяє реалізувати принцип “поділяй і володарюй”: завдяки обміну знаннями кожен агент може зосередитися на вивченні лише частини задачі, а спільно команда покриває весь простір станів більш ефективно.

Основним механізмом реалізації передачі узагальнених знань між нейромережевими агентами є дистиляція знань (knowledge distillation, KD) [81,

82]. Первісно метод дистиляції знань було запропоновано у контексті глибокого навчання для компресії моделей: менша студентська модель навчається відтворювати виходи більшої вчительської моделі, набуваючи подібних поведінкових характеристик. Навчання студента на “м’яких” виходах вчителя (ймовірностях класів, отриманих із логітів з підвищеною температурою) дозволяє передати тонкі відмінності у розподілі ймовірностей, які містять більше інформації, ніж жорсткі мітки максимального класу. Згодом автори [53] застосували цю ідею до навчання з підкріплення, ввівши поняття дистиляції політики (policy distillation) [83, 84] – передачі стратегій поведінки від однієї агента до іншої. За допомогою policy distillation можна, наприклад, витягти стратегію навченої великої мережі (агента-експерта) і використати її для тренування компактнішої нейромережі, що діятиме на рівні експерта. Більше того, цей метод дозволяє об’єднувати декілька політик – було продемонстровано успішну дистиляцію політик з різних завдань в одну універсальну політику, яка здатна виконувати всі вхідні завдання [85]. Таким чином, дистиляція знань проклала шлях до ефективного переносу навченої поведінки між різними моделями та задачами. У багатоагентному навчанні дистиляція знань знайшла застосування для обміну політиками та оцінками між агентами з метою підвищення їхньої точності та швидкості навчання. Цей підхід часто реалізують у форматі “вчитель-учень”.

Один із варіантів – використання централізованого вчителя, який тренується з розширеною інформацією (наприклад, маючи глобальний стан усієї системи), а потім спрямовує децентралізованих учнів (агентів) через дистиляцію політики. Така схема відповідає популярній парадигмі централізованого навчання з децентралізованим виконанням (CTDE): під час навчання централізований вчитель отримує повну інформацію про середовище і виводить оптимальну спільну стратегію, тоді як кожен агент-учень навчається відтворювати власну частину цієї стратегії, використовуючи лише локальні спостереження [86].

Інший підхід – взаємна дистиляція між агентами, коли всі агенти є рівноправними вчителями й учнями один для одного. Особливо це доцільно для

однорідних агентів, що мають спільну структуру моделі та подібні цілі. У такому випадку знання, набуті одним агентом, потенційно релевантні для інших. Наприклад, автори [44] запропонували метод об'єднання політик однотипних агентів через одночасну дистиляцію політик та вирівнювання функцій цінності (value matching) [87]. Ідея полягає в тому, що всі однорідні агенти мають однакову оптимальну функцію оцінки (за умови оптимальної координації), тож їхні оцінки можна узгоджувати: якщо один агент виявив високої цінності стан, інші агенти можуть використати цю інформацію, навіть якщо ще не відвідали цей стан. Алгоритм, запропонований авторами, здійснює дистиляцію політик агентів у спільну політику, а також поширює значення оцінок (Q- або V-функцій) між агентами, приводячи їх до єдиного узгодженого наближення. В результаті комбінована політика слугує спільним “банком знань”, з якого кожен агент може черпати інформацію і продовжувати навчання. Важливо, що навчання не зупиняється після дистиляції [88]: агенти надалі донавчаються в середовищі, але вже маючи кращу початкову основу за рахунок чужих знань. Це відрізняє підхід від класичного сценарію, де дистиляція виконується лише як фінальний етап для стискання моделі – натомість тут дистиляція інтегрована у процес спільного навчання, підтримуючи безперервне поширення знань протягом тренування.

Окрім політик (акторів), дистиляції можуть підлягати і оцінюючі функції (критики). Наприклад, замість (або доповнення до) імітації виборів дій учням можна передавати оцінки Q-величин для всіх можливих дій у стані. У цьому випадку студент навчається наближувати функцію цінності дії вчителя, мінімізуючи різницю між власними Q-значеннями та Q-значеннями вчителя (наприклад, за допомогою MSE-втрат). Такий підхід застосовується, зокрема, в згаданому вище методі [86]. (CTPDE), де компактна модель агента-учня вчиться відтворювати розподіл цінності дій (action-value distribution) центрального вчителя. Дистиляція на рівні критика може узгоджувати функції цінності станів $V(s)$ між агентами – якщо один агент оцінює певний стан як перспективний, інші агенти можуть підлаштовувати свої оцінки відповідно, навіть не відвідавши цей стан безпосередньо.

Однією з ключових переваг застосування дистиляції політики у багатоагентних системах навчання з підкріпленням є підвищення гнучкості архітектури агентів за збереження їх функціональної сумісності. Хоча агенти в межах системи мають бути однорідними з точки зору інтерфейсу взаємодії із середовищем, внутрішня реалізація політики може суттєво відрізнятися. Зокрема, моделі політики можуть базуватися на глибоких нейронних мережах із різною архітектурою, кількістю шарів або нейронів, за умови однакової семантики вхідного та вихідного просторів. Такий підхід дозволяє поєднувати в межах однієї багатоагентної системи агентів з різною обчислювальною складністю та узагальнювальною здатністю без необхідності уніфікації їх внутрішньої структури.

Крім того, у випадках, коли вхідні або вихідні простори агентів не є повністю ідентичними, можливе застосування додаткових механізмів трансформації інтерфейсів. Наприклад, окремі компоненти вхідного вектора стану можуть бути перетворені за допомогою заданих аналітичних або емпіричних формул, що забезпечує коректну передачу знань між агентами з частково відмінними спостереженнями. Це розширює область застосування дистиляції політики та підвищує адаптивність системи до гетерогенних середовищ.

Важливою практичною властивістю дистиляції політики є можливість динамічного масштабування та поступового розвитку багатоагентної системи. Зокрема, система може еволюціонувати шляхом поетапного додавання агентів із простішими або, навпаки, складнішими нейронними мережами, з меншою чи більшою кількістю параметрів. При цьому відсутня необхідність повторного навчання всієї системи або втрати вже накопичених знань, оскільки вони поступово поширюються на нові агенти за допомогою механізмів дистиляції політики.

За умови наявності механізмів контролю та вибіркового застосування процесу дистиляції нові агенти можуть інтегруватися в систему та прискорено навчатися на основі політик більш досвідчених агентів. Такий підхід дозволяє суттєво скоротити час початкового навчання нових агентів, зменшити кількість

необхідних взаємодій із середовищем та підвищити загальну стабільність процесу навчання в слабко пов'язаних багатоагентних середовищах. Крім того, можливе застосування одноразової інтенсивної процедури передачі знань для нових агентів, яка дозволяє швидко ініціалізувати їх політики на основі вже накопиченого досвіду системи. Такий підхід забезпечує передачу актуальних знань про середовище та знайдені на поточний момент ефективні стратегії, що дозволяє значно скоротити початкову фазу навчання нових агентів. У результаті дистиляція політики виступає не лише механізмом передачі знань, а й інструментом безперервного розвитку та масштабування багатоагентних систем без деградації вже досягнутої ефективності.

2.4 Метод децентралізованого вибору вчителя для контролю якості передачі знань

Як зазначалося вище, за умови вибіркового поширення знань стає можливим ефективне навчання нових агентів від уже навчених або більш досвідчених. У цьому випадку доцільним є цілеспрямований вибір агентів-вчителів, оскільки нові агенти на початкових етапах навчання володіють обмеженим обсягом корисної інформації. Поширення таких знань може бути нерелевантним або навіть негативно впливати на процес навчання інших агентів. Саме це обґрунтовує необхідність застосування механізмів вибору вчителя, які дозволяють підвищити якість і стабільність процесу дистиляції знань. Важливо зазначити, що запропонований підхід не обмежується вибором одного вчителя, а допускає формування множини агентів-вчителів, знання яких можуть бути агреговані в процесі навчання агента-учня.

Однією з фундаментальних переваг багатоагентних систем є їх децентралізована природа. У стаціонарних середовищах зі слабко пов'язаними агентами, які можуть бути апроксимовані як незалежні, операції вибору вчителя можуть виконуватися децентралізовано, без використання центрального координатора. Кожен агент-учень самостійно оцінює потенційних агентів-

вчителів на основі доступної локальної інформації, що зберігає масштабованість системи та зберігає високий рівень відмово стійкості.

Запропонований метод вибору вчителя ґрунтується на припущенні, що навчання від агента з вищою точністю або кращими показниками якості політики з високою ймовірністю сприятиме підвищенню ефективності агента-учня. Це припущення узгоджується з інтуїтивною ідеєю перенесення знань у навчанні з підкріпленням, а також із загальними принципами дистиляції знань у машинному навчанні.

Задачу вибору агента-вчителя або множини агентів-вчителів доцільно представити у вигляді двох послідовних операцій (рис. 2.4). На першому етапі агент-учень виконує ранжування потенційних агентів-вчителів відповідно до обраного критерію точності або ефективності політики. На другому етапі здійснюється фільтрація впорядкованої множини, що передбачає відкидання агентів із недостатнім рівнем ефективності та/або обмеження кількості обраних вчителів до заздалегідь визначеного значення. Така декомпозиція задачі забезпечує гнучкість реалізації методу та дозволяє адаптувати його до різних умов функціонування багатоагентної системи.

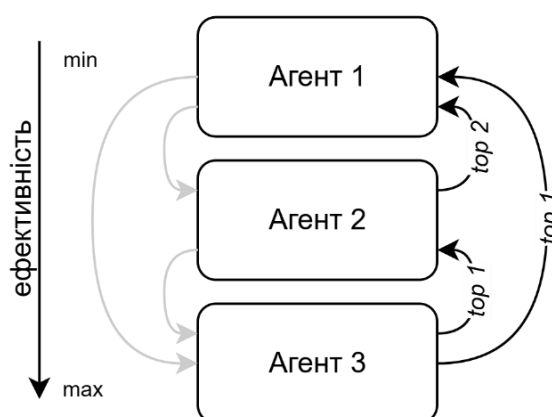


Рисунок 2.4 – Схема вибору вчителя(-ів) як задача децентралізованих пріоритизації та фільтрації запитів на поширення знань за продуктивністю

У межах даної роботи основна увага приділяється режиму безперервного навчання (continuous learning), за якого процес оновлення політики агентів та

поширення знань між ними не є одноразовою процедурою, а здійснюється ітеративно протягом усього часу функціонування системи. Агенти постійно навчаються на власному досвіді взаємодії із середовищем, водночас періодично обмінюючись знаннями з іншими агентами з метою узгодження або покращення апроксимації ціннісних функцій чи політик. Такий підхід дозволяє адаптуватися до змін у середовищі та поступово підвищувати якість прийняття рішень без необхідності повного перенавчання системи. Тобто, поширення знань відбувається багаторазово, а агенти можуть мінятися ролями з часом.

Нехай багатоагентна система містить набір агентів:

$$G = \{g_1, g_2, \dots, g_n\}. \quad (2.22)$$

Пропонується агент в цій системі описувати як:

$$g_i = \{R_i, D_i, \theta_i\}, \quad (2.23)$$

де R_i — сумарна зібрана винагорода цим агентом за минулий епізод (або історія зібраних винагород),

D_i — буфер досвіду,

θ_i — нейромережа, що визначає Q-функцію та політику.

Сумарна зібрана винагорода дозволить оцінювати продуктивність агенту та її зміну, буфер міститиме історію переходів, що важливо за умови різної ефективності агентів чи різниці в середовищах (наприклад, наприклад для фізично віддалених агентів, в яких логічно та сама система, але з різними параметрами). Для Q агента, цільове значення для дистиляції політики буде

$$y^{\text{target}} = Q(s_{\text{teacher}}, a_{\text{teacher}}, \theta_{\text{teacher}}). \quad (2.24)$$

Для формування набору даних дистиляції можуть використовуватися як онлайн-мережа, так і цільова мережа агента-вчителя. Онлайн-мережа оновлюється частіше та здатна швидше передавати нововиявлені стратегії, проте через це вона є менш стабільною порівняно з цільовою мережею, параметри якої змінюються повільніше.

Переходи, що використовуються для дистиляції, мають відбиратися з буфера досвіду агента-вчителя, що гарантує поширення знань виключно про ті стани та дії, з якими вчитель фактично взаємодівав у середовищі. Найпростішим підходом

до вибору досвіду, який було досліджено в межах цієї роботи, є випадковий неповторюваний вибір (семпл) переходів із буфера досвіду. Такий підхід успадковує переваги класичного випадкового відбору, що застосовується під час навчання агента на власному досвіді.

Зокрема, випадковий вибір переходів сприяє руйнуванню часових кореляцій між послідовними спостереженнями, зменшує ризик надмірної концентрації моделі на поточній стратегії завдяки використанню досвіду з попередніх епізодів, а також забезпечує покриття ширшого діапазону переходів у просторі станів і дій. Водночас такий підхід має потенціал до подальшого вдосконалення за аналогією з пріоритезованим буфером досвіду, у якому окремі переходи отримують вищу ймовірність відбору залежно від їхньої інформативності.

Одним із можливих критеріїв оцінювання цінності переходів у контексті дистиляції може бути відмінність між знаннями агента-учня та агента-вчителя або рівень переваги вчителя над учнем у певному підпросторі даних. Такий підхід відкриває можливості для адаптивного відбору досвіду, спрямованого на максимізацію користі від процесу поширення знань.

Ціль же студента — зменшити різницю між своєю та вчителя Q-функціями. Для прикладу, наведена функція втрат, що обраховується як середня квадратична похибка / Mean Square Error (MSE). Проте можна використовувати Huber loss.

$$L(\theta^{\text{student}}) = \text{MSE}(y^{\text{target}}, Q(s_{\text{teacher}}, a_{\text{teacher}}, \theta_{\text{student}})). \quad (2.25)$$

У межах запропонованого підходу зібрана агентом накопичена винагорода використовується як кількісна міра продуктивності його політики. Така метрика дозволяє здійснювати порівняльну оцінку агентів на основі результатів їхньої взаємодії із середовищем без залучення зовнішніх еталонів або централізованих механізмів оцінювання.

Агентом-вчителем визначається один або декілька агентів, які продемонстрували найвищу продуктивність протягом заданого періоду оцінки, наприклад окремого епізоду, фіксованого часового інтервалу або траєкторії взаємодії із середовищем. При цьому вибір вчителя не обмежується одним

агентом і може передбачати формування множини агентів-вчителів, знання яких використовуються в процесі дистиляції.

Визначення агентів-вчителів здійснюється децентралізовано — кожен агент-учень самостійно виконує оцінювання та відбір потенційних вчителів на основі отриманої інформації. Такий підхід узгоджується з принципами децентралізованих багатоагентних систем та не потребує централізованого контролю або агрегування глобального стану системи.

Разом з тим запропонований метод накладає певні обмеження на процес навчання та обміну знаннями. А саме, на період оцінювання продуктивності політика агента має залишатися незмінною, тобто процес онлайн-навчання тимчасово призупиняється. У разі, якщо агент змінює політику під час збору винагороди для оцінювання, наприклад унаслідок навчання на власному досвіді або дистиляції знань, отримана накопичена винагорода не буде коректно характеризувати жодну конкретну версію політики, а відображатиме комбінований ефект їх послідовної зміни. По-друге, для коректного ранжування агентів-вчителів у загальному випадку вимагається отримання інформації про продуктивність усіх агентів системи. Це означає, що в окремі моменти часу агент може тимчасово втратити зв'язок з іншими агентами, внаслідок чого він не зможе бути обраний як агент-вчитель, навіть якщо за період оцінки продемонстрував найвищу продуктивність. Така властивість є наслідком децентралізованої організації системи та обмежень комунікації між агентами.

Варто також зазначити, що вибір кількості агентів-вчителів є важливим гіперпараметром методу. Надмірна кількість вчителів за умови значного перекриття досвіду може призвести до перенавчання на окремих ділянках простору станів і дій, втрати впливу власного досвіду агента-учня або до надмірного усереднення політик. Натомість занадто мала кількість вчителів може уповільнити процес підвищення продуктивності та зменшити ефект від поширення знань.

Вибір тривалості періоду оцінювання також залежить від властивостей середовища та постановки задачі. Оцінювання на рівні окремих епізодів є

доцільним у середовищах зі скінченними та відносно короткими епізодами або за умови штучного обмеження їх довжини. У такому випадку процес дистиляції відбувається рідше, що потенційно може сповільнити навчання. Водночас це може бути компенсовано інтенсивнішим навчанням, контрольованим кількістю епох та використанням великого буфера досвіду, що підвищує стабільність процесу навчання та сприяє швидшому знаходженню ефективних стратегій.

Альтернативно, оцінювання продуктивності може виконуватися через фіксовану кількість кроків взаємодії із середовищем. Такий підхід є доцільним за наявності інформативної винагороди на кожному кроці та за відсутності суттєво відкладених наслідків дій агента.

Зазначені обмеження безпосередньо впливають на вибір методів оцінювання цінності та бутстрапінгу. Зокрема, запропонований підхід є сумісним із методами офлайн або пакетного навчання, за яких оновлення параметрів моделі не відбувається під час збору винагороди для оцінювання продуктивності політики.

У цій роботі використовується офлайн TD(0)-підхід, за якого оновлення ціннісної функції виконується після завершення епізоду на основі випадкової вибірки переходів із буфера досвіду. Такий підхід дозволяє розділити процес оцінювання продуктивності політики та процес навчання, забезпечуючи коректність зібраної накопиченої винагороди як міри якості політики.

Крім того, із запропонованим підходом узгоджуються методи Monte Carlo-оцінювання та n-крокові методи часової різниці, а також узагальнений підхід TD(λ), який поєднує властивості бутстрапінгу та використання багатокрокових повернень. Спільною властивістю цих методів є можливість виконання оновлень у пакетному режимі після завершення періоду оцінювання, що є критично важливим для коректного визначення агентів-вчителів на основі зібраної винагороди.

2.5 Метод динамічного децентралізованого вибору кількості епох дистиляції політики для контролю сили передачі знань

У межах даної роботи пропонується метод контролю інтенсивності навчання агентів шляхом динамічного децентралізованого визначення кількості епох дистиляції політики. На відміну від підходів, що фокусуються виключно на виборі джерела знань, запропонований метод також враховує необхідність регулювання сили впливу процесу дистиляції на політику агента-учня.

Окрім контролю якості передачі знань, зокрема шляхом вибору більш продуктивних агентів-вчителів, доцільним є також контроль інтенсивності такої передачі. Надто слабкий вплив дистиляції може призвести до передачі недостатнього обсягу корисних знань, тоді як надмірна інтенсивність навчання здатна спричинити перенавчання агента-учня, втрату впливу власного досвіду або надмірне узгодження політик агентів. Останнє, у свою чергу, зменшує різноманітність поведінки в багатоагентній системі, знижує потенціал до дослідження середовища та підвищує ризик застрягання у псевдооптимальних стратегіях [89].

У запропонованому підході інтенсивність дистиляції політики контролюється за допомогою кількості епох навчання, що виконуються під час одного акту обміну знаннями. Альтернативний механізм регулювання сили впливу шляхом пропуску обміну знаннями, як це запропоновано в окремих існуючих роботах, у даному випадку є малоприматним, оскільки процес поширення знань і так відбувається відносно рідко. Відповідно, використання нульової кількості епох означало б повну відсутність навчання в межах відповідного циклу обміну.

Використання фіксованої кількості епох дистиляції має обмежену адаптивність, оскільки не враховує зміну стану системи з часом, а також можливу значну різницю в продуктивності між агентами-учнями та агентами-вчителями. У таких умовах однакова інтенсивність навчання може бути недостатньою для

агентів із низьким рівнем підготовки та надмірною для агентів, політики яких уже є близькими за якістю.

Запропонований метод контролю інтенсивності навчання ґрунтується на припущенні, що навчання від агента з більшою різницею в продуктивності потенційно здатне забезпечити більший приріст якості політики агента-учня. Як кількісна міра продуктивності використовується накопичена винагорода, отримана агентом у попередньому епізоді, часовому періоді або іншому визначеному інтервалі оцінювання. Відповідно, різниця в продуктивності між агентом-учнем та кожним із потенційних агентів-вчителів використовується як сигнал для визначення інтенсивності дистиляції.

Визначення кількості епох дистиляції виконується децентралізовано та індивідуально для кожного агента-вчителя. Оскільки різні агенти-вчителі можуть суттєво відрізнятися за рівнем продуктивності, для кожного з них обчислюється власне значення кількості епох дистиляції, пропорційне різниці в продуктивності між агентом-учнем та відповідним агентом-вчителем (рис. 2.5). Такий підхід дозволяє диференційовано враховувати внесок кожного вчителя та уникати як недостатнього, так і надмірного впливу процесу дистиляції.

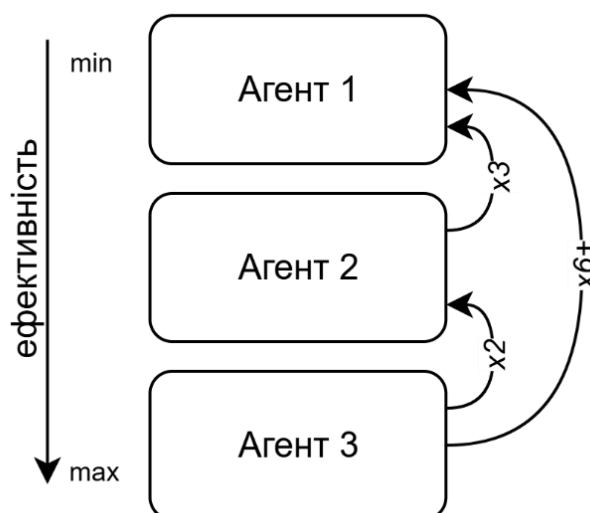


Рисунок 2.5 – Схема-приклад визначення інтенсивності поширення знань між агентами

Разом з тим запропонований метод накладає певні обмеження на процес навчання. По-перше, для його коректної роботи необхідна комунікація між агентами системи з метою обміну інформацією про їхню продуктивність, що є спільною вимогою і для раніше розглянутих методів вибору вчителя. По-друге, на період оцінювання продуктивності політика агента має залишатися незмінною, що виключає можливість онлайн-навчання в межах цього інтервалу та забезпечує коректність зібраної винагороди як міри якості політики.

Логічним є також припущення про нелінійний характер залежності між різницею в продуктивності агентів та доцільною кількістю епох дистиляції. Це пов'язано з тим, що надмірне збільшення інтенсивності навчання при великих відмінностях у продуктивності може мати обмежений або навіть негативний ефект, тоді як для малих відмінностей достатньою є незначна кількість епох.

Для виключно позитивних винагород пропонується така формула розрахунку кількості епох для дистиляції політики:

$$E^{\text{share}} = \left(\frac{R_{\text{teacher}}}{R_{\text{student}}} \right)^{\text{expPower}}, \quad (2.26)$$

де R_{teacher} — кумулятивна винагорода, обрана агентом, що вважається вчителем за обраний період оцінки.

R_{student} — кумулятивна винагорода, обрана агентом, що вважається учнем за обраний період оцінки. В цьому випадку $R_{\text{teacher}} \geq R_{\text{student}} > 0$.

expPower — параметр, що відповідає за лінійність, або ступінь нелінійності залежності відношення продуктивності агентів до кількості епох.

Для систем, у яких можливе отримання не лише додатної винагороди, але й штрафу (негативної винагороди), формула (2.26) втрачає коректність у низці випадків. Зокрема, вона не може бути застосована при $R_{\text{student}} = 0$, а також є некоректною за умов $0 > R_{\text{teacher}} \geq R_{\text{student}}$. оскільки в таких ситуаціях різниця між накопиченими винагородами агента-учня та агента-вчителя не відображає їх відносну продуктивність у коректному масштабі. Аналогічна проблема виникає у випадках, коли порівняння здійснюється виключно за модулем різниці накопичених винагород.

Для врахування зазначених випадків пропонується наступний спосіб обчислення кількості епох дистиляції політики. Спочатку визначається діапазон можливих значень кількості епох:

$$\Delta E = E_{\max} - E_{\min}. \quad (2.27)$$

Далі обчислюється нормалізована різниця продуктивності між агентом-вчителем та агентом-учнем:

$$\Delta R_{\text{norm}} = \frac{(R_{\text{teacher}} - R_{\text{student}})}{(R_{\max} - R_{\min})}. \quad (2.28)$$

Кінцева кількість епох дистиляції визначається за формулою

$$E^{\text{share}} = (E_{\min} + \Delta E * \Delta R_{\text{norm}})^{\text{expPower}}, \quad (2.29)$$

де $E_{\min}$ та $E_{\max}$ — відповідно мінімальна та максимальна кількість епох дистиляції політики, задані розробником системи.

$R_{\min}$ та $R_{\max}$ ($E_{\max} > E_{\min} > 0$) — відповідають мінімально та максимально можливій накопиченій винагороді за період оцінювання. Зазначені величини не обов'язково мають відповідати фактичним екстремальним значенням винагороди, а можуть задаватися оціночно. При цьому ширший діапазон між $R_{\min}$ та $R_{\max}$ призводить до зменшення середньої кількості епох дистиляції, тоді як його звуження — до її збільшення.

Кількість епох дистиляції в усіх розглянутих випадках обмежується мінімальним та максимальним значеннями $E_{\min}$ та $E_{\max}$. Отримане значення кількості епох додатково приводиться до додатного цілочисельного значення перед використанням у процесі навчання.

На рис. 2.6 наведена візуалізація формули 2.29 при $\text{expPower} = 1$ (лінійний зв'язок між різницею досвіду між агентами, та кількості епох навчання за умови $E_{\max} = 30$, $E_{\min} = 1$, $R_{\max} = 300$, $R_{\min} = -300$. Тут максимальна кількість епох буде застосована при максимальній різниці в продуктивності ($R_{\max} - R_{\min}$).

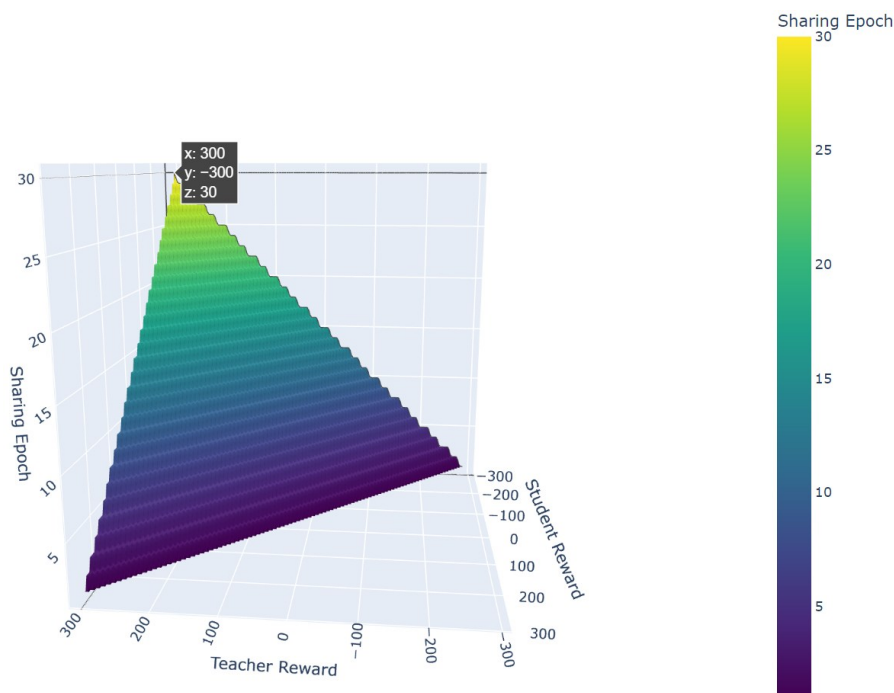


Рисунок 2.6 – Графік залежності кількості E^{share} (формула 2.29) від різниці ефективності вчителя та учня при $\text{expPower} = 1$

На рис. 2.7 наведена візуалізація формули 2.29 при $\text{expPower} = 2$ (лінійний зв'язок між різницею досвіду між агентами, та кількості епох навчання за тих самих умов, що і на малюнку 2.6. Тут максимальна кількість епох буде застосована при максимальній різниці у 110+ одиниць.

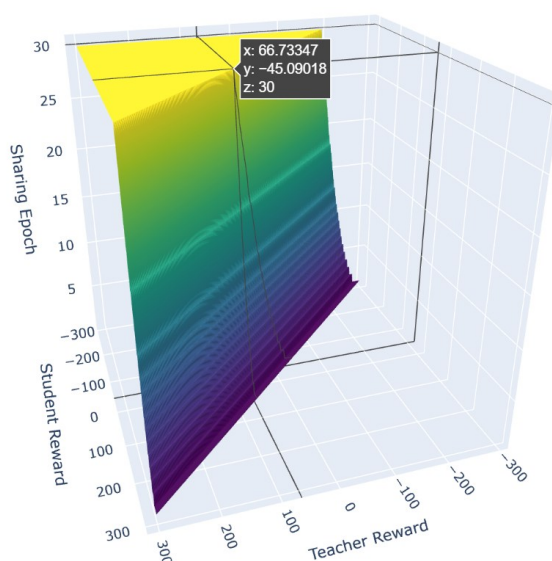


Рисунок 2.7 – Графік залежності кількості E^{share} (формула 2.29) від різниці ефективності вчителя та учня при $\text{expPower} = 2$

2.6 Поєднання методів контролю якості та інтенсивності передачі знань

Розроблені в межах цієї роботи методи контролю якості та інтенсивності передачі знань доповнюють один одного та можуть застосовуватися спільно в межах єдиного механізму поширення знань для досягнення максимальної ефективності. Метод вибору агента-вчителя спрямований на забезпечення якості переданих знань шляхом обмеження джерел дистиляції агентами з вищою продуктивністю. Окрім цього, він передбачає явне обмеження кількості агентів-вчителів, що зменшує ризик перенавчання та надмірного узагальнення політики агента-учня внаслідок одночасного навчання від великої кількості джерел.

Метод динамічного вибору кількості епох дистиляції, у свою чергу, дозволяє контролювати інтенсивність впливу процесу поширення знань. Адаптація кількості епох залежно від різниці в продуктивності між агентом-учнем та кожним агентом-вчителем підвищує ефективність дистиляції, водночас зменшуючи ризик перенавчання у випадках, коли політики агентів є близькими за якістю. Таким чином, перший метод відповідає за якість джерел знань, тоді як другий — за силу їх впливу на процес навчання.

Разом з тим запропоновані методи не враховують потенційну користь поширення знань від агентів з нижчою продуктивністю, але з якісно відмінним досвідом. Зокрема, можливі ситуації, у яких агенти взаємодіють із середовищем, що має однаковий інтерфейс, але складається з кількох логічно схожих підсередовищ або кластерів досвіду. У такому випадку окремі агенти можуть спеціалізуватися на різних кластерах, накопичуючи якісно відмінний досвід. Навіть якщо один із агентів демонструє вищу накопичену винагороду за епізод, інший агент може володіти корисними знаннями щодо підпросторів станів і дій, з якими перший агент взаємодіє рідше.

У подібних умовах поширення знань виключно від агентів з найвищою продуктивністю може обмежувати потенціал навчання, тоді як вибіркове залучення агентів із якісно іншим досвідом могло б сприяти підвищенню

загальної продуктивності системи. Аналогічні ефекти частково підтверджуються результатами досліджень у галузі дистиляції знань для моделей класифікації з незбалансованими наборами даних, де різні моделі володіють знаннями про різні підмножини класів. Крім того, подібні ідеї були використані в роботах [44, 86], зокрема в контексті дистиляції політик та узгодження ціннісних функцій у багатоагентних системах навчання з підкріпленням.

Водночас у межах даної роботи врахування якісно відмінного досвіду агентів не досліджується. Запропоновані методи не включають механізмів, які дозволяли б балансувати між навчанням від агентів з вищою продуктивністю та агентів із якісно іншим, але потенційно корисним досвідом. Реалізація такого підходу вимагала б розширення методу вибору агентів-вчителів із урахуванням додаткових критеріїв різноманітності досвіду, а також модифікації механізму визначення кількості епох дистиляції для врахування не лише кількісної, але й якісної різниці між агентами. Дослідження таких механізмів розглядається як перспективний напрям подальших робіт.

2.7 Висновки до розділу

У другому розділі було сформовано теоретичні та алгоритмічні засади дослідження децентралізованого обміну знаннями в системі слабо пов'язаних агентів, що навчаються з підкріпленням. Розділ забезпечує формальний зв'язок між класичною теорією марковських процесів прийняття рішень та сучасними методами глибокого навчання з підкріпленням, які використовуються в експериментальній частині роботи.

Окрему увагу приділено модифікаціям DQN. Проаналізовано проблему переоцінювання значень Q та показано, що Double DQN зменшує систематичне зміщення шляхом розділення операцій вибору й оцінювання дії. Також коротко охарактеризовано Prioritized Experience Replay, Dueling DQN та Rainbow DQN [90] як розширення, що підвищують ефективність навчання, але водночас ускладнюють інтерпретацію результатів. Обґрунтовано доцільність використання

у дослідженні саме Double DQN як компромісу між стабільністю та методологічною прозорістю.

У завершальній частині розділу було формалізовано модель Independent DQN у слабо пов'язаній системі. Показано, що у випадку повної факторизації динаміки та винагород система з декількох агентів еквівалентна декільком незалежним MDP, а навчання кожного агента є еквівалентним одноагентному DQN. Це дозволяє зберегти припущення стаціонарності середовища та застосовувати класичні результати теорії Q-learning. Таким чином, розглянута постановка забезпечує контрольоване середовище для подальшого аналізу механізмів передачі знань між агентами без змішування ефектів нестаціонарності та координації.

На основі проведеного аналізу запропоновано підхід до децентралізованого вибору агента-вчителя на основі показників продуктивності, а також механізм адаптивного контролю інтенсивності передачі знань, що реалізується через зміну кількості епох дистиляції політики. Сформульовано математичні моделі визначення інтенсивності передачі знань між агентами, що враховують різницю продуктивності агентів та діапазон можливих значень винагород середовища.

РОЗДІЛ 3. РОЗРОБКА МЕТОДУ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ЗНАННЯМИ З МЕХАНІЗМАМИ КОНТРОЛЮ ЯКОСТІ ТА ІНТЕНСИВНОСТІ ПЕРЕДАЧІ

У попередньому розділі було розроблено методи та моделі навчання мультиагентної системи на базі глибинного навчання з підкріпленням, а також визначено механізми взаємодії агентів та обміну знаннями. Запропоновані підходи потребують експериментальної перевірки з метою оцінки їх ефективності та практичної придатності.

У зв'язку з цим даний розділ присвячено дослідженню ефективності розроблених методів у різних умовах функціонування мультиагентного середовища. Особлива увага приділяється аналізу процесу навчання агентів, швидкості збіжності, стабільності роботи алгоритмів та якості отриманих стратегій.

У межах розділу розглядається організація експериментального середовища, описуються сценарії проведення досліджень, а також підходи до оцінювання результатів.

3.1 Методологія експериментального дослідження

Для проведення експериментального дослідження ефективності запропонованих методів необхідно визначити умови їх реалізації, включаючи характеристики середовища, параметри навчання та конфігурацію мультиагентної системи. Коректна постановка експерименту є важливою передумовою для отримання достовірних і відтворюваних результатів.

У даному підрозділі розглядається експериментальне середовище, що використовується для оцінювання підходу, а також описуються сценарії навчання агентів і ключові параметри алгоритмів. Особливу увагу приділено вибору середовищ типу CartPole та LunarLander, конфігурації агентів та налаштуванню процесу навчання.

3.1.1 Формулювання дослідницьких гіпотез

У межах даного експериментального дослідження розглядається повністю декомпозована багатоагентна система на основі IDQN [21, 23, 36, 115], у якій кожен агент навчається в окремому незалежному середовищі (між агентами відсутній вплив через стан чи винагород), а глобальна система може бути формально представлена як прямий добуток незалежних марковських процесів прийняття рішень. За таких умов взаємодія між агентами не зумовлена динамікою середовища, а виникає виключно через механізми передачі знань.

Метою дослідження є встановлення того, чи дозволяє децентралізований контроль якості та сили передачі знань підвищити ефективність навчання агентів порівняно з базовими схемами обміну або їх відсутністю. Для цього формулюються наступні дослідницькі гіпотези.

H1. Децентралізований вибір вчителя на основі показників продуктивності забезпечує вищу ефективність навчання порівняно з випадковим вибором або схемою all-to-all обміну.

Ця гіпотеза ґрунтується на припущенні, що якість джерела знань є критичним фактором результативності передачі. У випадку випадкового вибору або повного взаємного обміну між усіма агентами не враховується поточний рівень сформованості політики. Це може призводити до передачі нестабільних або менш ефективних стратегій, що знижує сумарну продуктивність системи. Натомість ранжування агентів за показниками винагороди дозволяє обмежити передачу знань лише від більш результативних агентів, що потенційно зменшує негативна передача та прискорює збіжність.

H2. Динамічний контроль кількості епох дистиляції перевершує фіксовану кількість епох з точки зору швидкості збіжності та стабільності навчання.

Фіксована кількість епох дистиляції не враховує поточну різницю продуктивності між учнем і вчителем. За незначної різниці надмірна кількість епох може призводити до перенавчання або втрати індивідуальних властивостей політики агента. За значної різниці — недостатня кількість епох обмежує

потенціал корисної передачі. Відповідно, адаптивна схема, у якій інтенсивність передачі знань є функцією різниці продуктивності, розглядається як більш узгоджена з поточним станом системи та здатна забезпечити кращий баланс між самонавчанням і зовнішнім впливом.

Н3. Поєднання контролю якості та сили передачі знань забезпечує більший приріст нормалізованої площі під кривою навчання (AUC), ніж застосування кожного з методів окремо.

Окреме застосування відбору вчителя не регулює масштаб впливу, тоді як ізольоване використання динамічної дистилляції не гарантує високої якості джерела знань. Передбачається, що одночасний контроль двох аспектів — джерела та інтенсивності передачі — формує адаптивний механізм регулювання передачі, здатний мінімізувати негативний вплив та підсилити позитивний. Таким чином, очікується наявність синергетичного ефекту, який проявляється у більшому прирості інтегральних показників ефективності порівняно з кожним із підходів окремо.

Н4. Обмін знаннями підвищує не лише середню, а й максимальну продуктивність системи.

У більшості досліджень основний акцент робиться на середній продуктивності агентів. Проте максимальна продуктивність характеризує здатність системи формувати високоефективні політики хоча б в одного агента. Якщо механізм передачі знань є ефективним, очікується прискорене поширення успішних стратегій та підвищення як середнього рівня результативності, так і найкращого досягнутого показника. Це дозволяє оцінити вплив обміну не лише як механізму вирівнювання агентів, але і як інструменту посилення їхнього потенціалу.

3.1.2 Опис середовищ та постановка задачі

Навчання відбувається епізодично. Кожен агент накопичує власний буфер досвіду та здійснює оновлення параметрів після завершення епізоду. У більшості

експериментів використовується синхронізація після завершення епізоду: агенти очікують завершення епізодів один одного перед виконанням процедур самонавчання та можливого обміну знаннями. Така схема забезпечує узгодженість моментів передачі знань без централізованого керування процесом навчання.

Для оцінювання ефективності запропонованих механізмів використовуються два класичні середовища з дискретною множиною дій.

CartPole-v1. Задача полягає в утриманні вертикального положення стрижня [91, 116], закріпленого на рухомій платформі. Стан середовища описується чотиривимірним вектором (позиція та швидкість платформи, кут і кутова швидкість стрижня). Простір дій є дискретним і складається з двох можливих дій (рух ліворуч або праворуч).

У рамках дослідження використовується модифікована функція винагороди: агент отримує +1 за кожен крок утримання стрижня та штраф при завершенні епізоду. Максимальна довжина епізоду обмежена 500 кроками. Така модифікація дозволяє підвищити чутливість системи до помилок політики та створює ширший діапазон значень сумарної винагороди. Для задачі приймається оцінний діапазон винагород $R_{\min} = 0$, $R_{\max} = 500$, що використовується в процедурах нормалізації при динамічній дистиляції.

LunarLander-v3. Задача полягає у м'якій посадці космічного апарата в заданій зоні [92]. Стан описується багатовимірним вектором, що включає координати, швидкості, кут нахилу, кутову швидкість та інформацію про контакт опор з поверхнею. Простір дій дискретний (чотири дії: відсутність тяги, головний двигун, лівий або правий бічний двигун).

Використовується стандартна функція винагороди середовища, яка враховує точність посадки, стабільність орієнтації, витрати палива та факт успішного завершення. Максимальна довжина епізоду становить до 1000 кроків. Для нормалізації різниці продуктивності приймається оцінний діапазон $R_{\min} = -300$, $R_{\max} = 300$, що відповідає типовим межам сумарної винагороди в даному середовищі.

Середовища CartPole-v1 та LunarLander-v3 реалізовані в бібліотеці Gym/Gymnasium [93, 94]. Обидва середовища обрано з таких причин:

- вони мають дискретний простір дій, що узгоджується з архітектурою DQN;
- характеризуються різною складністю динаміки;
- дозволяють дослідити вплив механізмів обміну знаннями як у відносно простій (CartPole), так і в складнішій (LunarLander) задачі керування.

Постановка задачі експериментального дослідження полягає у визначенні впливу децентралізованого контролю якості та сили передачі знань на ефективність навчання агентів у декомпованій IDQN-системі.

3.1.3 Метрики оцінювання

Для кількісної оцінки ефективності запропонованих методів використовується система метрик, що дозволяє проаналізувати як динаміку навчання, так і підсумкову продуктивність агентів. Оцінювання проводиться на основі результатів множинних незалежних запусків із різними початковими ініціалізаціями та випадковими зернами генератора.

Середня винагорода (Average Reward) характеризує інтегральну продуктивність системи з урахуванням усіх агентів. Для кожного епізоду обчислюється середнє значення сумарної винагороди всіх агентів:

$$R_{\text{avg}}(t) = \frac{1}{|G|} \sum_{i=1}^{|G|} R_i(t), \quad (3.1)$$

де $R_i(t)$ — сумарна винагорода i -го агента в епізоді t ,

$|G|$ — кількість агентів у системі.

Ця метрика дозволяє оцінити загальний рівень навчання системи та ступінь вирівнювання продуктивності між агентами. Для зменшення варіативності криві навчання додатково згладжуються за допомогою ковзного середнього (МА) з фіксованим вікном на графіках.

Оскільки досліджувана система є повністю декомпозованою та не містить конкурентної взаємодії між агентами (тобто не є грою з нульовою сумою), середня винагорода може використовуватися як інтегральний показник продуктивності системи. Збільшення значення $R_{avg}(t)$ може бути наслідком покращення політики одного, декількох або всіх агентів одночасно, і не супроводжується обов'язковим зниженням продуктивності інших агентів.

Максимальна винагорода (Max Reward) визначається як найбільше значення сумарної винагороди серед агентів у кожному епізоді:

$$R_{max}(t) = \max_{i \in \mathcal{A}} R_i(t). \quad (3.2)$$

Дана метрика відображає здатність системи формувати високоефективні політики хоча б в одного агента. На відміну від середнього показника, вона дозволяє оцінити верхню межу досягнутої продуктивності та виявити ефект прискореного поширення ефективних стратегій.

Додатково слід зазначити, що метрика $R_{max}(t)$ не означає фіксації ролі лідера за конкретним агентом протягом навчання. У процесі епізодичного навчання агенти можуть змінюватися в ролі найпродуктивнішого, що відображає стохастичний характер дослідження середовища та асиметрію накопиченого досвіду. Таким чином, Max Reward характеризує верхню межу досягнутої продуктивності системи в кожний момент часу, а не стабільну перевагу окремого агента.

Спільний аналіз метрик Average Reward та Max Reward дозволяє інтерпретувати структурні зміни у розподілі продуктивності між агентами:

- якщо R_{avg} зростає, а R_{max} залишається приблизно сталим, це свідчить про вирівнювання агентів: менш ефективні агенти наближаються до рівня найкращого без втрати максимальної продуктивності;
- якщо зростають обидві метрики, це означає загальне покращення системи та підвищення як середнього, так і найкращого досягнутого результату. Або ж тільки збільшення продуктивності найкращого агента, без суттєвого впливу на інших;

- якщо $R_{\max}$ зростає, а R_{avg} залишається сталою або зменшується, це може свідчити про збільшення дисперсії між агентами та концентрацію прогресу в одного або декількох агентів;
- якщо зменшується $R_{\max}$ при відносній стабільності R_{avg} , це може означати втрату пікової ефективності при одночасному вирівнюванні системи.

Для інтегральної оцінки швидкості та якості навчання використовується нормалізована площа під кривою навчання / Normalized AUC (Area Under Curve) [117]. Нехай T — загальна кількість епізодів. Тоді площа під кривою для середньої винагороди визначається як

$$\text{AUC} = \sum_{t=1}^T R_{\text{avg}}(t). \quad (3.3)$$

З метою коректного порівняння між різними середовищами та конфігураціями виконується нормалізація відносно теоретичного або емпірично визначеного діапазону винагород:

$$\text{AUC}_{\text{norm}} = \frac{\text{AUC} - TR_{\min}}{T(R_{\max} - R_{\min})}. \quad (3.4)$$

Нормалізована AUC приймає значення в інтервалі, наближеному до $[0, 1]$, що забезпечує порівнюваність результатів між середовищами різної складності. Ця метрика одночасно враховує швидкість збіжності та рівень досягнутої продуктивності.

Слід зазначити, що 95% довірчі інтервали обчислювалися для інтегральної метрики AUC_{norm} , оскільки саме вона використовується як основний критерій порівняння конфігурацій. Для Average Reward та Max Reward інтервали не розраховувалися, оскільки ці метрики застосовуються переважно для якісного аналізу динаміки навчання та інтерпретації структурних змін у системі.

Кількість незалежних запусків для кожної конфігурації визначалася з урахуванням вимоги до статистичної точності оцінки. У більшості випадків виконувалося не менше 30 запусків. У разі підвищеної варіативності результати повторювалися більше ніж 30 разів з метою зменшення відносної похибки оцінки середнього значення до рівня менше 5%. У поодиноких випадках кількість

запусків могла бути меншою за 30, якщо різниця між порівняними методами значна і довірчі інтервали [95, 119] не перетинаються.

Такий підхід забезпечує баланс між статистичною надійністю результатів і обчислювальними витратами, що є важливим з огляду на тривалість навчання глибоких нейронних мереж. Довірчі інтервали обчислювалися з використанням t -розподілу Стюдента [118]. Для великих вибірок ($n \geq 30$) критичне значення наближається до 1.96, що відповідає стандартному нормальному розподілу.

3.2 Архітектура та реалізація експериментальної системи

Особливості реалізації таких систем пов'язані з необхідністю узгодження процесів навчання окремих агентів, організації їх взаємодії, а також інтеграції механізмів передачі знань без порушення стабільності навчання. Це зумовлює необхідність побудови відповідної архітектури системи, яка забезпечує масштабованість, гнучкість та ефективність обчислень.

3.2.1 Архітектура IDQN-агента

Архітектура агента включає такі компоненти:

- нейронну мережу апроксимації Q -функції;
- replay buffer для збереження переходів [33, 95, 97];
- механізм ϵ -жадібної стратегії дослідження;
- процедуру пакетного оновлення параметрів після завершення епізоду;
- механізм заміру продуктивності агента;
- процедуру утворення sharable knowledge.

У більшості експериментів використовується класична одомережна схема DQN без окремої target-мережі. Такий вибір обумовлений необхідністю мінімізувати додаткові фактори впливу та ізолювати ефект механізмів міжагентної передачі знань. У частині експериментів також було протестовано варіант із Double DQN, однак базовою архітектурою залишається одомережна реалізація.

Архітектура мережі залежить від складності середовища. Для CartPole використовується компактна повнозв'язна мережа з двома прихованими шарами невеликої розмірності та нелінійністю ReLU [127]. Така структура є достатньою для апроксимації Q-функції в задачі з низькою розмірністю стану.

Для LunarLander застосовується глибша мережа з більшою кількістю нейронів у прихованих шарах, що обумовлено вищою складністю динаміки середовища та більшим простором станів.

Оновлення параметрів здійснюється за схемою Batch TD(0). Для мінімізації похибки при оновленні Q-функції використовувалася функція втрат Mean Squared Error (MSE). Для оптимізації параметрів мережі використовувався алгоритм Adam [129]. Коефіцієнт дисконтування у рівнянні Беллмана [98] був зафіксований на рівні 0,95, що дозволяло агенту враховувати довгострокові наслідки своїх дій.

Кожен агент зберігає переходи у власному буфері досвіду. Буфер використовується як для самонавчання, так і для формування набору станів при передачі знань іншому агенту.

Для балансування дослідження та експлуатації використовується ϵ -жадібна стратегія [69, 128]. Параметр ϵ зменшується за експоненційною схемою протягом навчання. Це дозволяє на початкових етапах активно досліджувати простір станів, а згодом зосередитися на використанні сформованої політики. Для складніших середовищ (більша глибина чи ширина графу переходів) може знадобитися більше дослідження [99].

У запропонованій реалізації використовується схема Batch TD(0), в якій оновлення параметрів не виконується на кожному кроці взаємодії з середовищем, а здійснюється після завершення епізоду. Такий підхід зменшує частоту оновлень, але підвищує їх стабільність за рахунок використання накопиченого буфера досвіду.

Оскільки навчання не відбувається покроково, зменшення частоти оновлень компенсується виконанням декількох ітерацій оптимізації на одному й тому самому буфері після завершення епізоду. У класичній схемі онлайн TD(0) можна вважати, що на кожен перехід припадає одна ітерація оновлення. У

запропонованій реалізації, навпаки, за один епізод виконується E_{self} епох навчання на вибірках із буфера. Таким чином, E_{self} регулює інтенсивність використання накопиченого досвіду.

Збільшення E_{self} :

- підсилює адаптацію до власного розподілу станів;
- дозволяє глибше мінімізувати функцію втрат на наявних даних;
- компенсує рідшу частоту оновлень у Batch TD(0).

Водночас надмірне значення E_{self} :

- може призводити до перенавчання на обмеженій підмножині буфера;
- знижує чутливість до нових переходів.

Зменшення E_{self} :

- підвищує стохастичність оновлень;
- може уповільнювати формування стабільної політики.

Оптимальне значення E_{self} залежить від:

- складності середовища (розмір простору станів, варіативність винагороди);
- розмірності та глибини нейронної мережі;
- розміру replay buffer;
- частоти обміну знаннями.

Окремо слід зазначити роль розміру міні-пакета (batch size) у процесі навчання як на власному досвіді агента, так і під час передачі знань. Для самонавчання використовується фіксований розмір пакета B_{self} . Менший розмір міні-пакета забезпечує більшу частоту стохастичних оновлень параметрів за однакової кількості зразків та підвищену варіативність оцінки градієнта. Це може призводити до швидшої адаптації політики на ранніх етапах навчання, однак супроводжується більшою дисперсією оновлень і потенційною нестабільністю. Збільшення розміру пакета, навпаки, зменшує дисперсію градієнтної оцінки, що підвищує стабільність навчання [90], але може уповільнювати реакцію мережі на нові зразки досвіду.

3.2.2 Реалізація навчання з механізмом обміну знаннями

Процес навчання багатоагентної системи організовано у вигляді повторюваних епізодів, кожен з яких включає послідовність етапів взаємодії агентів із середовищем та обміну знаннями. У межах одного циклу навчання порядок виконання дій є фіксованим і однаковим для всіх конфігурацій (за винятком спеціальних випадків):

1. Ініціалізація агентів та середовищ.
2. Взаємодія з середовищем. Кожен агент незалежно виконує дії відповідно до ϵ -жадібної стратегії до завершення епізоду. Кількість кроків у різних агентів може відрізнятися. Після завершення епізоду агенти синхронізуються: подальші процедури виконуються лише після того, як усі агенти завершили поточний епізод. Усі переходи зберігаються у відповідному replay buffer. Для кожного агента фіксується сумарна винагорода за епізод, що використовується для можливого ранжування.
3. Формування набору знань для поширення (Sharable Knowledge).
4. Вибір вчителя / вчителів.
5. Навчання на поширеному досвіді.
6. Навчання на власному досвіді

Знання для поширення утворюються до навчання на власному чи поширеному досвіді, для того, щоб використати оцінену політику.

Варіанти вибору вчителя:

- випадковий вибір одного з інших агентів;
- вибір усіх інших агентів (all-to-all);
- вибір за критерієм продуктивності (децентралізований контроль якості).

Агент може не навчатися ні від кого, якщо нема агента, чия політика була більше ефективною.

Навчання на поширеному досвіді включає в себе також обрахунок кількості епох для дистиляції політики E_{share} (інтенсивність передачі знань / навчання на даних вчителя). Також важливим параметром є розмір міні-пакета при дистиляції

B_{share} . Це кількість даних, що агент-вчитель обирає з власного досвіду для формування sharable knowledge.

3.3 Налаштування гіперпараметрів та стратегія підбору

Ефективність навчання в задачах глибокого підкріплення істотно залежить від вибору гіперпараметрів. У запропонованій системі всі параметри поділяються на три групи:

- a) параметри базового IDQN-агента;
 - 1) коефіцієнт дисконтування γ ;
 - 2) розмір буфера досвіду D ;
 - 3) розмір міні-пакета автономного навчання B_{self} ;
 - 4) кількість епох автономного навчання E_{self} ;
 - 5) параметри ϵ -жадібної стратегії;
 - 6) архітектура нейронної мережі;
- b) параметри механізму обміну знаннями;
 - 1) розмір міні-пакета дистиляції B_{share} ;
 - 2) межі кількості епох дистиляції $E_{min}^{share}, E_{max}^{share}$;
 - 3) показник ступеня масштабування $expPower$;
 - 4) стратегія вибору вчителя;
 - 5) кількість вчителів;
 - 6) кількість не вчителів, від яких можна навчатися;
 - 7) кількість агентів в системі;
- c) параметри стратегії дослідження;
 - 1) швидкість зменшення ϵ -жадібної стратегії.

Підбір параметрів здійснювався поетапно. Спочатку налаштовувалася автономна конфігурація без обміну знаннями (No Sharing) до досягнення стабільної збіжності. Після цього до системи інтегрувався базовий механізм передачі знань (один вчитель, одна епоха дистиляції, вибір найбільш продуктивного агента). Лише після стабілізації цієї конфігурації виконувалося

налаштування параметрів динамічної дистиляції. Такий підхід дозволив ізолювати вплив механізму обміну знаннями від впливу базових параметрів агента.

Коефіцієнт дисконтування $\gamma = 0,95$. Значення було зафіксовано для всіх експериментів як таке, що забезпечує стабільну збіжність у досліджуваних середовищах. Оптимізація цього параметра не виконувалася, оскільки він не є предметом дослідження.

Розмір буфера досвіду D . Фіксувався для кожного середовища окремо та обирався таким чином, щоб забезпечити достатнє різноманіття переходів без надмірного накопичення застарілих даних. Подальша оптимізація не проводилася.

Розмір міні-пакета автономного навчання B_{self} . Фіксувався для кожного середовища. Параметр підбирався на початковому етапі з урахуванням компромісу між стабільністю градієнтних оцінок і швидкістю навчання.

Кількість епох автономного навчання E_{self} . Перевага надавалася меншим значенням з метою зменшення часу обчислень та уникнення перенавчання на буфері досвіду. Збільшення E_{self} покращує мінімізацію функції втрат, але підвищує обчислювальні витрати та може зменшити вплив міжагентного обміну.

Параметри ϵ -жадібної стратегії. Швидкість зменшення ϵ підбиралася грубим перебором значень з множини $\{0,95; 0,995; 0,997\}$. Критерієм вибору була максимізація нормалізованої площі під кривою навчання. Для кожного середовища обране значення фіксувалося в усіх подальших експериментах.

Архітектура нейронної мережі. Кількість прихованих шарів та нейронів у кожному шарі підбиралася емпірично з акцентом на мінімально достатню складність моделі [120]. Використання компактніших мереж дозволяло скоротити час навчання та зменшити ризик перенавчання. Для кожного середовища архітектура фіксувалася та не змінювалася між експериментами.

Розмір міні-пакета дистиляції B_{share} . У всіх експериментах приймався рівним B_{self} для забезпечення порівнянної стабільності оновлень. Потенційно цей параметр може впливати на ступінь узагальнення переданих знань, однак у

даному дослідженні основний акцент зроблено на регулюванні інтенсивності передачі через кількість епох.

Межі кількості епох дистиляції $E_{\min}, E_{\max}$. Параметри підбиралися емпірично. У більшості випадків ефективним виявлялося значення $E_{\max}$, що не перевищувало або було близьким до E_{self} , що дозволяло уникнути домінування зовнішнього знання над автономним навчанням.

Показник ступеня масштабування expPower . Тестувалися значення 1 та 2 з метою порівняння лінійної та квадратичної залежності кількості епох від різниці продуктивності.

У роботі розглядаються дві формули визначення кількості епох передачі знань E^{share} , наведені у розділі 2: формула (2.26), що застосовується для задач із невід’ємними значеннями винагород, та узагальнена формула (2.29), яка враховує можливість від’ємних значень винагород.

Для середовища CartPole-v1, у якому сумарна винагорода є невід’ємною ($R_{\min} = 0$), експериментально були використані обидві формули. Проведене порівняння показало відсутність помітної різниці у поведінці алгоритму та підсумкових показниках ефективності, що пояснюється невід’ємним діапазоном винагород у даному середовищі. Наведені нижче значення отримані за використання формули (2.29) в усіх середовищах.

Для середовища LunarLander-v3, де сумарна винагорода може набувати від’ємних значень ($R_{\min} < 0$), використовувалася лише узагальнена формула (2.29), яка забезпечує коректну нормалізацію різниці продуктивності між агентами.

Стратегія вибору вчителя. Порівнювалися різні варіанти: випадковий вибір, all-to-all та вибір на основі показників продуктивності.

Для DDQN online та target мережі синхронізувались повним копіюванням вагів кожні 5 епізодів.

Специфічні параметри експериментів, залежно від середовища вказані в таблиці 3.1.

У межах експериментального дослідження кількість агентів у системі була фіксована на рівні трьох. Вибір саме такої кількості зумовлений компромісом між репрезентативністю багатоагентної взаємодії та обчислювальною складністю експериментів. Збільшення кількості агентів потенційно може підвищити ефективність механізмів передачі знань, оскільки розширює множину можливих джерел знань і підвищує ймовірність наявності більш продуктивного агента-вчителя. Зокрема, у разі обмеження максимальної кількості вчителів для одного агента, більша кількість агентів у системі збільшує ймовірність вибору агента з вищим рівнем сформованості політики, що може позитивно впливати на результативність передачі знань. Разом з тим збільшення кількості агентів призводить до суттєвого зростання обчислювальних витрат, оскільки кожен агент навчається у власному середовищі та має власну нейронну модель.

Таблиця 3.1 – Параметри експериментів залежно від середовища

	CartPole	LunarLander
Приховані шари	ReLU(24), ReLU(12)	ReLU(128), ReLU(64), ReLU(32)
B_{self}	128	64
B_{share}	128	64
ϵ_{decay}	0,95	0,995
Розмір буферу досвіду агентів	10000	15000
R_{min}	0	-300
R_{max}	500	300

3.4 Результати експериментального дослідження

Для представлення результатів експериментів використовується узгоджена система позначень конфігурацій навчання. Кожна конфігурація позначається у вигляді $E(E_{self}, E_{share})_SpecialConditions$, де E_{self} — кількість епох навчання агента на власному досвіді після завершення епізоду, а E_{share} — кількість епох навчання на поширених знаннях (дистиляції політики). Наприклад:

- $E(12, 0)$ — система, у якій агенти навчаються лише на власному досвіді протягом 12 епох без поширення знань;
- $E(3, 5)$ — система, у якій агенти виконують 3 епох автономного навчання та додатково 5 епох навчання на поширених знаннях.
- $E(12, eF_1_5_2)$ — система, у якій агенти виконують 12 епох автономного навчання та динамічно обирає кількість епох навчання на поширених знаннях за запропонованим методом, де $E_{min} = 1$, $E_{max} = 5$, $expPower = 2$.

У деяких експериментах використовуються додаткові модифікатори:

- **RandTeacher** — випадковий вибір вчителя замість вибору найпродуктивнішого агента;
- **Mixed5** — перші 100 епізодів використовується фіксоване значення $E_{share} = 5$, після чого застосовується динамічний механізм визначення кількості епох;
- **AllFromAll** — кожен агент навчається від усіх інших без вибору вчителя;
- **1Teacher&Others** — обирається один найкращий агент як вчитель із динамічним визначенням кількості епох, тоді як від інших агентів навчання виконується з фіксованою інтенсивністю.

Для кількісного порівняння конфігурацій використовується інтегральна метрика AUC_{norm} яка характеризує площу під кривою навчання, нормалізовану відносно можливого діапазону винагороди. Безпосереднє порівняння значень, отриманих у різних середовищах, потребує обережності. Зокрема, середовище LunarLander через свою складність вимагає тривалішої фази дослідження, що

спричиняє повільніший, ніж у CartPole, приріст винагороди на початкових етапах навчання. Це саме по собі зумовлює відмінність в абсолютних значеннях метрики між середовищами. Подальший аналіз результатів проводиться окремо для кожного досліджуваного середовища.

3.4.1 Середовище CartPoleV1

Вибір кількості епох автономного навчання E_{self} здійснювався методом грубого перебору. Метою цього етапу було не знаходження глобально оптимального значення параметра, а визначення режимів із вираженим недонавчанням та режиму, близького до оптимального. Це дозволило дослідити ефективність механізму поширення знань та запропонованих методів контролю якості і інтенсивності передачі знань у різних умовах автономного навчання.

Отримані результати (рис. 3.1) показали, що кількість епох автономного навчання є суттєвим фактором, який впливає на інтегральну ефективність системи. Некоректний вибір цього параметра може призводити як до недонавчання (повільного зростання продуктивності), так і до перенавчання.

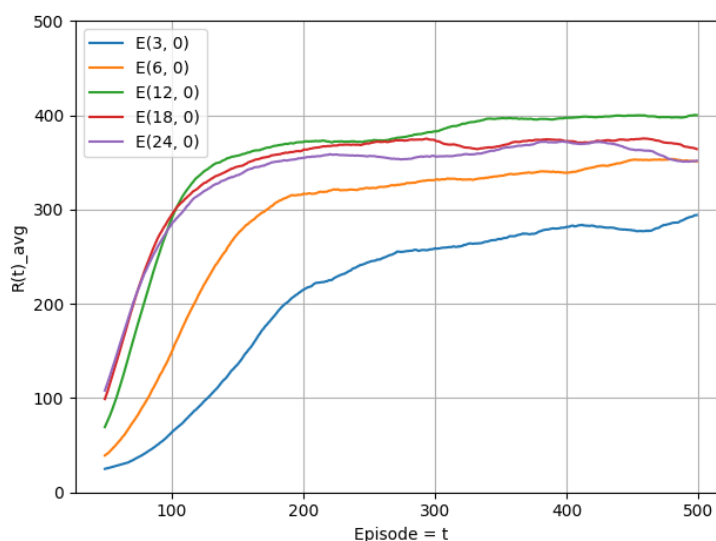


Рисунок 3.1 – МА(50) – Графіки середньої набраної за епізод сумарної винагороди системою з 3-х агентів залежно від E_{self} . CartPoleV1

Окремо було досліджено вплив механізму вибору вчителя на ефективність передачі знань між агентами (рис. 3.2). Для цього порівнювалися конфігурації з випадковим вибором вчителя та з вибором агента з найвищою продуктивністю.

Результати експериментів показують, що навіть випадковий вибір вчителя при невеликій інтенсивності поширення знань (наприклад, $E_{\text{share}} = 1$) здатний покращити інтегральну ефективність системи порівняно з конфігурацією без обміну знаннями. Це пояснюється тим, що агенти отримують додаткову інформацію про поведінку інших учасників системи, що може прискорювати формування ефективних стратегій.

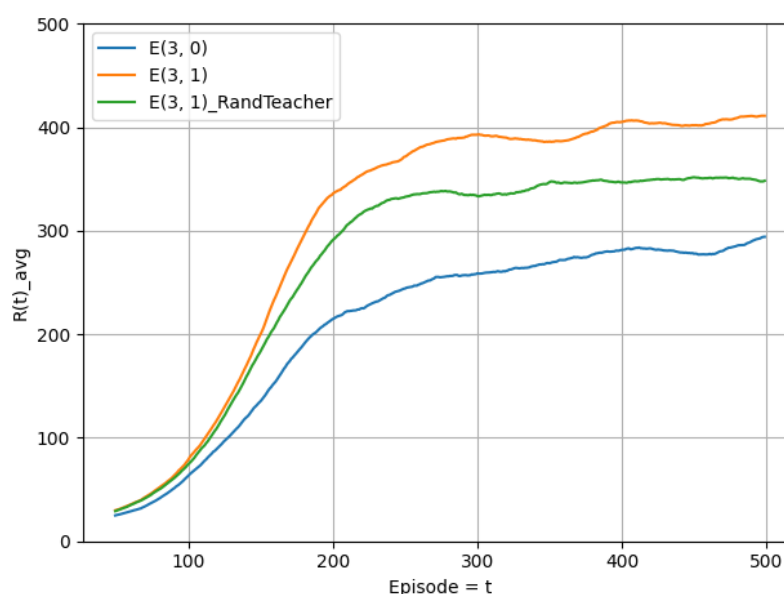


Рисунок 3.2 – МА(50) – Графіки середньої набраної за епізод сумарної винагороди системою з 3-х агентів залежно від механізму вибору вчителя.

CartPoleV1

Водночас використання випадкового вчителя демонструє значно менший позитивний ефект порівняно з конфігурацією, у якій вчителем обирається агент з найвищою продуктивністю. У цьому випадку поширення знань відбувається переважно від агентів, що вже сформували більш ефективні політики, що зменшує ризик негативної передачі знань.

У режимі недостатнього автономного навчання ($E_{\text{self}} = 3$) фіксована кількість епох поширення знань може призводити як до недонавчання, так і до перенавчання (рис. 3.3). Невелика кількість епох дистиляції дозволяє підвищити продуктивність, однак не забезпечує повного використання потенціалу передачі знань. Натомість надмірна інтенсивність поширення може призводити до зниження продуктивності на пізніх етапах навчання.

Фіксована велика кількість епох поширення дозволяє системі швидше досліджувати середовище на початкових етапах навчання, однак у подальшому може призводити до сповільнення зростання продуктивності.

Запропонований механізм динамічного визначення кількості епох дистиляції дозволяє частково поєднати переваги обох підходів: забезпечити швидке навчання на початкових етапах та уникнути перенавчання на пізніх етапах. Водночас на початкових етапах навчання агенти часто виконують випадкові дії внаслідок ϵ -жадібної стратегії дослідження, що зменшує різницю продуктивності між агентами. У таких умовах динамічний механізм може призначати недостатню кількість епох поширення.

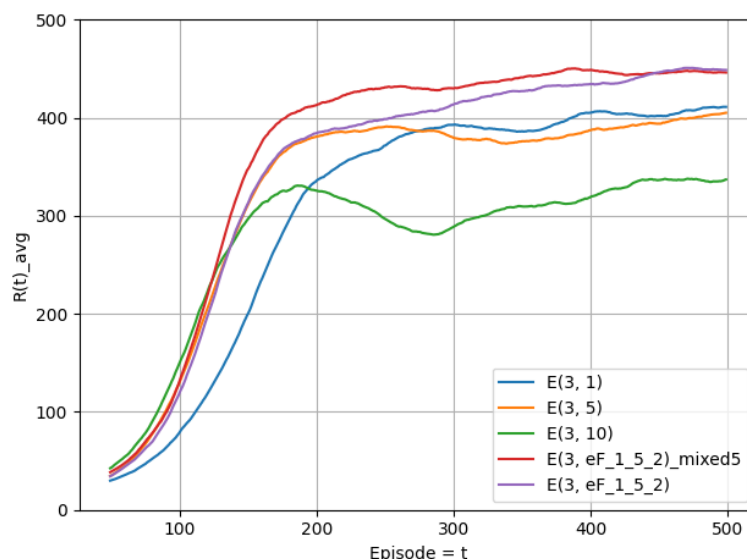


Рисунок 3.3 – МА(50) – Графіки середньої набраної за епізод сумарної винагороди системою з 3-х агентів залежно від E_{share} . CartPoleV1

Для компенсації цього ефекту було протестовано змішану схему

$E(3, eF_1_5_2)_mixed5$, у якій протягом перших епізодів використовується фіксована інтенсивність поширення, після чого застосовується динамічний механізм.

Порівняння з конфігурацією без обміну знаннями показало значний приріст інтегральної продуктивності:

- $AUC_{avg} = 0,714 \pm 0,030$ проти $0,412 \pm 0,033$;
- $\Delta mean = 0,3015$;
- 95% CI: $[0,2575; 0,3454]$;
- $t = 13,751$; $df = 52,6$; $p < 0,001$.

Відносний приріст продуктивності становить приблизно 73%.

Для максимальної продуктивності системи отримано:

- $AUC_{max} = 0,820 \pm 0,016$ проти $0,637 \pm 0,043$;
- $\Delta mean = 0,1826$;
- 95% CI: $[0,1368; 0,2284]$;
- $p < 0,001$.

Відносний приріст продуктивності становить приблизно 29%.

При використанні більшої кількості епох автономного навчання ($E_{self} = 12$) (рис. 3.4) спостерігаються аналогічні закономірності.

Фіксована велика кількість епох поширення ($E_{share} = 7$) демонструє тенденцію до перенавчання, тоді як мала кількість ($E_{share} = 3$) призводить до недонавчання.

Динамічний механізм вибору кількості епох демонструє результати, близькі до конфігурації з оптимальною фіксованою кількістю епох ($E_{share} = 5$). При цьому результати залишаються стабільними незалежно від діапазону можливих значень E_{share} (наприклад $[1-5]$, $[1-10]$ або $[3-7]$). Значення AUC_{avg} змінюється незначно, а графіки навчання практично не відрізняються.

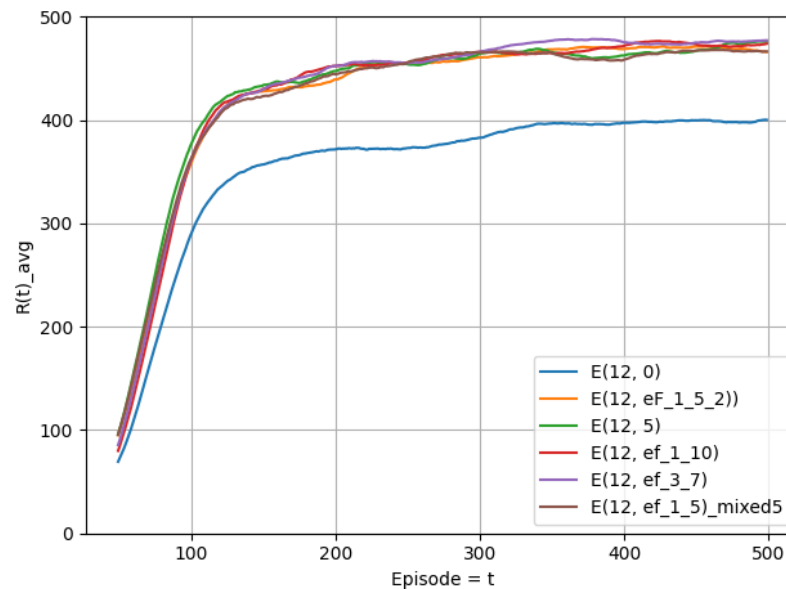


Рисунок 3.4 – МА(50) – Графіки середньої набраної за епізод сумарної винагороди системою з 3-х агентів залежно від E_{share} . CartPoleV1

Це свідчить про те, що динамічний механізм є менш чутливим до вибору гіперпараметрів порівняно з фіксованою кількістю епох.

Порівняння конфігурацій E(12, ef_1_10) та E(12, 0) показало:

- $AUC_{avg} = 0,826 \pm 0,023$ проти $0,687 \pm 0,036$;
- $\Delta mean = 0,1383$;
- 95% CI: [0,0969; 0,1797];
- $p < 0,001$.

Відносний приріст середньої продуктивності становить приблизно 20%.

Для максимальної продуктивності системи отримано:

- $AUC_{max} = 0,902 \pm 0,010$ проти $0,869 \pm 0,026$
- $\Delta mean = 0,0324$;
- 95% CI: [0,0053, 0,0595];
- $p = 0,0204$.

Відносний приріст максимальної продуктивності становить приблизно 4%.

Додатково було протестовано конфігурації, у яких агенти не відкидають вчителів із нижчою продуктивністю. У конфігурації E(12,1)_AllFromAll кожен агент навчається від усіх інших із однаковою інтенсивністю. Інший експеримент

E(12,eF_1_5_2)_Teacher&Others передбачає динамічний вибір одного найкращого вчителя, тоді як від інших агентів навчання відбувається з фіксованою інтенсивністю.

У обох випадках система не продемонструвала стабільного позитивного ефекту. Значення нормалізованої площі під кривою навчання статистично не відрізняється від конфігурації без обміну знаннями.

Схема AllFromAll концептуально відповідає механізму порад у фреймворку KnowSR. Вона прискорює навчання на початкових етапах, однак із часом негативний ефект передачі знань від менш ефективних агентів починає переважати позитивний вплив.

Це може бути пов'язано з відмінностями у постановці задачі. Фреймворк KnowSR орієнтований на нестационарні середовища з взаємозалежними агентами та використовує архітектуру Actor–Critic. У даному дослідженні розглядається повністю декомпозована система на основі IDQN із offline реалізацією Batch TD(0).

Додатково було протестовано схему копіювання вагів найкращого агента після кожного епізоду. Такий підхід також не продемонстрував стабільного підвищення продуктивності.

CartPoleV1 є відносно простим середовищем із обмеженою різноманітністю станів. Багато станів можуть повторюватися протягом усього епізоду, особливо коли агент успішно утримує стрижень близько до вертикального положення.

У таких умовах між агентами виникає відносно невелика якісна різниця в накопиченому досвіді [100]. Тому механізм поширення знань у цьому середовищі переважно прискорює засвоєння вже відомих стратегій, але рідко призводить до передачі принципово нових знань.

Також було протестовано варіант системи, у якій завершення епізодів агентів не синхронізується. У цьому випадку агенти можуть перебувати на різних етапах навчання (наприклад, один агент завершує 10-й епізод, тоді як інший перебуває на початку 8-го).

У такій конфігурації експеримент E(12, eF_1_10_2)_Async продемонстрував приріст інтегральної продуктивності AUC_{avg} лише на 6,3%, тоді як у синхронізованій системі приріст становив приблизно 20%.

Зниження ефективності може пояснюватися тим, що в асинхронному режимі агенти можуть повторно використовувати знання, отримані від інших агентів, навіть якщо вони вже навчалися на цих даних раніше. У результаті відбувається часткове перенавчання на однакових зразках досвіду, що зменшує ефективність передачі знань.

Було також протестовано варіант системи, у якому агент ігнорує поширені знання, на яких він уже навчався раніше. Такий підхід призводив до повільнішого зростання продуктивності на початкових етапах навчання, коли система перебуває у фазі активного дослідження середовища. Проте з часом продуктивність такої системи наближалася до продуктивності системи, що повторно використовує поширені знання.

3.4.2 Середовище LunarLanderV3

Середовище LunarLanderV3 є суттєво складнішим за CartPoleV1, оскільки має більший простір станів, складнішу динаміку середовища та більш тривалий період дослідження. У зв'язку з цим процес навчання агентів є повільнішим, а вплив міжагентного обміну знаннями проявляється інакше.

Кількість епох автономного навчання E_{self} підбиралася методом грубого перебору серед значень [12, 14, 16, 18, 20, 22, 24]. За результатами попередніх експериментів було встановлено, що значення $E_{self} = 18$ є близьким до оптимального для даної конфігурації системи. Менші значення призводять до недостатнього використання накопиченого досвіду, тоді як більші значення викликають перенавчання та погіршення узагальнюючих властивостей моделі.

У базовій конфігурації DQN із однією мережею було протестовано декілька варіантів механізму поширення знань.

Порівняння конфігурації з динамічним вибором кількості епох поширення знань $E(18, eF_1_15_2)_1Teacher\&Others$ із базовою системою без обміну знаннями $E(18, 0)$ показало статистично значущий приріст інтегральної продуктивності (рис. 3.5).

Для середньої інтегральної продуктивності системи отримано:

- $AUC_{avg} = 0,556 \pm 0,006$ проти $0,510 \pm 0,007$;
- $\Delta mean = 0,0469$;
- 95% CI: $[0,0375; 0,0562]$;
- $p < 0,001$.

Відносний приріст становить приблизно 9%.

Для максимальної продуктивності системи отримано:

- $AUC_{max} = 0,670 \pm 0,004$ проти $0,648 \pm 0,006$;
- $\Delta mean = 0,0218$;
- 95% CI: $[0,0148; 0,0287]$;
- $p < 0,001$.

Відносний приріст становить приблизно 3.4%.

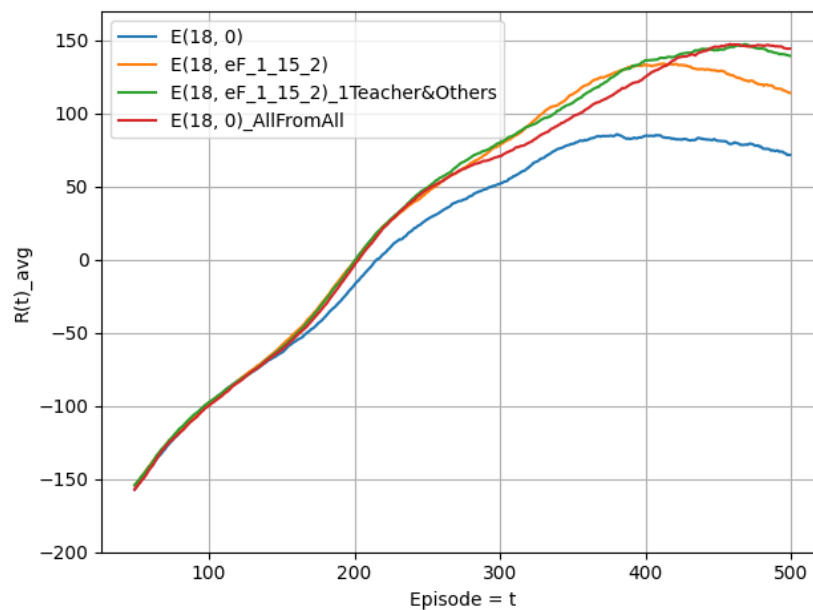


Рисунок 3.5 – МА(50) – Графіки впливу різних механізмів поширення знань на показник середньої набраної за епізод сумарної винагороди системою з 3-х DQN агентів. LunarLanderV3

Отримані результати свідчать про те, що застосування механізму контролю якості джерела знань та динамічного визначення інтенсивності навчання дозволяє підвищити інтегральну ефективність навчання системи агентів у складнішому середовищі.

Було також проведено порівняння конфігурації з динамічним вибором вчителя із схемою навчання від усіх агентів без фільтрації (E(18, 1)_AllFromAll). Різниця інтегральної продуктивності між цими конфігураціями є невеликою:

- $\Delta mean = 0,0056$;
- 80% CI: [0,0001; 0,0111];
- $p = 0,19082$.

Це свідчить про відсутність статистично значущої різниці між даними конфігураціями для середньої інтегральної продуктивності.

При цьому аналіз графіків навчання показує, що запропонований метод демонструє швидше зростання продуктивності на ранніх етапах навчання та більш стабільну поведінку на пізніх етапах.

Додатково було проведено експерименти із застосуванням Double DQN та механізму пріоритетного відтворення досвіду (Prioritized Experience Replay, PER). У цій конфігурації поширення знань відбувалося між online-мережами агентів (рис. 3.6 та 3.7).

Порівняння конфігурації E(18, eF_1_10_2) із базовою системою без обміну знаннями E(18, 0) показало суттєвий приріст інтегральної продуктивності.

Для середньої інтегральної продуктивності:

- $AUC_{avg} = 0,562 \pm 0,010$ проти $0,475 \pm 0,010$;
- $\Delta mean = 0,0864$;
- 95% CI: [0,0722; 0,1006];
- $p < 0,001$.

Відносний приріст становить приблизно 18,3%.

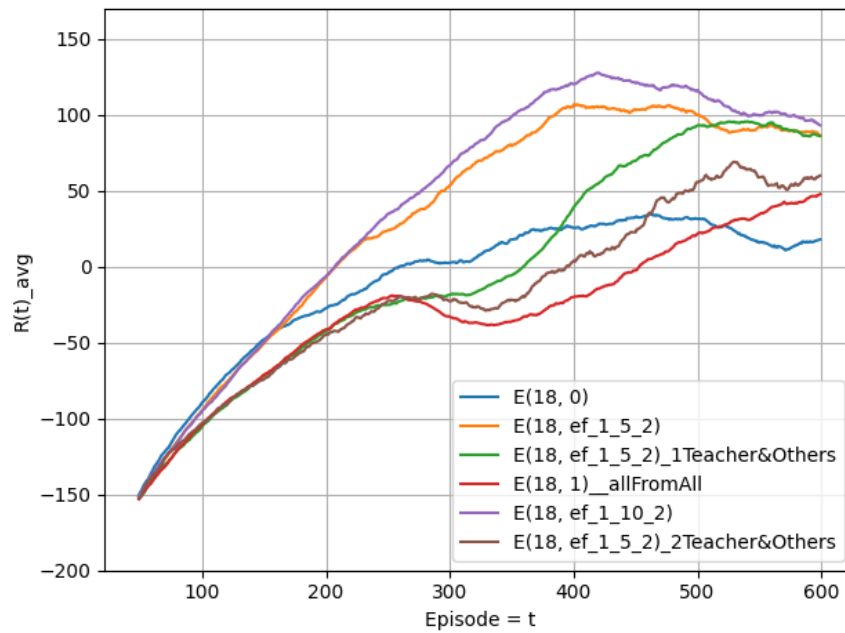


Рисунок 3.6 – MA(50) – Графіки впливу різних механізмів поширення знань на показник середньої набраної за епізод сумарної винагороди системою з 3-х DDQN агентів. LunarLanderV3

Для максимальної продуктивності:

- $AUC_{max} = 0,689 \pm 0,009$ проти $0,626 \pm 0,012$;
- $\Delta mean = 0,0634$;
- 95% CI: $[0,0487; 0,0781]$;
- $p < 0,001$.

Відносний приріст становить приблизно 10,1%.

Варто також зазначити, що за умови недостатнього дослідження $\varepsilon - decay > 0,995$ поширення знань в будь-якій конфігурації не дають позитивного ефекту.

Отримані результати показують, що в поєднанні з більш стабільними алгоритмами навчання, такими як Double DQN із PER, запропонований механізм обміну знаннями демонструє ще більш виражений позитивний ефект.

Значення метрики AUC у середовищі LunarLander є меншими порівняно з CartPole. Це пов'язано з тим, що агент значно довше перебуває у фазі дослідження середовища.

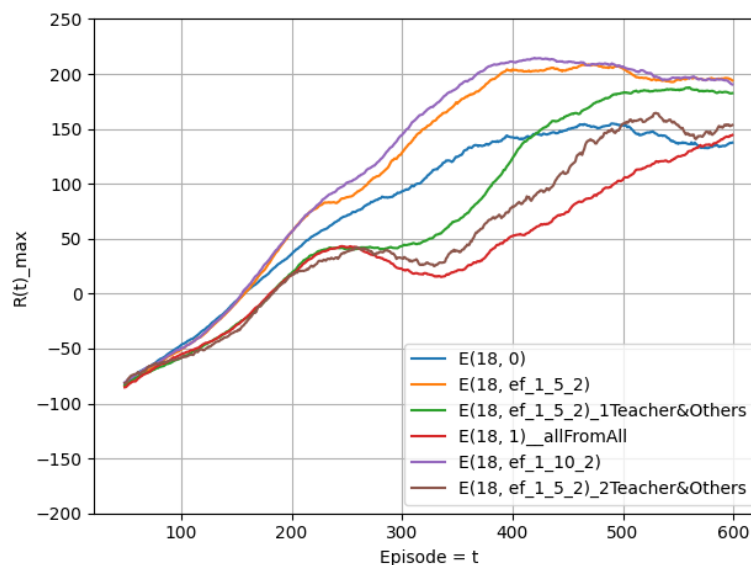


Рисунок 3.7 – МА(50) – Графіки впливу різних механізмів поширення знань на показник максимальної набраної за епізод сумарної винагороди системою з 3-х DDQN агентів. LunarLanderV3

На ранніх етапах навчання агенти часто виконують випадкові дії. Наприклад, приблизно на 150-му епізоді ймовірність випадкової дії становить близько 46,9%, а на 200-му епізоді — приблизно 36,5%. Для CartPole аналогічний показник знижується до менш ніж 1% вже приблизно на сотому епізоді.

Таким чином, у середовищі LunarLander триваліший період дослідження призводить до того, що агенти довше демонструють подібний рівень продуктивності. У таких умовах механізм поширення знань має менший вплив на ранніх етапах навчання, однак його ефект стає помітнішим на пізніших стадіях формування політики.

3.4.3 Аналіз та інтерпретація результатів

За результатами проведених експериментів можна виділити кілька закономірностей впливу міжагентного поширення знань на процес навчання системи.

Перш за все експерименти показали, що поширення знань між агентами здатне частково компенсувати недостатню інтенсивність автономного навчання. У випадках, коли кількість епох навчання на власному досвіді є недостатньою, передача знань дозволяє агентам отримувати додаткову інформацію про поведінку інших агентів і швидше формувати ефективні стратегії. Ефект такого компенсування є відносно невеликим на початкових етапах навчання, коли система перебуває у фазі активного дослідження середовища. У цей період агенти ще не сформували стабільних стратегій і здебільшого виконують випадкові дії, тому якість переданих знань є обмеженою. Отримані результати свідчать, що навіть за обмеженої якості знань міжагентний обмін може прискорювати формування ефективних стратегій, оскільки дозволяє агентам отримувати часткову інформацію про стани середовища, які вони ще не відвідали самостійно.

Другим важливим результатом є те, що навіть за умови достатньо ефективного автономного навчання поширення знань із застосуванням запропонованих механізмів контролю якості та інтенсивності передачі забезпечує додатковий приріст продуктивності системи. Хоча цей приріст є меншим порівняно з випадками недостатнього автономного навчання, він демонструє, що міжагентна передача знань здатний покращувати результати навіть у відносно добре налаштованих системах.

Експериментальні результати також показують, що найкраща продуктивність досягається при поєднанні оптимальної інтенсивності автономного навчання з механізмом поширення знань. У такій конфігурації агенти ефективно використовують власний досвід, а передача знань дозволяє прискорити процес навчання на етапі активного дослідження середовища.

Важливим компонентом ефективності запропонованого підходу є механізм вибору вчителя. Результати експериментів у досліджуваних середовищах показали, що вибір агента з найвищою продуктивністю як джерела знань у більшості випадків демонструє кращі результати порівняно з альтернативними схемами, такими як випадковий вибір вчителя або навчання від усіх агентів (AllFromAll). Разом з тим у деяких експериментах, зокрема у середовищі

LunarLander з використанням DQN, схема навчання від усіх агентів демонструвала результати, близькі до результатів вибору одного вчителя. Проте ці відмінності не були статистично значущими. При цьому використання двох або більше вчителів у більшості випадків призводило до перенавчання, що пов'язано з надмірною інтенсивністю передачі знань.

У таких умовах найкращі результати продемонструвала комбінована схема, у якій обирається один найпродуктивніший агент як основний вчитель із динамічним визначенням інтенсивності навчання, тоді як від інших агентів навчання здійснюється з мінімальною інтенсивністю (одна епоха). Такий підхід дозволяє поєднати переваги передачі знань від кількох джерел із контролем негативної передачі.

Окремо було проаналізовано вплив механізму визначення кількості епох дистиляції. Експерименти показали, що використання фіксованої кількості епох поширення знань у деяких середовищах може призводити до перенавчання на пізніх етапах навчання або до недостатнього використання потенціалу передачі знань. Динамічний механізм визначення кількості епох дозволяє уникнути цих проблем і забезпечує більш стабільні результати. Водночас у деяких випадках фіксована кількість епох може демонструвати результати, близькі до результатів динамічного механізму. Основною перевагою динамічного підходу є менша чутливість до вибору гіперпараметрів. Невеликі зміни значень $E_{\max}$ практично не впливають на результати, тоді як для фіксованої кількості епох навіть незначна зміна параметра може призводити до недонавчання або перенавчання.

Отримані результати також показали, що запропоновані механізми поширення знань покращують не лише середній рівень продуктивності системи, але й максимально досягнуту продуктивність окремими агентами. Це свідчить про те, що запропонований підхід не лише дозволяє менш ефективним агентам швидше наблизитися до більш продуктивних, але й сприяє загальному прискоренню навчання та досягненню кращих стратегій.

Водночас експерименти показали, що на початкових етапах навчання динамічний механізм визначення кількості епох може призначати недостатню

інтенсивність передачі знань. Це пов'язано з тим, що у фазі активного дослідження агенти здебільшого виконують випадкові дії, а різниця між їх продуктивністю є невеликою. У таких умовах корисним може бути використання фіксованої кількості епох поширення протягом перших епізодів навчання з подальшим переходом до динамічного механізму.

Разом із тим результати дослідження мають певні обмеження. По-перше, ефективність поширення знань залежить від рівня дослідження середовища. Якщо дослідження завершується занадто швидко і агенти переходять до експлуатації ще недостатньо вивчених стратегій, поширення знань може не давати позитивного ефекту.

По-друге, запропонований механізм вимагає оцінювання продуктивності агентів, що накладає обмеження на використання повністю онлайн-навчання без епізодичної структури.

По-третє, більшість експериментів проводилася у системах із синхронізацією завершення епізодів між агентами. У випадку асинхронного завершення епізодів ефективність поширення знань зменшувалася, що пов'язано з повторним використанням однакових знань різними агентами.

Крім того, експерименти виконувалися лише для системи з трьох агентів. У більших багатоагентних системах динаміка передачі знань може відрізнятися.

Ще одним обмеженням є використання архітектури Independent DQN, яка не передбачає централізованого критика. У більш складних архітектурах багатоагентного навчання результати можуть відрізнятися.

Також дослідження проводилося лише для середовищ із дискретним простором дій. У задачах із неперервним простором дій поведінка запропонованого методу може змінюватися.

Крім того, експерименти виконувалися для системи з повністю незалежними агентами. У багатоагентних системах із взаємодією агентів або нестационарними середовищами вплив поширення знань може бути іншим.

Нарешті, дослідження проводилося на обмеженій кількості середовищ. Хоча CartPole і LunarLander є стандартними тестовими задачами [29], для повнішої

оцінки ефективності методу необхідні додаткові експерименти на складніших середовищах.

Отримані результати свідчать про перспективність подальших досліджень у напрямку розширення запропонованого підходу на інші архітектури багатоагентного навчання, асинхронні системи навчання та середовища з неперервними просторами станів і дій.

3.5 Висновки до розділу

У розділі розроблено та реалізовано методологію експериментального дослідження міжагентного поширення знань у мультиагентних системах з незалежним навчанням агентів. Проведено серію експериментів у середовищах CartPoleV1 та LunarLanderV3, що дозволило дослідити вплив запропонованих механізмів вибору вчителя та контролю інтенсивності передачі знань на ефективність навчання системи.

У результаті проведеного дослідження встановлено таке.

- Поширення знань між агентами здатне підвищувати інтегральну ефективність навчання багатоагентної системи. Найбільше покращення продуктивності спостерігається у випадках у випадках недостатнього автономного навчання, коли міжагентна передача знань частково компенсує нестачу власного досвіду агентів.
- Поєднання автономного навчання з механізмом передачі знань забезпечує більш швидке формування ефективних стратегій, ніж використання лише одного з цих підходів.
- Використання агента з найвищою продуктивністю як джерела знань є більш ефективним порівняно з випадковим вибором вчителя або навчанням від усіх агентів, оскільки дозволяє зменшити негативну передачу знань.
- Динамічний механізм визначення кількості епох поширення знань забезпечує більш стабільні результати навчання та зменшує чутливість системи до вибору

гіперпараметрів порівняно з використанням фіксованої інтенсивності передачі знань.

- Запропонований підхід підвищує не лише середню інтегральну продуктивність системи, але й максимальну продуктивність окремих агентів, що свідчить про прискорення процесу формування ефективних стратегій.
- Ефективність міжагентного поширення знань залежить від умов навчання системи, зокрема від рівня дослідження середовища, різниці продуктивності між агентами та синхронізації завершення епізодів.

Отримані результати підтверджують доцільність використання механізмів контролю якості джерела знань та адаптивного регулювання інтенсивності передачі знань у багатоагентних системах з незалежним навчанням агентів.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОЇ АРХІТЕКТУРИ СИСТЕМИ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ЗНАННЯМИ У БАГАТОАГЕНТНОМУ НАВЧАННІ З ПІДКРІПЛЕННЯМ

У попередніх розділах було розроблено методи навчання мультиагентної системи та проведено їх експериментальну перевірку. Для підтвердження практичної цінності запропонованого підходу доцільним є розгляд особливостей його програмної реалізації та застосування у прикладних задачах.

У даному розділі розглядаються питання реалізації розроблених методів у вигляді програмного забезпечення, включаючи архітектуру системи, організацію взаємодії компонентів та особливості інтеграції алгоритмів навчання.

Окрему увагу приділено можливостям масштабування, адаптації до різних середовищ та використання у реальних задачах.

4.1 Вимоги до системи реалізації методів

Реалізація запропонованих методів децентралізованого контролю якості та інтенсивності передачі знань у багатоагентних системах навчання з підкріпленням потребує спеціалізованої програмної архітектури, здатної забезпечити взаємодію агентів, управління досвідом навчання та обмін знаннями.

На відміну від багатьох існуючих підходів до багатоагентного навчання, у яких використовується централізований критик або спільна модель середовища, у даному дослідженні розглядається повністю декомпована багатоагентна система, у якій кожен агент навчається у власному середовищі та має власну модель політики. Така постановка відповідає підходу Independent Reinforcement Learning, що широко застосовується у випадках слабо пов'язаних або незалежних середовищ [21].

У досліджуваній системі агенти можуть обмінюватися знаннями з метою підвищення ефективності навчання, однак цей обмін повинен здійснюватися без

централізованого координатора. Це накладає додаткові вимоги на архітектуру програмної реалізації.

Під час проєктування системи були враховані особливості алгоритмів глибокого навчання з підкріпленням, зокрема використання нейронних мереж для апроксимації функцій цінності, механізмів повторного використання досвіду та пакетних методів навчання [27, 101].

На основі аналізу задачі були сформульовані функціональні та нефункціональні вимоги до програмної системи реалізації запропонованих методів.

4.1.1 Функціональні вимоги

Функціональні вимоги визначають можливості системи щодо реалізації процесів навчання агентів та обміну знаннями між ними.

Кожен агент повинен мати можливість накопичувати власний досвід взаємодії із середовищем та використовувати його для оновлення параметрів моделі політики. Для цього система повинна підтримувати збереження переходів середовища та формування навчальних вибірок.

У більшості сучасних алгоритмів глибокого навчання з підкріпленням для цього використовується механізм *experience replay*, що дозволяє зменшити кореляцію між навчальними даними та підвищити стабільність процесу навчання [27].

Система повинна забезпечувати можливість формування підмножини досвіду агента, яка може використовуватися іншими агентами як джерело знань. Такі дані можуть формуватися у вигляді псевдорозмічених вибірок станів разом з оцінками функції цінності, що відповідає підходам дистиляції політики [53].

Система повинна забезпечувати механізм передачі поширюваних знань між агентами. Формат передачі у цьому випадку має містити:

- оцінку продуктивності агента за обраний період оцінювання;
- псевдо розмічений набір даних, сформований на основі досвіду агента.

Залежно від особливостей реалізації можуть також передаватися додаткові метадані, зокрема:

- ідентифікатор агента;
- часова мітка;
- номер епізоду або періоду оцінювання.

Такі дані можуть використовуватися для коректної обробки знань у випадках, коли періоди оцінювання агентів не синхронізовані або коли між агентами існує значна різниця у накопиченому досвіді. Крім того, використання метаданих дозволяє обмежити кількість повторних навчань на одних і тих самих даних.

Запропонований механізм передачі знань не вимагає однакової архітектури моделей глибокого навчання у всіх агентів. Агенти можуть використовувати моделі з різною кількістю прихованих шарів або нейронів. Водночас інтерфейс моделі (вхідний простір станів та вихідний простір дій) повинен залишатися узгодженим. У разі використання різних просторів дій система повинна підтримувати механізм однозначного відображення дій між різними представленнями.

Для реалізації контролю якості передачі знань система повинна забезпечувати можливість оцінювання продуктивності кожного агента за певний період навчання. У запропонованих методах таким показником виступає накопичена винагорода за епізод.

На основі цього показника система повинна підтримувати механізм вибору агентів, які виступають джерелами знань для інших агентів. Такий вибір здійснюється шляхом порівняння продуктивності агентів.

З метою обмеження надмірного впливу зовнішніх знань система повинна підтримувати параметр максимальної кількості вчителів, що можуть використовуватися одним агентом у процесі навчання.

Крім того, система може підтримувати можливість навчання від агентів, які не входять до множини обраних вчителів. У цьому випадку передбачається використання фіксованої кількості епох навчання на їхніх даних. Для цього система повинна підтримувати такі параметри:

- можливість увімкнення або вимкнення навчання від інших агентів;
- кількість епох навчання на даних інших агентів;
- максимальну кількість таких агентів.

Система повинна підтримувати можливість регулювання інтенсивності передачі знань. У запропонованих методах інтенсивність передачі визначається кількістю епох навчання на поширених даних.

Для цього система повинна підтримувати параметри:

- мінімальної кількості епох навчання на поширених знаннях;
- максимальної кількості епох навчання на поширених знаннях.

Система повинна підтримувати механізм визначення завершення епізоду навчання та запуску відповідних процедур:

- навчання на власному досвіді;
- обміну знаннями між агентами;
- навчання на поширених знаннях.

Додатково система може підтримувати механізм синхронізації періодів оцінювання. У цьому випадку агенти, що завершили епізод раніше, можуть очікувати завершення епізодів іншими агентами перед початком процедур навчання та обміну знаннями.

4.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають обмеження та характеристики програмної системи, що впливають на її архітектуру та організацію взаємодії компонентів.

Однією з ключових вимог є підтримка децентралізованої моделі взаємодії агентів, у якій відсутній централізований координатор або глобальний контролер. Такий підхід дозволяє зменшити складність системи, усунути єдину точку відмови та підвищити її стійкість до збоїв окремих компонентів.

Архітектура системи повинна забезпечувати масштабованість щодо кількості агентів [102, 103]. Зокрема, система має підтримувати можливість збільшення

кількості агентів без суттєвої зміни її структури або логіки взаємодії компонентів. У практичних задачах кількість агентів може значно варіюватися, тому важливо забезпечити можливість динамічного додавання нових агентів до системи під час її роботи.

Оскільки обмін знаннями відбувається безпосередньо між агентами, система повинна залишатися працездатною у випадку втрати зв'язку з одним або декількома агентами. У такій ситуації інші агенти повинні мати можливість продовжувати навчання на власному досвіді. Додатково може використовуватися навчання на вже отриманих та збережених наборах поширених знань.

Ще однією важливою вимогою є модульність архітектури. Система повинна складатися з незалежних компонентів, що дозволяє замінювати або розширювати окремі модулі без зміни загальної структури системи. Такий підхід є особливо важливим у дослідницьких системах, де може виникати необхідність експериментального порівняння різних алгоритмів навчання, стратегій обміну знаннями або способів формування навчальних вибірок.

Система повинна забезпечувати можливість інтеграції з існуючими бібліотеками глибокого навчання та навчання з підкріпленням, що широко використовуються у сучасних дослідженнях. Це дозволяє використовувати перевірені реалізації алгоритмів, зменшити складність розробки та підвищити відтворюваність експериментів.

При цьому архітектура системи повинна враховувати, що агенти можуть реалізовуватися як з використанням готових фреймворків побудови агентів, так і у вигляді повністю власних реалізацій. Тому механізми взаємодії агентів повинні базуватися на узгодженому інтерфейсі обміну знаннями, який не залежить від внутрішньої реалізації агента або використовуваної бібліотеки машинного навчання.

4.1.3 Архітектурні обмеження задачі

Окрім загальних вимог до системи, при проєктуванні архітектури необхідно враховувати специфічні обмеження досліджуваної задачі.

У даному дослідженні передбачається, що:

- кожен агент взаємодіє з власним незалежним середовищем;
- агенти не впливають на стан середовища один одного;
- навчання відбувається епізод за епізодом;
- найбільша продуктивність досягається при синхронізованому завершенні епізодів.

Такі обмеження дозволяють розглядати систему як набір незалежних процесів навчання, що взаємодіють лише через механізм передачі знань.

Хоча запропоновані методи (як і дистиляція політики в цілому) і не вимагає агентів мати власні незалежні середовища, а також працювати із незалежними/слабко пов'язаними агентами. Проте, в дослідженні експерименти проводилися саме в таких умовах. А тести проводилися з IDQN агентами, які погано працюють в багатоагентних системах з координацією, кооперацією чи конкуренцією. Це особливість саме IDQN, а не запропонованих методів. Проте, вплив параметрів та кожного методу за різних умов був досліджено лише в цих умовах.

4.2 Архітектурна модель системи реалізації методів

На основі сформульованих функціональних та нефункціональних вимог було розроблено архітектурну модель програмної системи, що забезпечує реалізацію запропонованих методів децентралізованого обміну знаннями між агентами у багатоагентному навчанні з підкріпленням.

Досліджувана система будується на основі підходу Independent Deep Q-Network (IDQN), у якому кожен агент навчається у власному незалежному середовищі та має власну модель політики. У такій постановці глобальна система

може розглядатися як сукупність незалежних процесів навчання, що взаємодіють між собою лише через механізм передачі знань. Подібна організація системи відповідає підходам незалежного навчання агентів, які широко застосовуються у багатоагентних системах зі слабкою або відсутньою взаємодією між середовищами [21].

Основною метою розробленої архітектури є забезпечення:

- децентралізованого обміну знаннями між агентами;
- реалізації механізмів контролю якості передачі знань;
- реалізації механізмів контролю інтенсивності передачі знань;
- можливості масштабування системи за кількістю агентів;
- стійкості до часткової втрати зв'язку між агентами.

Структура агента з підтримкою механізму обміну знаннями включає чотири основні модулі (рис. 4.1):

- AgentPolicy;
- ExperienceManager;
- LearningManager;
- Communicator.

Архітектура системи побудована за модульним принципом, що дозволяє розділити основні функції агента на окремі компоненти. Кожен агент у системі містить набір модулів, відповідальних за прийняття рішень, управління досвідом, координацію навчання та обмін знаннями.

Таке розділення дозволяє забезпечити незалежність компонентів та спростити розширення системи у випадку використання інших алгоритмів навчання з підкріпленням або альтернативних механізмів передачі знань.

Модуль AgentPolicy відповідає за реалізацію політики агента та процес прийняття рішень.

Основними функціями цього модуля є:

- визначення дії агента на основі поточного стану середовища;
- оновлення параметрів моделі політики;
- навчання на власному досвіді;

- навчання на поширених знаннях, отриманих від інших агентів.

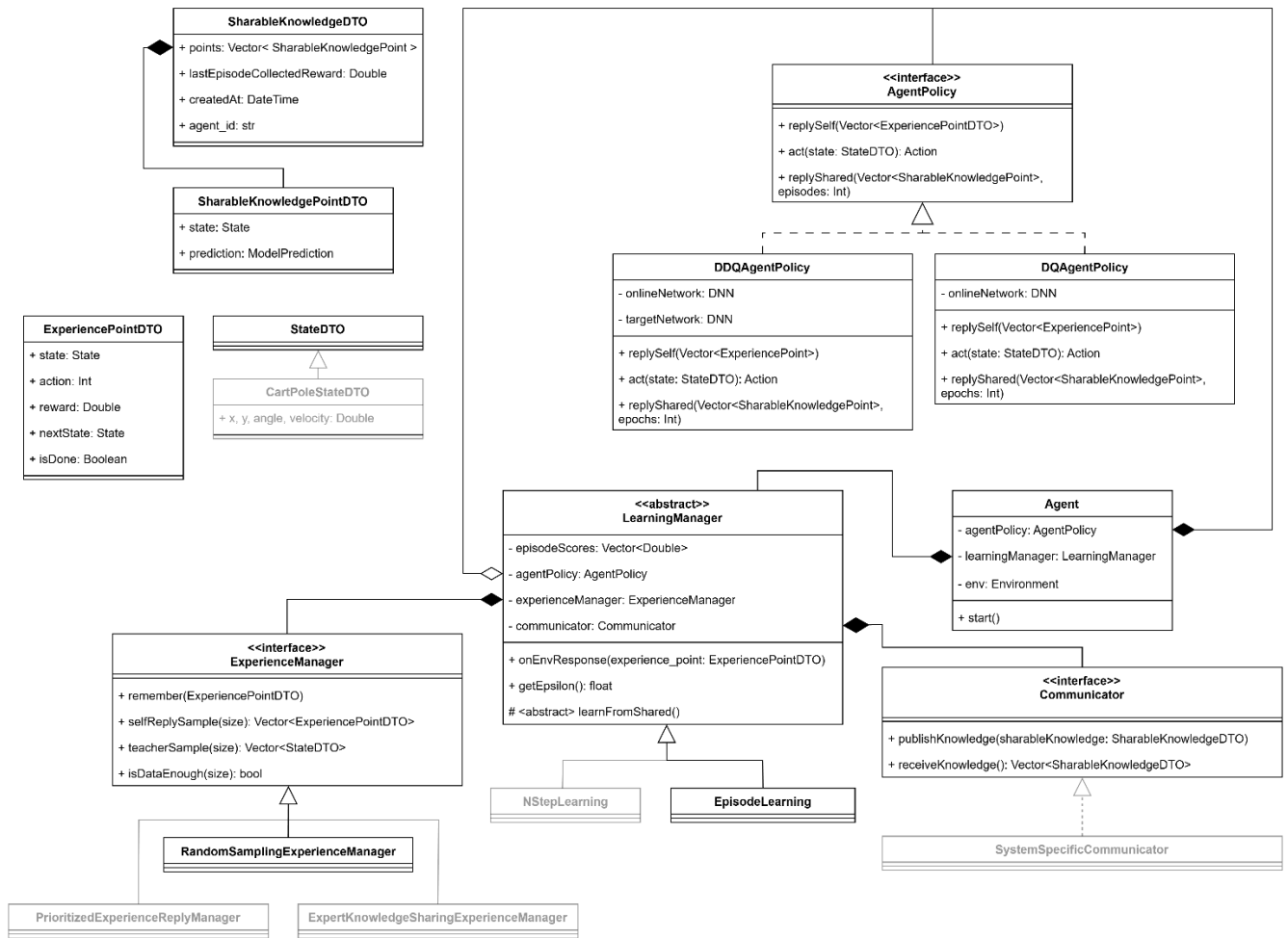


Рисунок 4.1 – Діаграма класів архітектури агента з підтримкою механізму обміну знаннями

У реалізованій системі політика агента представлена нейронною мережею, що апроксимує функцію цінності дій відповідно до підходу Deep Q-Network [27].

Навчання на власному досвіді здійснюється на основі пакетного варіанта алгоритму TD-навчання з використанням вибірок з буфера досвіду. Такий підхід дозволяє стабілізувати процес навчання та зменшити кореляцію між навчальними прикладами.

Крім навчання на власному досвіді, модуль підтримує навчання на поширених знаннях, отриманих від інших агентів, які формуються на основі оцінок функції цінності політики агента-вчителя. Такий підхід відповідає

концепції policy distillation, що використовується для передачі знань між моделями або агентами [53].

Модуль ExperienceManager відповідає за управління досвідом взаємодії агента із середовищем.

Основними функціями цього модуля є:

- збереження переходів середовища;
- формування буфера досвіду;
- вибірка навчальних пакетів;
- формування даних для поширення знань.

Використання буфера досвіду дозволяє повторно використовувати накопичені переходи та підвищує ефективність навчання, що є стандартним підходом у сучасних алгоритмах глибокого навчання з підкріпленням [27, 130].

Модуль також відповідає за формування підмножини станів, що можуть використовуватися як поширювані знання. У базовому варіанті реалізації така підмножина формується шляхом випадкової вибірки станів із буфера досвіду агента [104, 105].

Модуль LearningManager координує процес навчання агента та взаємодію між різними модулями системи.

Основними функціями цього модуля є:

- визначення завершення епізоду навчання;
- обчислення продуктивності агента за період оцінювання;
- запуск навчання на власному досвіді;
- організація обміну знаннями між агентами;
- визначення агентів-вчителів;
- визначення інтенсивності передачі знань.

Оцінка продуктивності агента здійснюється на основі накопиченої винагороди за епізод, яка використовується як метрика якості політики.

На основі цих значень модуль реалізує механізм децентралізованого вибору агентів-вчителів. Процедура вибору включає ранжування агентів за

продуктивністю та відбір підмножини агентів відповідно до обмеження максимальної кількості вчителів.

Після визначення агентів-вчителів модуль обчислює інтенсивність передачі знань, що визначається кількістю епох навчання на поширених даних.

Модуль Communicator відповідає за взаємодію агента з іншими агентами системи.

Основними функціями цього модуля є:

- передача метрик продуктивності агентів;
- передача поширюваних знань;
- отримання даних від інших агентів;
- управління процесом синхронізації обміну знаннями.

Комунікація між агентами може реалізовуватися різними способами, зокрема:

- прямим обміном повідомленнями між агентами [121] (рис. 4.2);
- використанням спільного середовища;
- використанням проміжного сервісу передачі повідомлень.

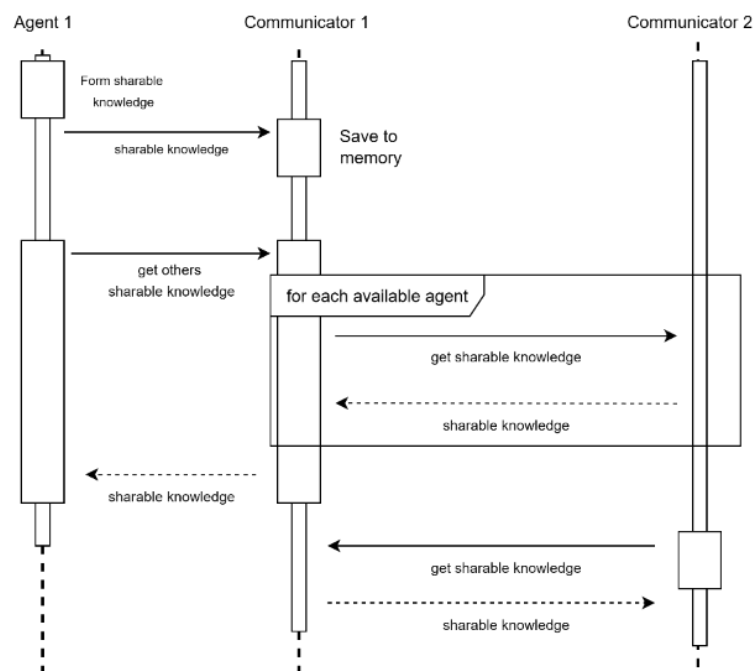


Рисунок 4.2 – Діаграма послідовності прикладу роботи комунікатора при обміні знаннями

Процес роботи агента у системі можна описати наступною послідовністю кроків:

1. Агент взаємодіє із середовищем та накопичує досвід.
2. ExperienceManager зберігає переходи у буфері досвіду.
3. Після завершення епізоду LearningManager обчислює продуктивність агента.
4. Communicator обмінюється метриками продуктивності між агентами.
5. LearningManager визначає агентів-вчителів.
6. Communicator отримує поширювані знання від інших агентів.
7. AgentPolicy виконує навчання на власному досвіді та на поширених знаннях.

Такий цикл повторюється після кожного епізоду навчання (рис. 4.3).

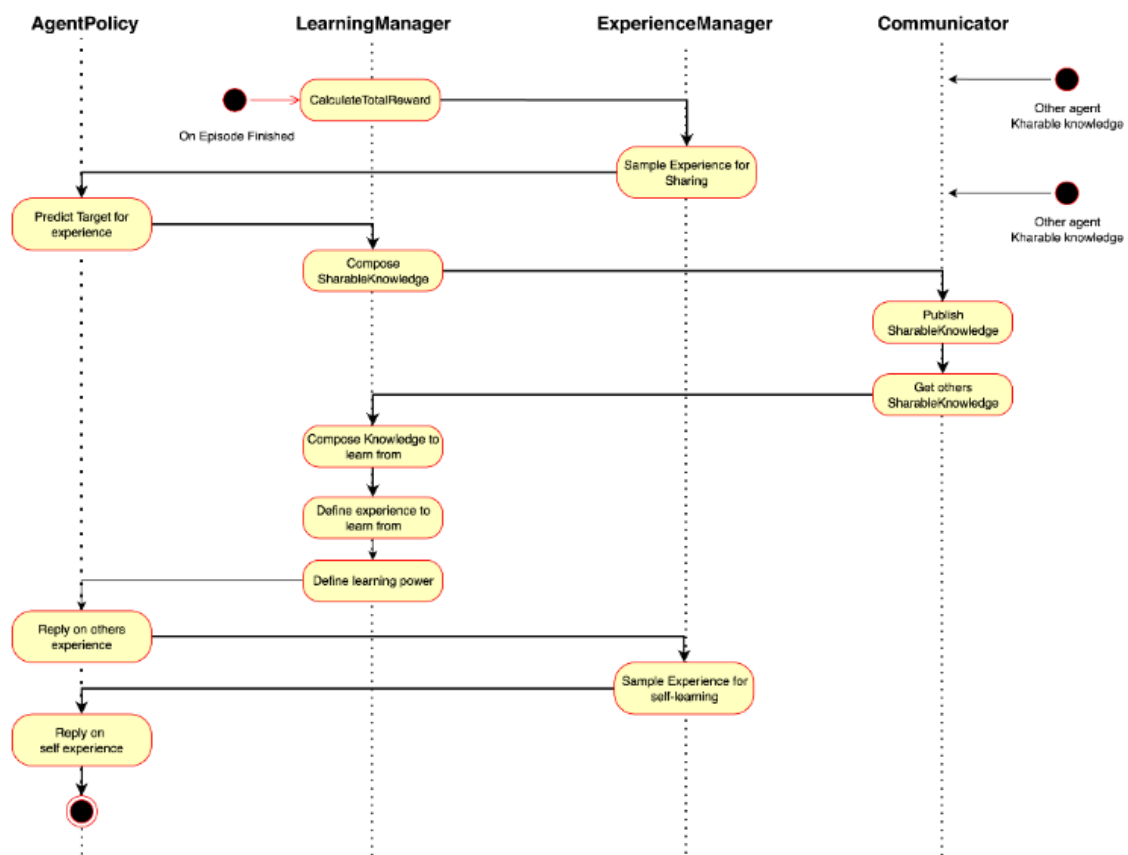


Рисунок 4.3 – Діаграма активності навчання агента з поширенням знань

Важливою властивістю архітектури є те, що вона не передбачає використання централізованого координатора. Це дозволяє зменшити складність системи та підвищити її стійкість до втрати окремих вузлів.

4.3 Модель інтеграції у зовнішні системи

Запропонована архітектура системи децентралізованого обміну знаннями у мультиагентному навчанні з підкріпленням розроблена з урахуванням можливості її інтеграції з існуючими програмними платформами для навчання з підкріпленням та інструментами побудови агентів. У сучасних дослідженнях алгоритми навчання з підкріпленням зазвичай реалізуються не як ізольовані системи, а як надбудови над бібліотеками глибокого навчання, симуляційними середовищами та фреймворками управління агентами.

У зв'язку з цим запропонована система орієнтована на використання адаптаційного підходу, при якому її основні компоненти можуть інтегруватися з зовнішніми програмними інструментами без необхідності повної зміни їхньої архітектури. Такий підхід дозволяє використовувати готові механізми взаємодії агента із середовищем, управління досвідом та організації навчального циклу, концентруючи реалізацію на механізмах передачі знань між агентами.

Інтеграція системи у зовнішні програмні платформи може здійснюватися на трьох основних рівнях:

- рівень середовищ навчання;
- рівень фреймворків навчання з підкріпленням;
- рівень бібліотек глибокого навчання.

Кожен із цих рівнів виконує власну функцію у загальній архітектурі програмної системи.

Для узагальнення можливостей інтеграції запропонованої архітектури з існуючими програмними інструментами у табл. 4.1 наведено відповідність основних компонентів системи та можливих програмних бібліотек.

Таблиця 4.1 – Інтеграція компонентів системи з програмними бібліотеками

Компонент системи	Функції	Можливі бібліотеки / фреймворки
AgentPolicy	Реалізація політики агента, обчислення функції цінності, навчання нейромереж	PyTorch, TensorFlow
ExperienceManager	Збереження переходів, формування replay buffer, вибірка пакетів досвіду	Tianshou, RLlib, AgileRL
LearningManager	Керування процесом навчання, вибір вчителів, визначення інтенсивності передачі знань	Власна реалізація або інтеграція з training loop фреймворків
Communicator	Передача метрик продуктивності та поширених знань між агентами	Власні модулі, RPC, message brokers
Environment Interface	Взаємодія агента із середовищем, отримання станів і винагород	Gymnasium, PettingZoo
Multi-Agent Management	Координація роботи кількох агентів, організація навчального циклу	Tianshou, RLlib, AgileRL
Neural Network Backend	Обчислення градієнтів, оптимізація параметрів моделей	PyTorch, TensorFlow

Використання такого підходу дозволяє поєднати авторські механізми передачі знань з готовими програмними інструментами, що вже реалізують базову інфраструктуру навчання з підкріпленням.

Зокрема, середовище може реалізовуватися з використанням стандартів Gym-подібних API або мультиагентних API, таких як PettingZoo [106, 107], що забезпечують уніфікований інтерфейс взаємодії агентів із середовищем. Управління агентами та навчальним циклом може здійснюватися за допомогою спеціалізованих фреймворків, наприклад Tianshou або AgileRL [108, 109], які підтримують мультиагентні сценарії та автоматизують збір досвіду і навчання моделей.

Завдяки такій інтеграційній моделі запропонована система передбачається можливість використання як надбудова над існуючими програмними платформами навчання з підкріпленням, не вимагаючи повної реалізації агентної інфраструктури з нуля.

Запропонована архітектура підтримує масштабування системи за кількістю агентів без суттєвих змін у її структурі. Кожен агент може функціонувати як окремий програмний модуль або процес.

У залежності від обчислювальних ресурсів агенти можуть розміщуватися:

- у межах одного програмного процесу;
- у вигляді незалежних процесів на одному комп'ютері;
- на різних обчислювальних вузлах.

Завдяки децентралізованій організації системи втрата окремих агентів або тимчасове порушення зв'язку між ними не призводить до зупинки всієї системи. У такому випадку інші агенти можуть продовжувати навчання на власному досвіді або використовувати вже отримані знання.

4.3.1 Інтеграція із середовищами навчання

Середовище навчання визначає правила взаємодії агента з оточенням та забезпечує генерацію станів, винагород і сигналів завершення епізоду. У більшості сучасних досліджень для моделювання таких середовищ використовуються стандартизовані програмні інтерфейси.

Найбільш поширеним інтерфейсом взаємодії агента із середовищем є Gym-подібний API, що визначає базові операції взаємодії:

- ініціалізацію середовища;
- виконання дії;
- отримання нового стану та винагороди;
- визначення завершення епізоду.

Використання такого стандарту дозволяє інтегрувати запропоновану систему з широким спектром симуляційних середовищ, зокрема:

- класичними задачами навчання з підкріпленням;
- фізичними симуляторами;
- ігровими середовищами.

У межах запропонованої архітектури взаємодія із середовищем відбувається на рівні циклу взаємодії агента. Модуль AgentPolicy визначає дію на основі поточного стану, а ExperienceManager зберігає отримані переходи середовища для подальшого навчання.

4.3.2 Інтеграція з фреймворками навчання з підкріпленням

У практичних системах навчання з підкріпленням часто використовуються спеціалізовані фреймворки, що автоматизують значну частину процесу навчання агентів. Такі системи забезпечують:

- організацію циклу взаємодії агента із середовищем;
- збір траєкторій;
- управління буферами досвіду;
- запуск алгоритмів навчання;
- експериментальне керування навчальними процесами.

Використання подібних фреймворків дозволяє значно спростити реалізацію мультиагентних експериментів і підвищити відтворюваність результатів дослідження.

У запропонованій системі такі фреймворки можуть використовуватися як базовий шар для реалізації агентів, тоді як модулі LearningManager та Communicator виступають як розширення, що реалізують механізми передачі знань.

У цьому випадку фреймворк забезпечує:

- створення та ініціалізацію агентів;
- взаємодію із середовищем;
- накопичення досвіду;
- виконання базових алгоритмів навчання.

Натомість запропонована система доповнює цей процес наступними функціями:

- оцінювання продуктивності агентів;
- вибір агентів-вчителів;
- формування поширюваних знань;
- навчання на поширених знаннях.

Такий підхід дозволяє інтегрувати механізми обміну знаннями у вже існуючі системи навчання з підкріпленням без необхідності змінювати їхню внутрішню логіку.

4.3.3 Інтеграція з бібліотеками глибокого навчання

На нижньому рівні програмної реалізації використовуються бібліотеки глибокого навчання, що забезпечують реалізацію нейронних мереж та процедур оптимізації параметрів моделей.

Найбільш поширеними бібліотеками такого типу є:

- PyTorch [110];
- TensorFlow [111].

У межах запропонованої архітектури ці бібліотеки використовуються для реалізації нейронних мереж, що апроксимують функцію цінності або політику

агента. Відповідні моделі інтегруються у модуль AgentPolicy, який виконує роль адаптера між архітектурою системи та конкретною реалізацією моделі.

Завдяки такому підходу внутрішня архітектура нейронної мережі може змінюватися без впливу на інші компоненти системи. Це дозволяє використовувати різні архітектури моделей, оптимізатори або функції втрат, що широко застосовуються у сучасних алгоритмах навчання з підкріпленням [27].

4.4 Аналіз застосовності

Отримані результати та їх аналіз дозволяють перейти до оцінювання практичної застосовності запропонованого підходу. Важливим аспектом є визначення умов, у яких використання розробленої архітектури мультиагентної системи є доцільним, а також оцінка її сильних сторін і потенційних обмежень.

Аналіз застосовності передбачає розгляд як теоретичних характеристик підходу, так і практичних аспектів його реалізації, зокрема масштабованості, гнучкості та ефективності використання обчислювальних ресурсів.

У цьому контексті доцільним є виокремлення основних переваг запропонованої архітектури, що визначають її потенціал для використання у різних класах задач.

4.4.1 Переваги архітектури

Запропонована архітектура програмної реалізації мультиагентної системи навчання з підкріпленням має низку важливих переваг, що зумовлені використанням модульної організації компонентів, децентралізованої моделі взаємодії агентів та адаптації до сучасних підходів розробки систем машинного навчання.

Однією з ключових переваг архітектури є модульність програмної структури. Функціональні компоненти агента, зокрема AgentPolicy, ExperienceManager, LearningManager та Communicator, реалізують окремі аспекти поведінки агента та

взаємодії з іншими компонентами системи. Такий підхід відповідає принципам модульного проєктування програмного забезпечення, які передбачають розділення системи на незалежні компоненти з чітко визначеними інтерфейсами взаємодії.

Завдяки цьому окремі модулі можуть змінюватися або розширюватися без необхідності модифікації інших частин системи. Наприклад, модуль AgentPolicy може використовувати різні алгоритми навчання з підкріпленням (DQN, Double DQN, Actor–Critic тощо), тоді як інші компоненти системи залишаються незмінними.

Ще однією важливою перевагою є масштабованість архітектури. Оскільки кожен агент реалізує власний набір модулів та взаємодіє з іншими агентами через стандартизований інтерфейс обміну знаннями, кількість агентів у системі може змінюватися без необхідності перебудови архітектури. Подібна властивість є критично важливою для сучасних мультиагентних систем, де кількість агентів може значно варіюватися залежно від задачі або доступних обчислювальних ресурсів [107, 114, 123].

Важливою характеристикою запропонованої архітектури є також відмовостійкість. У системі відсутній централізований координатор, що усуває єдину точку відмови. Кожен агент виконує навчання автономно та може продовжувати взаємодію із середовищем навіть у випадку тимчасової втрати зв'язку з іншими агентами. Такий підхід відповідає принципам побудови розподілених інтелектуальних систем, у яких автономність компонентів дозволяє підвищити стійкість системи до збоїв окремих вузлів [112].

Архітектура також забезпечує гнучкість експериментування та розширення системи. Завдяки розділенню функцій управління досвідом, навчанням та комунікацією між агентами можлива незалежна модифікація різних аспектів алгоритму. Наприклад, механізми передачі знань між агентами можуть змінюватися без необхідності змінювати реалізацію політики агента або структуру буфера досвіду. Подібна організація системи є характерною для сучасних дослідницьких платформ у галузі навчання з підкріпленням, де

експериментальна перевірка різних алгоритмів та гіперпараметрів потребує гнучкої архітектури програмної реалізації.

Ще однією перевагою є узгодженість архітектури з сучасними підходами до побудови систем машинного навчання. У запропонованій моделі чітко відокремлюються компоненти, відповідальні за прийняття рішень (AgentPolicy), управління даними (ExperienceManager), координацію навчального процесу (LearningManager) та взаємодію між агентами (Communicator). Подібна декомпозиція відповідає принципам побудови масштабованих систем машинного навчання, у яких окремі функціональні рівні системи реалізуються незалежними програмними модулями.

Окремо слід відзначити підтримку різноманітності моделей агентів. Оскільки передача знань між агентами відбувається у формі навчальних вибірок або оцінок функції цінності, архітектура не накладає жорстких обмежень на внутрішню структуру моделей агентів. Це дозволяє використовувати різні архітектури нейронних мереж або різні алгоритми навчання у різних агентів системи. Така властивість є важливою для багатьох практичних сценаріїв, де агенти можуть використовувати різні стратегії дослідження або різні моделі апроксимації політики [104].

4.4.2 Класи задач, для яких доцільне використання запропонованого підходу

Запропоновані методи децентралізованого контролю якості та інтенсивності передачі знань орієнтовані на мультиагентні системи навчання з підкріпленням, у яких декілька агентів можуть накопичувати досвід паралельно та обмінюватися знаннями для підвищення ефективності навчання.

Експериментальні результати показали, що застосування запропонованого підходу насамперед сприяє підвищенню середньої продуктивності системи агентів. Це досягається за рахунок поширення знань від більш ефективних агентів

до менш ефективних, що дозволяє поступово вирівнювати рівень їхньої продуктивності.

З огляду на це підхід є найбільш доцільним для задач, що характеризуються такими властивостями:

- наявність декількох агентів, які навчаються паралельно;
- подібність або сумісність середовищ, у яких діють агенти;
- можливість передачі досвіду між агентами;
- значна варіативність ефективності агентів у процесі навчання.

Особливо ефективним є застосування запропонованих методів у системах безперервного або довготривалого навчання, у яких агенти постійно взаємодіють із середовищем та поступово вдосконалюють свою політику. У таких системах агенти можуть накопичувати різний досвід взаємодії із середовищем, що створює передумови для ефективної передачі знань [114].

Ще одним можливим сценарієм використання є паралельне навчання декількох агентів для розв'язання однієї задачі. У цьому випадку агенти навчаються незалежно, але періодично обмінюються знаннями. Такий підхід дозволяє зменшити вплив випадкових факторів, зокрема випадкової ініціалізації параметрів моделі або стохастичності середовища. Крім того, різні агенти можуть досліджувати різні області простору станів, що сприяє швидшому накопиченню корисного досвіду.

4.4.3 Особливості використання запропонованого підходу

Запропонований механізм передачі знань передбачає навчання агентів на поширених вибірках даних, сформованих на основі досвіду інших агентів. На відміну від підходів, що передбачають пряме копіювання параметрів моделей, такий механізм дозволяє зберігати різноманітність стратегій агентів і зменшує ризик передчасної конвергенції системи.

Експериментальні результати показали, що пряме копіювання параметрів найуспішнішого агента може призводити до погіршення результатів у порівнянні

з використанням запропонованих механізмів передачі знань. Це пояснюється тим, що копіювання політики може поширювати не лише корисні знання, але й локальні особливості або помилки моделі.

Додатковою перевагою запропонованого підходу є відносна робастність до вибору гіперпараметрів навчання. Навіть у випадках, коли обсяг навчання на власному досвіді є недостатнім, агенти можуть частково компенсувати цей недолік шляхом використання знань інших агентів.

Практичні експерименти показали, що ефективною евристикою для налаштування параметрів навчання є співвідношення: $E_{\max}^{share} < E_{\text{self}}$. Таке співвідношення дозволяє підтримувати баланс між індивідуальним навчанням агентів та використанням колективного досвіду. До того ж, в деяких випадках на стадії активного дослідження з жадібною стратегією має сенс встановити константну кількість епох, адже різниця між продуктивністю агентів є невеликою, через високий шанс випадкового вибору дії.

4.4.4 Обмеження застосування

Незважаючи на переваги запропонованого підходу, його застосування має ряд обмежень.

По-перше, експериментальна перевірка методів виконувалася у межах архітектури Independent Deep Q-Network (IDQN), у якій кожен агент навчається незалежно від інших. У такій постановці агенти не впливають на середовище один одного, а загальна система може розглядатися як набір незалежних процесів навчання.

По-друге, дослідження проводилося на задачах, що можуть бути представлені як декомпозовані марковські процеси прийняття рішень, у яких кожен агент взаємодіє з власним незалежним середовищем. У більш складних багатоагентних системах, де агенти впливають один на одного, середовище стає нестационарним. У таких умовах незалежне навчання агентів може призводити до нестабільності або погіршення результатів.

По-третє, ефективність обміну знаннями залежить від ступеня дослідження середовища агентами. Якщо агенти мають недостатній обсяг досвіду взаємодії із середовищем, поширювані знання можуть бути недостатньо інформативними.

По-четверте, запропоновані методи передбачають періодичне оцінювання продуктивності агентів, що вимагає тимчасової фіксації (заморожування) політики на час збору досвіду. У зв'язку з цим оновлення політики відбувається рідше, ніж у випадку повністю онлайн-навчання.

Така організація навчального процесу є більш природною для алгоритмів, що використовують епізодичну або пакетну структуру навчання, зокрема Monte Carlo методів, n-step TD або пакетних варіантів TD-навчання [124].

По-п'яте, ефективність передачі знань передбачає сумісність просторів станів та дій агентів або можливість їх однозначного відображення.

4.4.5 Потенціал подальшого розвитку

Подальший розвиток запропонованого підходу може здійснюватися у декількох напрямках.

По-перше, перспективним є розширення методів на більш складні мультиагентні системи, у яких агенти взаємодіють у спільному середовищі. У таких системах передача знань може поєднуватися з кооперативним або конкурентним навчанням.

По-друге, можливим є використання більш складних критеріїв оцінювання якості знань, що передаються між агентами. Зокрема, оцінювання може враховувати не лише накопичену винагороду, але й інші характеристики поведінки агентів.

По-третє, перспективним напрямом є розширення механізмів вибору агентів-вчителів, які враховують не лише поточну ефективність агентів, але й різноманітність накопиченого досвіду.

Крім того, подальші дослідження можуть бути спрямовані на інтеграцію запропонованих методів із іншими підходами трансферного навчання у навчанні з підкріпленням.

4.5 Платформа для проведення експериментів

Проведення експериментальних досліджень у задачах навчання з підкріпленням пов'язане з необхідністю виконання великої кількості обчислювальних експериментів із різними наборами параметрів. Зокрема, при дослідженні запропонованих методів передачі знань між агентами виникла потреба у систематичному тестуванні великої кількості конфігурацій алгоритмів, параметрів навчання та варіантів реалізації моделей.

Однією з особливостей таких досліджень є значна кількість параметрів, що впливають на результати навчання. До них належать параметри алгоритмів навчання, параметри механізмів передачі знань, характеристики середовищ, а також параметри нейронних мереж. Це створює необхідність проведення великої кількості експериментів для аналізу впливу окремих факторів на ефективність навчання.

Крім того, у процесі дослідження може виникати потреба у тестуванні різних програмних реалізацій алгоритмів. Наприклад, зміна типу алгоритму навчання (DQN, DDQN), використання різних середовищ або модифікація архітектури моделі може вимагати змін у коді, які не можуть бути описані лише параметрами конфігурації.

Ще однією важливою особливістю експериментів у навчанні з підкріпленням є значна залежність результатів від випадкових факторів, зокрема від початкової ініціалізації параметрів моделей або особливостей дослідження середовища. Використання фіксованих значень випадкових генераторів (seed) сприяє відтворюваності експериментів, однак не гарантує отримання подібних результатів за інших початкових умов. У зв'язку з цим для коректного оцінювання

впливу параметрів алгоритму необхідно виконувати декілька повторів одного й того самого експерименту.

У задачах навчання з підкріпленням окремий експеримент може виконуватися від декількох десятків хвилин до десятків годин залежно від складності середовища, параметрів навчання та доступної обчислювальної інфраструктури. Крім того, експерименти можуть вимагати значних обчислювальних ресурсів, включаючи оперативну пам'ять та графічні процесори.

За наявності великої кількості експериментів ручне керування їх запуском стає неефективним. Тому виникає потреба у створенні програмної системи, яка забезпечує:

- автоматизоване планування запуску експериментів;
- контроль використання обчислювальних ресурсів;
- автоматичне збереження результатів;
- зручне групування та аналіз експериментів;
- можливість повторного запуску експериментів з однаковими параметрами.

Додатковою вимогою до системи є забезпечення відтворюваності експериментів, включаючи збереження параметрів запуску, програмної реалізації та отриманих результатів. Крім того, система, у якій виконується велика кількість експериментів з різними параметрами, повинна забезпечувати можливість швидкої інтерпретації змісту кожного експерименту без необхідності детального аналізу його реалізації або параметрів.

4.5.1 Архітектура системи запуску експериментів

Для автоматизації проведення експериментів була розроблена спеціалізована програмна платформа, що забезпечує керування запуском експериментів, моніторинг їх виконання та збереження результатів.

Архітектура платформи базується на концепції задач експерименту (Task). Кожна задача представляє окрему програмну реалізацію експерименту та містить усі необхідні залежності для її виконання.

Кодова реалізація задачі зберігається у каталозі tasks у вигляді окремої папки з унікальною назвою. Кожна така папка містить файл Dockerfile, який визначає середовище виконання експерименту, включаючи необхідні програмні бібліотеки та залежності [125].

Використання контейнеризації дозволяє забезпечити ізольоване виконання експериментів та спрощує перенесення системи між різними обчислювальними платформами.

Параметри експерименту передаються у програмний код через змінні середовища. Такий підхід дозволяє використовувати одну й ту саму реалізацію задачі для проведення великої кількості експериментів з різними конфігураціями.

У межах розробленої системи окремий запуск експерименту називається Run. Один експеримент може містити декілька запусків, які відповідають повторним виконанням одного і того самого експерименту з однаковими параметрами.

Кожен запуск експерименту виконується у вигляді окремого Docker-контейнера. Перед запуском експерименту для відповідної задачі автоматично створюється Docker-образ, який надалі використовується для всіх запусків цієї задачі.

Кожному контейнеру призначається окрема директорія у каталозі outputs, що використовується для збереження результатів експерименту. Це забезпечує ізоляцію результатів різних запусків та спрощує їх подальший аналіз. Це дозволяє зберігати результати експерименту незалежно від життєвого циклу контейнера.

Контейнери можуть запускатися як на центральному процесорі, так і з використанням графічного процесора. Це дозволяє ефективно використовувати доступні обчислювальні ресурси.

4.5.2 Реалізація системи запуску експериментів

Система керування експериментами реалізована у вигляді консольної утиліти, що підтримує декілька основних команд.

Команда `serve` запускає сервіс керування експериментами, який періодично перевіряє стан виконання запущених задач, оцінює доступні обчислювальні ресурси та запускає нові експерименти відповідно до запланованої черги.

Команда `status` відображає інформацію про поточний стан системи, включаючи список виконаних, запущених та запланованих експериментів.

Команда `submit` використовується для додавання нового експерименту до системи. У якості аргументу передається файл конфігурації у форматі JSON, який містить опис експерименту, включаючи:

- унікальний ідентифікатор експерименту;
- текстовий опис;
- назву задачі;
- кількість запусків;
- набір параметрів експерименту.

Уся інформація про експерименти зберігається у локальній базі даних, що дозволяє відстежувати історію запусків та забезпечує зручний доступ до результатів.

Однією з ключових вимог до експериментальної платформи є забезпечення відтворюваності результатів дослідження [126]. Для цього система автоматично зберігає інформацію про параметри експерименту, використану реалізацію задачі та результати виконання.

Завдяки використанню контейнеризації кожен експеримент виконується у чітко визначеному програмному середовищі. Це дозволяє повторно запускати експерименти навіть після зміни програмної інфраструктури.

Крім того, система зберігає історію виконаних експериментів, що дозволяє виконувати повторний аналіз результатів та відтворювати попередні експериментальні конфігурації.

Розроблена платформа дозволила значно спростити проведення експериментальних досліджень. Зокрема, вона забезпечила можливість автоматичного запуску великої кількості експериментів, ізоляції їх виконання та систематичного збереження результатів.

Використання контейнерів дозволило легко переносити експериментальне середовище між різними обчислювальними платформами. Для перенесення експериментів достатньо скопіювати каталог із задачами, конфігураціями та результатами.

Крім того, система дозволила гнучко змінювати кількість запусків експериментів, тестувати різні реалізації алгоритмів та використовувати різні програмні бібліотеки і залежності.

4.6 Висновок до розділу

У розділі розглянуто питання програмної реалізації запропонованих методів децентралізованого обміну знаннями між агентами та створення програмного середовища для проведення експериментальних досліджень.

Сформульовано функціональні та нефункціональні вимоги до системи реалізації методів. Визначено основні можливості системи, необхідні для підтримки процесів навчання агентів, накопичення досвіду, передачі поширюваних знань та керування інтенсивністю їх використання. Окрему увагу приділено вимогам до модульності архітектури, масштабованості, відмовостійкості та можливості інтеграції з існуючими програмними бібліотеками та фреймворками навчання з підкріпленням.

На основі сформульованих вимог запропоновано архітектурну модель системи реалізації методів обміну знаннями між агентами. Архітектура базується на модульному підході та включає компоненти управління політикою агента, керування процесом навчання, зберігання досвіду та комунікації між агентами. Така структура забезпечує гнучкість програмної реалізації, спрощує модифікацію

окремих компонентів системи та дозволяє інтегрувати різні алгоритми навчання з підкріпленням.

Розглянуто можливості інтеграції запропонованої системи з існуючими програмними бібліотеками та фреймворками розробки мультиагентних систем. Показано, що запропонована архітектура може використовуватися у поєднанні з сучасними інструментами розробки систем навчання з підкріпленням та не накладає жорстких обмежень на реалізацію моделей агентів.

Проведено аналіз застосовності запропонованих методів та визначено класи задач, у яких їх використання є доцільним. Показано, що підхід є найбільш ефективним у системах з незалежними або слабко пов'язаними агентами, які навчаються паралельно та можуть обмінюватися накопиченим досвідом. Також визначено основні обмеження застосування методів, зокрема залежність від ступеня дослідження середовища агентами, сумісності просторів станів і дій та особливостей організації процесу навчання.

Крім того, розроблено програмну платформу для автоматизації проведення експериментальних досліджень. Платформа забезпечує керування запуском великої кількості експериментів, контроль використання обчислювальних ресурсів, ізоляцію експериментальних середовищ та автоматичне збереження результатів. Використання контейнеризації дозволило забезпечити відтворюваність експериментів та спростити перенесення експериментального середовища між різними обчислювальними платформами.

У результаті виконаних досліджень у межах розділу отримано такі результати.

- Вперше розроблено архітектуру програмного забезпечення для реалізації механізму децентралізованого обміну знаннями між агентами у мультиагентних системах навчання з підкріпленням, що відрізняється підтримкою механізмів контролю якості та інтенсивності передачі знань, які реалізуються без використання централізованого координатора, що дозволяє гнучко керувати процесом поширення знань між агентами та

забезпечує адаптивне використання колективного досвіду під час навчання.

- Удосконалено модель програмної організації мультиагентної системи навчання з підкріпленням шляхом виділення функціональних модулів управління політикою агента, керування процесом навчання, зберігання досвіду та міжагентної комунікації, що забезпечує модульність програмної архітектури, спрощує модифікацію окремих компонентів системи та дозволяє інтегрувати різні алгоритми навчання і механізми передачі знань.
- Розроблено програмну платформу для автоматизації проведення експериментальних досліджень у задачах навчання з підкріпленням, що забезпечує керування великими наборами експериментів, контроль використання обчислювальних ресурсів та відтворюваність результатів завдяки використанню контейнеризованого середовища виконання, що дозволяє значно спростити організацію експериментальних досліджень і підвищити надійність отриманих результатів.

ВИСНОВКИ

У дисертаційній роботі розв'язано актуальну науково-прикладну задачу підвищення продуктивності багатоагентних систем навчання з підкріпленням у стаціонарних середовищах зі слабо пов'язаними агентами шляхом розроблення методів та програмних засобів децентралізованого обміну знаннями між агентами. Запропонований багатоагентний підхід забезпечує адаптивний контроль якості та інтенсивності передачі знань між агентами, що дозволяє зменшити негативну передачу знань і підвищити продуктивність процесу навчання.

Актуальність дослідження зумовлена зростанням складності сучасних розподілених інформаційних систем, у яких значну роль відіграють багатоагентні підходи до моделювання та оптимізації процесів прийняття рішень. Багатоагентні системи широко застосовуються у робототехніці, системах автономного керування, логістиці, комп'ютерних іграх, системах підтримки прийняття рішень та інших галузях. Одним із найбільш ефективних механізмів адаптації агентів у таких системах є навчання з підкріпленням, зокрема його багатоагентний варіант. Проте продуктивність навчання багатоагентних систем часто обмежується необхідністю великої кількості взаємодій із середовищем, дублюванням досвіду між агентами та складністю організації ефективної передачі знань.

У процесі дослідження встановлено, що існуючі методи обміну знаннями між агентами у багатоагентному навчанні з підкріпленням часто базуються на централізованих архітектурах або не забезпечують достатнього контролю якості переданих знань. Це може призводити до негативної передачі знань, зниження ефективності навчання або нестабільності поведінки агентів. У зв'язку з цим виникає необхідність розроблення децентралізованих механізмів обміну знаннями, здатних забезпечувати адаптивний контроль передачі знань у системах зі слабо пов'язаними агентами.

У межах дисертаційної роботи проведено аналіз сучасного стану досліджень у галузі багатоагентних систем та навчання з підкріпленням, зокрема підходів до

передачі знань між агентами, таких як спільні буфери досвіду, дистиляція політики, централізоване навчання з децентралізованим виконанням та інші. За результатами аналізу встановлено, що більшість існуючих методів або орієнтована на централізовані архітектури, або не враховує необхідності контролю якості та інтенсивності передачі знань у процесі навчання.

Для розв'язання поставленої наукової задачі у дисертації запропоновано методи та програмні засоби децентралізованого обміну знаннями між агентами у системах багатоагентного навчання з підкріпленням. Основною ідеєю запропонованого підходу є використання механізмів вибору агента-вчителя та адаптивного регулювання інтенсивності передачі знань на основі оцінки поточної продуктивності агентів.

У роботі отримано такі основні результати.

1. Набуто подальшого розвитку метод децентралізованого вибору вчителя для передачі знань, що дозволяє контролювати якість поширюваних знань уникаючи негативного переносу.
2. Вперше розроблено метод динамічного децентралізованого контролю інтенсивності передачі знань між агентами, що дозволяє ефективно використовувати набуті системою знання та протидіяти перенавчанню
3. Вперше розроблено архітектуру програмного забезпечення для реалізації механізму обміну знаннями в системі багатоагентного навчання, характерною особливістю якої є можливість децентралізованого обміну знань з застосуванням механізмів контролю якості та інтенсивності передачі.
4. Практичне використання результатів роботи, розроблених методів та програмних засобів дозволило підвищити продуктивність багатоагентної системи за метрикою нормалізованої площі під кривою навчання (normalized AUC) сумарної винагороди за епізод на 4–20% для середньої інтегральної продуктивності системи та на 3,4–18,3% для максимальної продуктивності окремого агента залежно від конфігурації механізму обміну знаннями та режиму автономного навчання. Експериментальні

дослідження проведено у тестових середовищах CartPole-v1 та LunarLander-v3 бібліотеки Gym/Gymnasium. Запропонований механізм обміну знаннями продемонстрував результативність як для алгоритмів Deep Q-Network (DQN), так і Double Deep Q-Network (DDQN).

5. Мета досліджень, яка полягала в підвищенні продуктивності багатоагентної системи слабко пов'язаних агентів у стаціонарному середовищі за допомогою механізму поширення знань між агентами на базі децентралізованого підходу, вирішена повністю. Наукові результати дослідження є значним внеском у розвиток теоретичних і прикладних основ багатоагентного навчання з підкріпленням, зокрема у частині розроблення методів контролю якості та інтенсивності передачі знань між агентами, а також створення програмних засобів децентралізованого обміну знаннями, що забезпечують підвищення ефективності навчання у системах зі слабко пов'язаними агентами в стаціонарних середовищах. Отримані результати можуть бути використані під час розроблення інтелектуальних розподілених систем, зокрема у робототехнічних системах, системах автономного керування, інтелектуальних транспортних системах та інших задачах, де застосовуються багатоагентні підходи та методи навчання з підкріпленням.
6. В якості можливих напрямків продовження дослідження можна відмітити розширення запропонованих методів децентралізованого обміну знаннями на інші алгоритми навчання з підкріпленням, зокрема методи на основі політик та актор-критик підходів, а також їх застосування у різних типах багатоагентних середовищ, включаючи кооперативні, конкурентні та змішані сценарії. Перспективним є дослідження альтернативних підходів до оцінювання продуктивності агентів, зокрема на основі безперервної (rolling) оцінки показників навчання без фіксації політики на період оцінювання. Окремим напрямом подальших досліджень є розроблення механізмів якісної оцінки відмінностей у досвіді агентів та їх використання для підвищення ефективності передачі знань і зменшення

ризикі негативної передачі в багатоагентних системах навчання з підкріпленням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kubera Y., Mathieu P., Picault S.. Everything can be Agent!. 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), May 2010, Toronto, Canada. p.1547-1548
2. Wooldridge M. An Introduction to MultiAgent Systems. John Wiley & Sons, 2002. 366 с.
3. Leitão, Paulo; Karnouskos, Stamatis (March 26, 2015). Industrial agents: emerging applications of software agents in industry. Leitão, Paulo, Karnouskos, Stamatis. Amsterdam, Netherlands. ISBN 978-0128003411. OCLC 905853947.
4. Volkmer, G. G. A., & Alsabah, N. (2022). Group cohesion in Multi-Agent scenarios as an emergent behavior. arXiv (Cornell University). <https://doi.org/10.48550/arxiv.2211.02089>
5. M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo, «Swarm robotics: A review from the swarm engineering perspective», Swarm Intell., vol. 7, no. 1, pp. 1-41, 2013.
6. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. arXiv (Cornell University). <https://doi.org/10.48550/arxiv.1312.5602>
7. The Arcade Learning Environment: An Evaluation Platform for General Agents, 2012–2013, ID 1207.4708.
8. StarCraft II: A New Challenge for Reinforcement Learning, 2017, ID 1708.04782.
9. The StarCraft Multi-Agent Challenge, 2019, ID 1902.04043.
10. OpenSpiel: A Framework for Reinforcement Learning in Games, 2019, ID 1908.09453
11. Tichý P., Staron R.J. Multi-agent technology for fault tolerant and flexible control. // In: D. Srinivasan, L.C. Jain (eds.). Innovations in Multi-Agent Systems and Applications – 1. Studies in Computational Intelligence, vol. 310. Berlin, Heidelberg: Springer, 2010. C. 223–246.

12. Yang, X., Liu, Z., Jiang, W., Zhang, C., Zhao, L., Song, L., & Bian, J. (2023). A versatile Multi-Agent reinforcement learning benchmark for inventory management. arXiv (Cornell University). <https://doi.org/10.48550/arxiv.2306.07542>
13. Бочок В.О., Федорова Н.В. Багатоагентні системи та проблеми їх оптимізації. Вчені Записки Таврійського національного університету імені В.І. Вернадського, 2023. Том 34 (73) № 2. С.131-137.
14. Бочок, В.О, Федорова, Н.В. Централізоване навчання для deep Q-learning моделей. Інформаційні технології та суспільство, Випуск 2 (13). 2024. С. 6-11.
15. Бочок, В., & Федорова, Н. Обмін знаннями в Independent DQN багатоагентній системі та особливості його реалізації. Записки Таврійського національного університету імені В.І. Вернадського, 2025. Том 36 (75) № 3.
16. Бочок В.О., Федорова Н.В. Обмін досвідом між агентами в багатоагентній системі. Матеріали І-ї міжнародної науково – практичної конференції «Сучасні аспекти інженерії програмного забезпечення». – м. Київ, 14 грудня 2023 р. – С.92
17. Бочок В.О., Федорова Н.В. Обмін та збереження інформації між агентами, здатними до навчання. Збірник матеріалів III Міжнародної науково-технічної конференції “Системи і технології зв’язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку”, 30 листопада 2023 року, м. Київ. Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, 2023. С. 89.
18. Бочок В.О., Федорова Н.В. Оптимізація багатоагентних систем. XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики» / XXI International scientific and practical conference of young scientists and students "Modern problems of scientific support of power engineering". 2024
19. Бочок В., Федорова Н. Визначення якісної та кількісної різниці в досвіді агентів для його передачі. XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики». 2024.

- 20.S. V. Albrecht, F. Christianos, and L. Schäfer, Multi-Agent Reinforcement Learning: Foundations and Modern Approaches. Cambridge, MA, USA: MIT Press, 2024.
- 21.R. Sutton and A. Barto, Reinforcement Learning: An Introduction, 2nd ed. MIT Press, 2018.
- 22.L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- 23.M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 2nd ed. Wiley, 2014.
- 24.D. P. Bertsekas, Reinforcement Learning and Optimal Control. Belmont, MA, USA: Athena Scientific, 2019.
- 25.Y. Li, "Deep reinforcement learning: An overview," arXiv:1701.07274, 2017.
- 26.M. Dulac-Arnold et al., "Challenges of real-world reinforcement learning," arXiv:1904.12901, 2019.
- 27.V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
- 28.K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- 29.P. Henderson et al., "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artificial Intelligence (AAAI)*, New Orleans, LA, USA, 2018, pp. 3207–3214.
- 30.J. Achiam, "Spinning Up in Deep Reinforcement Learning," OpenAI, 2018.
- 31.A. Barreto et al., "Successor features for transfer in reinforcement learning," *NeurIPS*, 2017.
- 32.J. Foerster et al., "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1146–1155.
- 33.T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.
- 34.A. Gupta et al., "Cooperative multi-agent control using deep reinforcement learning," *Proc. AAMAS*, 2017.

- 35.K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*, Studies in Systems, Decision and Control, vol. 325. Cham, Switzerland: Springer, 2021, pp. 321–384.
- 36.L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 38, no. 2, pp. 156–172, 2008.
- 37.R. Lowe et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 6379–6390.
- 38.T. Hester et al., "Deep Q-learning from demonstrations," *Proc. AAAI Conf. Artificial Intelligence*, 2018.
- 39.J. Hernandez-Leal, B. Kartal, and M. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, pp. 750–797, 2019.
- 40.S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, Feb. 2022.
- 41.C. Hernandez-Garcia and R. S. Sutton, "Understanding multi-agent cooperation from the perspective of multi-armed bandits," *arXiv preprint arXiv:1912.10081*, 2019.
- 42.T. Yang et al., "An efficient transfer learning framework for multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.
- 43.M. Gerstgrasser, T. Danino, and S. Keren, "Selectively sharing experiences improves multi-agent reinforcement learning," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, 2023, pp. 2433–2435.
- 44.S. Wadhwanian et al., "Policy distillation and value matching in multiagent reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 5871–5878.

- 45.W.-C. Tseng, T.-H. J. Wang, S.-W. Liu, Y.-C. Liu, Y.-H. Lin, and S.-D. Lin, "Offline Multi-Agent Reinforcement Learning with Knowledge Distillation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- 46.R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Reincarnating reinforcement learning: Reusing prior computation to accelerate progress," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022.
- 47.F. da Silva, M. Taylor, A. Reali Costa, "Autonomously Reusing Knowledge in Multiagent RL," in *Proc. IJCAI*, pp. 4543–4549, 2018
- 48.F. L. Da Silva, G. Warnell, A. H. R. Costa, and P. Stone, "Agents teaching agents: A survey on inter-agent transfer learning," *Auton. Agents Multi-Agent Syst.*, vol. 34, no. 1, art. 9, 2020.
- 49.M. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.
- 50.S. Lazarcic, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning*, Springer, 2012.
- 51.Y. Mai, Y. Zang, Q. Yin, W. Ni, and K. Huang, "Deep multitask multiagent reinforcement learning with knowledge transfer," *IEEE Trans. Games*, vol. 16, no. 3, pp. 566–576, Sep. 2024.
- 52.D. Shi, J. Tong, Z. Chen, S. Deng, and K. Chen, "Knowledge reuse of multi-agent reinforcement learning in cooperative tasks," *Applied Sciences*, vol. 12, no. 24, p. 12723, 2022, doi: 10.3390/app122412723.
- 53.A. A. Rusu et al., "Policy distillation," *Proc. ICLR*, 2016.
- 54.R. Teh et al., "Distral: Robust multitask reinforcement learning," *NeurIPS*, 2017.
- 55.I.-J. Liu, U. Jain, R. A. Yeh, and A. G. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, PMLR vol. 139, 2021, pp. 6826–6836.
- 56.E. Ilhan, J. Gow, and D. Perez-Liebana, "Action advising with advice imitation in deep reinforcement learning," in *Proc. 20th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Online, May 2021, pp. 629–637.

- 57.Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," in Proc. 8th Int. Conf. Learn. Represent. (ICLR), Virtual, Apr. 2020.
- 58.S. Omidshafiei, D.-K. Kim, M. Liu, G. Tesauro, M. Riemer, C. Amato, M. Campbell, and J. P. How, "Learning to teach in cooperative multiagent reinforcement learning," in Proc. 33rd AAAI Conf. Artif. Intell. (AAAI), Honolulu, HI, USA, 2019, pp. 6128–6136.
59. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in Proc. 34th AAAI Conf. Artif. Intell. (AAAI), New York, NY, USA, 2020, pp. 5792–5799.
- 60.S. Ganapathi Subramanian, M. E. Taylor, K. Larson, and M. Crowley, "Multi-agent advisor Q-learning," *J. Artif. Intell. Res.*, vol. 74, pp. 1–74, May 2022.
- 61.D. Amir et al., "Interactive teaching strategies for agent training," *IJCAI*, 2016.
- 62.F. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," *AAMAS*, 2013.
- 63.J. Zhao, X. Hu, M. Yang, W. Zhou, J. Zhu, and H. Li, "CTDS: Centralized teacher with decentralized student for multi-agent reinforcement learning," *IEEE Trans. Games*, vol. 16, no. 1, pp. 140–150, Mar. 2024.
- 64.A. Tsao et al., "A Survey of Multi-Agent Reinforcement Learning: Federated, Cooperative and Noncooperative Regimes," *arXiv:2402.12345*, 2024
- 65.M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in Proc. 10th Int. Conf. Mach. Learn. (ICML), Amherst, MA, USA, 1993, pp. 330–337.
- 66.M. Hessel et al., "Deep Reinforcement Learning at the Edge of the Statistical Precipice," in Proc. International Conference on Learning Representations (ICLR), 2018.
- 67.C. Yu, A. Velu, E. Vinitzky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 24611–24624.

- 68.V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2000.
- 69.C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
- 70.R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, 1992.
- 71.I. Goodfellow et al., "Deep Learning," MIT Press, 2016
- 72.K. Zhang et al., "Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents," *ICML*, 2018.
- 73.H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," in *Proc. AAAI Conference on Artificial Intelligence*, 2016.
- 74.M. L. Littman, "Markov Games as a Framework for Multi-Agent Reinforcement Learning," in *Proc. International Conference on Machine Learning (ICML)*, 1994.
- 75.J. N. Foerster et al., "Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning," in *Proc. International Conference on Machine Learning (ICML)*, 2017.
- 76.M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2nd ed. Hoboken, NJ, USA: Wiley, 2014.
- 77.Z. Wang et al., "Dueling Network Architectures for Deep Reinforcement Learning," in *Proc. International Conference on Machine Learning (ICML)*, 2016.
- 78.F. Christianos, L. Schäfer, and S. V. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 10707–10717.
- 79.K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, Jun. 2019, vol. 97, pp. 5887–5896.
- 80.T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 178, pp. 1–51, 2020.

81. G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in NIPS Deep Learning and Representation Learning Workshop, 2015, arXiv:1503.02531.
82. J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.
83. D. Silver et al., "Deterministic Policy Gradient Algorithms," in *Proc. International Conference on Machine Learning (ICML)*, 2014.
84. W. M. Czarnecki, R. Pascanu, S. Osindero, S. M. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *Proc. 22nd Int. Conf. Artif. Intell. Statist. (AISTATS)*, Naha, Japan, Apr. 2019, vol. 89, pp. 1331–1340.
85. Z. Gao, K. Xu, B. Ding, and H. Wang, "KnowRS: Knowledge reuse via knowledge distillation in multi-agent reinforcement learning," *Entropy*, vol. 23, no. 8, p. 1043, 2021.
86. Y. Pei, T. Ren, Y. Zhang, Z. Sun, and M. Champeyrol, "Policy distillation for efficient decentralized execution in multi-agent reinforcement learning," *Neurocomputing*, vol. 626, art. no. 129617, 2025.
87. P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst. (AAMAS)*, Stockholm, Sweden, 2018, pp. 2085–2087.
88. Z. Xu, K. Wu, Z. Che, J. Tang, and J. Ye, "Knowledge transfer in multi-task deep reinforcement learning for continuous control," in *Proc. 34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, Canada, 2020.
89. L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.
90. M. Hessel et al., "Rainbow: Combining Improvements in Deep Reinforcement Learning," in *Proc. AAAI Conference on Artificial Intelligence*, 2018.

91. R. S. Sutton, "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding," Proc. NeurIPS, 1996.
92. OpenAI, "LunarLander Environment Documentation," 2023. [Online]. Available: <https://gymnasium.farama.org> (accessed 2026.02.17)
93. G. Brockman et al., "OpenAI Gym," arXiv:1606.01540, 2016.
94. Farama Foundation, "Gymnasium: A Standard API for Reinforcement Learning Environments," 2023. [Online]. Available: <https://gymnasium.farama.org> (accessed 2026.02.17)
95. J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," JMLR, vol. 7, pp. 1–30, 2006.
96. S. Ahilan and P. Dayan, "Correcting experience replay for multi-agent communication," in Proc. 9th Int. Conf. Learn. Represent. (ICLR), Virtual, 2021.
97. J. Jeon, W. Kim, W. Jung, and Y. Sung, "MASER: Multi-agent reinforcement learning with subgoals generated from experience replay buffer," in Proc. 39th Int. Conf. Mach. Learn. (ICML), Baltimore, MD, USA, 2022, pp. 10041–10052.
98. R. Bellman, Dynamic Programming. Princeton University Press, 1957.
99. J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang, "Exploration in deep reinforcement learning: From single-agent to multiagent domain," IEEE Trans. Neural Netw. Learn. Syst., vol. 35, no. 7, pp. 8762–8782, Jul. 2024.
100. M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, Long Beach, CA, USA, 2017.
101. Y. Zhu et al., "Transfer Learning in Deep Reinforcement Learning: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 11, pp. 13344–13362, 2023.
102. W. Hasselbring, G. Steinacker, "Microservice Architectures for Scalability, Agility and Reliability," in Proc. IEEE ICSE Archit. Workshops, pp. 243–246, 2017
103. S. Meshram, R. Kulkarni et al., "Design Patterns for Scalable E-Commerce Application," IJIRCT, vol. 7, no. 1, pp. 477–482, 2021

104. H. Jo, M.-J. Kim, “Evolutionary Sampling for Knowledge Distillation in Multi-Agent RL,” *Math.*, vol. 13, no. 17:2734, 2025
105. J. Jeon, W. Kim, W. Jung, and Y. Sung, "MASER: Multi-agent reinforcement learning with subgoals generated from experience replay buffer," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, Baltimore, MD, USA, 2022, pp. 10041–10052.
106. Farama Foundation, “PettingZoo: An API standard for multi-agent reinforcement learning,” 2024. [Online]. Available: <https://pettingzoo.farama.org> (accessed 2026.02.17)
107. C. Hill, S. Greydanus et al., “PettingZoo: Gym for Multi-Agent Reinforcement Learning,” in *Proc. NeurIPS Workshop on RL (ICLRL)*, vol. 1, 2020
108. Tianshou Contributors, “Tianshou: A Deep Reinforcement Learning Library,” Documentation, 2024. [Online]. Available: <https://tianshou.org> (accessed 2026.02.17)
109. AgileRL Contributors, “AgileRL Documentation,” 2024. [Online]. Available: <https://docs.agilerl.com> (accessed 2026.02.17)
110. J. Raffin et al., “Stable-Baselines3: Reliable Reinforcement Learning Implementations,” *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021
111. M. Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning,” 2016, arXiv:1605.08695
112. W. Hasselbring, G. Steinacker, “Microservice Architectures for Scalability, Agility and Reliability,” in *Proc. IEEE ICSE Archit. Workshops*, pp. 243–246, 2017
113. Y.-W. Chong et al., “Applications of Multi-Agent Deep Reinforcement Learning: Models and Algorithms,” *Appl. Sci.*, vol. 11, no. 22:10870, 2021
114. T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020, doi: 10.1109/TCYB.2020.2977374
115. J. Pineau et al., "Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program)," *J. Mach. Learn. Res.*, vol. 22, no. 164, pp. 1–20, 2021.

116. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–846, Sep. 1983.
117. S. C. Y. Chan, S. Fishman, J. Canny, A. Korattikara, and S. Guadarrama, "Measuring the reliability of reinforcement learning algorithms," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020. arXiv:1912.05663.
118. C. Colas, O. Sigaud, and P.-Y. Oudeyer, "A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms," arXiv:1904.06979, Apr. 2019.
119. R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. G. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 29304–29320. arXiv:2108.13264.
120. C. Lyle, Z. Zheng, E. Nikishin, B. Avila Pires, R. Pascanu, and W. Dabney, "Understanding plasticity in neural networks," in *Proc. 40th Int. Conf. Mach. Learn. (ICML)*, Honolulu, HI, USA, Jul. 2023, vol. 202, pp. 23190–23211.
121. S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 29, Barcelona, Spain, Dec. 2016, pp. 2244–2252. arXiv:1605.07736.
122. S. Amershi et al., "Software engineering for machine learning: A case study," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng.: Softw. Eng. Pract. (ICSE-SEIP)*, Montreal, QC, Canada, May 2019, pp. 291–300.
123. Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, vol. 80, pp. 5571–5580. arXiv:1802.05438.
124. S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," arXiv:2005.01643, Nov. 2020.
125. C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Oper. Syst. Rev.*, vol. 49, no. 1, pp. 71–79, Jan. 2015, doi: 10.1145/2723872.2723882.

126. A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019, pp. 8024–8035. arXiv:1912.01703.
127. V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, Haifa, Israel, Jun. 2010, pp. 807–814.
128. P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Inf. Fusion*, vol. 85, pp. 1–22, Sep. 2022, doi: 10.1016/j.inffus.2022.03.003.
129. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015. arXiv:1412.6980.
130. L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3–4, pp. 293–321, May 1992, doi: 10.1007/BF00992699.

ДОДАТОК А СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Бочок В., Федорова Н. Багатоагентні системи та проблеми їх оптимізації. Вчені Записки Таврійського національного університету імені В.І. Вернадського. Серія. Технічні науки. 2023. Том 34 (73) № 2. С.131-137.
2. Бочок В., Федорова Н. Централізоване навчання для deep Q-learning моделей. Інформаційні технології та суспільство. 2024. Вип. 2 (13). С. 6-11.
3. Бочок В., Федорова Н. Обмін знаннями в Independent DQN багатоагентній системі та особливості його реалізації. Записки Таврійського національного університету імені В.І. Вернадського. Серія. Технічні науки. 2025. Том 36 (75) № 3. С. 113-119.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. Бочок В., Федорова Н. Обмін досвідом між агентами в багатоагентній системі. Матеріали І-ї міжнародної науково – практичної конференції «Сучасні аспекти інженерії програмного забезпечення». – м. Київ, 14 грудня 2023 р. – С.92
5. Бочок В., Федорова Н. Обмін та збереження інформації між агентами, здатними до навчання. Збірник матеріалів III Міжнародної науково-технічної конференції “Системи і технології зв’язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку”, 30 листопада 2023 року, м. Київ. Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, 2023. С. 89.
6. Бочок В., Федорова Н. Оптимізація багатоагентних систем. XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики» / XXI International scientific and practical conference of young scientists and students "Modern problems of scientific support of power engineering". 2024

7. Бочок В., Федорова Н. Визначення якісної та кількісної різниці в досвіді агентів для його передачі. XXI міжнародна науково-практична конференція молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики». 2024.

ДОДАТОК Б АКТИ ВПРОВАДЖЕННЯ



ЗАТВЕРДЖУЮ
професор з навчальної роботи
КПІ ім. Ігоря Сікорського,
кандидат технічних наук, доцент

Тетяна ЖЕЛЯСКОВА

« 20 » 03 2026 р.

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційного дослідження аспіранта кафедри інженерії програмного забезпечення в енергетиці, навчально-наукового інституту атомної та теплової енергетики Бочка В'ячеслава Олександровича за темою «Методи та програмні засоби децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням» на здобуття наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення» в освітній процес Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Комісія у складі:

- голови комісії, в. о. завідувача кафедри інженерії програмного забезпечення в енергетиці, професора, д. т. н. Ковалю О. В.;
- професора, д. т. н. Гаврилка С. В.;
- професора, д. т. н. Недашківського О. Л.;
- доцента, к.т.н. Варави І. А.

засвідчує, що результати дисертаційного дослідження Бочка В'ячеслава Олександровича на тему «Методи та програмні засоби децентралізованого обміну знаннями в системах багатоагентного навчання з підкріпленням», а саме: розроблені теоретичні підходи до побудови архітектури програмного забезпечення IoT як агентної системи, опис основних модулів та розподіл відповідальності; агентний підхід до побудови децентралізованих систем, запровадження механізмів обміну знаннями та його контролю для забезпечення оптимізації процесів, впроваджено у лекційні та практичні заняття навчальної дисципліни «Проектування кібер-фізичних систем», що входить до нормативної підготовки здобувачів ступеня бакалавра за ОПП «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» за спеціальністю 121 «Інженерія програмного забезпечення»:

Лекція 7. Архітектура програмного забезпечення для IoT-систем. Об'єктно-орієнтований підхід, модульність, взаємодія сенсорів і виконавчих блоків, логування та тестування.

Лекція 9. Мультиагентні системи та колективне керування. Теорія агентів, комунікація, узгодження рішень, приклади в розподілених енергосистемах.

Лекція 14. Самоорганізація та штучне життя в енергетичних мережах. Моделі колективної поведінки, рої та клітинні автомати для оптимізації процесів.

Комп'ютерний практикум 8. Мультиагентна взаємодія. Обмін повідомленнями для розподілу «права дії» за допомогою Raspberry Pi Pico.

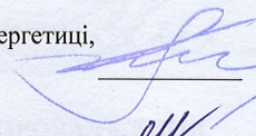
Посилання на силабус: <https://ipze.kpi.ua/>

Зазначене актуалізує зміст навчальної дисципліни до потреб практики, розкриває агентний підхід до проектування кібер-фізичних систем, показує вплив механізму обміну знаннями на швидкість навчання.

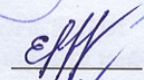
В. о. завідувача кафедри

інженерії програмного забезпечення в енергетиці,

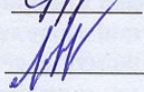
доктор технічних наук, професор

 Олександр КОВАЛЬ

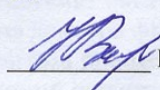
Професор, д. т. н.

 Євген ГАВРИЛКО

Професор, д. т. н.

 Олексій НЕДАШКІВСЬКИЙ

Доцент, к.т.н.

 Іван ВАРАВА



ТОВ "Квалітек"
03062, м. Київ, вул. Чистяківська, 2, оф. 206
+380678657007

Код ЄДРПОУ 38366304

mail@qualitek.com.ua

IBAN UA693007110000026008052604417

ІПН 383663026575



ЗАТВЕРДЖУЮ

Директор ТОВ «Квалітек»
Д.О. Дьомін
«26» 02 2026 р

АКТ ВПРОВАДЖЕННЯ

Цей акт складено про те, що на підприємстві ТОВ «Квалітек» були використані результати дисертаційного дослідження аспіранта Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» Бочка В'ячеслава Олександровича, поданої на здобуття ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення».

У процесі виконання дослідницьких та експериментальних робіт, пов'язаних із моделюванням та аналізом процесів передачі даних у телекомунікаційних мережах, було застосовано окремі результати дисертаційної роботи, зокрема:

- 1) **Метод децентралізованого вибору вчителя для передачі знань**, що дозволяє контролювати якість поширюваних знань, уникаючи негативного переносу.
- 2) **Метод динамічного децентралізованого вибору кількості епох дистиляції політики**, що дозволяє контролювати інтенсивність передачі знань між агентами за допомогою посилення чи послаблення передачі знань.
- 3) **Архітектуру програмного забезпечення** для реалізації механізму обміну знаннями, характерною особливістю якої є можливість децентралізованого обміну знань з застосуванням механізмів контролю якості та інтенсивності передачі.
- 4) **Система для організації повторюваних експериментів**.

Використання запропонованих у дисертаційній роботі методів обміну знаннями між агентами дозволило:

- прискорити процес навчання агентів у різних мережах або тестових конфігураціях;
- вирівняти продуктивність агентів у різних середовищах;
- підвищити ефективність пошуку оптимальних стратегій керування параметрами мережі.
- підвищити рівень автоматизації експериментальних досліджень

Результати дисертаційного дослідження можуть бути використані для подальшої розробки інтелектуальних систем моніторингу та оптимізації параметрів телекомунікаційних мереж, а також для автоматизації тестування якості обслуговування в мобільних мережах.

Від ТОВ «Квалітек»
Головний технолог

Л.В. Ліпницький

Від НТУУ «КПІ імені Ігоря Сікорського»
Аспірант

В. О. Бочок

ДОДАТОК В ЛІСТИНГ КОДУ ПРОГРАМНИХ РЕАЛІЗАЦІЙ

Код системи організації експериментів. Main.py

```
import argparse
import sys
from pathlib import Path

# Add src directory to path
sys.path.insert(0, str(Path(__file__).parent))

from src.commands import submit_command, serve_command, status_command
from src.database import init_db


def main():
    # Initialize database
    init_db()

    parser = argparse.ArgumentParser(
        description="Experiments Orchestrator - Manage and run containerized experiments"
    )

    subparsers = parser.add_subparsers(dest='command', help='Available commands')

    # Submit command
    submit_parser = subparsers.add_parser('submit', help='Submit a new experiment')
    submit_parser.add_argument(
        'definition_file',
        type=str,
        help='Path to experiment definition JSON file'
    )

    # Serve command
    serve_parser = subparsers.add_parser('serve', help='Start the experiment runner')
```

```

# Status command
status_parser = subparsers.add_parser('status', help='Show experiments and server status')
status_parser.add_argument(
    '--task_name',
    type=str,
    help='Filter by task name',
    default=None
)
status_parser.add_argument(
    '--hide_finished',
    action='store_true',
    help='Hide finished experiments'
)

args = parser.parse_args()

if args.command == 'submit':
    submit_command(args.definition_file)
elif args.command == 'serve':
    serve_command()
elif args.command == 'status':
    status_command(args.task_name, args.hide_finished)
else:
    parser.print_help()
    sys.exit(1)

if __name__ == "__main__":
    main()

```

Код системи організації експериментів. Commands.py

```

import json
import re
import sys

```

```

import time
from datetime import datetime
from pathlib import Path
from typing import Dict, Optional

from .database import (
    create_experiment, get_experiments, get_runs_by_experiment,
    get_experiments_needing_runs, create_run, update_run_status,
    get_all_runs
)
from .docker_manager import DockerManager
from .system_monitor import SystemMonitor

CPU_THRESHOLD = 90
MEMORY_THRESHOLD = 95
MAX_RUNNING_CONTAINERS = 10

SLEEP_TIME = 60

def validate_alias(alias: str) -> bool:
    """Validate experiment alias format"""
    pattern = r'^[a-zA-Z0-9][a-zA-Z0-9_]*$'
    return bool(re.match(pattern, alias))

def validate_task_exists(task_name: str) -> bool:
    """Check if task folder exists"""
    task_path = Path("tasks") / task_name
    return task_path.exists() and task_path.is_dir()

def submit_command(definition_file: str):
    """Submit a new experiment from definition file"""

```

try:

```

    # Read and parse definition file
    with open(definition_file, 'r') as f:
        definition = json.load(f)

    # Validate required fields
    required_fields = ['alias', 'task_name', 'instances']
    for field in required_fields:
        if field not in definition:
            print(f'Error: Missing required field '{field}' in definition file")
            sys.exit(1)

    # Extract fields
    alias = definition['alias']
    description = definition.get('description', "")
    task_name = definition['task_name']
    instances = definition['instances']
    envs = definition.get('envs', {})

    # Validate alias format
    if not validate_alias(alias):
        print(f'Error: Invalid alias format. Must match [a-zA-Z0-9][a-zA-Z0-9_-.]+")
        sys.exit(1)

    # Validate task exists
    if not validate_task_exists(task_name):
        print(f'Error: Task folder 'tasks/{task_name}' does not exist")
        sys.exit(1)

    # Validate instances
    if not isinstance(instances, int) or instances < 1 or instances > 100:
        print(f'Error: 'instances' must be a positive integer between 1 and 100")
        sys.exit(1)

    # Create experiment

```

```

experiment_id = create_experiment(alias, description, task_name, instances, envs)
print(f'✓ Experiment submitted successfully (ID: {experiment_id})')
print(f' Task: {task_name}')
print(f' Alias: {alias}')
print(f' Instances: {instances}')

except FileNotFoundError:
    print(f'Error: Definition file '{definition_file}' not found')
    sys.exit(1)
except json.JSONDecodeError as e:
    print(f'Error: Invalid JSON in definition file: {e}')
    sys.exit(1)
except Exception as e:
    print(f'Error: {e}')
    sys.exit(1)

def serve_command():
    """Start the experiment runner service"""
    print("Starting Experiments Orchestrator service...")
    print("Press Ctrl+C to stop\n")

    docker_manager = DockerManager()
    system_monitor = SystemMonitor()

    try:
        while True:
            print(f'\n[ {datetime.now().strftime('%Y-%m-%d %H:%M:%S')} ] Running service cycle...')

            # Update status of running containers
            print("Updating run statuses...")
            runs = get_all_runs()
            active_runs = [r for r in runs if r['status'] == 'Started']

            for run in active_runs:

```

```

container_status = docker_manager.get_container_status(run['container_name'])
if container_status == 'exited':
    exit_code = docker_manager.get_container_exit_code(run['container_name'])
    if exit_code == 0:
        update_run_status(run['id'], 'Finished')
        print(f' ✓ Run {run['id']} ({run['container_name']}) finished successfully')
    else:
        update_run_status(run['id'], 'Failed')
        print(f' ✗ Run {run['id']} ({run['container_name']}) failed with exit code
{exit_code}')
elif container_status == 'not_found':
    update_run_status(run['id'], 'Failed')
    print(f' ✗ Run {run['id']} ({run['container_name']}) container not found")

# Check system resources and start new runs if possible
cpu_usage, ram_usage = system_monitor.get_usage()
active_runs_count = len([r for r in get_all_runs() if r['status'] == 'Started'])

print(f"\nSystem usage - CPU: {cpu_usage:.1f}%, RAM: {ram_usage:.1f}%")
print(f"Containers running: {active_runs_count}")

if cpu_usage < CPU_THRESHOLD and ram_usage < MEMORY_THRESHOLD and
active_runs_count < MAX_RUNNING_CONTAINERS:
    experiments_needing_runs = get_experiments_needing_runs()

    if experiments_needing_runs:
        print("\nStarting new runs...")

        for experiment, needed_runs in experiments_needing_runs:
            # Check resources again before each run
            cpu_usage, ram_usage = system_monitor.get_usage()
            if cpu_usage >= CPU_THRESHOLD or ram_usage >= MEMORY_THRESHOLD:
                print(" System resource limit reached, stopping new runs")
                break

```

```

        # Start one run for this experiment
        try:
            run_id = docker_manager.start_run(experiment)
            if run_id:
                print(f" ✓ Started run {run_id} for experiment '{experiment['alias']}'")
            else:
                print(f" ✗ Failed to start run for experiment '{experiment['alias']}'")
        except Exception as e:
            print(f" ✗ Error starting run for '{experiment['alias']}': {e}")

        # Only start one run per cycle to avoid overwhelming the system
        break

    else:
        print("\nNo experiments need new runs")

    else:
        print(f"System resources above {CPU_THRESHOLD}% threshold or max running
experiments reached, skipping new runs")

        # Wait 10 minutes before next cycle
        print("\nWaiting 5 minutes until next cycle...")
        time.sleep(SLEEP_TIME)

except KeyboardInterrupt:
    print("\n\nService stopped by user")
    sys.exit(0)

def status_command(task_name: Optional[str], hide_finished: bool):
    """Show experiments and server status"""
    experiments = get_experiments(task_name)

    if not experiments:
        if task_name:
            print(f"No experiments found for task '{task_name}'")
        else:

```

```

    print("No experiments found")
    return

print("EXPERIMENTS STATUS")
print("=" * 80)

for exp in experiments:
    runs = get_runs_by_experiment(exp['id'])

    # Count run statuses
    started = len([r for r in runs if r['status'] == 'Started'])
    finished = len([r for r in runs if r['status'] == 'Finished'])
    failed = len([r for r in runs if r['status'] == 'Failed'])
    total = len(runs)

    # Skip if hiding finished and all runs are complete
    if hide_finished and total >= exp['instances'] and started == 0:
        continue

    print(f"\nExperiment: {exp['alias']} (ID: {exp['id']})")
    print(f"Task: {exp['task_name']}")
    print(f>Description: {exp['description'] or 'N/A'})
    print(f"Progress: {total}/{exp['instances']} runs")
    print(f>Status: ✓ Finished: {finished}, ⚡ Running: {started}, ✗ Failed: {failed}")

    if runs:
        print("\n Recent runs:")
        for run in runs[-5:]: # Show last 5 runs
            status_icon = {
                'Started': ' ⚡ ',
                'Finished': ' ✓ ',
                'Failed': ' ✗ '
            }.get(run['status'], '?')

```

```

        print(f"    {status_icon} Run {run['id']}: {run['container_name']} - {run['status']}")

print("\n" + "=" * 80)

# Show system status
system_monitor = SystemMonitor()
cpu_usage, ram_usage = system_monitor.get_usage()
print(f"\nSYSTEM STATUS")
print(f"CPU Usage: {cpu_usage:.1f}%")
print(f"RAM Usage: {ram_usage:.1f}%")

# Show Docker status
docker_manager = DockerManager()
if docker_manager.is_docker_running():
    print("Docker: ✓ Running")
else:
    print("Docker: ✗ Not running")

```

Код системи організації експериментів. `Docker_manager.py`

```

import subprocess
import socket
from pathlib import Path
from typing import Dict, Optional

from .database import create_run

class DockerManager:
    def __init__(self):
        self.hostname = socket.gethostname()

    def is_docker_running(self) -> bool:
        """Check if Docker daemon is running"""
        try:
            result = subprocess.run(

```

```

        ['docker', 'info'],
        capture_output=True,
        text=True,
        check=False
    )
    return result.returncode == 0
except FileNotFoundError:
    return False

def image_exists(self, image_name: str) -> bool:
    """Check if Docker image exists locally"""
    try:
        result = subprocess.run(
            ['docker', 'images', '-q', image_name],
            capture_output=True,
            text=True,
            check=False
        )
        return bool(result.stdout.strip())
    except:
        return False

def build_image(self, task_name: str) -> bool:
    """Build Docker image for a task"""
    task_path = Path("tasks") / task_name
    image_name = f'task_{task_name}'

    print(f' Building Docker image '{image_name}'...')
    # print("task_path", task_path)
    # print("command:", *['docker', 'build', '-t', image_name, '.'])
    try:
        result = subprocess.run(
            ['docker', 'build', '-t', image_name, '.'],
            cwd=task_path,
            capture_output=True,

```

```

        text=True,
        check=False
    )

    if result.returncode == 0:
        print(f' ✓ Successfully built image '{image_name}''')
        return True
    else:
        print(f' ✗ Failed to build image: {result.stderr}')
        return False
except Exception as e:
    print(f' ✗ Error building image: {e}')
    return False

def get_container_status(self, container_name: str) -> str:
    """Get the status of a container (running, exited, not_found)"""
    try:
        result = subprocess.run(
            ['docker', 'inspect', '-f', '{{.State.Status}}', container_name],
            capture_output=True,
            text=True,
            check=False
        )

        if result.returncode == 0:
            return result.stdout.strip()
        else:
            return 'not_found'
    except:
        return 'not_found'

def get_container_exit_code(self, container_name: str) -> Optional[int]:
    """Get the exit code of a container"""
    try:
        result = subprocess.run(

```

```

        ['docker', 'inspect', '-f', '{{.State.ExitCode}}', container_name],
        capture_output=True,
        text=True,
        check=False
    )

    if result.returncode == 0:
        return int(result.stdout.strip())
    else:
        return None
except:
    return None

def start_run(self, experiment: Dict) -> Optional[int]:
    """Start a new run for an experiment"""
    task_name = experiment['task_name']
    image_name = f'task_{task_name}'

    # Check if image exists, build if not
    if not self.image_exists(image_name):
        if not self.build_image(task_name):
            return None

    # Create run in database first to get run_id
    # Temporary container name, will update after we have run_id
    temp_container_name = f'temp_{task_name}_{experiment["id"]}'
    run_id = create_run(experiment['id'], self.hostname, temp_container_name)

    # Create actual container name
    container_name = f'{task_name}_{experiment["alias"]}_{run_id}'

    # Create output directory
    output_dir = Path("output") / task_name / str(experiment['id']) / str(run_id)
    output_dir.mkdir(parents=True, exist_ok=True)

```

```

# Prepare docker run command
docker_cmd = [
    'docker', 'run',
    '-d', # Detached mode
    '--gpus=all',
    '--name', container_name,
    '-v', f'{output_dir.absolute()}:/output'
]

# Add environment variables
for key, value in experiment['envs'].items():
    docker_cmd.extend(['-e', f'{key}={value}'])

docker_cmd.append(image_name)

# Run container
try:
    result = subprocess.run(
        docker_cmd,
        capture_output=True,
        text=True,
        check=False
    )

    if result.returncode == 0:
        # Update container name in database
        from .database import sqlite3, DB_PATH
        conn = sqlite3.connect(DB_PATH)
        cursor = conn.cursor()
        cursor.execute(
            "UPDATE runs SET container_name = ? WHERE id = ?",
            (container_name, run_id)
        )
        conn.commit()
        conn.close()

```

```

        return run_id
    else:
        print(f' X Failed to start container: {result.stderr}')
        # Update run status to failed
        from .database import update_run_status
        update_run_status(run_id, 'Failed')
        return None

except Exception as e:
    print(f' X Error starting container: {e}')
    from .database import update_run_status
    update_run_status(run_id, 'Failed')
    return None

```

Код системи організації експериментів. System_monitor.py

```

import psutil
from typing import Tuple

class SystemMonitor:
    def get_usage(self) -> Tuple[float, float]:
        """Get current CPU and RAM usage percentages"""
        # CPU usage over 1 second interval
        cpu_percent = psutil.cpu_percent(interval=1)

        # RAM usage
        memory = psutil.virtual_memory()
        ram_percent = memory.percent

        return cpu_percent, ram_percent

    def get_detailed_stats(self) -> dict:
        """Get detailed system statistics"""
        cpu_percent = psutil.cpu_percent(interval=1)
        memory = psutil.virtual_memory()

```

```
return {  
  'cpu': {  
    'percent': cpu_percent,  
    'count': psutil.cpu_count()  
  },  
  'memory': {  
    'percent': memory.percent,  
    'total': memory.total,  
    'available': memory.available,  
    'used': memory.used  
  }  
}
```