Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна

Кваліфікаційна наукова
праця на правах рукопису

Новіков Артем Олександрович

УДК 004.94

ДИСЕРТАЦІЯ

ІНФОМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ ТЕСТУВАННЯ АЛГОРИТМІВ КЕРУВАННЯ РОБОТИЗОВАНИМИ ПРИСТРОЯМИ У СЦЕНАРІЯХ КОМАНДНОЇ ВЗАЄМОДІЇ У МУЛЬТИАГЕНТНИХ СИСТЕМАХ

Спеціальність 122 Комп'ютерні науки
Галузь знань 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

_____Новіков А.О.

Науковий керівник: Яковлев Сергій Всеволодович, член-кореспондент НАН України, доктор фізико-математичних наук, професор

Харків – 2025

АНОТАЦІЯ

Новіков А.О. Інфомаційна технологія для тестування тестування алгоритмів керування роботизованими пристроями у сценаріях командної взаємодії у мультиагентних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 122 Комп'ютерні науки (Галузь знань 12 Інформаційні технології). – Харківський національний університет імені В. Н. Каразіна Міністерства освіти і науки України, Харків, 2025.

Дисертаційна робота присвячена розробці інформаційної технології для тестування алгоритмів керування роботизованими пристроями у сценаріях командної взаємодії в мультиагентних системах, що є актуальною задачею сучасної робототехніки та штучного інтелекту. У роботі представлено платформу моделювання на основі гри Pac-Man, яка дозволяє досліджувати кооперативні та антагоністичні стратегії взаємодії агентів у 2D-середовищах. Запропоновано інноваційні модифікації алгоритмів пошуку, зокрема MCTS, а також проаналізовано ефективність нейромережевих моделей, згенерованих за допомогою MADDPG. Результати дисертації мають як наукову новизну, так і практичне значення для побудови автономних систем управління в реальних умовах.

Зміст дисертації. У **вступі** обґрунтовано актуальність теми, сформульовано мету та задачі досліджень, розкрито наукову новизну та практичну цінність роботи, представлено її загальну характеристику.

У **розділі 1** проведено комплексний аналіз наукових джерел з питань комп'ютерного моделювання та алгоритмів управління автономними агентами, із детальним розглядом сучасних методів моделювання взаємодії роботизованих

систем. На основі отриманих даних сформульовано та ключові напрямки і завдання дослідження.

У **розділі 2** здійснено систематизацію та формалізацію завдань агентів, де чітко окреслено їх ключові ролі та взаємодію у заданому контексті, що імітує реальний контекст застосування роботизованих систем. Для комп'ютерного моделювання запропоновані моделі, що включають інформаційну, структурну та функціональну складові, які надають можливість побудувати архітектуру та розробити двовимірне середовище для тестування алгоритмів керування роботизованими пристроями в умовах, наближених до реальних місій.

У **розділі 3** проведено детальний аналіз пошукових алгоритмів, призначених для визначення оптимального шляху до цільових позицій на території. Обґрунтовано використання як класичних методів, так і запропонованих модифікацій, адаптованих для мультиагентної взаємодії з урахуванням актуальних загроз. Аргументовано використання функцій оцінки рішень і евристичних методів, запропоновано використання методів машинного навчання для синтезу адаптивних нейромережевих моделей, здатних реагувати на різноманітні сценарії руху та взаємодії агентів, здійснено систематизацію методів розробки та збору статистичних даних з метою подальшої оцінки ефективності підходів керування за допомогою розробленої платформи.

У **розділі 4** представлено результати експериментальних досліджень, у ході яких проведено детальний аналіз ефективності як традиційних алгоритмів, так і навченої моделі нейронної мережі для керування роботизованими пристроями. Обґрунтовано експериментальні умови, запропоновано структуру тестового середовища та здійснено порівняльну оцінку отриманих результатів з обгрунтуванням на основі існуючих наукових досліджень. Запропоновано подальші кроки та напрямки удосконалення розглянутих підходів керування.

У **розділі 5** проведено детальний аналіз альтернативних стратегій керування агентами у складних середовищах, які підлягають подальшому

тестуванню із застосуванням комп'ютерного моделювання у розробленій платформі. Особливу увагу приділено сучасним нейронним мережевим архітектурам для прогнозування траєкторій та гібридним методам, що інтегрують нейронні мережі з алгоритмами навчання з підкріпленням.

У **висновках** підведені підсумки та запропоновані напрямки та перспективи подальших досліджень, а саме додавання підтримки 3D-моделей сероедовища з розподілом висот польоту з метою надання можливості використання технології для моделювання процесу керування БПЛА у наближених до реальних умов, інтеграція аспектів реальних загрозу, включно з кібербезпекою, оптимізація та додавання нових алгоритмів в рамках системи, інтеграція програмно-алгоритмічних підходів до формалізації та оптимізації конфігурацій покриття, додавання функціональності для гібридних підходів з участю людини.

Ключові слова: інтелектуальні стратегії управління, штучний інтелект, моделювання в реальному часі, глибоке навчання, нейронні мережі, ройові алгоритми, машинне навчання, навчання з підкріпленням, мультиагентні системи, безпілотні літальні апарати (БПЛА), роботизовані та мобільні робототехнічні системи, інформаційно-аналітичні системи, прийняття рішень та функції, OpenAI

ABSTRACT

Novikov A.O. Information technology for testing algorithms for controlling robotic devices in scenarios of team interaction in multi-agent systems. - Qualification scientific work with the manuscript copyright.

Dissertation for a Doctor of Philosophy Degree by specialty 122 Computer science (Field of knowledge 12 Information Technologies). – V. N. Karazin Kharkiv National University of the Ministry of Education and Science of Ukraine, Kharkiv, 2025.

The dissertation is dedicated to the development of an information technology for testing control algorithms of robotic devices in team interaction scenarios within multi-agent systems, which is a relevant task in modern robotics and artificial intelligence. The work presents a simulation platform based on the Pac-Man game, enabling the exploration of cooperative and antagonistic agent interaction strategies in 2D environments. Innovative modifications to search algorithms, particularly MCTS, are proposed, and the effectiveness of neural network models generated using MADDPG is analyzed. The results of the dissertation possess both scientific novelty and practical significance for the development of autonomous control systems in real-world conditions.

Contents of the dissertation. The **introduction** substantiates the relevance of the topic, formulates the purpose and objectives of the research, reveals the scientific novelty and practical value of the work, and presents its general characteristics.

Chapter 1 provides a comprehensive analysis of scientific sources on computer modeling and control algorithms for autonomous agents, with a detailed consideration of modern methods for modeling the interaction of robotic systems. Based on the findings, key research directions and objectives are formulated.

Chapter 2 systematizes and formalizes the agents' tasks, clearly defining their main roles and interactions in a specified context that simulates real-world applications

of robotic systems. For computer modeling, models are proposed that include informational, structural, and functional components, enabling the construction of an architecture and the development of a two-dimensional environment for testing control algorithms for robotic devices in conditions approximating real missions.

Chapter 3 offers a detailed analysis of search algorithms designed to determine the optimal path to target positions within a given area. Both classical methods and proposed modifications adapted for multi-agent interaction under current threats are justified. The use of decision-evaluation functions and heuristic methods is substantiated, and the application of machine learning methods for synthesizing adaptive neural network models—capable of responding to various scenarios of agent movement and interaction—is proposed. A systematization of methods for development and data collection is also presented, with the aim of further evaluating the effectiveness of the control approaches using the developed platform.

Chapter 4 presents the results of experimental studies, which include a detailed analysis of the effectiveness of both traditional algorithms and a trained neural network model for controlling robotic devices. The experimental conditions are grounded, a structure of the test environment is proposed, and a comparative assessment of the obtained results is performed, with supporting arguments derived from existing scientific research. Further steps and directions for enhancing the discussed control approaches are also suggested.

Chapter 5 offers a detailed analysis of alternative strategies for agent control in complex environments, which will be applicable to further testing through computer modeling on the developed platform. Special attention is devoted to modern neural network architectures for trajectory prediction and to hybrid methods that integrate neural networks with reinforcement learning algorithms.

The **Conclusions** summarize the findings and propose directions and prospects for further research. These include the addition of support for 3D models of the environment with altitude distribution to enable the use of the technology for simulating

UAV control processes under near-real conditions, the integration of aspects of real threats including cybersecurity, the optimization and incorporation of new algorithms within the system, the integration of software-algorithmic approaches for the formalization and optimization of coverage configurations, and the addition of functionality for hybrid approaches involving human participation.

Keywords: intelligent control strategies, artificial intelligence, real-time modeling, deep learning, neural networks, swarm algorithms, machine learning, reinforcement learning, multiagent systems, unmanned aerial vehicles (UAVs), robotized and mobile robotics systems, information and analytical systems, decision making and functions, OpenAI

Список наукових публікацій здобувача

Публікації, в яких опубліковано основні наукові результати дисертації

Статті у наукових фахових виданнях, що входять до міжнародних наукометричних баз:

1. Yakovlev S., **Novikov A.,** Gushchin I. Exploring the possibilities of MADDPG for UAV swarm control by simulating in Pac-Man environment. *Radioelectronic and Computer Systems*. 1(113). 2025. P. 327-337. http://doi.org/10.32620/reks.2025.1.21 (Scopus).

Статті у наукових фахових виданнях України:

2. **Новіков А.О.,** Маций О.Б. Потенціал використання нейронних мереж для передачення траекторії руху у мультиагентних системах на прикладі гри Pac-Man. *Вісник Кременчуцького національного університету імені Михайла Остроградського*. 2024. 4. С. 92–104. https://doi.org/10.32782/1995-0519.2024.4.12 *(Особистий внесок здобувача: проведення аналізу існуючих методів прогнозування траєкторії руху у динамічних середовищах, розробка запропонованого методу прогнозування траєкторії руху Pac-Man за допомогою нейронних мереж, порівняння традиційних та сучасних підходів до підвищення ефективності стратегій агентів.Особистий внесок Ольги Маций: перевірка наукової достовірності отримуваних результатів, перевірка тексту роботи.)*

3. **Novikov A.O.,** Yanovsky V.V. Analysis of Search and Multi-Agent Algorithms in the Pac-Man Game. *Control Systems and Computers*. 2024. 308(4). P. 19–33. https://doi.org/10.15407/csc.2024.04.019

(*Особистий внесок здобувача: програмна розробка середовища та імплементація алгоритмічних підходів, аналіз ефективності класичних пошукових алгоритмів, таких як A* та BFS, а також мультиагентних підходів, таких як Alpha-Beta, Expectimax і MCTS, у контексті модифікованого середовища на основій гри Pac-Man. Проведення експериментів з різними умовами гри, включаючи складність лабіринту, поведінку привидів і динаміку середовища. Оцінка впливу цих факторів на час виконання, оцінку та відсоток успішних заверень завдань агента. Відповідні результати наведені в практичній частині роботи.*

Особистий внесок Володимира Яновського: перевірка наукової достовірності отримуваних результатів, перевірка тексту роботи)

4. **Novikov A.O.,** Yanovsky V.V. Exploring the limits of mcts in pac-man: maze size, simulations, and performance. *Herald of Khmelnytskyi National University. Technical sciences*. 341(5). 2024. P. 351–359. https://doi.org/10.31891/2307-5732-2024-341-5-52

(Особистий внесок здобувача: програмна розробка середовища на основі гри Pac-Man та імплементація алгоритму пошуку дерева Монте-Карло (MCTS), аналіз продуктивності та обмежень алгоритму в різних умовах гри, включаючи змінну складність лабіринтів та кількість симуляцій алгоритму. Проведення експериментів з різними розмірами та конфігураціями лабіринтів, оцінка компромісів між обчислювальною вартістю та результативністю агента. Оцінка впливу середовища на ефективність алгоритму та пропозиція щодо покращень за рахунок адаптивних симуляцій та комбінування MCTS з іншими методами. Відповідні результати наведені в практичній частині роботи. Особистий внесок Володимира Яновського: перевірка наукової достовірності отриманих результатів, рецензування тексту статті, внесення корективів у формулювання та структуру роботи, перевірка правильності теоретичних основ дослідження)

5. **Novikov A.O.,** Yanovsky V.V. Analysis of the decision-making algorithm efficiency in complex game environments on the example of Pac-Man. Information Technologies and Computer Engineering. 21(3), P. 108–118. https://doi.org/10.63341/vitce/3.2024.108

(Особистий внесок здобувача: програмна розробка середовища на основі гри Pac-Man та імплементація алгоритмів Expectimax, MCTS та Alpha-Beta Pruning. Проведення експериментів з різними умовами гри, включаючи складність лабіринтів, кількість перешкод та їх вплив на ефективність алгоритмів. Оцінка ефективності алгоритмів на основі таких показників, як кількість балів, час гри та відсоток виграшів. Аналіз результатів експериментів для визначення найефективнішого підходу до ухвалення рішень у складних середовищах. Результати наведені в практичній частині роботи. Особистий внесок Володимира Яновського: перевірка наукової достовірності отриманих результатів, рецензування тексту статті, перевірка правильності інтерпретації результатів експериментів та оптимізації алгоритмів для складних середовищ.)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. **Novikov A.O.,** Matsyi O.B. Enhancing decision-making in multi-agent systems through neural network-based trajectory prediction. *Матеріали XII Міжнародної науково-практичної конференції «Актуальні проблеми сучасної науки та освіти»*. Львів, Україна. 2024. С. 54-56.
2. **Novikov A.O.,** Matsyi O.B. Comparative analysis of trajectory prediction techniques in multi-agent systems: traditional vs neural network approaches. *Матеріали XIII Міжнародної Науково-Практичної Конференції «Актуальні питання розвитку науки та освіти»*. Львів, Україна. 2024. С. 174-177. URL:
3. **Novikov A.,** Yanovsky V. Identification and analysis of shadow zones in artificial neurons. *Science and society: modern trends in a changing world. Proceedings of the 10th International scientific and practical conference*. MDPC Publishing. Vienna, Austria. 2024. P. 45-50.

ЗМІСТ

ВСТУП

У сучасному світі моделювання та тестування алгоритмів керування роботизованими пристроями є ключовими етапами їх розробки. Автономні системи дедалі частіше застосовуються у сферах логістики, промислової автоматизації, медицини, військової справи та кібербезпеки. Оптимізація їхньої поведінки в умовах змінного середовища потребує ефективних методів тестування, що забезпечують можливість аналізу алгоритмів ухилення, навігації та командної взаємодії.

Актуальність теми. Сучасний розвиток автономних систем відкриває широкі можливості для їхнього використання в різних сферах діяльності. Однак ефективне керування такими системами у динамічних умовах залишається складним завданням. Серед ключових викликів – адаптація алгоритмів до мінливого середовища, забезпечення злагодженої командної взаємодії між автономними агентами та розробка стратегій протидії в умовах конкурентного середовища. Ці аспекти є критично важливими у військових, логістичних та рятувальних операціях, де точність та оперативність прийняття рішень безпосередньо впливають на ефективність виконання завдань.

Розроблена інформаційна технологія надає можливість тестування алгоритмів у сценаріях, які моделюють як кооперативну, так і антагоністичну взаємодію автономних агентів. Це дозволяє визначити оптимальні підходи для виконання різних задач, зокрема:

- Розмінування територій, де автономні роботи аналізують навколишнє середовище та формують стратегії для ефективного очищення небезпечних зон.
- Логістика та автономна доставка, що вимагають оптимального планування маршрутів з урахуванням динамічних змін у середовищі та перешкод.

- Військові задачі БПЛА, у яких автономні безпілотні літальні апарати використовуються як для розвідки, так і для протидії ворожим системам.
- Керування роями дронів та групами роботизованих систем, що дозволяє розробляти узгоджені тактики дій та оптимізувати взаємодію між численними агентами.
- Протидія ворожим автономним системам, що є критично важливим у сценаріях кібербезпеки та оборони.

Одним із підходів до перевірки алгоритмів керування є використання симуляційних платформ. Відомі середовища, такі як Gazebo, Webots, MORSE та CoppeliaSim, забезпечують високу точність фізичного моделювання, однак їхня складність може ускладнювати тестування певних типів алгоритмів, як зазначається у роботі [1]. Альтернативою є 2D середовища, які, фокусуючись саме на дозволяють швидко змінювати умови експерименту, вводити сценарії з перешкодами, супротивниками та динамічними обмеженнями. Наприклад, гра Pac-Man часто використовується для дослідження алгоритмів керування агентами, як це показано у роботах [2, 3], а тому має перспективи, при певних модифікаціях, бути використанною як платформа для комп’ютерного моделювання процесів взаємодії рухомих пристроїв, що мають аналоги у задачах виконання місій і навігації роботизованих та безпілотних систем.

Сучасні підходи керування автономними агентами можна поділити на традиційні (використання пошукових алгоритмів та їх модифікацій під багатоагентну взаємодію), згадані у роботах [2, 3], керування за допомогою нейромереж, що ґрунтуються на використанні методів навчання з підкріпленням, освітлених у роботах [3, 4] (наприклад, Multi-Agent Deep Deterministic Policy Gradient – MADDPG) [5] та гібридних підходів, запропонованих у роботах [6, 7], що суміщають вище зазначені мтоди. Вибір відповідного алгоритму залежить від вимог до швидкості прийняття рішень, стратегії в залежності від задач та гнучкості адаптації. Для оцінки ефективності цих алгоритмів необхідно створити

інформаційну технологію, яка дозволить тестувати та порівнювати їх у різних сценаріях та на різних моделях територій, що дозволить синтезувати нові методи та підходи курування.

Для подібних досліджень активно використовуються моделі та фреймворки розроблені провідними компаніями, зокрема, MADDPG gym середовища від OpenAI, як ті, що використовувались у роботах [5, 8], а також та моделі нейромереж, як у роботі [9] та фремфорк CUDA від NVIDIA, про який згадано у роботі [10].

Мета дослідження — розробити інформаційну технологію для тестування алгоритмів керування роботизованими пристроями у сценаріях командної взаємодії у мультиагентних системах, оцінити ефективність традиційних та навчальних алгоритмів у змодельованих середовищах та визначити їхню придатність для реальних застосувань.

Об’єкт дослідження – це процеси моделювання та тестування алгоритмів керування роботизованими пристроями у сценаріях командної взаємодії в мультиагентних системах.

Предмет дослідження – це методи, моделі та інформаційна технологія, що забезпечує ефективне тестування алгоритмів керування роботизованими пристроями в умовах командної взаємодії мультиагентних систем за допомогою комп'ютерного моделювання.

Основні завдання дисертації:

1. Розробити інформаційну технологію для тестування алгоритмів керування роботизованими пристроями на базі 2D середовища гри Pac-Man.
2. Імплементувати існуючі пошукові алгоритми з модифікаціями для забезпечення координації автономних агентів у контексті виконання місій, спираючись на сучасні дослідження.
3. Згенерувати різні варіації 2D моделей середовища різного масштабу та складності, наближених до реальних умов, для проведення експериментів.

4. Провести експерименти з тестування імплементованих алгоритмів, проаналізувати отримані результати та порівняти їх із результатами наукових досліджень з метою підтвердження коректності роботи моделювання в розробленій інформаційній технології.
5. Забезпечити підтримку нейромережевих моделей для керування агентами в рамках розробленої інформаційної технології.
6. Синтезувати натреновану нейромережеву модель для подальшого тестування.
7. Провести експерименти та порівняльний аналіз отриманих результатів із даними інших досліджень для підтвердження коректності та ефективності моделювання в рамках розробленої інформаційної технології.

Важливо зазначити, що розроблена інформаційна технологія забезпечує можливість моделювання як з боку протагоніста, який виконує поставлене завдання (наприклад, доставку або розвідку), так і з боку антагоніста, який протидіє виконанню завдання (наприклад, блокує переміщення або організовує перехоплення). Це дозволяє створювати складні сценарії взаємодії 1:N та N:1, що значно розширює сферу застосування розробленої системи.

Методи дослідження. В процесі розв'язання поставлених задач використовувались наступні методи:

- Комп'ютерне моделювання та симуляції: Використання симуляційного 2D середовища (на базі гри Pac-Man) для відтворення сценаріїв командної взаємодії агентів, що дозволяє експериментально перевірити ефективність розроблених алгоритмів та підходів керування агентами.
- Алгоритмічна розробка та модифікація: Розробка та імплементація пошукових алгоритмів із адаптацією під мультиагентні системи, зокрема модифікація існуючих алгоритмів з урахуванням специфіки командної взаємодії.

- Методи машинного навчання та нейронних мереж: Застосування сучасних підходів машинного навчання для навчання моделей, здатних адаптуватися до різних сценаріїв руху та взаємодії агентів.
- Статистичний аналіз та валідація результатів: Проведення кількісного збору статистики та аналізу отриманих експериментальних даних із використанням критеріїв ефективності та порівняння з даними з літератури для підтвердження достовірності результатів.

Наукова новизна отриманих результатів. Наукову новизну в роботі мають такі практичні та теоретичні результати:

1) Представлено вперше:
   a) Розроблено платформу для тестування алгоритмів керування роботизованими пристроями у сценаріях командної взаємодії в мультиагентних системах на базі середовища гри Pac-Man, що дозволило моделювати реалістичні умови колективної взаємодії та об’єктивно оцінити ефективність застосовуваних підходів керування.
   b) Запропоновано модифікацію гри Pac-Man як інноваційний фреймворк для автоматизації досліджень та оцінювання ефективності алгоритмів керування роботизованими пристроями в антагоністичних та кооперативних стратегіях, що дозволило інтегрувати елементи гейміфікації з аналітичними підходами до моделювання поведінки агентів.
2) Набуло подальшого розвитку:
   a) Модифіковано алгоритм MCTS (Monte Carlo Tree Search) шляхом використання специфічної функції оцінки потенційного стану у процесі симуляції при плануванні маршруту та динамічному реагуванні на загрози, що дозволило підвищити точність прогнозування траєкторій і забезпечити оперативне реагування на змінні умови середовища.

b) Проведено детальний порівняльний аналіз та обгрунтовано ефективність розроблених алгоритмів пошуку та планування маршрутів для керування рухомими пристроями на 2D-моделях рельєфу різної складності, що надало змогу аргументувати ефективність роботи платформи на основі результатів суміжних наукових досліджень, а також ідентифікувати оптимальні алгоритмічні рішення під різні варіації моделей місцевості, що сприяє їх потенційному застосуванню у роботизованих системах.

c) За допомогою розробленої платформи, доведено адекватність синтезоваої моделі нейромережі, навчання якої здійснювалось за допомогою алгоритму MADDPG, у вирішенні задач протидії агентам, керованим базовими алгоритмами пошуку, у порівнянні з класичними правило-орієнтованими підходами перехоплення цілей. Це дозволило визначити переваги та обмеження методу навчання. Обгрунтовано потенціал використання алгоритму навчання MADDPG для синтезу моделей під реальне застосування у керуванні роботизованими системами.

Практичне значення отриманих результатів. Розроблена інформаційна технологія є інструментом для аналізу, тестування та синтезу нових моделей і методів керування роботизованими пристроями. Підбір і тестування алгоритмів для вирішення реальних задач дозволяє синтезувати нові методи та моделі, що знаходять практичне застосування у різних галузях, включаючи розробку роботизованих систем та БПЛА, логістику і військову справу, наприклад, розмінування, розвідка, рятуавальні операції тощо. Запропонована платформа є універсальним фреймворком, який не лише сприяє оцінці ефективності існуючих алгоритмів у змодельованих умовах, а й відкриває можливості для розширення набору методів, моделей середовищ та інтеграції нових стратегій автоматизованого керування у реальні автономні системи. Ретельний аналіз

алгоритмів, проведений у рамках дослідження, робить свій внесок у розширення наявної теоретичної та практичної основи, що може надати дослідникам та науковцям додаткові деталі для модифікації існуючих або розробки альтернативних рішень, що сприятиме подальшому розвитку теоретичних підходів. Завдяки цьому, результати дослідження мають високий потенціал для практичного впровадження та подальшого розвитку сучасних технологій керування рухомими пристроями у мультиагентних системах.

Особистий внесок здобувача. Автором самостійно отримано основні результати дисертаційного дослідження або за участю його керівника. В опублікованих в співавторстві наукових працях здобувачем здійснено огляд існуючих робіт та їх аналіз, розробку моделей та методів для імплементації інформаційної технології, створення 2D моделей середовищ для експериментів, синтез навченої нейромережевої моделі, проведення експериментів із збором статистичних даних та подальшим аналізом отриманих результатів, а також розробку програмного засобу на основі існуючих напрацювань, а саме [5, 11]. Співавторам статей належить постановка задачі, ідея і методологія проведення досліджень.

Апробація результатів дисертації. Результати дисертації доповідались та обговорювались на:

- Науково-методичниму семінарі аспірантів ННІ комп’ютерних наук та штучного інтелекту Харківського національного університету імені В. Н. Каразіна, 14 лютого 2024 року, Харків, Україна.
- XII Міжнародній науково-практичній конференції «Актуальні проблеми сучасної науки та освіти», 29-30 серпня 2024 р., Львів, Україна.
- 10th International scientific and practical conference “Science and society: modern trends in a changing world”, September 2-4, 2024, Vienna, Austria (Online).

- XIII Міжнародній науково-практичній конференції «Актуальні питання розвитку науки та освіти», 29-30 жовтня 2024 року, Львів, Україна.
- X Науково-технічній міжнародній конференції «Комп'ютерне моделювання у наукоємних технологіях (КМНТ –2024)», 27-29 листопада 2024 року, Харків, Україна.

Публікації. Основні наукові результати дисертаційної роботи у повній мірі викладено в 5 публікаціях, з яких: 4 статті опубліковано в фахових виданнях України, 1 з них опубліковано англійською мовою та проіндексовано в наукометричній базі SCOPUS у журналі Radioelectronic and Computer Systems, 2025, no. 1(113), Kharkiv, Ukraine, ISSN 1814-4225 (print), ISSN 2663-2012 (online).

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, п'яти розділів, загальних висновків та списку використаних літературних джерел, який містить 98 найменувань. Загальний обсяг дисертаційних досліджень викладено на 170 сторінках друкованого тексту, де обсяг основного тексту – 138 сторінок. Дисертація включає 29 рисунків, 21 таблицю та 1 додаток.

РОЗДІЛ 1. ОГЛЯД СТАНУ ПРОБЛЕМИ І ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз процесів моделювання в контексті управління роботизованими пристроями у мультиагентних системах.

Сьогодні моделювання переміщення роботизованих систем є однією з ключових сфер автоматизації та застосування алгоритмічних рішень. З розвитком технологій автономні пристрої все частіше використовуються для виконання складних завдань у динамічних середовищах. Від автономних транспортних засобів до дронів, що здійснюють розвідку, як загадно у роботах [12, 13] чи рятувальні операції з роботи [14], існує зростаюча потреба у точних та ефективних алгоритмах для планування руху, уникнення зіткнень і прийняття оптимальних рішень у режимі реального часу. У цьому контексті особливу роль відіграють алгоритми керування, які забезпечують адаптацію до змінних умов середовища та ефективну взаємодію між компонентами системи, як зазначає робота [15].

Моделювання переміщення роботизованих систем є актуальним завданням, оскільки воно дозволяє розвивати методи адаптації до мінливих умов середовища. Класичні алгоритми пошуку, наприклад A* чи алгоритми на основі дерев рішень, що досліджені у роботі [3], підлягають вирішенню задач актуальних у системах навігації та керування. Вони можуть бути ефективними для визначення оптимального маршруту або прийняття рішень у статичних середовищах, проте мають обмеження у динамічних умовах, що вимагають швидкої адаптації.

Наприклад, у сфері логістики автономні роботизовані засоби використовують алгоритми пошуку для вибору оптимального маршруту, враховуючи трафік і мінливі умови середовища, як зазначається у статті [16]. У рятувальних операціях дрони можуть швидко адаптуватися до складних умов,

знаходячи найкоротший шлях до постраждалих у зонах катастроф, що висвітлено у роботі [14].

Важливим аспектом напрямку є інтеграція методів пошуку та оптимізації з технологіями автономного навчання, вирішенню цих задач присвячена робота [4]. Цей підхід дозволяє агентам вивчати оптимальні стратегії через взаємодію із середовищем. Наприклад, як зазначається у роботі [4], у сценаріях координації груп дронів методи пошуку допомагають узгоджувати дії окремих агентів для досягнення спільної мети.

Можливість ефективного моделювання процесу переміщення та взаємодії роботизованих систем для підбору і подальшого використання алгоритмів та підходів керування відкриває нові перспективи в багатьох галузях. Це стосується не лише автоматизації, але й інших сфер, таких як медицина, де роботизовані хірургічні системи потребують високої точності та адаптивності, як у роботі [17], навігація в міській місцевості, наприклад задачі з роботи [18] або військова справа, де автономні системи можуть виконувати завдання в умовах ризику, що розглянуто у роботах [19, 20].

Розвиток методів моделювання переміщення роботизованих систем є важливим кроком до створення більш безпечних, ефективних і універсальних технологій, які матимуть вплив на багато аспектів сучасного життя. Завдяки використанню алгоритмів керування, оптимізації та пошуку, такі системи стають дедалі більш адаптивними та ефективними, відкриваючи нові горизонти для їхнього застосування в різних сферах, що розглядається детально у роботі [15].

Швидкий розвиток безпілотних літальних апаратів (БПЛА) змінив численні галузі, починаючи від логістики та реагування на стихійні лиха до спостереження та сільського господарства. БПЛА все частіше використовуються через їх здатність працювати автономно, навіть у складних середовищах із перешкодами, забезпечуючи такі програми, як надійна передача даних і зв’язок у важкодоступних місцях, автори робіт [21, 22] присвятили цьому окрему увагу.

Однак ефективні стратегії координації та навігації для команд БПЛА залишаються критичними проблемами, особливо в динамічному та змагальному середовищі, де потрібні швидке прийняття рішень та здатність до адаптації, ці проблеми рзглянуті в роботі [23]. Крім того, нещодавні досягнення в технології створення роїв БПЛА підкреслюють важливість високошвидкісних каналів зв'язку та гнучких стратегій керування для покращення спільного прийняття рішень та координації між БПЛА, особливо в густонаселених міських середовищах, де ефективна автономна робота підвищує надійність і скорочує час місії, як зазначається у дослідженні [24].

У цьому контексті методи багатоагентного підсилювального навчання (MARL), такі як багатоагентний глибокий детермінований градієнт політики (MADDPG), набули популярності завдяки своєму потенціалу для вирішення складних проблем координації, як зазначено у роботі [4] і цільового відстеження, згідно з роботою [25], через використання методів спільного навчання.

Застосування рішень MARL, в тому числі і MADDPG [5], для управління БПЛА продемонструвало багатообіцяючі результати в таких сферах, як оптимізація шляху, що висвітлено у роботах [6, 26], уникнення зіткнень, як зазначається в роботах [4, 28] і розподіл ресурсів, відповідно до робіт [28, 29]. Ці методи дозволяють агентам діяти з певним ступенем автономності, враховуючи як індивідуальні цілі, так і цілі команди, важливі для завдань, таких як місії спостереження згідно з матеріалами робіт [12, 13] або операції з надання допомоги при стихійних лихах, що досліджувалося у роботі [14]. Незважаючи на ці досягнення, зберігаються проблеми в оцінці підходів MARL в середовищах, які точно імітують сценарії реального світу, особливо ті, що включають суперницьких агентів або непередбачувану динаміку, як відзначаються у роботі [30].

Спеціалізовані програмні рішення, такі як Gazebo, Webots, MORSE та CoppeliaSim, що існують для моделювання процесу переміщення роботизованих

систем, забезпечують високу точність фізичного моделювання та широко використовуються в дослідженнях робототехніки, їх детальний огляд зроблено у роботі [1]. Однак складність цих засобів у налаштуванні та інтеграції може ускладнювати тестування алгоритмів алгоритмів керуавння, особливо в умовах коли пріорітетною є швидкість та проведення великої кількості експерментів. З огляду на цю проблему, виникає необхідність розробки альтернативного підходу, який дозволить спростити процес налаштування середовища, інтеграції різних варіацій методів керування та пришвидшити текстування, надавши можливість проводити велику кількість експериментів за менші проміжки часу. Тому в рамках дисертаційного дослідження запропоновано використання двовимірного ігрового середовища як бази для розробки інформаційної технології тестування алгоритмів керування роботизованими пристроями. Таке средовище забезпечує контрольовані умови для тестування та вдосконалення алгоритмів керування, значно знижуючи витрати, пов’язані з експериментами на реальних роботизованих пристроях. Зазначена платформа дозволяє відтворювати складні сценарії, що включають динамічні зміни умов, обмеження руху, необхідність ухвалення стратегічних рішень і врахування багатьох змінних.

Ця платформа особливо корисна для дослідження кооперативної та конкурентної поведінки агентів у мультиагентних системах. Вона дає змогу тестувати здатність алгоритмів адаптуватися до непередбачуваних ситуацій, координувати дії з іншими агентами та ефективно працювати у змінних середовищах. У реальних задачах, таких як управління транспортними потоками або координація дронів, подібні навички є ключовими для досягнення успіху, як зазначено роботах [16, 18].

Окрім того, ігрові середовища створюють простір для інтеграції різних підходів, таких як поєднання нейронних мереж із класичними алгоритмами планування. Це дозволяє вирішувати завдання, які потребують як високого рівня абстракції, так і точних розрахунків. У результаті дослідники отримують

універсальний інструмент для перевірки нових ідей і адаптації їх до реальних умов.

Гра Pac-Man, широко відома своєю складною динамікою навігації, пропонує унікальний тестовий стенд для вивчення стратегій координації роботизованих систем в спрощеному, але репрезентативному середовищі, подібним чином його застосували у дослідженнях [2, 31]. Використовуючи це середовище з певними модіфікаціями, ми можемо імітувати сценарії, коли автономні агенти повинні орієнтуватися, ухилятися від супротивників і досягати цілей місії — аналогічно застосуванням реального світу, згаданим раніше.

Платформа для тестування методів керування роботизованими пристроями на базі модифікованого середовища Pac-Man, запропонована в цьому дослідженні, виступає рішенням, що допоможе впоратися з вищеописаними викликами та сприяти таким чином розвитку зазначених напрямків. Вона представляє новий підхід до моделювання взаємодії автономних агентів у складних динамічних середовищах, здатний до адаптації під різні сценарії за допомгою налаштувань 2D середовища, а також розширення фунекціоналу, слугуючи у ролі фреймворку для подальших розробок. У цій конфігурації ролі агентів були переосмислені: Pac-Man виступає як автономний роботизований пристрій, що виконує завдання навігації та прийняття рішень у змінних умовах, зокрема орієнтується в середовищі для досягнення заданих цілей (наприклад, збирання капсул як контрольних точок). Водночас Привиди представляють супротивні сили або перешкоди, які створюють змінне середовище та ускладнюють виконання місії.

Такий підхід дозволяє як тестувати ефективність класичних алгоритмів пошуку шляху для навігації, такими як пошук у ширину (BFS) і A* та алгоритмів з евристичними стратегіями, так і порівнювати їх у протидії моделям нейронних мереж, наприклад, навченими методами багатокористувацького навчання з підкріпленням (MARL), таких як MADDPG, або іншим класичним алгоритмам,

як правило-орієнтовані методи переслідуваннями, що використовуються привидами з оригінальної гри, ці алгоритми описані в статтях [32, 33].

1.2 Аналіз сучаних підходів керування та взаємодії агентів у контексті завдань роботизованих систем

Розробка та тестування алгоритмів навігації та взаємодії агентів у динамічних середовищах потребують використання відповідних симуляційних платформ. Серед найпоширеніших середовищ, що застосовуються для досліджень у сфері робототехніки, можна виділити Gazebo, Webots, MORSE та CoppeliaSim, розглянутих у роботі [1]. Ці платформи широко використовуються для моделювання роботизованих систем, забезпечуючи можливість високоточного відтворення фізичних процесів та взаємодії агентів із середовищем. Проте, незважаючи на свою функціональність, такі середовища мають певні обмеження, зокрема складність налаштування та недостатню адаптивність для абстрагування місій із навігаційними та змагальними елементами, особливо що стосується військового контексту, як розмінування, розвідка або виконання інших місій.

Альтернативним підходом є використання 2D середовищ, які спрощують налаштування, надають гнучкість у налаштуванні контексту та дозволяють швидко тестувати алгоритми у змодельованих сценаріях із чіткими критеріями оцінювання, на великій кількості ітерацій. Одним із найпоширеніших середовищ такого типу є середовище гри Pac-Man, яке активно застосовується для дослідження пошукових та мультиагентних алгоритмів у контексті гри, а отже має потенціал при модифікації бути застосованим для перевірки алгоритмів ухилення, переслідування, навігації та багатокомпонентної взаємодії агентів. Завдяки можливості регулювання складності та визначенню метрик ефективності, ця платформа має потенціал надати дослідникам та розробникам

зручний інструмент для тестування традиційних алгоритмів керування та методів машинного навчання.

Гра Pac-Man, спочатку розроблена як розважальна погоня, стала цінною платформою для вивчення навігації та прийняття рішень у контрольованому середовищі, таким чином вона була використана у дослідженях [2, 3, 31]. У грі представлено динамічне сіткове середовище, де агент проходить складними лабіринтами, збалансовуючи такі цілі, як захоплення цілі та ухилення від ворогів. Ці атрибути роблять Pac-Man ефективною абстракцією для завдань навігації в реальному світі, де БПЛА може знадобитися перетинати міське середовище, уникати перешкод і виконувати цілі місії за часових обмежень, як це показано у роботі [34, 35].

Дослідження [2, 3] показують, як Pac-Man можна використовувати як інструмент комп'ютерного моделювання для тестування та оцінки навігаційних алгоритмів. Його структурована сітка імітує навігаційні виклики в реальному світі, а його змагальна динаміка дає можливість досліджувати алгоритми в конкурентних умовах. Наприклад у роботі [3], гра була використана для тестування стратегій пошуку оптимального шляху та навчання з підкріпленням, дозволяючи дослідникам оцінювати компроміси між розвідкою та експлуатацією під час прийняття рішень.

Алгоритм пошуку в ширину (BFS), класичний метод обходу графа, широко відомий завдяки своїй простоті та оптимальності в ненавантажених середовищах. BFS працює, досліджуючи всі можливі шляхи від заданого вузла рівень за рівнем, гарантуючи, що знайдено найкоротший шлях до цілі, як зазначають в дослідженні [36]. У контексті навігації БПЛА BFS використовується для завдань, що вимагають вичерпного пошуку в структурованих середовищах, таких як планування шляху на основі сітки або виявлення перешкод, наприклад у роботах [37, 38].

Незважаючи на те, що обчислювально дорогий у великомасштабних або високорозмірних налаштуваннях, BFS залишається одним з класичних еталонів для порівняння більш складних методів пошуку шляху. У середовищі Pac-Man BFS часто використовується для моделювання детермінованих навігаційних стратегій, дозволяючи дослідникам оцінювати його продуктивність відносно змагальної динаміки та інших алгоритмів, наприклад, як це зробили у дослідженнях [2, 3], використовуючи гру Pac-Man як платформу для комп'ютерного моделювання.

Алгоритм A*, розширення BFS, містить евристику для оптимізації ефективності пошуку. Поєднуючи фактичну вартість досягнення вузла з оціночною ціною, A* досягає чудової продуктивності в середовищах, де обчислювальна ефективність є критичною, що обгрунтовано роботою [39]. Це робить A* особливо придатним для завдань пошуку шляху БПЛА, де рішення в реальному часі важливі для уникнення зіткнень і досягнення визначених маршрутних точок, згідно з роботою [40].

У застосуваннях БПЛА A* часто використовується для планування оптимальних шляхів у середовищах, багатих перешкодами, включаючи міські місцевості та зони лиха, наприклад, цьому питанню присвячені роботи [41, 42]. Адаптивність алгоритму до різноманітних евристик ще більше покращує його застосовність до сценаріїв, які вимагають як точності, так і гнучкості.

У складніших сценаріях важливу роль відіграють стратегічні алгоритми прийняття рішень, такі як Expectimax, Alpha Beta Pruning та Monte Carlo Tree Search (MCTS). Вони дозволяють агентам прогнозувати наслідки своїх дій, що висвітлено у роботах [43, 44], цей аспект особливо актуальний у конкурентних середовищах, таких як контекст місій роботизованих систем або БПЛА, де супротивники можуть змінювати стратегію поведінки в реальному часі, наприклад, динамічні зміни, розглянуті у роботі [42].

Monte Carlo Tree Search (MCTS) є одним із найбільш перспективних алгоритмів для розв'язання завдань у симуляційних середовищах. Алгоритм показав здатність враховувати багатофакторні обмеження та приймати оптимальні рішення за мінімальний час у задачах управління залізничним рухом, демонструючи можливість використання у реальному часі, що обгрунтовують у роботі [45]. Самонавчальна версія MCTS (SL-MCTS), що є однією із модіфікацій, забезпечує швидше виконання задач та покращену ефективність у плануванні шляху, використовуючи нейронну мережу для оптимізації пошукового процесу. Ця модифікація значно покращує ефективність роботи алгоритму в умовах обмеженого часу, як зазначають автори роботи [46].

Але також існує і необхідність адаптації MCTS до складних середовищ, наголошуючи на важливості інтеграції проблемно-орієнтованих модифікацій автори виклали свої пропозиції у роботі [47], зокрема розглядаються гібридні підходи для складних ігор із високим розгалуженням та реальних застосувань у транспорті.

Alpha-Beta Pruning також привертає увагу дослідників. Наприклад, показуючи свою ефективність при покращенні продуктивності NPC у грі Triple Triad значно збільшуючи шанси на перемогу в багатокористувацьких середовищах, що висвітлено у роботі [48]. Також, згідно з матеріалами ролботи [49], альфа-бета відсікання суттєво оптимізує час розрахунків у задачах із великою кількістю можливих рішень, особливо в іграх із двома гравцями. Важлливо також враховувати обмеження Minimax і Alpha-Beta Pruning, зазначені у роботі [50], у випадках, коли протилежні агенти не діють оптимально, що вимагає адаптації цих алгоритмів до середовищ із невизначеністю. Це підтверджує здатність алгоритму покращувати ефективність прийняття рішень у складних середовищах із великою кількістю стратегічних факторів, що робить його перспективним для досліджень щодо застосування у керуванні роботизованими системами.

Системи штучного інтелекту досягають людського рівня у виконанні складних завдань, таких як розпізнавання зображень, обробка природної мови та ігри, що підтверджується оглядом у роботі [51]. Однак більшість систем штучного інтелекту сьогодні базуються на статистичних методах машинного навчання, які вивчають закономірності на основі великих обсягів даних, як зазначається у роботі [52].

Застосування методів глибокого навчання та навчання з підкріпленням надає можливість обробляти великі об’єми даних, згідно тверджень роботи [52], що при інтеграцію у керування агентами, дозволяє їм адаптувати свою поведінку на основі накопиченого досвіду та взаємодії із середовищем. Це відкриває можливість не тільки моделювати складні стратегії, а й оптимізувати рішення агентів відповідно до змінних умов середовища. Окрім базового використання нейронних мереж для реалізації соло-агентів, на сьогоднішній день широко використовуються мультиагентні мережі, в яких агенти не працюють окремо, а взаємодіють між собою, наприклад розглянутий підхід у роботі [5].

Multi-Agent Deep Deterministic Policy Gradient (MADDPG) є розширенням алгоритму Deep Deterministic Policy Gradient (DDPG) для середовищ з багатьма агентами. Цей підхід поєднує централізоване навчання з децентралізованим виконанням, дозволяючи кожному агенту приймати рішення на основі власних спостережень, використовуючи при цьому інформацію про дії інших агентів під час навчання, що сприяє ефективній координації та взаємодії між агентами в складних середовищах, що розглядається авторами статті [5].

У змагальних ситуаціях централізоване навчання MADDPG дозволяє агентам вчитися на спільному досвіді, підвищуючи їх здатність передбачати та протидіяти стратегіям опонентів, як зазначається у роботі [5]. В рамках розробленої інформаційної технології модель, навчена за допомогою MADDPG, використовується для контролю агентів суперників у середовищі, з метою

длслідження його потенціалу для реальних застосувань у роботизованих систамах або БПЛА, які потребують координації та адаптації.

Системне моделювання - це процес створення абстрактних уявлень систем для аналізу їхньої поведінки та функціонування, виявлення потенційних проблем та оптимізації їхньої архітектури та реалізації. Воно включає в себе різні методи, такі як комп'ютерне моделювання, математичне моделювання, імітація та оптимізація, які дозволяють дослідникам вивчати взаємодію між різними компонентами системи та прогнозувати їхню поведінку в різних сценаріях. Системне моделювання є важливою складовою напрямку роботизованих систем, як приклд виступають роботи [53, 54], тому в рамках розробки інформаційної системи комп'ютерне моделювання виступає як фундаментальний метод, що дозволяє симулювати взаємодії між агентами та середовищем.

Пропонується розглянути два основних підходи до спілкування автономного програмного агента з ігровим середовищем: пряма та опосередкована взаємодія.

Під час прямої взаємодії (рис. 1.1) програмний агент зазвичай спілкується з грою через програмні інтерфейси, які він підтримує. З міркувань безпеки та авторських прав не всі програми підтримують цю функцію. Інший варіант прямої взаємодії - інтеграція програмного забезпечення в код шляхом маніпуляцій з кодом гри.

Рис. 1.1 – Безпосередня взаємодія автономного програмного агента з грою

Непряма взаємодія (рис. 1.2) – цей підхід, при якому програмний агент взаємодіє з середовищем, в якому грають в ігри, отримує інформацію від середовища і через неї впливає на гру. Агенти можуть розташовуватися як всередині, так і поза середовищем. Інтерфейси програмування та стандартизовані інструменти походять із середовища і не є специфічними для гри. Багато існуючих емуляторів для різних операційних систем можуть бути використані як середовище. Основним недоліком непрямої взаємодії є неможливість використання спеціалізованих засобів керування грою.

Перевагою є можливість побудувати універсальну технологію для автоматизованого управління агентами, незалежно від того, чи надають вони стандартні механізми, чи ні.

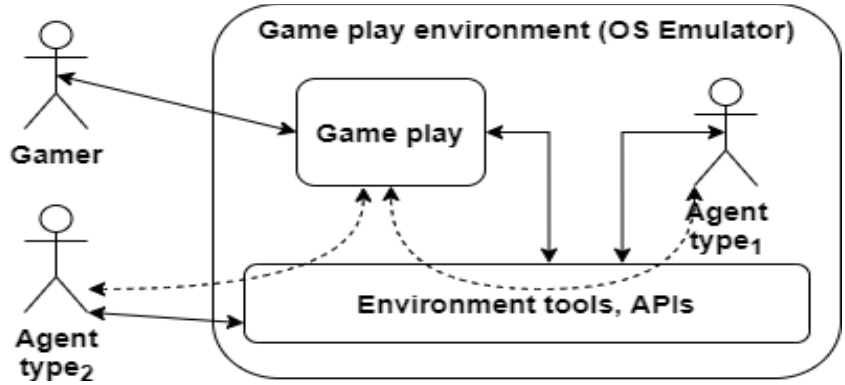


Рис. 1.2 – Непряма взаємодія автономного програмного агента з грою

1.3 Постановка задач дисертаційного дослідження

З урахуванням результатів проведеного огляду джерел різних напрямків, який підтвердив високу актуальність проблеми підбору та тестування методів керування роботизованими системами у складних середовищах, постає потреба у розробці 2D-платформи, що дозволить тестувати відповідні методи, відтворюючи реалістичні сценарії взаємодії агентів, в тому числі з подоланням перешкод, завдяки використанню комп'ютерного моделювання у двовимірному середовищі, що сприятиме проведенню великої кількості експериментів, збору

необхідних статистичних показників та подальшому об’єктивному оцінюванню ефективності різних підходів керування під специфічні задачі. Створення такої платформи передбачає розробку та аналіз структурних, інформаційних і функціональних моделей, які забезпечують синтез ефективних рішень у динамічних і антагоністичних умовах, а також використання актуальних методів, таких як пошукові алгоритми, методи навчання з підкріпленям для синтезу моделі нейромережі, збір статистичних даних та методи розробки ПЗ. У ході роботи виділено такі основні кроки для вирішення, наведеної проблеми:

1. **Розробка структурних моделей** для побудови архітектури системи та інтеграції агентів у багатокомпонентне середовище. Це передбачає візуалізацію взаємодії рухомих пристроїв, рельєфу, перешкод і цілей. Такі моделі дозволяють вивчити топологію середовища та ефективність шляхів і стратегій навігації.
2. **Синтез інформаційних моделей** для реалізації процесів прийняття рішень. Інформаційні моделі забезпечують опис алгоритмів сприйняття середовища, обробки отриманих даних і адаптації агентів до змінюваних умов. Це дозволяє аналізувати ефективність передачі інформації та інтеграцію сенсорних даних для покращення результатів.
3. **Розробка функціональних моделей алгоритмів керування**. Основна увага приділена реалізації алгоритмів, що базуються як на традиційних методах (A*, BFS), так і на сучасних підходах навчання з підкріпленням (MADDPG). Функціональні моделі мають визначати взаємодію між компонентами системи, описують послідовність виконання операцій та відображають ключові функції, необхідні для реалізації алгоритмів керування агентами.
4. **Імплементація пошукових алгоритмів з модифікаціями.** Реалізувати функціональні моделі пошукових алгоритмів, таких як A*, BFS, Expectimax, Alpha Beta Prunning та MCTS, із відповідними адаптаціями для

забезпечення координації автономних агентів у контексті виконання місій, виходячи із сучасних наукових досліджень.

5. **Генерація варіацій 2D моделей середовища.** Потрібно розробити різні варіації 2D середовищ різного масштабу та складності, що наближені до реальних умов, для експериментальної оцінки ефективності алгоритмічних рішень.
6. **Експериментальна валідація результатів.** Необхідно провести серію експериментів із тестування реалізованих алгоритмів, зібрати та проаналізувати отримані дані, а також здійснити порівняльний аналіз з результатами інших наукових досліджень для підтвердження коректності роботи системи.
7. **Підтримка нейромережевих моделей.** Треба забезпечити інтеграцію нейромережевих моделей для керування агентами в рамках розробленої системи, що дозволить адаптувати стратегії до динамічних умов середовища.
8. **Синтез тренованої нейромережевої моделі.** Потрібно розробити та натренувати нейромережеву модель на базі MADDPG з використанням MLP, яка буде готова до подальшого тестування в системі.
9. **Проведення експериментального аналізу.** Необхідно виконати порівняльний аналіз отриманих результатів з даними інших досліджень для підтвердження ефективності моделювання та практичної цінності розробленої інформаційної технології.

Результати цього дослідження створюють основу для використання розробленої інформаційної технології в реальних сценаріях, таких як міська навігація, рятувальні операції, розміновування, розвідувальні завдання, а також у напрямках робототехніки й автономних систем. Це дозволяє не лише оптимізувати процеси керування, але й відкриває нові можливості для їх масштабування та інтеграції в сучасні автоматизовані платформи.

Висновки до розділу 1

Використання 2D платформ, таких як модифіковане ігрове середовиище гри Pac-Man, як полігону для розробки та тестування ефективності взаємодії алгоритмів і підходів керування роботизованими системами дозволяє досліджувати складні сценарії переслідування, ухилення та взаємодії між агентами при певних налаштуваннях.

У рамках цієї роботи гра Pac-Man розглядається як платформа для тестування алгоритмів керування роботизованими агентами у багатокомпонентному середовищі. Ігрові агенти виконують функцію мобільних роботів, які повинні координувати свої дії в умовах обмеженого простору, де вони взаємодіють між собою та з ігровими об'єктами. Завдання ухилення від супротивників та досягнення цілей моделює типові виклики, які зустрічаються у реальних застосуваннях роботизованих систем та БПЛА.

На підставі проведеного аналізу можна зробити наступні висновки:

- Розвиток алгоритмів навігації та керування автономними роботизованими системами є надзвичайно актуальним завданням, оскільки сучасні технології вимагають адаптивних, ефективних та безпечних рішень для роботи в динамічних умовах.
- Класичні алгоритми пошуку, такі як A* і BFS, мають обмеження при роботі в змінних середовищах, що стимулює пошук нових підходів, зокрема на основі методів навчання з підкріпленням та мультиагентних систем (MARL).
- Ігрові середовища, зокрема модифікована гра Pac-Man, демонструють свій потенціал як платформа для тестування та валідації алгоритмів навігації та координації, дозволяючи моделювати складні сценарії з динамічною поведінкою агентів.

- Вибір адекватних симуляційних платформ є критичним для перевірки ефективності алгоритмів, проте існуючі системи часто мають складнощі в налаштуванні та інтеграції, що підсилює актуальність альтернативних підходів із застосуванням ігрових середовищ.

Таким чином, проведений огляд підтверджує необхідність розробки нових інформаційних технологій та методів моделювання, які дозволять підвищити ефективність управління автономними агентами в складних динамічних середовищах. Розроблена інформаційна технологія демонструє приклад інтеграції алгоритмів керування та неромережевих моделей у сценарії багатоагентної взаємодії у віртуальному середовищі. Такий підхід має перспективи для подальшого застосування в роботизованих системах при розробці нових стратегій та підходів, або оптимізації існуючих.

РОЗДІЛ 2 РОЗРОБКА МОДЕЛЕЙ ПЛАТФОРМИ ДЛЯ ТЕСТУВАННЯ АЛГОРИТМІВ КЕРУВАННЯ У МУЛЬТИАГЕНТНИХ СИСТЕМАХ

2.1 Концептуалізація взаємодії агентів та їх ролей у системі

Гра Pac-Man може бути розглянута як спрощена симуляція задач управління автономними системами, де агенти мають виконувати певні завдання у двовимірному просторі, стикаючись із рухомими перешкодами. У цьому контексті лабіринт у грі може бути інтерпретований як 2D-модель місцевості, де автономні системи, такі як дрони або наземні роботи, повинні оптимізувати свої шляхи, зокрема, в обмеженому просторі з перешкодами. Гра Ms. Pac-Man є відмінною тестовою задачею у проблемі переслідування-втечі з кількома активними суперниками, які адаптують свої стратегії переслідування на основі стану та рішень Pac-Man, це більш детально розглядається у роботі [55]. Науковий інтерес до цієї теми зростає, і численні напрями досліджень активно розвиваються, включаючи роботи в галузях робототехніки, біології, соціології та психології. Найактивнішою є область обчислювального інтелекту, не в останню чергу через популярні академічні дослідження по грі Pac-Man, розглянуті у роботі [56].

Pac-Man виконує роль автономного агента, чия мета полягає у виконанні критичних завдань — збиранні капсул, що можна порівняти з реальними операціями дронів, які збирають дані або доставляють ресурси до певних точок. Капсули можна розглядати як цілі, розташовані на місцевості, що потребують виконання конкретних дій, таких як збір даних, обслуговування або моніторинг певних ділянок. Агент визначається як проста скінченна автоматна машина з набором правил, які регулюють ймовірність руху агента з урахуванням обмежень лабіринту в певний момент часу, що пояснено у роботі [57]. Завдяки обчисленню шляхів в режимі реального часу, агент може швидко адаптуватися до

неочікуваних рухів супротивників або динамічного середовища, наприклад, як у дослідженні [58].

Привиди, в свою чергу, представляють динамічні загрози або ворожі агенти, як-от інші дрони або рухомі об'єкти, які можуть перешкоджати виконанню місії. Взаємодія з такими перешкодами вимагає від автономного агента постійного ухилення та коригування свого шляху в реальному часі. У реальних сценаріях це може включати адаптацію алгоритмів для уникнення перехоплення або зіткнення з ворожими чи іншими рухомими об'єктами. Роботи [59, 2, 3], що досліджували ефективность алгоритмів ухвалення рішень у складних середовищах підтверджують, що продуктивність систем керування автономними агентами суттєво залежить від складності оточення та кількості взаємодіючих об’єктів. У простих сценаріях класичні методи пошуку забезпечують ефективність, проте із зростанням рівня динамічності середовища найкращих результатів досягають алгоритми, що комбінують стохастичні методи з навчанням з підкріпленням. Це підтверджує необхідність адаптивного підходу до вибору алгоритмів залежно від умов системи.

Отже, гра Pac-Man надає чудову платформу для моделювання і тестування алгоритмів, які можна застосовувати в реальних автономних системах для виконання завдань у складних умовах із динамічно змінними перешкодами. Взаємодія автономних агентів із перешкодами та виконання критичних завдань вимагають від алгоритмів оптимізації шляху та адаптації до змін середовища, що безпосередньо відповідає реальним викликам у сфері робототехніки та автономних систем. Наприклад, у дослідженні [60] обговорюється, що зв'язані методи розглядають агентів як складну систему з високою розмірністю та використовують повні оптимальні планувальники (наприклад, A* та його варіанти). Однак наївний підхід швидко стає непрактичним через експоненційну складність, тому пропонуються конкретні субоптимальні алгоритми з поліноміальною складністю для обчислення фактичного шляху в важливих

підзадачах загальної проблеми, які використовують примітиви одного агента, але повертають послідовні шляхи, де лише один агент рухається в один момент часу.

Застосування алгоритму Multi-Agent Deep Deterministic Policy Gradient (MADDPG) для навчання моделей, які будуть керувати агентами, дозволяє моделювати складні сценарії взаємодії між агентами, як зазначають автори роботи [5], зокрема це можна інтегрувати у кооперацію привидів для ефективного переслідування Pac-Man'a. У цьому контексті кожен привид розглядається як окремий агент, який навчається оптимізувати свою стратегію з урахуванням дій інших привидів та поведінки Pac-Man'a. Використання MADDPG сприяє розвитку кооперативної поведінки серед команди агентів, що підвищує їхню ефективність у досягненні спільної мети — зловити опонента,, представленого як Pac-Man. Цей підхід демонструє потенціал алгоритмів багатокористувацького навчання з підкріпленням у складних динамічних середовищах (Див. Розділ 4).

2.2 Синтез інформаційних моделей для реалізації процесів прийняття рішень агентами

У мультиагентних системах для тестування алгоритмів керування роботизованими пристроями необхідно створення узгоджених інформаційних моделей, які описують середовище, агентів та їхню взаємодію. Інформаційні моделі дозволяють стандартизувати представлення даних, формувати єдину структуру взаємодії між об'єктами та ефективно тестувати алгоритми.

Для цього у роботі пропонується наступна інформаційна модель мультиагентної системи (Див. Рис 2.1), що будується на трьох основних рівнях:

1. Модель середовища (Environment Model)
    - Визначає просторові обмеження, об'єкти та їхнє розташування.
    - Містить карти, структури для представлення стін, їжі, капсул.
    - Фіксує динамічний стан середовища під час взаємодії з агентами.

2. Модель агента (Agent Model)
    - Визначає характеристики агентів, їхні дії та стани.
    - Описує взаємодію агентів із середовищем.
    - Розділяється на Pac-Man, привидів та зовнішні алгоритми навчання.
3. Модель взаємодії (Interaction Model)
    - Визначає логіку прийняття рішень та правила взаємодії агентів.
    - Включає алгоритми планування, оцінки ситуацій та ухвалення рішень.

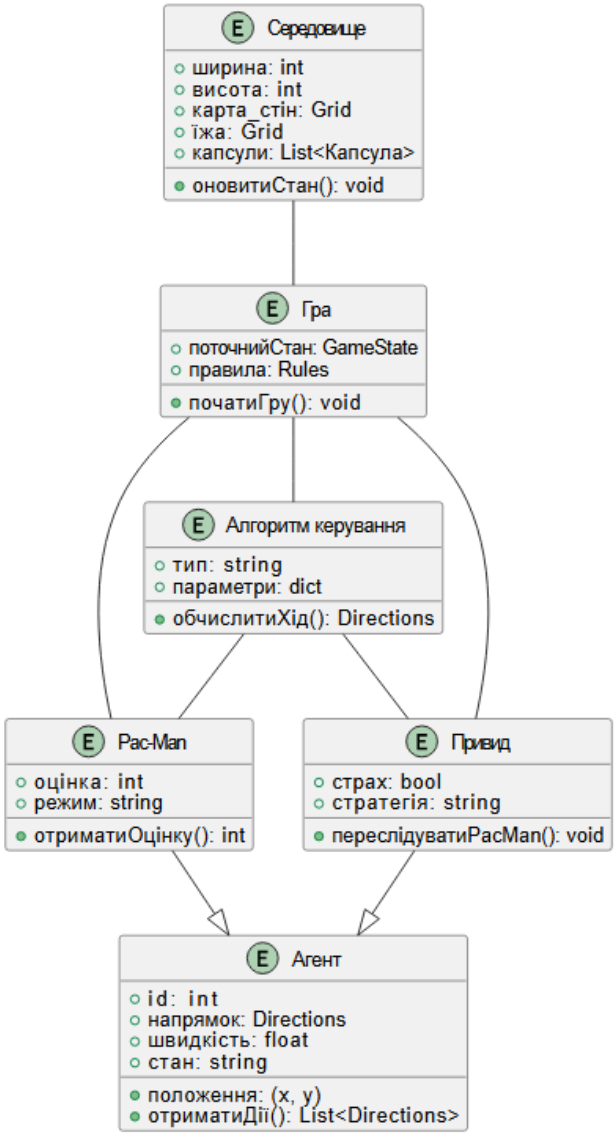


Рис. 2.1 — Запропонована інформаційна модель середовища та взаємодії агентів

1. Середовище (Environment)

- Визначає розташування стін, їжі та капсул.
- Змінюється у процесі гри та оновлює стан після кожної ітерації.
- Дозволяє отримувати стан світу агентами через API.

2. Агент (Agent)
   - Є узагальненим класом для всіх активних учасників гри.
   - Містить координати, напрямок руху, швидкість та можливі дії.
   - Використовується як базовий клас для Pac-Man та привидів.
3. Pac-Man
   - Є основним керованим агентом, що може приймати рішення на основі алгоритму.
   - Має внутрішню оцінку стану, яка використовується у навчанні.
   - Підтримує різні режими гри (агресивний, ухилення, оптимізація балів).
4. Привид (Ghost)
   - Використовує алгоритм прийняття рішень для вибору дій.
   - Має параметр "страх", який змінює поведінку після з’їдання капсули.
   - Використовує різні стратегії (переслідування, блокування, хаотичний рух).
5. Гра (Game)
   - Контролює перебіг симуляції та правила взаємодії агентів.
   - Визначає переможця та оновлює стан кожного агента.
6. Алгоритм керування (Control Algorithm)
   - Використовується для визначення стратегій агентів.
   - Може базуватися на класичних алгоритмах (BFS, A*) або методах навчання.
   - Передає результати розрахунків агентам для виконання дій.

Розроблена інформаційна модель дозволяє формалізувати структуру мультиагентної системи та забезпечує ефективне тестування алгоритмів

керування. Виділені компоненти, такі як середовище, агенти та алгоритми, створюють гнучку архітектуру, що підтримує модифікацію та розширення функціональності.

2.3 Розробка структурних моделей для побудови архітектури системи та інтеграції агентів у багатокомпонентне середовище

Структурне моделювання є основним етапом у розробленні інформаційної технології тестування алгоритмів керування агентами в мультиагентних системах. На основі структурних моделей формалізується архітектура програмного середовища, визначаються ключові компоненти системи, їхні взаємозв’язки та логіка функціонування.

Завдання структурного моделювання включає:

1. Формалізацію представлення середовища, у якому працюють агенти.
2. Побудову моделі агентів, що включає опис їхніх характеристик та механізмів ухвалення рішень.
3. Розробку моделей управління, що визначають алгоритми координації та навчання агентів.

У межах цієї роботи пропонується структурна модель системи, що будується за трьома рівнями:

- Рівень середовища (Environment Level) – моделює фізичні параметри ігрового простору та взаємодію агентів із середовищем (Див. Рис. 2.2).
- Рівень агентів (Agent Level) – визначає класи агентів, їхні характеристики та методи ухвалення рішень (Див. Рис. 2.3).
- Рівень алгоритмів управління (Control Level) – містить алгоритми пошуку, нейромережеві моделі та правилоорієнтовані алгоритми переслідування, включно з фуункціями оцінки для прийняття рішень (Див. Рис. 2.4).

Середовище є основним компонентом, що визначає обмеження для агентів, задає правила гри та відображає змінні параметри простору. Воно містить карту,

структури для представлення стін, їжі, капсул та інших об’єктів, що впливають на ухвалення рішень агентами.

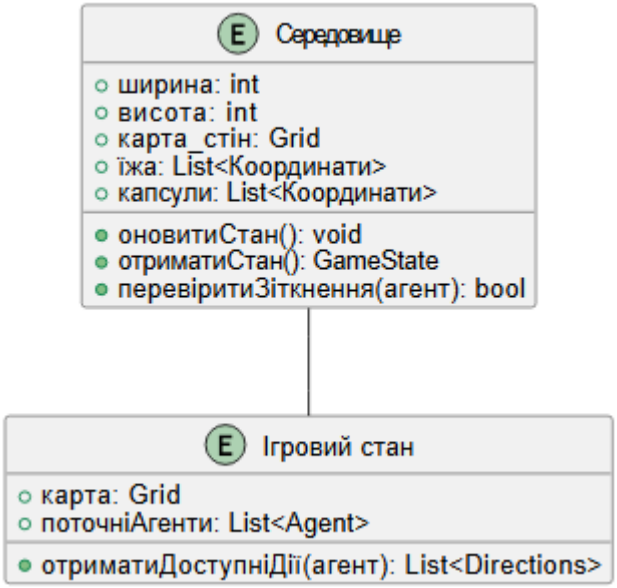


Рис. 2.2 — Структурна модель середовища

Середовище (Environment) моделює простір, у якому працюють агенти, забезпечує їхню взаємодію з об’єктами та зберігає правила фізичного простору.

Ігровий стан (GameState) містить дані про поточні координати агентів, доступні дії та глобальні обмеження середовища.

Оновлення стану відбувається через функцію оновитиСтан(), яка змінює розташування об’єктів у відповідь на дії агентів.

Агенти взаємодіють із середовищем та один з одним, використовуючи різні алгоритми ухвалення рішень. Вони можуть бути Pac-Man або привидами, кожен з яких має свою логіку поведінки.

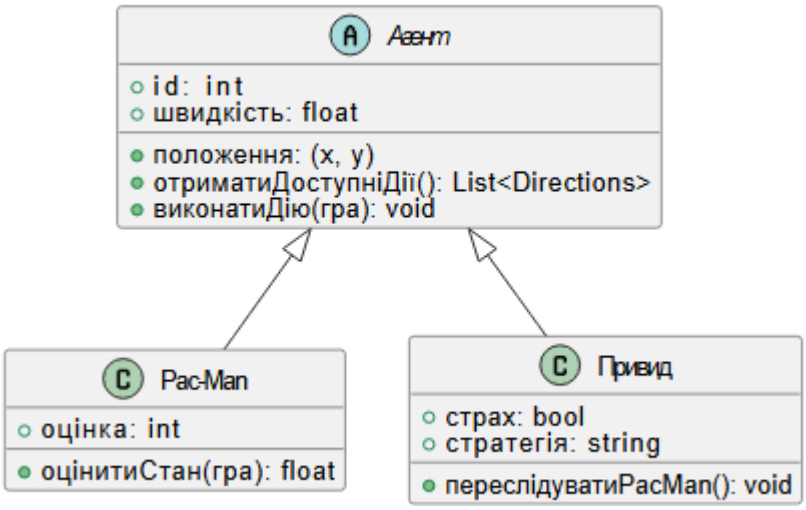


Рис. 2.3 — Структурна модель агента

Pac-Man приймає рішення на основі алгоритмів пошуку або навчання з підкріпленням.

Привиди (Ghost) використовують стратегічні алгоритми ухвалення рішень, такі як Minimax або Expectimax.

Усі агенти отримують доступ до середовища та ухвалюють рішення через функцію отриматиДоступніДії().

Алгоритми керування агентами відіграють ключову роль у тестуванні мультиагентних систем, визначаючи, як агенти ухвалюють рішення, координують свої дії та адаптуються до змін середовища. Представлена структурна модель чітко розмежовує агентів, що використовуються різні алгоритми. Які слугують для керуванням привидами або Pac-Man`ом, відображаючи реальні сценарії їхнього застосування.

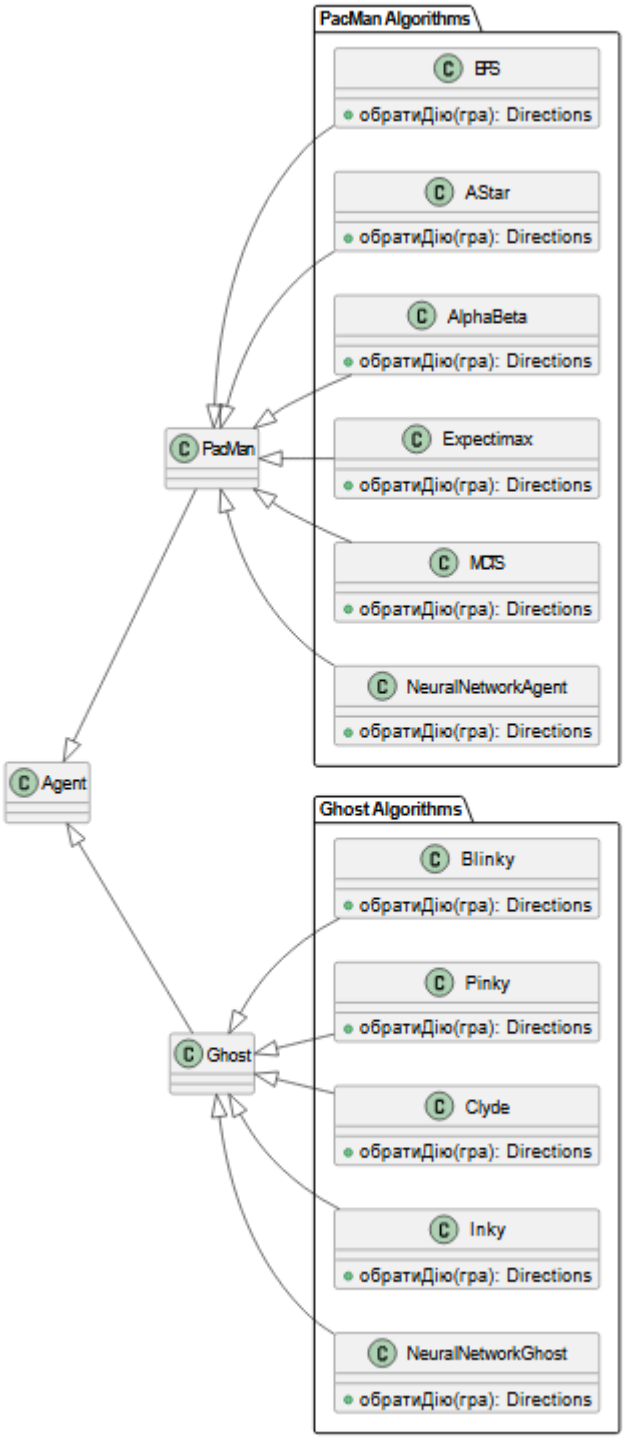


Рис. 2.4 — Структурна модель алгоритмів керування агентами

У системі реалізовано дві основні групи агентів:

1. Pac-Man – автономний агент, що ухвалює рішення самостійно.

2. Привиди (Ghosts) – багатокористувацькі агенти, які можуть діяти індивідуально або координуватися на основі попередньо навченої моделі.

Для кожного типу агентів використовуються свої методи ухвалення рішень.

1. Керування Pac-Man

Pac-Man виконує стратегічні маневри, ухиляючись від привидів та збираючи цілі на карті. Його ухвалення рішень базується на таких підходах:

- Пошукові алгоритми:
  - BFS (Breadth-First Search) – знаходження найкоротшого шляху в середовищі.
  - A* (A-star) – оптимізований алгоритм пошуку з використанням евристики.
- Стратегічні алгоритми:
  - Minimax – стратегія вибору найкращого рішення за умови протидії супротивника.
  - AlphaBeta – вдосконалений Minimax із відсіканням нерелевантних ходів.
  - Expectimax – розширена версія Minimax, що враховує ймовірні події.
- Модель, навчена з використанням глибокого навчання:
  - Neural Network Agent – агент, що приймає рішення на основі моделі, навченої в іншому середовищі.

Привиди (Ghosts) реалізують різні стратегії переслідування Pac-Man, які підлягають розширенню та наповненню під конкретні потреби. Як базові пропонуються наступні методи:

- Оригінальні алгоритми гри:
  - Blinky – класична стратегія активного переслідування.
  - Pinky – передбачення майбутнього положення Pac-Man та перехоплення.

- Inky – комбінування положення Blinky та Pac-Man для оптимізації маршруту.
- Clyde – змінює поведінку залежно від відстані до Pac-Man.

- Алгоритмічні підходи:
  - Monte Carlo Tree Search (MCTS) – прийняття рішень на основі симуляцій для знаходження найкращої дії.
  - MADDPG (Multi-Agent Deep Deterministic Policy Gradient) – модель координації кількох агентів, навчена методом глибокого навчання з підкріпленням, що дозволяє привидам діяти узгоджено для ефективного переслідування Pac-Man.
- Модель нейромережі:
  - Neural Network Ghost – агент, що підтримує керування за допомогою моделі нейромережі, яку пропонується інтегрувати у систему під конкретні потреби.

Для моделювання агентів використовується підхід об’єктно-орієнтованого представлення, де базовий клас Agent розширюється двома підкласами:

- Pac-Man, що реалізує алгоритми навігації, ухилення та пошуку оптимального маршруту.
- Привид (Ghost), що моделює різні стратегії переслідування та координації.

Розроблена модель агентів забезпечує гнучкість у виборі методів керування завдяки використанню пошукових алгоритмів, стратегічних моделей і навченої нейромережі. Це дозволяє адаптувати поведінку Pac-Man і привидів залежно від цільових завдань, створюючи реалістичні сценарії для тестування стратегій ухилення та переслідування.

2.4 Розробка функціональних моделей алгоритмів керування.

Функціональні моделі визначають взаємодію між компонентами системи, описують послідовність виконання операцій та відображають ключові функції,

необхідні для реалізації алгоритмів керування агентами. Ці моделі забезпечують формалізацію логіки роботи системи, що є основою для її подальшої реалізації та тестування.

Запропоновано функціональну модель алгоритмів керування, що охоплює такі основні аспекти:

1. Обробка станів середовища: оновлення станів карти та передача даних агентам.
2. Прийняття рішень агентами: вибір оптимальних дій на основі алгоритмів керування.
3. Виконання дій: зміна стану середовища у відповідь на виконані дії.
4. Оцінка ефективності: аналіз дій агентів і коригування алгоритмів.

Узагальнена функціональна модель (Див. Рис 2.5) відображає взаємодію між середовищем, агентами та алгоритмами.

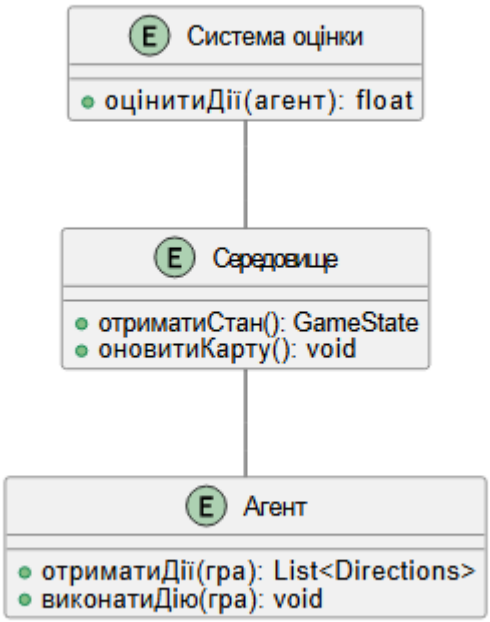


Рис. 2.5 — Узагальнена функціональна модель

Середовище (Environment): забезпечує актуальні дані про стан системи та оновлює карту у відповідь на дії агентів.

Агент (Agent): отримує доступ до середовища, приймає рішення та виконує дії.

Алгоритм керування (ControlAlgorithm): надає логіку вибору дій для агентів.

Система оцінки (Evaluation): оцінює завершеність гри, підраховує кількість очок і результат гри.

Модель обробки станів середовища (Див. Рис 2.6) відображає послідовність дій для оновлення карти та взаємодії з агентами.

Рис. 2.6 — Модель обробки станів середовища

Кожен агент виконує дії відповідно до алгоритму керування (Див. Рис 2.7).

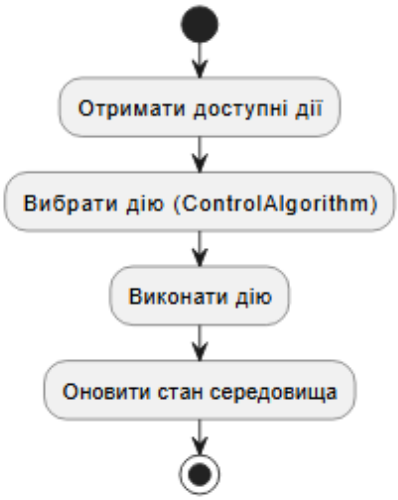


Рис. 2.7 — Функціональна модель роботи агентів

Агенти спочатку отримують доступні дії з середовища.

На основі алгоритму (наприклад, A*, MADDPG) обирається оптимальна дія.

Виконана дія передається назад у середовище для оновлення стану.

Оцінка ефективності (Див. Рис 2.8) є важливим етапом для аналізу продуктивності алгоритмів керування.

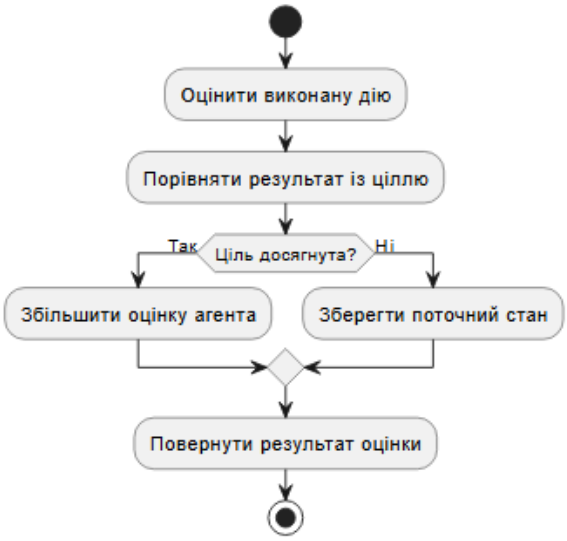


Рис. 2.8 — Функціональна модель оцінки ефективності

Оцінка базується на порівнянні результатів дій із заданою метою.

При досягненні цілі агент отримує позитивну оцінку, яка використовується для адаптації алгоритму.

Функціональна модель тестування алгоритмів керування агентами забезпечує детальне представлення логіки роботи системи. Виділені ключові аспекти, такі як обробка станів середовища, прийняття рішень агентами та оцінка ефективності, створюють основу для реалізації та вдосконалення алгоритмів керування.

Розроблені моделі дозволяють ефективно формалізувати процеси, забезпечуючи узгодженість між компонентами системи та їхню інтеграцію у єдину архітектуру.

Висновки до розділу 2

У другому розділі, присвяченому розробці моделей тестування алгоритмів керування агентами у сценаріях командної взаємодії, було побудовано інформаційні, структурні та функціональні моделей, розглянуто їх ключові аспекти. Основна увага приділялась забезпеченню узгодженості компонентів системи, їхньої інтеграції та оптимізації взаємодії між агентами та середовищем.

Основні висновки включають:

1. Інформаційні моделі: дозволяють формалізувати структуру середовища, визначаючи параметри карти, положення об’єктів та характеристики агентів. Це створює основу для передачі даних між компонентами системи та ефективного моделювання сценаріїв взаємодії.
2. Структурні моделі: забезпечують організацію системи на рівні середовища, агентів та алгоритмів. Розроблена архітектура є гнучкою, модульною та підтримує масштабованість, що дозволяє легко адаптувати систему під різні сценарії.

3. Функціональні моделі: детально описують послідовність взаємодій між компонентами, включаючи обробку станів середовища, вибір дій агентами та оцінку ефективності алгоритмів. Такий підхід сприяє підвищенню ефективності роботи системи та її адаптивності.

Розроблені моделі демонструють значні переваги у забезпеченні узгодженості та інтеграції системи, дозволяючи ефективно тестувати алгоритми керування агентами. Впровадження цих підходів відкриває нові можливості для реалізації складних сценаріїв взаємодії, забезпечуючи високу адаптивність та реалістичність поведінки агентів.

Отримані результати підкреслюють перспективи подальшого використання представлених моделей у розробці мультиагентних систем, орієнтованих на створення інтелектуальних динамічних середовищ, зокрема у сферах ігрових симуляцій, робототехніки та оптимізації процесів.

РОЗДІЛ 3 РОЗРОБКА МЕТОДІВ ЗАБЕЗПЕЧЕННЯ ВЗАЄМОДІЇ МІЖ КОМПОНЕНТАМИ СИСТЕМИ

3.1 Імплементація взаємодії середовища та агентів за допомогою комп'ютерного моделювання

В оригнальній версії гри Pac-Man, про яку розповідає автор у статті [32], основне завдання — зібрати всі точки у лабіринті, уникаючи привидів, які переслідують гравця.

У запропонованій модифікації гри, основна мета інша — зібрати всі капсули на карті, які можна розглядати як цілі або задачі для виконання в реальному середовищі, наприклад, розвідка території або збір даних за допомогою дронів або роботизованих систем. Капсули виступають як критичні об'єкти, які необхідно віднайти і зібрати. У цьому контексті лабіринт стає двовимірною моделлю місцевості, де необхідно оптимізувати шлях для виконання завдання, водночас уникаючи зіткнення з перешкодами або іншими агентами.

З метою наближення симуляційного середовища до реальних умов виконання місій роботизованих систем та забезпечення ефективності процесу комп'ютерного моделювання пропонується розгляд наступних аспектів механіки, як ключових:

1. Лабіринт:
   - має прямокутну форму з фіксованим розміром;
   - містить стін, які визначають шляхи, по яких може пересуватися Pac-Man та привиди;
   - в межах лабіринту розташовані точки (pellets) та капсули, які розглядаються як ключові елементи місій і надають гнучкість налаштування контексту задач, за рахунок пріоритизації одних чи інших;

- налаштовується як верстка у форматі ‘*.lay*’

2. Pac-Man:
   - Pac-Man може рухатися у чотирьох напрямках;
   - Pac-Man тимчасово отримує можливість знищувати привидів після збору капсул;
   - початкова позиція задається у верстці лабіринту;
   - для керування доступні різноманітні алгоритми, підтримується додавання нових;
3. Точки (Pellets):
   - кожна зібрана точка додає певну кількість балів;
   - капсули дають тимчасову здатність полювати на привидів.
4. Привиди:
   - рухаються в чотирьої напрямках;
   - виконують задачі протидії Pac-Man;
   - відновлюються після нейтралізації і починають зі стартової позиції, що задається у верстці лабіринту;
   - можуть бути використані різноманітні алгоритми та моделі нейромереж, підтримується додавання нових;
   - імена привидів в оригінальній грі – Blinky, Pinky, Inky, Clyde [32];
   - при налаштуваннях за замовчуванням використані стратегії поведінки з оригінальної гри, що описані у статтях [32, 33]:
     - Blinky — переслідує Pac-Man;
     - Pinky — намагається перехопити Pac-Man, передбачаючи його маршрут;
     - Inky — комбінує свої дії з іншими привидами, створюючи непередбачувану поведінку;
     - Clyde — рухається випадковим чином.

5. Система очок: кожна точка або нейтралізація привида додає певну кількість внутрішніх балів у Pac-Man, також пропонується штраф за відсутність резуцльтативних дій;

Механіка привидів у ролі опонентів**:**

1. **Привиди як мобільні перешкоди**: Привиди в грі Pac-Man можна розглядати як рухомі перешкоди або вороги в реальному середовищі. Це можуть бути інші автономні системи (дрони, наземні роботи) або навіть агресивні елементи, такі як ворожі дрони у військових сценаріях. Їхня агресивна поведінка, яка полягає у переслідуванні Pac-Man, нагадує динамічні загрози, які необхідно враховувати при плануванні шляху і ухиленні від них.
2. **Зляканий стан**: Уразливість привидів під час зляканого стану може бути інтерпретована як тимчасова слабкість в системі або агенті, яку можна використати для досягнення цілі. Це, наприклад, може стосуватися моментів, коли інші дрони або транспортні засоби стають тимчасово недієздатними через технічні проблеми, і основний агент (Pac-Man) отримує можливість діяти більш агресивно для завершення місії.

Штрафи та нагороди (Підтримується гнучке налаштування)**:**

1. **Штраф за бездіяльність**: У реальних задачах, наприклад, розвідці за допомогою дронів, час є критичним ресурсом. Бездіяльність або затримки під час виконання завдань можуть призвести до втрати ефективності або навіть до провалу місії. Подібно до того, як Pac-Man втрачає очки за кожен зайвий хід, у реальних системах оптимізація часу і енергії є ключовими для успіху.
2. **Нагорода за збирання капсул**: Капсули можна трактувати як критичні точки збору даних або важливі об’єкти, які необхідно ідентифікувати та обробити. Наприклад, дрони можуть бути запрограмовані на збір

інформації з певних точок на карті, розмінування або виконання певних дій в місцях з високою концентрацією об'єктів.

3. **Нагорода за знищення уразливого привида**: Зляканий привид у контексті реальних систем може означати тимчасову вразливість іншого агента або системи. Наприклад, це може бути час, коли рухомий об’єкт (інший дрон чи транспортний засіб) є вразливим через технічну проблему, втрату зв’язку, обмежений запас енергії тощо, що дозволяє основному агенту скористатися моментом для нейтралізації або завершення завдання. Така ситуація може виникати, коли, наприклад, один дрон тимчасово втрачає зв'язок або потужність, що створює можливість для інших агентів швидко діяти.
4. **Збір точок (pellets) як другорядне завдання**: Як і в реальних автономних системах, Pac-Man, крім основних капсул, збирає точки, що приносить менші, але стабільні очки. Це можна порівняти з другорядними завданнями в реальних автономних місіях, де система виконує додаткові операції, що приносять менші, але важливі результати, наприклад, збір другорядних даних або виконання нескладних допоміжних дій, які сприяють загальному успіху місії, наприклад розвідка додаткових шляхів.

Важливим пріоритетом агентів є **оптимізація шляху і часу**: Втрата очок за кожен зайвий хід в грі імітує реальні ситуації, де час і ресурси є обмеженими. Це обґрунтовано тим, що у таких системах, як дрони або автономні роботи, кожен хід має свою вартість — енергетичну або часову. Неефективне планування маршруту може призвести до перевитрати енергії або до невиконання місії в заплановані терміни. Наприклад, автономний дрон, що проводить моніторинг або доставку, повинен постійно оптимізувати свій маршрут, оскільки кожен непотрібний рух призводить до зменшення ресурсу батареї, що може негативно вплинути на виконання завдань. Така ж ситуація й у Pac-Man: кожен

неефективний хід, за який знімаються очки, наближає гравця до поразки, що робить пошук оптимальних маршрутів критично важливим.

Розробка вище зазначених аспектів здійснена за допомогою методології життєвого циклу розробки програмного забезпечення (SDLC), із застосуванням Scrum як основного підходу до організації робочого процесу. Це дозволило розбити роботу на короткі спринти, регулярно переглядати пріоритети та проводити ретроспективи для постійного вдосконалення процесів. Такий підхід сприяв чіткій комунікації з науковим керівником та консультантами по роботі, забезпечував гнучке реагування на зміни вимог та дозволяв систематизувати весь процес розробки, включаючи планування, аналіз вимог, проєктування, реалізацію, тестування та проведення експериментів. Особливу увагу приділено оформленню результатів експериментів у наукові роботи, що сприяє динамічності процесу та поетапній валідації функціонування системи.

Програмні засоби та бібліотеки. Для реалізації середовища використовувалися такі бібліотеки та технології:

- *Python* – основна мова програмування, що забезпечує гнучкість реалізації алгоритмів, є найпоширенішою в академічному сегменті та широко застосовується у розробці штучного інтелекту.
- *Tkinter* – бібліотека для візуалізації середовища, що дозволяє відображати рух агентів, взаємодію між ними та зміну ігрового поля.
- *NumPy* – використовується для обробки матриць та числових розрахунків, необхідних для реалізації алгоритмів пошуку та нейромережевих моделей.
- *Matplotlib* – застосовується для побудови графіків та діаграм.
- *TensorFlow* – використовується для відтворення нейромережевих моделей.
- *OpenAI MADDPG Gym* – окреме середовище, що використовувалось для синтезу моделі нейромережі для кооперативного керування групою агентів за допомогою алгоритму MADDPG [5].

Архітектура симуляційного середовища. Структура інформаційної технології побудована за принципом модульності, що дозволяє легко змінювати параметри алгоритмів, тестувати різні підходи та порівнювати результати.

Основні компоненти системи пропонуються наступні:

1. Модуль ігрової логіки – реалізує правила гри, збереження стану середовища та механіку взаємодії агентів.
2. Модуль середовища – основна імплементація та логіка запуску симуляцій
3. Модуль керування агентами – містить реалізації алгоритмів ухвалення рішень для Pac-Man та привидів.
4. Модуль утиліт - містить утилітні функції та структури даних
5. Модуль збору статистики – відповідальний за логування подій, обчислення ключових метрик та аналіз продуктивності алгоритмів.
6. Модуль візуалізації – забезпечує графічне представлення середовища та агентів у реальному часі.
7. Модуль генерації лабіринтів – надає можливість створювати нові лабіринти під потрібні параметри ширини, висоти та відсотку площі зайнятого стінами лабіринту.
8. Модуль експериментів – містить преналаштовані шаблони для автоматизації циклічного запуску симуляцій під потрібні параметри.

Графіка та візуалізація. Для наочного аналізу взаємодії агентів у середовищі використовується система візуалізації, що підтримує як текстове відображення, яке обґрунтоване необхідністю зменшенням витрат ресурсів за потреби, так і графічне відображння за допомогою 2D візуалізації. Розроблено відображення наступних аспектів:

- Поточне положення агентів та елементів на карті.
- Взаємодію між агентами, включаючи ухилення, переслідування та зіткнення.
- Актуальний рахунок.

Гнучкість системи та розширюваність. Для забезпечення гнучкості системи та можливості розширювати наявні методи та компоненти, з метою використання платформи як фреймворк, запропоновано використання модульної архітектури. Модульна архітектура дозволяє легко додавати нові алгоритми керування, змінювати параметри симуляції та розширювати функціональність середовища. Завдяки використанню гнучких інструментів програмування система може бути адаптована для подальших досліджень та тестування інших підходів до керування агентами в мультиагентних системах.

Основні параметри з наявних у середовищі:

1. *-n, --numGames*: Кількість ігор для гри (за замовчуванням: 1)
2. *-l, --layout*: Файл розкладки лабіринту для використання (за замовчуванням: 'mediumClassic')
3. *-p, --pacman*: тип агента Pacman (за замовчуванням: 'KeyboardAgent' для керування людиною)
4. *-g, --ghosts*: Тип агентів-привидів для використання, підтримує передачу параметрів рядку через кому, для використання одразу різних агентів (за замовчуванням: 'RandomGhost')
5. *-k, --numghosts*: Максимальна кількість привидів у грі (за замовчуванням: 4)
6. *-t, --textGraphics*: Запускати гру у текстовому режимі без графіки
7. *-q, --quietTextGraphics*: Мінімальний режим виведення без графіки
8. *--zoom*: Масштабний коефіцієнт для графічного вікна (за замовчуванням: 1.0)
9. *--frameTime*: Затримка між кадрами у секундах (за замовчуванням: 0.1, від'ємне значення означає керування з клавіатури)
10. *-a, --agentArgs*: Аргументи, що передаються агенту (формат: «opt1=val1,opt2,opt3=val3»)

11. *-m, --model*: Шлях до файлу моделі ШІ для завантаження (за замовчуванням: «Не використовується»)
12. *-r, --recordActions*: Зберігати історію гри у файл
13. *--replay*: Відтворити раніше записаний файл гри
14. *-c, --catchExceptions*: Увімкнути обробку виключень під час гри
15. *-timeout*: Максимальний час обчислень для однієї гри у секундах (за замовчуванням: 30)

Приклад команди запуску симуляцій: *python pacman.py -l smallNonOptimized40w -p AStarCapsulesSearchAgent -g BlinkyGhost,PinkyGhost -k 2 -n 100*

3.2 Інтеграція класичних пошукових алгоритмів для побудови агентом найкоротшого шляху до цільових точок середовища

У класичних підходах до пошуку шляхів, таких як A* та BFS, мета полягає у знаходженні найкоротшого шляху до точок або інших об’єктів на змодельованій території. Алгоритм A* використовує евристику для оцінки відстані до цілі, що дозволяє йому оптимізувати час пошуку, зменшуючи кількість розглянутих вузлів. Цей підхід особливо ефективний в умовах статичних перешкод, таких як стіни лабіринту, проте не враховує динамічні загрози на кшталт привидів. З іншого боку, BFS пропонує інший підхід, забезпечуючи повний пошук усіх можливих шляхів у лабіринті. Однак його недоліком є те, що він може бути менш ефективним у великих лабіринтах через високу вартість пошуку. Хоча класичні пошукові алгоритми є ефективними для статичних середовищ, вони втрачають свою ефективність у динамічних середовищах, таких як гра Pac-Man, де привиди рухаються та створюють змінювані загрози. Як зазначають у роботі [15], багато алгоритмів не мають належного методу адаптації до швидкостей рухомих об'єктів у процесі прийняття рішень щодо навігації. Це особливо помітно в великих і складних лабіринтах, де

класичні алгоритми часто призводять до локально оптимальних, але не глобально ефективних рішень, оскільки не враховують змінну поведінку привидів. Та тим не менш через популярність цих класичних підходів серед дослідників, наприклад у роботах [2, 3] та наявність досліджень щодо застосувань, подібних до робіт [37, 61], використання їх є доцільним задля валідації корректності роботи системи.

Breadth-First Search (BFS) є одним із класичних алгоритмів для пошуку шляхів у графах і широко використовується у завданнях, пов’язаних із навігацією та плануванням. Цей алгоритм забезпечує обхід графа в ширину, гарантуючи знаходження найкоротшого шляху в умовах, де всі ребра графа мають однакову вагу.

Алгоритм BFS базується на ідеї поступового дослідження вузлів графа, починаючи з початкового вузла. Він використовує чергу (FIFO) для обробки вузлів, що дозволяє уникнути рекурсивної глибини та зберегти всі можливі шляхи до моменту їх обробки (Див. Рис 3.1). У контексті ігрового середовища Pac-Man, BFS може бути застосований для планування руху Pac-Man до найближчого капсули або уникнення зіткнень із привидами.

```
BFS(start, goal):
    Створити порожню чергу Q
    Додати стартовий вузол до черги Q
    Встановити стартовий вузол як відвіданий

    поки Q не порожня:
        поточний = видалити перший елемент із Q
        якщо поточний == goal:
            повернути шлях до поточного вузла
```

```
    для кожного сусіда поточного вузла:
        якщо сусід не відвіданий:
            відмітити сусіда як відвіданий
            додати сусіда до черги Q
            зберегти шлях до сусіда

    повернути "Шлях не знайдено"
```

1. Алгоритм починається з початкового вузла, який додається до черги, та його відмічають як відвіданий.
2. На кожній ітерації алгоритм видаляє перший вузол із черги та перевіряє, чи є він цільовим.
3. Якщо ціль досягнута, повертається шлях до цільового вузла.
4. Для кожного сусіднього вузла поточного вузла перевіряється, чи був він відвіданий. Якщо ні, то вузол додається до черги та відмічається як відвіданий.
5. Якщо черга спорожніє, а ціль не буде знайдена, алгоритм повертає повідомлення про відсутність шляху.

Рис. 3.1 — Схема роботи BFS

1. Асимптотична складність:
   - Часова складність: O(V + E), де V — кількість вузлів, E — кількість ребер графа.
   - Просторова складність: O(V), оскільки алгоритм зберігає вузли в черзі та таблиці відвіданих вузлів.

2. Переваги BFS:
    - Гарантоване знаходження найкоротшого шляху в графах із рівномірною вагою ребер.
    - Простота реалізації завдяки використанню черги.
3. Недоліки BFS:
    - Не підходить для графів із великим числом вузлів через високу вимогу до пам’яті.
    - Ефективність знижується у графах із неоднорідною вагою ребер, оскільки не враховує вартість шляху.
4. Застосування в Pac-Man:
    - BFS може бути використаний для швидкого знаходження оптимального шляху до найближчої їжі, або капсули за рахунок преналаштованої функції проблеми пошуку, яка може бути замінена іншими імплементаціями під конкретні потреби.

Алгоритм A* (A-star) є розширенням BFS, що використовує евристичні функції для підвищення ефективності пошуку шляхів. Цей алгоритм комбінує вартість поточного шляху з оцінкою відстані до цільового вузла, що дозволяє знаходити оптимальні маршрути навіть у складних середовищах із неоднорідною вагою ребер (Див. Рис 3.2).

A* є одним із найефективніших алгоритмів для знаходження найкоротшого шляху. Його ключовою перевагою є використання функції оцінки $f(n)=g(n)+h(n)$, де:

- $g(n)$ — вартість шляху від початкового вузла до поточного nnn.
- $h(n)$ — евристична оцінка вартості шляху від вузла nnn до цільового вузла.

У грі Pac-Man цей алгоритм можна використовувати для пошуку оптимальних шляхів до капсул, їжі або уникнення привидів.

A*(start, goal):

```
Створити порожній відкритий список Open
Додати стартовий вузол до Open зі значенням f = h(start)
Створити закритий список Closed

поки Open не порожній:
    поточний = вузол з найменшим значенням f у Open
    якщо поточний == goal:
        повернути шлях до поточного вузла

    Видалити поточний із Open, додати його до Closed

    для кожного сусіда поточного вузла:
        якщо сусід у Closed:
            пропустити
        якщо сусід не в Open:
            обчислити g, h, f для сусіда
            додати сусіда до Open
        інакше:
            якщо новий шлях до сусіда кращий:
                оновити значення g, h, f сусіда

повернути "Шлях не знайдено"
```

1. Алгоритм починається з додавання стартового вузла до списку Open із початковим значенням f=h(start).
2. На кожній ітерації вибирається вузол із найменшим значенням f зі списку Open.
3. Якщо вибраний вузол є цільовим, повертається шлях.

4. Сусіди поточного вузла обробляються для оновлення їхніх значень g, h, f.
5. Алгоритм завершується, якщо список Open спорожніє, а шлях не буде знайдений.

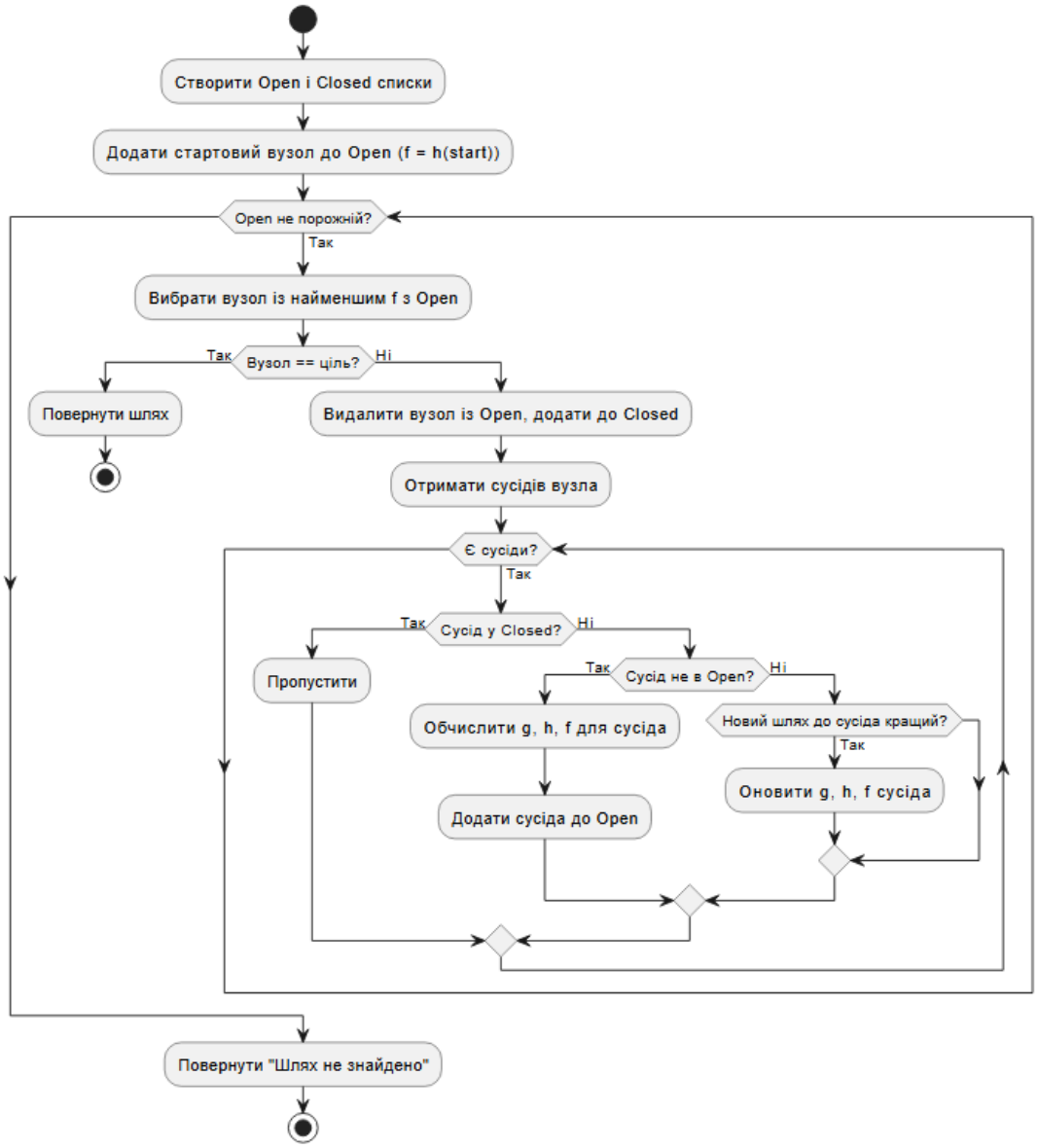


Рис. 3.2 — Схема роботи A*

1. Асимптотична складність:
   - Часова складність: $O(b^d)$, де d — глибина розв'язку (довжина найкоротшого шляху), а b - коефіцієнт розгалуження (максимальна

кількість наступників для стану), оскільки він зберігає в пам'яті всі згенеровані вершини.

- Просторова складність: $O(b^d)$, оскільки потрібно відстежувати кожну вершину графа, навіть ті, які ми ніколи не відвідували і не потрібно відвідувати.

2. Евристичні функції:
   - Використання функції h(n) має вирішальне значення. Для гри Pac-Man часто використовують манхеттенську відстань (Manhattan Distance) як евристику, оскільки вона відповідає дискретній природі ігрової карти.
3. Переваги A*:
   - Гарантоване знаходження оптимального шляху (за умови адекватної евристики).
   - Підвищена ефективність у порівнянні з BFS завдяки зменшенню кількості вузлів для обробки.
4. Недоліки A*:
   - Високі вимоги до пам’яті в умовах великих графів.
   - Чутливість до якості обраної евристичної функції.
5. Застосування в Pac-Man:
   - A* ідеально підходить для задач швидкого планування руху до найближчих капсул або їжі. Також може бути доналаштований шляхом заміни імплементації searchProblem функції, яка може бути замінена іншими імплементаціями під конкретні потреби, без зміни класу агенту.

3.3 Розробка модифікацій алгоритмів пошуку для урахування та уникання загроз в змодельованому середовищі

Для вирішення задач у динамічних середовищах використовуються алгоритми, які здатні моделювати взаємодію з іншими агентами, зокрема привидами. Алгоритм Expectimax дозволяє моделювати ймовірнісні дії агентів супротивників, що дає можливість передбачати можливі загрози та ухилятися від них.

Однак варто зазначити, що моделювання можливих дій усіх привидів у межах Expectimax може значно збільшити обчислювальну складність, особливо в великих лабіринтах, що обмежує застосування цього алгоритму в реальних умовах. Для вирішення цієї проблеми використовується Alpha-Beta Pruning, яке дозволяє оптимізувати пошук, відсікаючи непотрібні гілки дерева можливих ходів. Як зазначають автори роботи [62], цей підхід використовує значення MiniMax для відсікання піддерева, коли є впевненість, що хід не вплине на рішення в кореневому вузлі, що дозволяє Pac-Man швидко приймати рішення, зменшуючи кількість розглянутих варіантів. Дослідження також вказує на те, що після його відкриття, метод Alpha-Beta було розширено на інші типи ігор та алгоритми пошуку дерев, що свідчить про його широке застосування в різних контекстах, наприклад, як у роботах [48, 62].

Крім того, варто звернути увагу на алгоритми, що використовують підхід симуляції, як-от Monte Carlo Tree Search (MCTS). Цей метод дозволяє агенту досліджувати можливі варіанти дій через симуляцію майбутніх станів. MCTS є алгоритмом, що знаходить "оптимальні" дії шляхом випадкових рухів у просторі дій і побудови їх у деревовидну структуру, що розглянутоо у роботі [63].

Пошук по дереву Монте-Карло (MCTS) останніми роками набув значного поширення в галузі штучного інтелекту (ШІ), завдяки таким дослідженням як [44, 46], особливо для вирішення проблем прийняття рішень в умовах невизначеності, що розглядається у роботі [47]. Окрім застосувань в ігровій сфері, як це зроблено

у роботах [63, 64], MCTS використовується в різних складних середовищах, де агенти повинні ефективно досліджувати величезні простори рішень, наприклад, у роботах [45, 65].

Алгоритм Minimax є основою для прийняття рішень у змагальних середовищах, де агенти мають протилежні цілі. У грі Pac-Man цей алгоритм можна використовувати для керування Pac-Man з ухиленням від привидів.

Minimax ґрунтується на теорії ігор і моделює гру між двома сторонами — гравцем (Pac-Man) і противником (привиди). Мета алгоритму — максимізувати виграш Pac-Man (або мінімізувати втрати), передбачаючи дії супротивника. Алгоритм виконується рекурсивно, оцінюючи можливі стани гри на декілька кроків уперед (Див. Рис 3.3).

Основна ідея:

- Max-вузли відповідають діям Pac-Man і спрямовані на максимізацію виграшу.
- Min-вузли представляють дії привидів, які намагаються мінімізувати виграш Pac-Man.

```
Minimax(node, depth, maximizingPlayer):
    якщо depth == 0 або node є термінальним:
        повернути оцінку(node)

    якщо maximizingPlayer:
        maxEval = -∞
        для кожної дії дитини у node:
            eval = Minimax(дитина, depth-1, False)
            maxEval = max(maxEval, eval)
        повернути maxEval
    інакше:
```

```
minEval = +∞
для кожної дії дитини у node:
    eval = Minimax(дитина, depth-1, True)
    minEval = min(minEval, eval)
повернути minEval
```

1. Базовий випадок: алгоритм завершується, якщо досягнута максимальна глибина або термінальний стан (виграш/програш).
2. Max-вузли: оцінюють стан із точки зору Pac-Man, максимізуючи виграш.
3. Min-вузли: представляють дії привидів, спрямовані на мінімізацію виграшу Pac-Man.
4. Алгоритм працює рекурсивно, обчислюючи значення для кожного вузла на основі значень дочірніх вузлів.

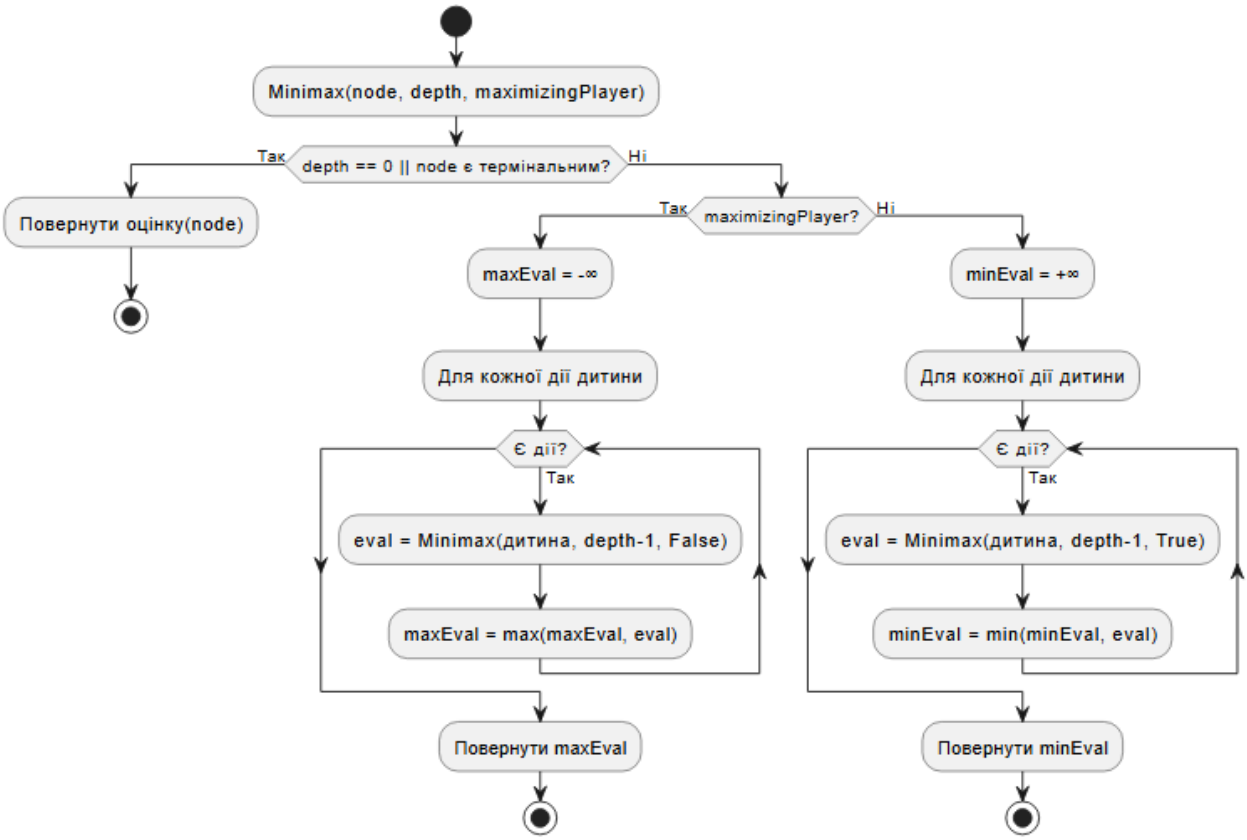


Рис. 3.3 — Схема роботи Minimax

1. Асимптотична складність:
    - Часова складність: $O(b^d)$, де b — фактор розгалуження (кількість дій на кожному рівні), d — глибина пошуку.
    - Просторова складність: O(d) для зберігання вузлів у стеку рекурсії.
2. Переваги Minimax:
    - Гарантоване знаходження оптимальної стратегії в середовищі з повною інформацією.
    - Ефективність для стратегічних ігор, де сторони діють раціонально.
3. Недоліки Minimax:
    - Експоненційне зростання кількості обчислень із глибиною пошуку.
    - Потребує повного перебору всіх можливих дій, що є проблемою для великих ігрових просторів.
4. Оптимізація за допомогою відсікання Alpha-Beta:
    - Використання відсікання значно зменшує кількість вузлів, які потрібно обробити.
5. Застосування в Pac-Man:
    - Minimax може використовуватися Pac-Man для вибору найкращої стратегії ухилення від привидів.

Expectimax є модифікацією алгоритму Minimax, який враховує ймовірності випадкових подій у процесі прийняття рішень. У грі Pac-Man цей алгоритм може використовуватися для передбачення дій привидів із врахуванням їхнього випадкового або непередбачуваного руху, що додає реалістичності та складності сценарію.

Алгоритм Expectimax замінює Min-узли на "випадкові узли", у яких рішення ґрунтується на очікуваному значенні стану, розрахованому за допомогою математичного сподівання (Див. Рис 3.4). Це дозволяє обробляти випадкові елементи в середовищі, наприклад, невизначений рух привидів.

Основна ідея:

- Max-вузли моделюють дії Pac-Man, спрямовані на максимізацію виграшу.
- Випадкові вузли обчислюють середнє очікуване значення можливих дій привидів, беручи до уваги їхню ймовірність.

```
Expectimax(node, depth, maximizingPlayer):
    якщо depth == 0 або node є термінальним:
        повернути оцінку(node)

    якщо maximizingPlayer:
        maxEval = -∞
        для кожної дії дитини у node:
            eval = Expectimax(дитина, depth-1, False)
            maxEval = max(maxEval, eval)
        повернути maxEval
    інакше:
        totalEval = 0
        для кожної дії дитини у node:
            prob = Ймовірність(дитина)
            eval = Expectimax(дитина, depth-1, True)
            totalEval += prob * eval
        повернути totalEval
```

1. Базовий випадок: алгоритм завершується, якщо досягнута максимальна глибина або термінальний стан.
2. Max-вузли: представляють дії Pac-Man, які максимізують виграш.
3. Випадкові вузли: розраховують середнє очікуване значення для кожної дії, враховуючи ймовірності.

4. Рекурсія завершується, коли досягається термінальний вузол або максимальна глибина дерева.

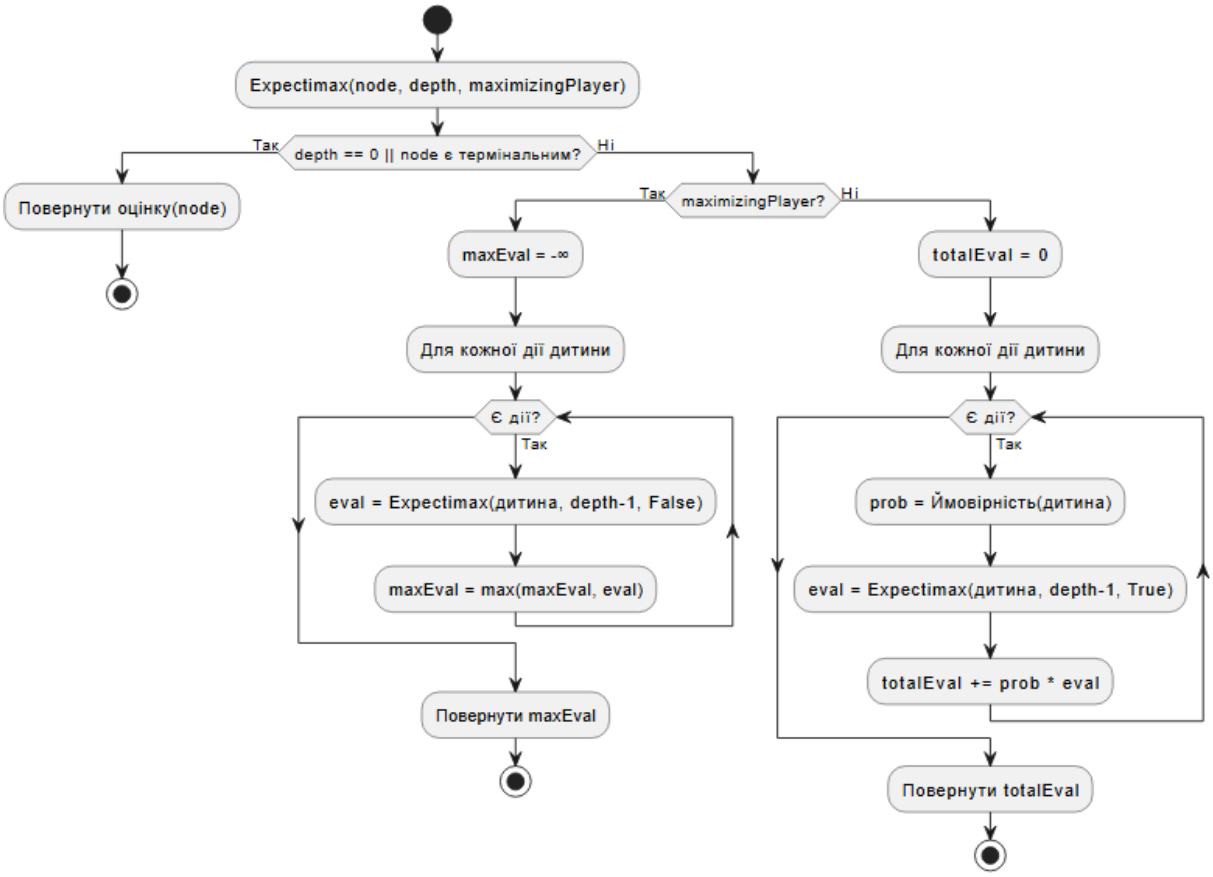


Рис. 3.4 — Схема роботи Expectimax

1. Асимптотична складність:
   - Часова складність: $O(b^d)$, де b — фактор розгалуження, d — глибина дерева.
   - Просторова складність: O(d), оскільки алгоритм зберігає вузли у стеку рекурсії.
2. Переваги Expectimax:
   - Обробляє випадковість у діях супротивників, що підвищує точність прийняття рішень.

- Більш реалістично моделює поведінку у багатофакторному середовищі.

3. Недоліки Expectimax:
   - Експоненційне зростання кількості обчислень із глибиною дерева.
   - Потребує точних ймовірностей для коректної роботи.
4. Застосування в Pac-Man:
   - Expectimax дозволяє Pac-Man передбачати ймовірний рух привидів, враховуючи випадковість у їхній поведінці.

Monte Carlo Tree Search (MCTS) є одним із найпоширеніших алгоритмів для прийняття рішень у середовищах із невизначеністю. Цей метод використовується у багатьох ігрових і стратегічних сценаріях, де необхідно обирати оптимальні дії на основі випадкових симуляцій. MCTS особливо ефективний у ситуаціях, коли кількість можливих ходів є великою, а розрахунок усіх можливих варіантів занадто витратний з погляду обчислювальних ресурсів.

Метод MCTS базується на побудові дерева пошуку, в якому кожен вузол відповідає стану гри, а ребра — можливим діям агента. Алгоритм використовує випадкові симуляції (метод монте-карло) для оцінки перспективності ходів, що дозволяє швидко наближатися до оптимального рішення (Див. Рис 3.5).

MCTS складається з чотирьох основних етапів:

1. Розширення дерева (Selection) – вибір найперспективнішої гілки для подальшого аналізу на основі певної політики (наприклад, UCT - Upper Confidence Bound for Trees).
2. Додавання нового вузла (Expansion) – створення нового вузла у дереві пошуку.
3. Симуляція (Simulation) – випадкове програвання гри з нової позиції для отримання оцінки результату.
4. Оновлення значень вузлів (Backpropagation) – оновлення оцінок вузлів дерева відповідно до результату симуляції.

```
MCTS(root, iterations):
    для і від 1 до iterations:
        node = Select(root)
        якщо node не є кінцевим станом:
            node = Expand(node)
        result = Simulate(node)
        Backpropagate(node, result)
    повернути найкращий хід з root

Select(node):
    поки node має дітей:
        node = обрати найкращий вузол за UCT
    повернути node

Expand(node):
    створити нового нащадка для node
    повернути нового нащадка

Simulate(node):
    поки node не є кінцевим станом:
        node = випадковий нащадок node
    повернути оцінку node

Backpropagate(node, result):
    поки node ≠ root:
        оновити статистику node за result
        node = батьківський вузол node
```

Основний алгоритм MCTS працює в ітеративному режимі, проводячи випадкові симуляції та оновлюючи дерево пошуку на основі отриманих результатів.

Функція Select вибирає найперспективніший вузол для дослідження, використовуючи політику UCT, яка збалансовує дослідження нових варіантів і експлуатацію вже знайдених хороших ходів.

Функція Expand додає новий вузол до дерева, якщо вибраний вузол має можливість розширення.

Функція Simulate виконує випадкову гру, щоб оцінити якість поточного ходу.

Функція Backpropagate оновлює статистику вузлів дерева відповідно до результату симуляції, коригуючи значення виграшу для майбутніх ітерацій.

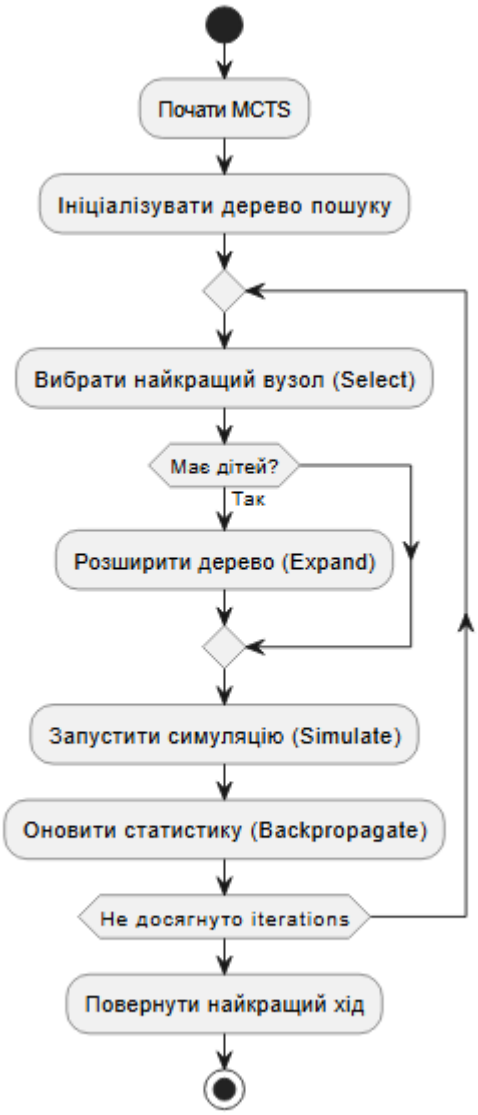


Рис. 3.5 — Схема роботи MCTS

1. Асимптотична складність:
    - Часова складність: O(mkI/C), де де m - кількість випадкових дітей, які потрібно розглянути за один пошук, а k - кількість паралельних пошуків, а I - кількість ітерацій, а C - кількість доступних ядер.
2. Переваги MCTS:
    - Підходить для середовищ із великою кількістю можливих ходів.
    - Гнучко адаптується до різних стратегічних сценаріїв.

- Враховує ймовірності можливих майбутніх станів.

3. Недоліки MCTS:
   - Чутливий до кількості ітерацій: при недостатній кількості симуляцій результат може бути далеким від оптимального.
   - Висока обчислювальна вартість порівняно з детермінованими алгоритмами (наприклад, Expectimax).
4. Застосування в Pac-Man:
   - MCTS використовується для прийняття рішень Pac-Man щодо оптимальної траєкторії ухилення від привидів.
   - Завдяки можливості передбачати майбутні ходи противника, MCTS забезпечує гнучку та адаптивну поведінку Pac-Man у грі.

Alpha-Beta Pruning (α-β обрізання) є вдосконаленням алгоритму Minimax, що використовується для пошуку оптимальних ходів у стратегічних іграх із повною інформацією. Його головна перевага полягає у зменшенні кількості вузлів, які необхідно переглядати, завдяки виключенню гілок дерева пошуку, що не впливають на кінцевий результат.

Alpha-Beta Pruning дозволяє значно скоротити кількість розглянутих варіантів, «обрізаючи» гілки дерева, які гарантовано не вплинуть на кінцеве рішення (Див. Рис 3.6). Це досягається введенням двох меж:

- α (альфа) — найкраще знайдене значення для гравця Max.
- β (бета) — найкраще знайдене значення для гравця Min.

Якщо у процесі пошуку значення вузла стає гіршим, ніж знайдені α або β, то подальший аналіз гілки припиняється.

```
AlphaBeta(node, depth, α, β, maximizingPlayer):
  якщо depth == 0 або node є кінцевим станом:
    повернути оцінку(node)
```

```
якщо maximizingPlayer:
    value = -∞
    для кожної дитини node:
        value = max(value, AlphaBeta(child, depth-1, α, β, False))
        α = max(α, value)
        якщо α ≥ β:
            break  # Обрізання гілки
    повернути value
інакше:
    value = +∞
    для кожної дитини node:
        value = min(value, AlphaBeta(child, depth-1, α, β, True))
        β = min(β, value)
        якщо α ≥ β:
            break  # Обрізання гілки
    повернути value
```

Алгоритм починається з оцінки вузла, якщо досягнута максимальна глибина або кінцевий стан.

Max-гравець намагається знайти найкраще можливе значення, використовуючи α як поточне найкраще. Якщо значення вузла перевищує β (оптимальний хід Min-гравця), подальший аналіз не має сенсу — виконується обрізання.

Min-гравець діє аналогічно, намагаючись зменшити значення і використовуючи β як критерій обрізання.

Якщо α ≥ β, виконується обрізання гілки, що скорочує загальну кількість обчислень.

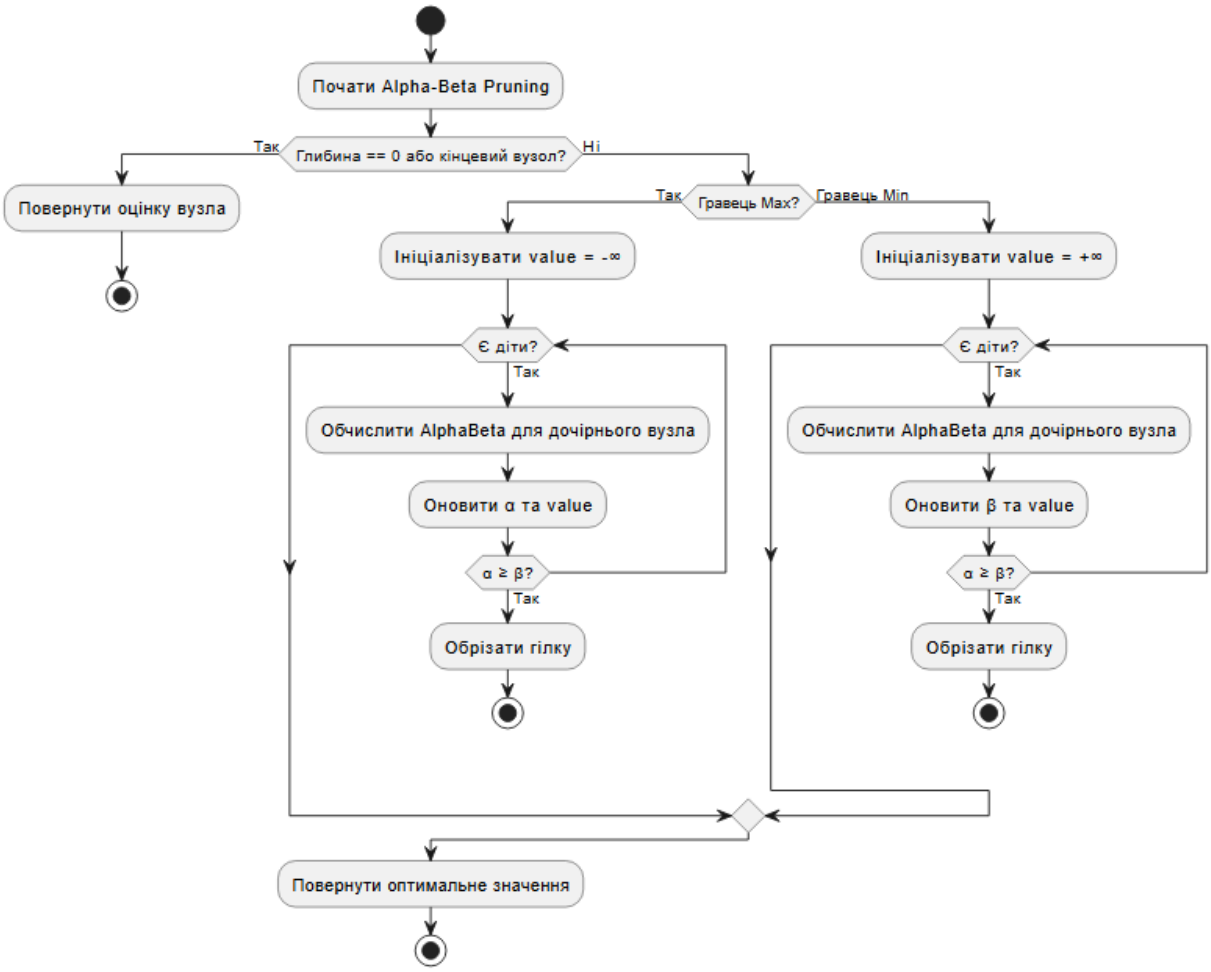


Рис. 3.6 — Схема роботи Alpha-Beta Pruning

1. Асимптотична складність:
   - У найгіршому випадку (без обрізання) алгоритм має складність $O(b^d)$, де b — середня кількість гілок на вузол, а d — глибина дерева.
   - У найкращому випадку (оптимальне обрізання) складність скорочується до $O(\sqrt{b^d})$, що дає значне покращення.
2. Переваги Alpha-Beta Pruning:
   - Економія ресурсів: Обрізання гілок значно зменшує кількість необхідних розрахунків.
   - Гарантовано оптимальне рішення: За правильної реалізації алгоритм дає той самий результат, що і Minimax, але швидше.

- Ефективність у стратегічних іграх: Використовується у шахах, шашках, Pac-Man та інших іграх з чітко визначеними правилами.

3. Недоліки Alpha-Beta Pruning:
   - Залежність від порядку перебору вузлів: Якщо вузли не впорядковані оптимально, ефективність обрізання може бути суттєво нижчою.
   - Потреба у точних оцінювальних функціях: Якість гри залежить від того, наскільки добре алгоритм оцінює проміжні стани.
4. Застосування в Pac-Man:
   - Alpha-Beta Pruning використовується для керування Pac-Man, вибираючи найбільш вигідний шлях, враховуючи дії привидів.

3.4 Забезпечення гнучкого налаштування пріоритизації задач агентами через функції оцінки та евристики

Для забезпечення гнучкості налаштування пріоритизації цілей та адаптації поведінки агентів у середовищі пропонується використання окремих реалізацій функцій оцінки, евристичних функції та функцій постановки задач пошуку для алгоритмів прийняття рішень. Розширюючи цю ідею, слід зазначити, що кожна з цих функцій виконує свою ключову роль у моделюванні поведінки агентів. Функції оцінки дозволяють визначати ефективність обраного шляху чи стратегії, евристичні функції спрямовані на прискорення пошуку оптимальних рішень шляхом встановлення пріоритетів, а функції постановки задач пошуку допомагають формалізувати завдання у вигляді чітко окреслених проблем для подальшого вирішення.

Завдяки наданій можливості налаштовувати ці функції під різні завдання та пріоритети поза межами алгоритмів керування, систему можна адаптувати для вирішення конкретних задач, інтегруючи необхідні версії функцій залежно від поставлених завдань. Такий підхід значно підвищує гнучкість і варіативність, дозволяючи користувачам налаштовувати поведінку агентів під різні сценарії у

процесі розробки та підбору методів керування, надаючи можливість агентам ефективно орієнтуватися в середовищі, адаптувати свою поведінку та оптимізувати прийняття рішень відповідно до поточних умов.

Зокрема, на розробленій платформі запропоновано три базові функції, що сприяють пріоритизації виконання місій (збирання капсул у середовищі). Ці функції допомагають агентам віддавати перевагу новим шляхам для розвідки перед уже пройденими, а також уникають зіткнень із супротивниками:

1. **capsulesSearchHeuristic** – функція, що імплементує базовий метод “problemSearch” для пошукових агентів;
2. **capsulesEvaluation** – базовий метод оцінки поточного стану і варіантів ходів для класів мультиагентних алгоритмів;
3. **heuristic** – метод, що використовується агентом **MCTSAgentWithHeuristic**, який може бути замінений імплементаціями класів наслідників під конкретні потреби.

Функція capsulesSearchHeuristic визначає, наскільки вигідно Pac-Man відвідувати капсули, що може впливати на його стратегію ухилення від привидів (Див. Рис 3.7):

```
capsulesSearchHeuristic(state, problem):
    знайти найближчу капсулу
    знайти відстань до цієї капсули
    якщо є загроза привидів:
        збільшити вагу відстані
    повернути значення евристики
```

Рис. 3.7 — Схема роботи capsulesSearchHeuristic

Функція capsulesEvaluation використовується у алгоритмах з підтримкою ухилення від переслідувачів для визначення корисності того чи іншого стану гри (Див. Рис 3.8).

```
capsulesEvaluationFunction(state):
    отримати поточний рахунок
    знайти капсули, що залишилися
    якщо капсул мало:
        підвищити пріоритет їх збору
    якщо є загроза привидів:
        скоригувати оцінку ризику
    повернути фінальну оцінку стану
```

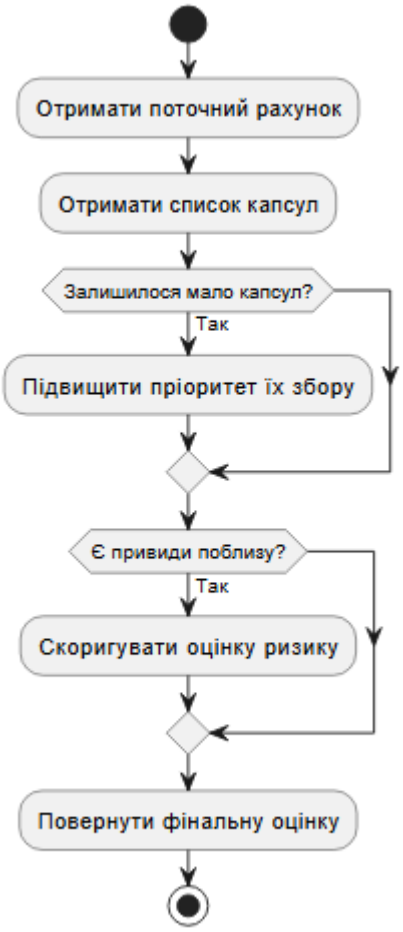


Рис. 3.8 — Схема роботи capsulesEvaluationFunction

Звичайна імплементація MCTS покладається лише на евристичну функцію для направлення вибору вузлів, проте впроваджена інтеграція, базового для всіх класів мультиагентних алгоритмів в системі, методу evaluationFunction дозволяє отримувати більш детальну оцінку поточних станів дерева пошуку. Це розширення зроблено з метою підвищити точність оцінювання та адаптивність алгоритму до специфічних умов задачі, забезпечуючи більш гнучке управління процесом прийняття рішень.

Метод heuristic дозволяє оцінювати можливі стани на основі додаткових факторів, таких як відстані до їжі та привидів (Див. Рис 3.9).

heuristic(state):

обчислити мінімальну відстань до їжі

обчислити відстань до найближчого привида

якщо привид загрожує:

збільшити ризик-фактор

повернути скориговану евристику

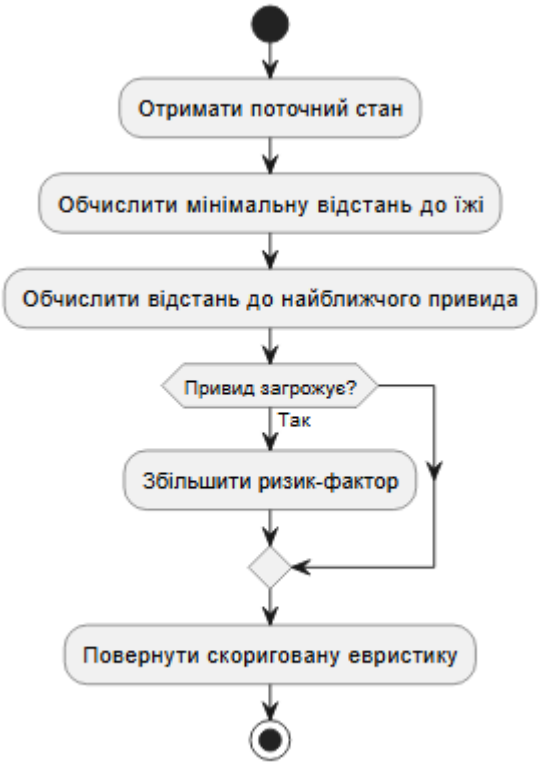


Рис. 3.9 — Схема роботи heuristic в MCTSAgentWithHeuristic

3.5 Використання машинного навчання для синтезу тренованої моделі нейромережі

Для навчання моделей нейронних мереж спрямованих на керування роботизованими системами зазвичай використовуються рішення на основі Multi-Agent Reinforcement Learning (MARL), оскільки вони дозволяють численним

агентам ефективно взаємодіяти та вчитися в умовах високої динамічності та невизначеності, як у роботах [6, 26]. MARL забезпечує адаптивність, координацію та оптимізацію дій кожного агента завдяки спільному навчальному процесу, що сприяє обміну інформацією та синергії між компонентами системи, що висвітлено авторами робіт [26, 66]. Такий підхід дозволяє роботизованим системам швидко адаптуватися до змін у зовнішньому середовищі, покращувати загальну продуктивність та забезпечувати надійне виконання складних завдань, що підтверджується результатими роботи [28].

MADDPG (Multi-Agent Deep Deterministic Policy Gradient) — це алгоритм навчання з підкріпленням, який дозволяє агентам взаємодіяти та координувати свої дії в багатокористувацьких сценаріях. MADDPG є важливим інструментом для розв'язання завдань мультиагентного навчання з підкріпленням, особливо у середовищах зі змішаною кооперативно-конкурентною взаємодією. Цей алгоритм базується на поєднанні централізованого навчання та децентралізованого виконання, що дозволяє ефективно оптимізувати політики агентів, враховуючи дії та спостереження інших учасників середовища, як стверджують його автори у статті [5].

У розробленій інформаційній технології пропонується використання цього алгоритму для навчання моделей кооперативного керування агентами супротивниками (спільна стратегія переслідування), у вигляді окремого середовища. Також має перспективи бути впровадженим для використання для навчання нейромережевих моделей для керування Pac-Man (уникнення привидів та оптимізація руху) у подальших дослідженнях.

MADDPG розширює стандартний алгоритм DDPG, додаючи централізованого критика, який оцінює дії всіх агентів на основі глобальної інформації (Див. Рис 3.10). Актори залишаються децентралізованими, дозволяючи кожному агенту приймати рішення самостійно. Такий підхід забезпечує координацію між агентами, зменшуючи конфлікти в їхніх діях.

Основна ідея:

- Централізований критик: оцінює дії всіх агентів одночасно.
- Децентралізовані актори: використовують лише локальну інформацію для прийняття рішень.
- Глибокі нейронні мережі: забезпечують параметризацію політики та функції цінності.

MADDPG(агенти, середовище):
  Ініціалізувати мережі акторів і критиків для кожного агента
  Ініціалізувати буфер відтворення R

  поки не досягнуто максимального числа епох:
    Отримати початковий стан середовища
    поки не завершиться епізод:
      Для кожного агента обрати дію за допомогою політики (актор)
      Виконати дії та спостерігати новий стан і винагороду
      Зберегти перехід (стан, дія, винагорода, новий_стан) у R
    Якщо достатньо даних у R:
      Для кожного агента:
        Вибрати випадковий міні-батч із R
        Оновити критика за допомогою функції втрат
        Оновити актора, максимізуючи очікувану винагороду
        Оновити ваги мереж за допомогою градієнтного спуску

1. Ініціалізація акторів і критиків для кожного агента, а також буфера відтворення.
2. У кожному епізоді агенти вибирають дії на основі своєї політики, а середовище повертає новий стан і винагороду.

3. Перехідні дані зберігаються в буфері відтворення.
4. Мережі оновлюються за допомогою градієнтного спуску на основі вибраних міні-батчів.

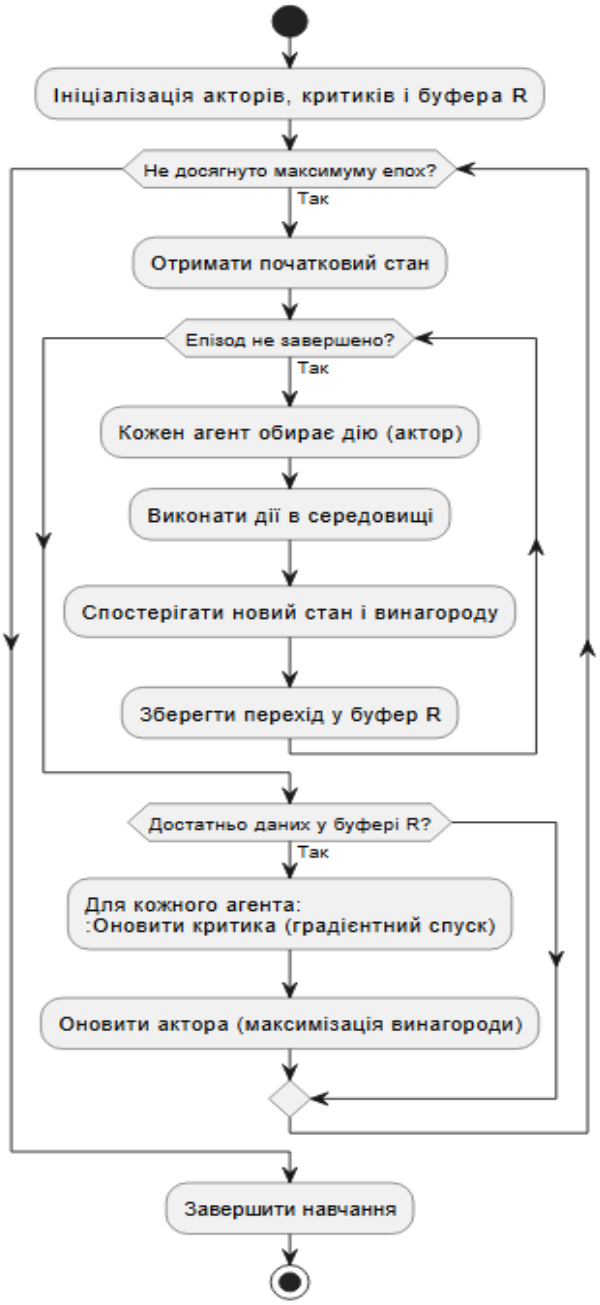


Рис. 3.10 — Схема роботи MADDPG

1. Переваги MADDPG:

- Забезпечує координацію між агентами за рахунок централізованого критика.
- Ефективний для сценаріїв із неповною інформацією.
- Здатність до адаптації та навчання в реальному часі.

2. Недоліки MADDPG:
   - Високі вимоги до обчислювальних ресурсів через використання декількох нейронних мереж.
   - Складність налаштування гіперпараметрів, таких як швидкість навчання та розмір буфера.
3. Застосування в Pac-Man:
   - Привиди можуть використовувати модель нейромережі навченої за допомогою MADDPG для координації своїх стратегій переслідування, взаємодіючи як команда.
   - В перспективі може бути синтезована і впроваджена модель для керування Pac-Man, яка зможе навчитися ухилятися від групи привидів, оптимізуючи свої шляхи.

3.6 Розробка методів збору статистики для аналізу ефективності методів керування агентами

Для оцінки ефективності алгоритмів керування агентами у розробленій інформаційній технології впроваджено збір статистики через логування, що дозволяє аналізувати та обгрунтовувати продуктивність агентів, їхню поведінкову адаптивність та ефективність ухвалення рішень на основі зібраних статистичних показників. Аналіз статистики базується на даних, отриманих під час роботи алгоритмів у симуляційному середовищі та подальшому порівнянні ключовими метриками, серед яких пропонуються наступні:

1. Середній рахунок за симуляційний запуск – цей показник відображає середню суму балів або оцінку, яку набирають агенти під час одного

симуляційного запуску. Він допомагає оцінити ефективність алгоритмів управління в роботизованих системах та їх здатність досягати поставлених головних та другорядних цілей.

2. Середній час симуляції – даний параметр визначає середню тривалість одного запуску симуляції. Він дозволяє аналізувати швидкість виконання алгоритмів та ефективність і якість прийнятих рішень в умовах реального часу.
3. Відсоток успішних завершень – цей показник характеризує ефективність роботи агентів у досягненні головної мети симуляції, яка полягає у поїданні всіх капсул. Високий відсоток успішних завершень свідчить про надійність та стабільність застосованих рішень для керування агентами.

Засоби збору статистики. Для збору статистичних даних пропонується модуль, що здійснює автоматичний запис параметрів під час роботи проведення експериментів. Модуль підтримує розширення на нові параметри у разі необхідності. Параметри, які збираються через логування по кожній симуляції:

- *layout*: Назва лабіринту, що використовується
- *pacman*: Тип використовуваного агента Pacman
- *numGames*: Загальна кількість ігор, зіграних в експерименті
- *scores*: Список балів з кожної гри (симуляції)
- *averageScore*: Середній бал серед всіх симуляцій запуску
- *times*: Список часу проходження кожної гри
- *averageTime*: Середній час серед всіх симуляцій запуску
- *winRate*: Відсоток виграшів у відсотках
- *record*: Список логічних значень, що вказують на перемоги/поразки. Запис виграшу/програшу у форматі «Виграш, програш, виграш, ...»
- *scaredTime*: Час, протягом якого привиди залишаються у наляканому/вразливому стані

- *modelName*: Назва моделі нейромережі, що використовується (у випадку використання)

Дані відформатовані за допомогою чітких міток і впроваджено зберігання двома способами:

1. Кожен запис одразу записується у файл журналу при виклику функції *log*
2. Записи також зберігаються в пам'яті у списку даних, який може бути записаний за допомогою функції *save_summary*

Отримані статистичні показники дозволяють об’єктивно оцінити різні підходи до керування агентами відносно поставлених задач та визначити найефективніші алгоритми для конкретного контексту.

Висновки до розділу 3

У третьому розділі розглянуто комплекс підходів та методів, спрямованих на розробку інформаційної технології, що дозволить використовувати комп’ютерне моделювання з метою тестування алгоритмів та підходів управління агентами у 2D середовищі в контексті виконання реалістичних задач роботизованих систем. Було представлено основні деталі роботи агентів, середовища, механіку лабіринту, а також поведінку та ролі агентів у середовищі.

Розглянуто як методи традиційні пошукові алгоритми, такі як BFS та A*, а також їх модифікації для мультиагентної взаємодії, зокрема алгоритми Expectimax, Minimax, MCTS та Alpha-Beta Pruning. Особливу увагу приділено використанню Multi-Agent Deep Deterministic Policy Gradient (MADDPG) для навчання нейромережевої моделі, яка забезпечує координацію привидів та їхню взаємодію в у середовищі, як одна із доступних опцій.

Крім того, у розділі висвітлено роль функцій оцінки, евристик та методів збору статистики, які є ключовими для контролю якості та адаптивності системи. Запровадження системи логування основних параметрів симуляційних запусків

дозволило об'єктивно оцінити ефективність застосованих рішень та оптимізувати подальшу розробку.

Розробка зазначених методів здійснювалася із застосуванням SDLC та Scrum, що забезпечило гнучкість, чітку організацію робочого процесу та можливість швидкої адаптації до змін вимог.

РОЗДІЛ 4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ ПРОВЕДЕНИХ У РОЗРОБЛЕНІЙ ІНФОРМАЦІЙНІЙ ТЕХНОЛОГІЇ

4.1 Важливість тестування агентних систем у складних середовищах

У сучасних системах роботизованих платформ та автономних пристроїв особливе значення має розробка та тестування алгоритмів ухвалення рішень, здатних адаптуватися до змінних умов середовища. Важливість цього процесу зумовлена необхідністю забезпечення стабільної та надійної роботи роботизованих систем у реальному часі, де кожна секунда прийняття рішення може впливати на кінцевий результат виконання завдання. Одним із ключових завдань досліджень у цій галузі є тестування та оцінка ефективності алгоритмів, що дозволяє визначити їхню стабільність, продуктивність та гнучкість в умовах змінного середовища.

Основним завданням тестування є аналіз здатності алгоритмів ефективно виконувати планування маршруту, ухилення від загроз, координацію групових дій та адаптацію до динаміки середовища. Ці алгоритми повинні забезпечувати:

- **Адаптивність**: алгоритми повинні оперативно реагувати на змінні умови та несподівані ситуації, що виникають у процесі експлуатації.
- **Продуктивність**: визначення продуктивності рішень дозволяє виявити обмеження обчислювальних ресурсів та забезпечити оптимальне використання можливостей системи.
- **Стабільність**: перевірка роботи алгоритмів в умовах високої динаміки середовища дозволяє оцінити їхню стійкість до змін та зовнішніх впливів.

Автономні системи, що функціонують у динамічних середовищах, мають вирішувати широкий спектр завдань, включаючи планування маршруту, ухилення від загроз, координацію групових дій та адаптацію до змінних умов. Pac-Man виступає як автономний агент, що має ухвалювати оптимальні рішення для досягнення своїх цілей, а привиди реалізують мультиагентну систему, яка

координує свої дії задля протидії Pac-Man. Для ефективного виконання цих завдань, в рамках розробленої інформаційної технології, використовуються алгоритми пошуку оптимальних рішень, мультиагентні алгоритми, а також моделі нейромереж натреновані за допомогою навчання з підкріпленням.

У рамках даного дослідження особлива увага приділяється аналізу поведінки агентів у симуляційному середовищі, яке імітує різні сценарії взаємодії між автономними пристроями. Використання тестового майданчику, заснованого на середовищі Pac-Man, пропонує наступні можливості:

1. **Моделювати багатогранну взаємодію**: гра включає як кооперативні, так і конкурентні сценарії, що сприяє вивченню різних аспектів групової поведінки, що також дає можливість випробовувати алгоритми керування для сценаріїв протидії командної взаємодії (1 до N) або з іншої сторони задачі – протидіяти супротивнику групою агентів (N до 1).
2. **Відтворювати складну топологію**: лабіринтоподібна структура рівнів створює умови, наближені до реальних середовищ, де агентам доводиться вирішувати завдання з навігації та ухилення. Завдяки гнучкості імплементації інформаційної технології вона дає можливість розширювати наявний набір лабіринтів, додаючи нові під потрібні задачі.
3. **Забезпечувати контрольовані експерименти**: можливість варіювати масштаби лабіринтів (від Small до XLarge), кількість агентів, кількість цілей, розмірність нагороди для пріоритизації задач та інші параметри дозволяє оцінити ефективність алгоритмів при різних рівнях складності та різноманітних сценаріях взаємодії.
4. **Доступність та відтворюваність експериментів:** симуляційне середовище забезпечує контрольовані умови, що дозволяє точно порівнювати результати роботи алгоритмів.

Експерименти в таких умовах дозволяють не лише порівняти продуктивність традиційних алгоритмів (BFS, A*, AlphaBeta, Expectimax,

MCTS), але й дослідити потенціал навчальних моделей, наприклад, MADDPG, в сценаріях протидії та координації дій. Це дає змогу зробити висновки щодо придатності розроблених алгоритмів для реальних застосувань у роботизованих системах під конкретні задачі та умови.

4.2 Прикладне застосування та практична значимість інформаційної технології

Розроблена інформаційна технологія, що базується на симуляційному середовищі Pac-Man, має значний прикладний потенціал завдяки широким можливостям моделювання різноманітних сценаріїв взаємодії автономних агентів. Серед отриманих результатів експериментів особливе місце займає спрямованість на практичні аспекти використання цієї технології у реальних умовах, що стосуються вирішення завдань оптимізації маршрутів, адаптації поведінки агентів під динамічні зміни середовища та координації їх дій у групових сценаріях. Завдяки можливості точно контролювати умови експерименту (зміна розмірів лабіринтів, кількості агентів, різноманітності цілей та параметрів винагороди) система стає універсальним інструментом для дослідження як традиційних алгоритмічних рішень, так і моделей нейромереж, навчених за допомогою сучасних методів навчання з підкріпленням.

Окрім базових характеристик, технологія дозволяє моделювати як кооперативні, так і конкурентні сценарії, що є надзвичайно важливим для аналізу реальних застосувань. Наприклад, у середовищі Pac-Man можна зобразити ситуації, коли автономні агенті працюють разом для досягнення спільної мети або ж, навпаки, взаємодіють як суперники, що створює основу для тестування алгоритмів ухилення від загроз, оптимізації маршрутів та розподілу ресурсів у випадку конфлікту. Така гнучкість дозволяє адаптувати підходи тестування під специфічні вимоги різних галузей, що потребують високої автономності та оперативного прийняття рішень.

Нижче наведено основні сфери застосування цієї технології та детальний розгляд її практичного значення:

- Вирішення завдань оптимізації маршрутів, що є критично важливим у роботизованих системах для розмінування.
- Координація руху та взаємодія агентів у складних, часто змінних середовищах, наприклад, при управлінні дронами.
- Реалізація систем безпеки та моніторингу, де автономні агенти виконують завдання розвідки та оперативного реагування.
- Дослідження адаптивних стратегій для агентів у змодельованих сценаріях, що дозволяє аналізувати та вдосконалювати алгоритми керування при різних умовах експлуатації.

Таким чином, завдяки своїй модульності та гнучкості, розроблена технологія не лише забезпечує високий рівень контролю над експериментальними умовами, але й створює базу для подальшого впровадження інноваційних алгоритмічних рішень у різних прикладних сферах робототехніки та автономних систем.

4.3 Умови експериментів та структура гри

Для дослідження ефективності алгоритмів керування автономними агентами було створено експериментальну платформу, засновану на класичній грі Pac-Man. Ця платформа дозволяє змоделювати різноманітні сценарії взаємодії між агентами в контрольованих умовах, що є важливим для порівняння продуктивності як традиційних пошукових алгоритмів, так і сучасних методів мультиагентного навчання.

Опис структури гри та завдань. Основною метою гри є виконання місій Pac-Man, які включають:

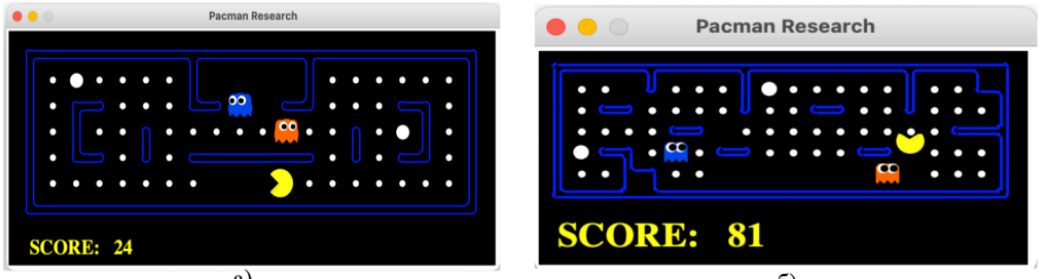
- **Збір капсул:** Головне завдання – зібрати всі капсули (ключові точки, які необхідні для виконання місії), що розташовані по лабіринту. Ці капсули

виступають стратегічними цілями, що вимагають від агента швидкого прийняття рішень для оптимізації маршруту. Кожне з'їдання капсули надає агенту 500 очок в системі оцінювання та робить приивидів уразливими для нейтралізації.

- **Уникання привидів:** Привиди діють як динамічні загрози. Вони переслідують Pac-Man, застосовуючи різні стратегії руху, що ускладнює завдання уникання зіткнень. Така поведінка змушує алгоритми враховувати не лише статичні особливості середовища, а й змінну поведінку противників.
- **Знищення уразливих привидів:** При збиранні капсули привиди стають уразливими протягом 10 ходів, що відкриває можливість для Pac-Man знищувати їх за додаткові очки. Цей механізм сприяє тестуванню алгоритмів з урахуванням тактики агресивного або оборонного стилю гри. За нейтралізацію привида в системі оцінювання нараховується 200 очок.
- **Збір їжі:** Додаткове завдання – збирання їжі, що дає можливість агенту отримувати додаткові очки за дослідження нових ділянок лабіринту. Оцінюється в 10 очок.

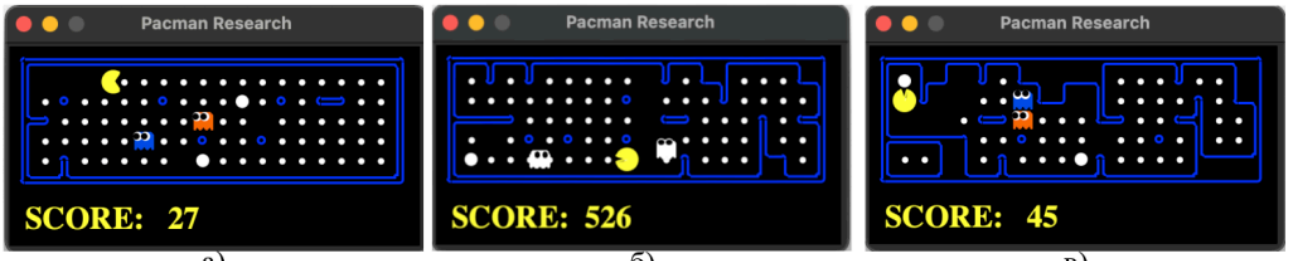
Типи лабіринтів та конфігурації середовища. Для оцінки ефективності алгоритмів впроваджено різноманітні типи лабіринтів, що відрізняються за структурною складністю, розміром і динамікою взаємодії агентів, на яких проводилися експерименти:

Small (Малі лабіринти): 7x20 розмірність, 2 противники, 2 цілі. Містить вертикально-симметричну конфігурацію (Див Рис 4.1). Для тестування нейромережевої моделі навченої за допомогою MADDPG використовувались окремі конфігурації з площею перешкод у 10% від площі лабіринту, 25% та 40% (Див. Рис 4.2).



а) б)

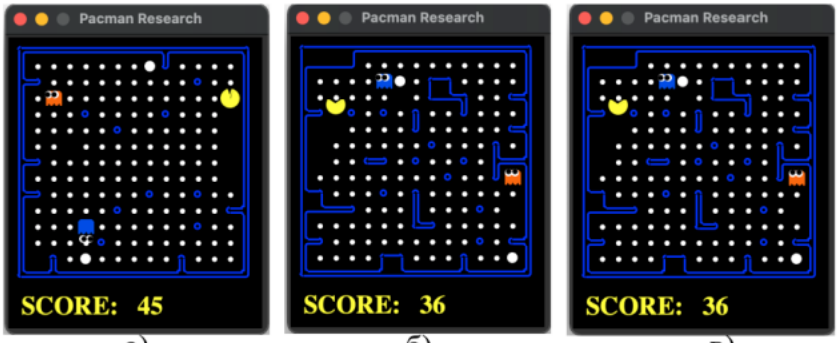
Рис. 4.1 — Макети малих лабіринтів, що використовувалися в експериментах: а) Small Maze 1 – вертикально симметричний, б) Small Maze 2 – хаотичний розподіл наближений до реальної місцевості



а) б) в)

Рис. 4.2 — Макети малих лабіринтів, що використовувалися в експериментах з нейромережевою моделлю: а) Small 10w - низької складності (10% стін), б) Small 25w - середньої складності (25% стін), в) Small 40w - високої складності (40% стін).

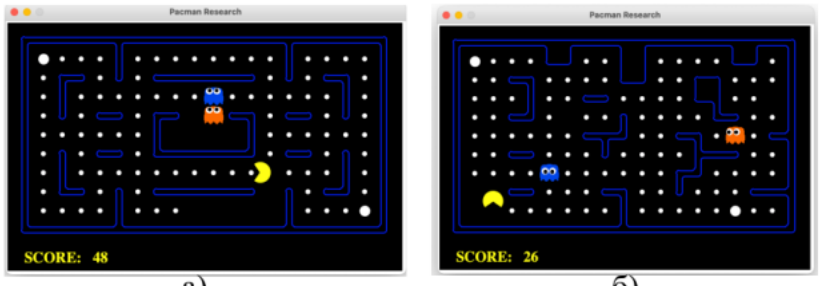
SM (Small-to-Medium): 15x15 розмірність, 2 противники, 2 цілі. Унікальні конфігурації, розроблені для перевірки навченої нейромережевої моделі MADDPG. Для експерименту використовуваись у варіаціях (Див. Рис. 4.3).



а) б) в)

Рис. 4.3 — Макети SM лабіринтів, що використовувалися в експериментах з нейромережевою моделлю: а) SM1 - низької складності (10% стін), б) SM2 - середньої складності (25% стін), в) SM3 - високої складності (40% стін).

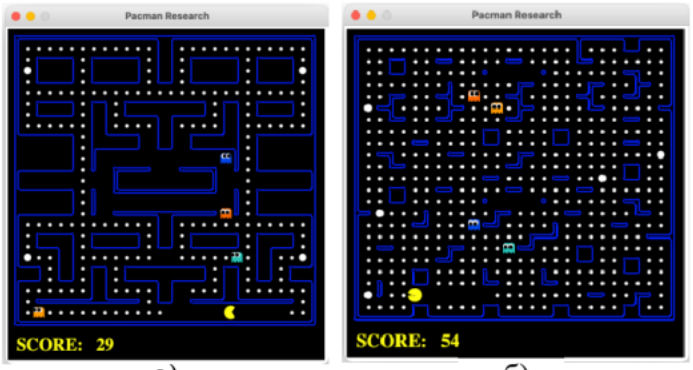
Medium (Середні лабіринти): 11x20 розмірність, 2 противники, 2 цілі. Лабіринти з більш складними маршрутами та обмеженим простором для маневру, що створює додаткове навантаження на алгоритми (Див. Рис. 4.4).



а) б)

Рис. 4.4 — Макети середніх лабіринтів, що використовувалися в експериментах: а) Medium Maze 1 – вертикально симметричний, б) Medium Maze 2 – хаотичний розподіл наближений до реальної місцевості

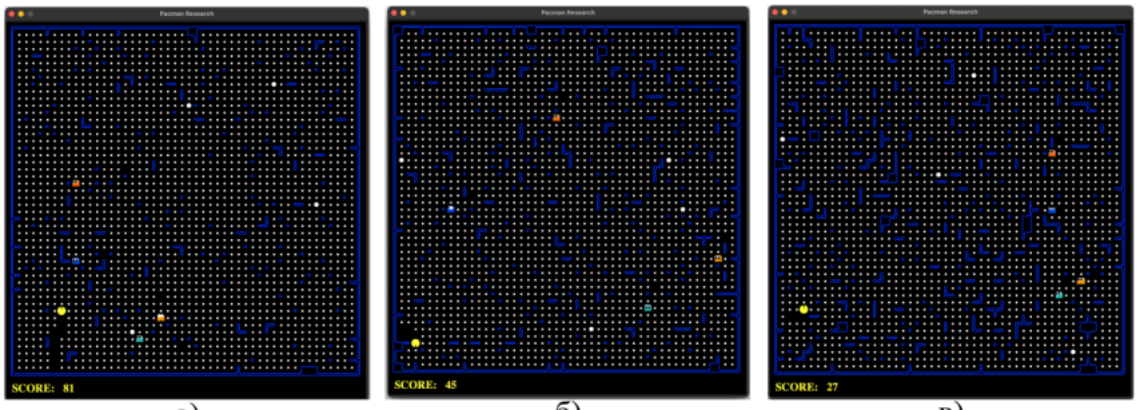
Large (Великі лабіринти): 27x28 розмірність, 4 противники, 4 цілі. Розширена топологія з численними варіантами маршрутів, що дозволяє тестувати алгоритми на довготривале планування та стратегічну адаптацію (Див. Рис. 4.5).



а) б)

Рис. 4.5 — Макети великих лабіринтів, що використовувалися в експериментах: а) Large Maze 1 – вертикально симметричний, б) Large Maze 2 – хаотичний розподіл наближений до реальної місцевості

XLarge (Надвеликі лабіринти): 50x50 розмірність, 4 противники, 4 цілі. Дуже просторі лабіринти, які ставить перед алгоритмами завдання стратегічного ухвалення рішень у масштабованих сценаріях (Див. Рис. 4.6).



а) б) в)

Рис. 4.6 — Макети надвеликих лабіринтів, що використовувалися в експериментах: а) XLarge Maze 1 - низької складності (10% стін), б) XLarge Maze 2 - середньої складності (15% стін), в) XLarge Maze 3 - високої складності (20% стін).

Запропоновані лабіринти, в рамках дослідження, вважаються наближеними до реальних ландшафтів, як міських, так і природних, з перешкодами, що представляють собою будівлі, дерева та інші перешкоди. Ці перешкоди імітують

умови реальних застосувань таких як розміновування роботизованими системами або умови польоту БПЛА на певних висотах, де навігація та уникнення перешкод є критично важливими. Такий підхід дозволяє тестувати адаптивність алгоритмів у сценаріях, що максимально наближені до реальних викликів, з якими стикаються роботизовані системи або БПЛА в динамічному та обмеженому середовищі.

Кожен тип лабіринту тестувався по 100 експериментів на кожного з агентів, що дозволяло отримати статистично показові результати для подальшого аналізу.

Алгоритмічні підходи та варіанти експериментів. Для оцінки ефективності роботи системи використовувались різні групи алгоритмів у ролі агентів, що дозволило охопити широкий спектр методів ухвалення рішень:

- **Пошукові алгоритми (Search Algorithms):**
  - *BFS (Breadth-First Search):* Забезпечує обхід у ширину та знаходження найкоротшого шляху.
  - *A-star (A**): Використовує евристику для оптимізації навігації, що дозволяє прискорити пошук оптимального маршруту.
- **Мультиагентні алгоритми (Multi-Agent Algorithms):**
  - *Alpha-Beta Pruning:* Модифікація алгоритму Minimax із скороченням непотрібних гілок дерева рішень, що покращує швидкодію.
  - *Expectimax:* Розширює принцип Minimax з урахуванням ймовірнісних подій, забезпечуючи адаптивність до невизначеності в поведінці противників.
  - *MCTS (Monte Carlo Tree Search):* Комбінує випадковий пошук та евристичну оцінку, що дозволяє приймати рішення в умовах високої складності сценаріїв.

- **Правило-орієнтовані алгоритми для поведінки привидів:** Привиди з гри Pac-Man реалізовані за класичними алгоритмами гри, описаними в статтях [32, 33]:
  - *Blinky (Червоний привид):* Агресивно переслідує Pac-Man, змушуючи останнього швидко ухилятися.
  - *Pinky (Рожевий привид):* Стратегічно блокує шлях Pac-Man, прогнозуючи його напрямок руху.
  - *Inky (Блакитний привид):* Їх поведінка залежить від позиції як Pac-Man, так і Blinky, що створює непередбачувану загрозу.
  - *Clyde (Помаранчевий привид):* Змінює стратегію залежно від відстані до Pac-Man, що ускладнює планування маршруту останнього.
- **Нейромережеві моделі натреновані за допомогою навчання з підкріпленням:**
  - *MADDPG (Multi-Agent Deep Deterministic Policy Gradient):* Забезпечує координацію дій агентів, навчаючи їх стратегічній поведінці та адаптації в мультиагентних середовищах і детально описаний в статті [5].
- **Random Agent:** Використовується як базовий еталон для порівняння ефективності алгоритмів.

Метрики оцінки ефективності. Для об’єктивного аналізу роботи різних алгоритмів використовувалися наступні ключові метрики:

- **Кількість часу до завершення гри:** Цей показник відображає ефективність алгоритму у досягненні поставленої мети – збиранні всіх капсул або уникненні загроз. Менша кількість часу свідчить про оптимальніший маршрут, високу продуктивність та швидкість прийняття рішень.

- **Відсоток перемог:** Ця метрика показує, скільки експериментальних запусків завершилося успішним виконанням завдання Pac-Man (збір усіх капсул). Високий відсоток перемог свідчить про стабільність та адаптивність алгоритму в умовах змінної динаміки середовища.
- **Кількість набраних очок Pac-Man:** Очки враховують не тільки базове завдання збирання капсул, а й додаткові елементи, як знищення уразливих привидів та збір їжі. Ця метрика дозволяє оцінити комплексну ефективність алгоритму, враховуючи його тактичні та стратегічні рішення протягом гри.

Використання зазначених метрик є важливим, оскільки вони дозволяють отримати багатовимірну оцінку роботи алгоритмів. Поєднання показників швидкості, стабільності та загального результату гри створює основу для комплексного порівняльного аналізу та дозволяє виявити як сильні, так і слабкі сторони кожного підходу.

Підсумок експериментальних умов. Структура гри та умови експерименту ретельно сплановані для того, щоб максимально точно відобразити різноманіття реальних сценаріїв взаємодії автономних агентів. Варіювання розмірів лабіринтів, конфігурацій середовища, типів загроз та чисельності агентів дозволяє:

- Дослідити адаптивність алгоритмів до змінних умов.
- Провести порівняльний аналіз традиційних пошукових та мультиагентних алгоритмів, а також сучасних моделей навчання з підкріпленням.
- Забезпечити контрольоване середовище для тестування, що сприяє точній верифікації отриманих результатів.

Таким чином, умовив експерименту та структура гри сприяють отриманню достовірних і репрезентативних даних, що дозволяють зробити обґрунтовані висновки щодо ефективності розробленої інформаційної технології для

тестування алгоритмів керування роботизованими пристроями у мультиагентних системах.

4.4 Порівняння пошукових і мультиагентних алгоритмів у протидії команді роботизованих систем

Головна мета дослідження полягала у визначенні та обґрунтуванні переваг та недоліків класичних пошукових алгоритмів та методів мультиагентного ухвалення рішень у різних середовищах із варіативною складністю лабіринтів.

Як було зазначено вище, кожен агент виконав 100 ігор у кожному типі середовища, а результати оцінювалися за наступними критеріями:

- Середній рахунок (чим вищий – тим ефективніше агент виконував свої завдання).
- Час гри (середня тривалість гри в секундах).
- Відсоток виграшів (кількість ігор, де Агент вижив і виконав свою місію).

Таблиця 4.1. Результати у Small Maze 1 (Вертикально симметричному)

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	35.88	2.66	0
A* Agent	1063.27	4.12	37
BFS Agent	1071.49	4.08	37
AlphaBeta Agent	1272.29	7.09	62
Expectimax Agent	1410.85	7.91	72

Таблиця 4.2. Результати агента MCTS у Small Maze 1 (Вертикально симметричному)

Кількість симуляцій	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
101	585.18	7.59	12
200	611.07	9.68	13
300	569.33	11.14	13
400	569.33	12.99	18
500	666.89	14.99	17

У таблиці 4.1 представлено результати роботи агентів у Small Maze 1 з вертикально симетричною конфігурацією. Серед стандартних агентів **Random Agent** демонструє дуже низьку ефективність (середній рахунок 35.88, 0% перемог), що і очікувано через випадковий характер прийняття рішень. Алгоритми **A*** та **BFS** забезпечують подібний рівень продуктивності з середніми рахунками близько 1063–1071, часом виконання приблизно 4 секунди та відсотком перемог 37%. Значне покращення видно при застосуванні стратегій, що враховують опонентську поведінку: агент **AlphaBeta** демонструє середній рахунок 1272.29, час роботи 7.09 секунд і перемоги у 62% випадків, тоді як агент **Expectimax** досягає найкращих показників – середній рахунок 1410.85, час роботи 7.91 секунд і 72% перемог. Це свідчить про те, що розширене прогнозування та врахування невизначеності (як у випадку Expectimax) забезпечують кращу адаптацію в умовах складних сценаріїв.

Таблиця 4.2 демонструє результати агента, що використовує алгоритм **MCTS**, при різній кількості симуляцій у тому ж Small Maze 1. Зі збільшенням кількості симуляцій спостерігається поступове зростання часу виконання (від 7.59 секунд при 101 симуляції до 14.99 секунд при 500 симуляціях), проте середній рахунок коливається в діапазоні 569–666, а відсоток перемог

залишається значно нижчим (від 12% до максимуму 18%). Це може свідчити про те, що, незважаючи на можливість глибшого аналізу за допомогою симуляцій, агент на базі **MCTS** в даних умовах не досягає конкурентної ефективності порівняно з алгоритмами, орієнтованими на прогнозування опонентської поведінки.

Таким чином, результати свдчать, що серед досліджуваних підходів агентів з алгоритмами, орієнтованими на пряме прогнозування (**Expectimax**, **AlphaBeta**), демонструють кращу продуктивність у термінах середнього рахунку та відсотка успішного виконання, поставлених перед агентом, задач. Алгоритм **MCTS**, хоча й є перспективним методом для моделювання невизначених сценаріїв, потребує додаткової оптимізації для досягнення конкурентних результатів у порівнянні з пошуковими та мультиагентними алгоритмами в умовах вертикально симетричної моделі території.

Таблиця 4.3. Результати у Small Maze 2

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	61.73	2.68	0
A* Agent	963.92	3.62	49
BFS Agent	936.20	3.58	40
AlphaBeta Agent	1098.27	7.24	17
Expectimax Agent	1212.56	17.03	44

Таблиця 4.4. Результати агента MCTS у Small Maze 2

Кількість симуляцій	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
101	784.41	9.28	9
200	717.09	10.81	10
300	732.33	12.98	9
400	797.49	16.16	7
500	900.73	21.22	12

У таблиці 4.3 наведено результати роботи роботизованих систем у 2D моделі середовища типу Small Maze 2. Серед базового варіанту роботизована система, що працює за випадковим алгоритмом (**Random Agent**), демонструє дуже низьку ефективність (середній рахунок 61.73, 0% успішних виконань завдань), що пояснюється відсутністю будь-якого алгоритмічного планування. Алгоритми **A*** та BFS забезпечують подібну продуктивність – середні рахунки знаходяться у діапазоні 936.20–963.92, час роботи близько 3.6 секунд, а відсоток успішного виконання завдань коливається від 40% до 49%.

Цікаво відзначити, що роботизована система, що використовує стратегії, засновані на прогнозуванні поведінки роботизованих систем супротивників, демонструє змішану ефективність:

- Агент з алгоритмом **AlphaBeta** досягає середнього рахунку 1098.27 при часі роботи 7.24 секунд, проте її показник успішності становить лише 17%.
- Натомість агент, що працює за алгоритмом **Expectimax**, отримує найвищий середній рахунок – 1212.56, проте час її виконання зростає до 17.03 секунд, а відсоток успішних завдань – 44%.

Ці результати свідчать про те, що підходи, орієнтовані на прогнозування поведінки супротивних систем, здатні забезпечити вищу якість прийняття

рішень, але при цьому можуть вимагати більших обчислювальних ресурсів, що відображається у збільшеному часі роботи.

Таблиця 4.4 демонструє результати роботи роботизованої системи з використанням алгоритму **MCTS** у тій же 2D моделі середовища (Small Maze 2) при різній кількості симуляцій. Зі збільшенням кількості симуляцій спостерігається поступове зростання часу обчислень (від 9.28 секунд при 101 симуляції до 21.22 секунд при 500 симуляціях). Середній рахунок коливається від 717.09 до 900.73, а відсоток успішних виконань завдань залишається низьким – в межах 7–12%. Це свідчить про те, що, незважаючи на потенціал алгоритму **MCTS** для глибшого аналізу стратегічних сценаріїв, у даних умовах його ефективність значно поступається системам, що застосовують пряме прогнозування поведінки супротивників.

Таким чином, результати таблиць 4.3 і 4.4 вказують на те, що роботизовані системи, які використовують алгоритми на базі пошуку (**A*, BFS**) демонструють стабільну продуктивність у 2D моделях малого середовища з мінімальними обчислювальними витратами. Підходи, орієнтовані на прогнозування (**Expectimax, AlphaBeta**), дозволяють досягати вищих рахунків, хоча й за рахунок збільшення часу обробки, що може бути критичним у реальних застосуваннях. Алгоритм **MCTS**, попри перспективність для моделювання невизначених ситуацій, потребує додаткової оптимізації, оскільки його продуктивність поки що не відповідає вимогам щодо швидкого прийняття рішень у конкурентному середовищі роботизованих систем супротивників.

Таблиця 4.5. Результати у Medium Maze 1 (Вертикально симметричному)

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	86.13	5.77	0
A* Agent	847.12	4.84	5
BFS Agent	863.19	4.86	6
AlphaBeta Agent	1427.44	9.12	68
Expectimax Agent	1424.39	9.46	75

Таблиця 4.6. Результати агента MCTS у Medium Maze 1 (Вертикально симметричному)

Кількість симуляцій	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
101	1093.71	17.95	34
200	1103.57	23.71	33
300	1122.53	39.19	34
400	1126.94	49.39	36
500	1188.56	60.29	41

У таблиці 4.5 наведено результати роботи роботизованих систем у 2D моделі середовища типу Medium Maze 1 з вертикально симетричною конфігурацією. Серед базових систем, що працюють за випадковими алгоритмами (**Random Agent**), ефективність є мінімальною (середній рахунок 86.13, 0% успішних виконань завдань), що пояснюється відсутністю планування та адаптивних стратегій. Алгоритми **A*** та **BFS** демонструють дещо кращі результати – середні рахунки становлять близько 847–863, час роботи складає приблизно 4.84–4.86 секунд, а відсоток успішного виконання завдань не перевищує 6%.

Натомість, роботизовані системи, які використовують алгоритми, орієнтовані на прогнозування поведінки систем супротивників, досягають значного покращення показників. Система з алгоритмом **AlphaBeta** отримує середній рахунок 1427.44, час роботи 9.12 секунд і демонструє 68% успішних виконань завдань, тоді як система, що працює за алгоритмом **Expectimax**, досягає майже таких же рахунків (1424.39) при трохи більш тривалому часі (9.46 секунд) і демонструє найкращий показник – 75% успішних виконань. Це свідчить про те, що врахування сценаріїв поведінки супротивних роботизованих систем забезпечує кращу адаптацію до умов середовища та підвищує ефективність прийняття рішень.

Таблиця 4.6 демонструє результати роботи роботизованої системи з використанням алгоритму **MCTS** у Medium Maze 1 з вертикально симетричною конфігурацією при різній кількості симуляцій. Зі збільшенням кількості симуляцій спостерігається поступове зростання часу обчислень – від 17.95 секунд при 101 симуляції до 60.29 секунд при 500 симуляціях. Середній рахунок також зростає (від 1093.71 до 1188.56), а відсоток успішного виконання завдань підвищується з 34% до 41%. Проте, незважаючи на деяке покращення показників з додатковою кількістю симуляцій, загальна ефективність системи на базі **MCTS** все ще поступається роботизованим системам з алгоритмами, орієнтованими на пряме прогнозування поведінки супротивників (**AlphaBeta** та **Expectimax**).

Таким чином, результати таблиць 4.5 і 4.6 свідчать, що в умовах 2D моделі вертикально симметричного середовища типу Medium Maze 1 роботизовані системи, які використовують алгоритми прямого прогнозування (**AlphaBeta**, **Expectimax**), демонструють значно вищу продуктивність за показниками середнього рахунку та відсотка успішного виконання завдань. Алгоритм **MCTS**, хоча й показує тенденцію до покращення при збільшенні кількості симуляцій, потребує подальшої оптимізації для досягнення конкурентних результатів у

порівнянні з традиційними пошуковими та мультиагентними підходами в умовах симетричної 2D моделі середовища.

Таблиця 4.7. Результати у Medium Maze 2

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	43.54	2.48	0
A* Agent	1078.98	4.95	54
BFS Agent	1115.53	4.98	61
AlphaBeta Agent	989.38	7.73	18
Expectimax Agent	1129.20	10.61	30

Таблиця 4.8. Результати агента MCTS у Medium Maze 2

Кількість симуляцій	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
101	766.24	15.31	5
200	773.37	25.85	6
300	746.08	24.26	6
400	813.50	56.12	12
500	789.85	73.79	9

У таблиці 4.7 наведено результати роботи роботизованих систем у 2D моделі середовища типу Medium Maze 2. Серед базового варіанту, **Random Agent**, демонструє дуже низьку ефективність (середній рахунок 43.54, 0% успішних виконань завдань), що обґрунтовано відсутністю алгоритмічного планування. Роботизовані системи, що використовують пошукові алгоритми, показують значно кращі результати:

- **A* Agent** досягає середнього рахунку 1078.98 за 4.95 секунд та демонструє успішність 54%,
- **BFS Agent** отримує трохи вищий рахунок – 1115.53, час роботи становить 4.98 секунд, а відсоток успішних виконань завдань дорівнює 61%.

Ці показники свідчать про ефективність традиційних пошукових підходів у моделюванні керування, де прості евристики дозволяють забезпечити швидке планування та адаптацію в 2D середовищі.

Результати роботизованих систем, що орієнтуються на прогнозування поведінки систем супротивників, демонструють дещо інший характер роботи:

- **AlphaBeta Agent** отримує середній рахунок 989.38 при часі роботи 7.73 секунд, але успішність виконання завдань становить лише 18%,
- **Expectimax Agent** демонструє середній рахунок 1129.20, час роботи 10.61 секунд та 30% успішних виконань завдань.

Це свідчить про те, що підходи, орієнтовані на прогнозування, не завжди гарантують кращу ефективність у середовищах з підвищеною структурною складністю, ймовірно через додаткове обчислювальне навантаження, яке впливає на час реагування.

У таблиці 4.8 представлені результати роботи роботизованої системи з використанням алгоритму **MCTS** у Medium Maze 2 при різній кількості симуляцій. Спостерігається, що із збільшенням кількості симуляцій від 101 до 500:

- Середній рахунок варіюється від 746.08 до 813.50,
- Час виконання значно зростає – від 15.31 секунд при 101 симуляції до 73.79 секунд при 500 симуляціях,
- Відсоток успішних виконань завдань залишається на низькому рівні – від 5% до 12%.

Це вказує на те, що хоча алгоритм MCTS дозволяє збільшити глибину аналізу через більшу кількість симуляцій, приріст точності планування

супроводжується значним збільшенням часу обробки, що робить його менш придатним для сценаріїв, де потрібна швидка реакція.

Таким чином, результати таблиць 4.7 і 4.8 вказують на те, що роботизовані системи з використанням пошукових алгоритмів (**A* Agent**, **BFS Agent**) демонструють стабільну продуктивність у 2D моделях середовища з мінімальними обчислювальними витратами. Підходи, орієнтовані на прогнозування поведінки систем супротивників (**AlphaBeta Agent**, **Expectimax Agent**), можуть досягати високих рахунків, проте їхня ефективність часто обмежується збільшеним часом обробки. Алгоритм **MCTS**, хоча й перспективний для моделювання глибших стратегічних сценаріїв, потребує додаткової оптимізації для забезпечення конкурентоспроможної швидкості прийняття рішень у даному середовищі.

Таблиця 4.9. Результати у Large Maze 1 (Вертикально симметричному)

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	80.63	14.31	0
A* Agent	2156.62	18.08	31
BFS Agent	2166.65	18.09	43
AlphaBeta Agent	2382.58	28.56	25
Expectimax Agent	2272.06	33.45	11

Таблиця 4.10. Результати агента MCTS у Large Maze 1 (Вертикально симметричному)

Кількість симуляцій	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
50	1255.41	36.23	1
101	1190.48	41.32	2
200	1369.72	60.83	2
300	1480.29	91.49	0

У таблиці 4.9 представлені результати роботи роботизованих систем у 2D моделі вертикально симметричного середовища типу Large Maze 1 з вертикально симетричною конфігурацією. Серед базових систем, що працюють за випадковими алгоритмами, **Random Agent** демонструє низьку ефективність (середній рахунок 80.63, 0% успішних виконань завдань) через відсутність алгоритмічного планування. Роботизовані системи, які використовують пошукові алгоритми, забезпечують значно вищу продуктивність:

- **A* Agent** досягає середнього рахунку 2156.62 за 18.08 секунд з успішністю 31%,
- **BFS Agent** отримує трохи вищий рахунок – 2166.65, час роботи 18.09 секунд і 43% успішних виконань завдань.

Ці результати свідчать про те, що традиційні підходи до пошуку, засновані на евристичних методах, дозволяють ефективно планувати маршрути та адаптуватися до умов складної 2D моделі середовища, забезпечуючи стабільний результат.

У таблиці 4.10 наведено результати роботи роботизованої системи з використанням алгоритму **MCTS** у Large Maze 1 при різній кількості симуляцій. При збільшенні кількості симуляцій спостерігається наступне:

- При 50 симуляціях система отримує середній рахунок 1255.41 за 36.23 секунд, успішність виконання завдань – лише 1%,
- При 101 симуляції – середній рахунок 1190.48 за 41.32 секунд з успішністю 2%,
- При 200 симуляціях – середній рахунок зростає до 1369.72, час роботи 60.83 секунд, успішність залишається на рівні 2%,
- При 300 симуляціях – середній рахунок досягає 1480.29, але час виконання збільшується до 91.49 секунд і успішність падає до 0%.

Таким чином, результати таблиць 4.9 і 4.10 свідчать про те, що роботизовані системи, які використовують пошукові алгоритми (**A* Agent, BFS Agent**), демонструють кращу продуктивність у 2D моделях середовища типу Large Maze 1 завдяки здатності забезпечувати високі рахунки при помірних обчислювальних витратах. Алгоритм **MCTS**, незважаючи на потенціал для глибшого аналізу стратегічних сценаріїв, в умовах великих моделей середовища вимагає значних обчислювальних ресурсів, що призводить до суттєвого збільшення часу обробки та не дозволяє досягти конкурентних показників за успішністю виконання завдань.

Таблиця 4.11. Результати у Large Maze 2

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	62.61	4.31	0
A* Agent	2854.78	15.44	77
BFS Agent	2836.62	17.08	72
AlphaBeta Agent	2985.85	21.27	14
Expectimax Agent	3307.09	30.23	17

Таблиця 4.12. Результати агента MCTS у Large Maze 2

Кількість симуляцій	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
50	3109.16	37.12	29
101	3129.72	49.7	30
200	3138.23	85.54	32
300	2836.98	906.89	22

У таблиці 4.11 представлені результати роботи роботизованих систем у 2D моделі середовища типу Large Maze 2 з вертикально симетричною конфігурацією. Серед інших систем **Random Agent** демонструє дуже низьку ефективність (середній рахунок 62.61, 0% успішних виконань завдань), що обґрунтовано відсутністю алгоритмічного планування. Роботизовані системи, які використовують пошукові алгоритми, показують значно кращі результати:

- **A* Agent** отримує середній рахунок 2854.78, час роботи 15.44 секунд та досягає успішності 77%,
- **BFS Agent** демонструє трохи вищий рахунок – 2836.62, з часом роботи 17.08 секунд і 72% успішних виконань завдань.

Ці результати свідчать про те, що традиційні пошукові методи ефективно забезпечують планування та прийняття рішень у великих 2D моделях середовища, що відповідають умовам реального застосування роботизованих систем.

Результати роботи роботизованої системи з використанням алгоритму **MCTS** у тому ж середовищі наведені у таблиці 4.12. При збільшенні кількості симуляцій спостерігається наступна динаміка:

- При 50 симуляціях система отримує середній рахунок 3109.16 за 37.12 секунд та досягає успішності 29%,

- При 101 симуляції — середній рахунок 3129.72, час роботи 49.7 секунд і успішність 30%,
- При 200 симуляціях — середній рахунок зростає до 3138.23, час роботи збільшується до 85.54 секунд, а успішність досягає 32%.
- При 300 симуляціях спостерігається зниження середнього рахунку до 2836.98 із різким збільшенням часу виконання до 906.89 секунд і зниженням успішності до 22%.

Таким чином, результати таблиць 4.11 і 4.12 вказують на те, що роботизовані системи, які використовують традиційні пошукові алгоритми (**A* Agent**, **BFS Agent**), демонструють високу продуктивність у 2D моделях великого середовища, забезпечуючи високі рахунки та значну успішність виконання завдань при помірних обчислювальних витратах. Підхід з використанням алгоритму **MCTS**, попри потенціал для моделювання глибших стратегічних сценаріїв, потребує значних обчислювальних ресурсів, що відображається у суттєвому збільшенні часу обчислень. Хоча при оптимальному виборі параметрів **MCTS** може забезпечити конкурентні результати, наразі його ефективність не відповідає вимогам щодо швидкого прийняття рішень у складних умовах конкурентного середовища роботизованих систем супротивників.

Таблиця 4.13. Результати у XLarge Maze 1 (10% стін)

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	120.17	29.14	0
Alpha-Beta Agent	3174.03	192.96	34
Expectimax Agent	2219.85	278.77	8
MCTS Agent (50s)	3906.96	484.85	38
MCTS Agent (101s)	3383.16	354.64	35

У таблиці 4.13 наведено результати роботи роботизованих систем у 2D моделі середовища типу XLarge Maze 1 з конфігурацією, де стіни займають 10% площі лабіринту. Серед базового варіанту, **Random Agent** демонструє низьку ефективність (середній рахунок 120.17, 0% успішних виконань завдань), що пояснюється відсутністю алгоритмічного планування.

Системи, які використовують методи пошуку та прогнозування, показують значне покращення результатів. Зокрема:

- **Alpha-Beta Agent** отримує середній рахунок 3174.03 при часі роботи 192.96 секунд та досягає успішності 34%.
- **Expectimax Agent** демонструє нижчий середній рахунок – 2219.85, але час роботи зростає до 278.77 секунд із успішністю лише 8%.

Результати роботизованих систем, що використовують алгоритм **MCTS**, виглядають досить перспективно:

- **MCTS Agent (50 симуляцій)** показує найвищий середній рахунок – 3906.96, але це досягається за час 484.85 секунд із успішністю 38%.
- **MCTS Agent (101 симуляцій)** отримує середній рахунок 3383.16 при менших обчислювальних витратах (354.64 секунд) та 35% успішних виконань завдань.

Таким чином, результати таблиці 4.13 свідчать про те, що роботизовані системи з використанням алгоритмів на базі прогнозування (як **Alpha-Beta Agent** та **Expectimax Agent**) демонструють кращі результати порівняно з випадковим підходом, проте з різною ефективністю та витратами часу. Алгоритми **MCTS**, попри потенціал для досягнення високих рахунків, потребують значних обчислювальних ресурсів, що впливає на час прийняття рішень. Вибір між конфігураціями **MCTS** (50 симуляцій або 101) залежить від компромісу між досягнутою продуктивністю та витратами часу, що є критичним при моделюванні управління роботизованими системами у великих 2D середовищах.

Таблиця 4.14. Результати у XLarge Maze 2 (15% стін)

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	110.07	31.79	0
Alpha-Beta Agent	2602.32	698.16	12
Expectimax Agent	2183.97	254.42	6
MCTS Agent (50s)	3361.54	290.2	19
MCTS Agent (101s)	3215.14	599.82	25

У таблиці 4.14 наведено результати роботи роботизованих систем у 2D моделі середовища типу XLarge Maze 2 з конфігурацією, де стіни займають 15% площі. Серед базового варіанту, **Random Agent** демонструє низьку ефективність (середній рахунок 110.07, 0% успішних виконань завдань), що пояснюється відсутністю алгоритмічного планування.

Системи, що використовують методи пошуку та прогнозування, показують покращені результати, проте з різними характеристиками щодо швидкості та успішності:

- **Alpha-Beta Agent** отримує середній рахунок 2602.32, але його час обчислень досить високий – 698.16 секунд, що забезпечує лише 12% успішних виконань завдань.
- **Expectimax Agent** демонструє середній рахунок 2183.97 з відносно меншим часом роботи – 254.42 секунд, проте успішність виконання завдань становить лише 6%.

Результати роботизованих систем із застосуванням алгоритму MCTS свідчать про потенціал цього підходу, хоча і з певними компромісами:

- **MCTS Agent (50 симуляцій)** показує середній рахунок 3361.54 за 290.2 секунд із 19% успішних виконань завдань, що є досить перспективним показником у порівнянні з алгоритмами на базі прогнозування.

- **MCTS Agent (101 симуляцій)** отримує трохи нижчий середній рахунок – 3215.14, проте час обчислень зростає до 599.82 секунд, а успішність виконання завдань підвищується до 25%.

Таким чином, результати таблиці 4.14 свідчать про те, що роботизовані системи з використанням традиційних алгоритмів пошуку та прогнозування демонструють певну ефективність у великих 2D моделях середовища з високою кількістю перешкод. Проте підхід із застосуванням алгоритму MCTS, незважаючи на значні обчислювальні витрати, дозволяє досягати високих рахунків та покращувати показники успішності завдань. Вибір між конфігураціями MCTS (50 симуляцій чи 101 симуляції) визначається компромісом між досягнутою продуктивністю та витратами часу, що є критичним аспектом при моделюванні управління роботизованими системами у складних 2D середовищах.

Таблиця 4.15. Результати у XLarge Maze 3 (20% стін)

Агент	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random Agent	183.90	80.48	0
Alpha-Beta Agent	1672.59	200.09	2
Expectimax Agent	1685.06	253.68	0
MCTS Agent (50s)	2042.03	460.64	1
MCTS Agent (101s)	2221.03	298.61	2

У таблиці 4.15 наведено результати роботи роботизованих систем у 2D моделі середовища типу XLarge Maze 3 з конфігурацією, де стіни займають 20% площі. **Random Agent** очікувано демонструє низький результат (середній рахунок 183.90, 0% успішних виконань завдань), що пояснюється відсутністю алгоритмічного планування.

Системи, які використовують алгоритми прогнозування, показують кращі результати, хоча й залишаються низькими у такому складному середовищі:

- **Alpha-Beta Agent** отримує середній рахунок 1672.59 за 200.09 секунд, але досягає лише 2% успішних виконань завдань, що свідчить про труднощі оптимального планування в умовах підвищеної щільності перешкод.
- **Expectimax Agent** демонструє трохи вищий середній рахунок – 1685.06 при часі роботи 253.68 секунд, проте його успішність залишається на рівні 0%, що може бути пов’язано з надмірною обчислювальною затримкою при обробці невизначеності.

Результати роботизованих систем із застосуванням алгоритму MCTS виглядають дещо перспективнішими:

- **MCTS Agent (50 симуляцій)** показує середній рахунок 2042.03 за 460.64 секунд із досягненням 1% успішних виконань завдань.
- **MCTS Agent (101 симуляцій)** отримує найвищий середній рахунок – 2221.03 за 298.61 секунд та досягає 2% успішних виконань завдань.

Таким чином, результати таблиці 4.15 свідчать про те, що навіть у складних умовах 2D моделі середовища з високою щільністю перешкод роботизовані системи, які використовують алгоритми прогнозування та MCTS, демонструють покращені результати порівняно з випадковим підходом. Проте загальна ефективність залишається низькою за показниками успішності виконання завдань, що вказує на значну складність моделювання управління роботизованими системами у середовищах з великою кількістю перешкод. Вибір між конфігураціями MCTS (50 симуляцій або 101) вказує на компроміс між досягнутою продуктивністю та витратами часу, що є критичним фактором при розробці систем управління в умовах високої складності середовища.

Аналіз отриманих результатів дозволяє сформулювати такі основні висновки:

1. **Низька ефективність базових підходів -** Роботизовані системи, що працюють за випадковим алгоритмом (**Random Agent**), у всіх 2D моделях середовища демонструють низький середній рахунок та 0% успішних виконань завдань. Це свідчить про необхідність застосування алгоритмічного планування для досягнення прийнятних результатів.
2. **Переваги пошукових алгоритмів -** У невеликих (Small Maze) та середніх (Medium Maze) 2D моделях середовища алгоритми типу A* та BFS забезпечують стабільну продуктивність з відносно невеликими обчислювальними витратами. У великих моделях (Large Maze) ці підходи дозволяють досягати високих рахунків (наприклад, близько 2156–2166 у Large Maze 1 та понад 2800 у Large Maze 2) з помірним часом виконання, що робить їх ефективним вибором для сценаріїв з реальними обмеженнями ресурсів.
3. **Переваги підходів, орієнтованих на прогнозування поведінки супротивників -** Системи з алгоритмами **Alpha-Beta** та **Expectimax** демонструють значне підвищення середніх рахунків та, в деяких випадках, вищий відсоток успішного виконання завдань, особливо в моделях середнього масштабу (Medium Maze 1). Проте, ці підходи часто супроводжуються збільшенням часу обчислень, що може бути критичним у реальних застосуваннях, де час реакції має велике значення.
4. **Потенціал та обмеження алгоритму MCTS -** Алгоритм MCTS, при збільшенні кількості симуляцій, дозволяє досягати високих рахунків, що свідчить про його потенціал для глибшого аналізу стратегічних сценаріїв. Проте, збільшення кількості симуляцій суттєво впливає на час виконання, а рівень успішних виконань завдань залишається значно нижчим порівняно з підходами, орієнтованими на прогнозування. У великих 2D моделях середовища (XLarge Maze) вибір між конфігураціями MCTS (50 чи 101

симуляцій) визначається компромісом між досягнутою продуктивністю та витратами часу.

5. **Вплив складності середовища -** Зі збільшенням розміру 2D моделі середовища та кількості перешкод (від 10% до 20% стін) спостерігається як загальне зниження ефективності роботизованих систем, так і збільшення часу обчислень. Це вказує на необхідність подальшої оптимізації алгоритмічних підходів для роботи у більш складних умовах, де критично важливо швидко приймати рішення.

Таким чином, отримані результати свідчать, що для моделювання керування роботизованими системами у складних 2D середовищах найбільш ефективними є алгоритми, засновані на пошуку (A*, BFS) та прямому прогнозуванні поведінки супротивних систем (Alpha-Beta, Expectimax). Алгоритм MCTS, попри свій потенціал для аналізу невизначених сценаріїв, потребує додаткової оптимізації, оскільки його обчислювальні витрати можуть бути надто високими для застосування у реальному часі. Ці висновки дозволяють окреслити подальші напрямки досліджень щодо оптимізації алгоритмічних рішень для ефективного управління роботизованими системами у мультиагентних середовищах.

4.5 Використання навченої моделі нейромережі для протидії автономним агентам

У попередніх розділах було проведено детальний аналіз ефективності класичних пошукових та мультиагентних алгоритмів ухвалення рішень при тестуванні у розробленій інформаційній системі на базі гри Pac-Man. Проте всі ці алгоритми мають істотні обмеження, пов’язані з відсутністю механізмів адаптації до динамічно змінюваних умов середовища та стратегій супротивника. Щоб подолати ці недоліки у перспективі, а також показати можливість застосування розробленої технології для тестування методів протидії автономним агентам

групою зкоординованих роботозованих систем, у цьому розділі розглядається підхід, заснований на використанні навченої за допомогою MADDPG (Multi-Agent Deep Deterministic Policy Gradient) нейромережевої моделі, для керування командою агентів.

Основна мета експерименту – оцінити здатність нейромережевої моделі адаптуватися до змін у середовищі, а також порівняти її ефективність із традиційними алгоритмами переслідування (Rule-Based Pursuit and Prediction Algorithms). У даному контексті перевіряється не ефективність алгоритмів автономного агента, представленого Pac-Man'ом, а те, наскільки команда агентів керованих нейромережею у порівнянні з класичними Rule-Based алгоритмами.

Для тестування було використано SM (small-to-medium) лабіринти, що містять різні рівні складності:

1. SM Maze 1 – простий лабіринт із низькою щільністю стін.
2. SM Maze 2 – середній рівень складності, більша кількість перешкод.
3. SM Maze 3 – складний лабіринт із 40% перешкод, що імітує навігацію в обмеженому просторі.

Варіанти керування привидами:

- Rule-Based Ghosts – класичні алгоритми переслідування (Blinky - Rule-Based Pursuit, Pinky - Rule-Based Prediction [32, 33]), які діють за жорстко визначеними правилами.
- MADDPG-Trained Model – привиди, керовані моделлю нейромережі, навчені в автономному середовищі для кооперативного переслідування атономного агента, представленого як Pac-Man.

Метричні показники оцінки залишаються тими ж самими, що і у попредньому розділі.

1. Середній рахунок.
2. Середня тривалість гри (у секундах).
3. Відсоток перемог агента противника.

Таблиця 4.16 – Результати тестування агентів у Small Maze 10w (10% стін)

Агент	**Команда противників**	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random	Rule-Based Algorithms	33.63	2.43	0.00
A* Search	Rule-Based Algorithms	806.91	2.76	68.00
BFS Search	Rule-Based Algorithms	688.45	2.59	58.00
Random	MADDPG-Trained Model	97.63	5.38	0.00
A* Search	MADDPG-Trained Model	408.03	2.41	32.00
BFS Search	MADDPG-Trained Model	393.71	2.51	26.00

У таблиці 4.16 наведено результати тестування роботизованих систем у 2D моделі середовища типу Small Maze 10w з конфігурацією, де стіни займають 10% площі. Результати показують вплив типу команди супротивників на продуктивність систем, що управляються різними алгоритмічними підходами:

- **Random Agent** у комбінації з класичними алгоритмами переслідування (Rule-Based Algorithms) демонструє дуже низьку ефективність (середній рахунок 33.63, час 2.43 секунд, 0% успішних виконань завдань). Аналогічно, при використанні супротивників, керованих нейромережею (MADDPG-Trained Model), **Random Agent** покращує свій середній рахунок до 97.63, проте показник успішних виконань залишається 0%, що підкреслює важливість алгоритмічного планування для досягнення успіху.
- **A* Search** у парі з класичними алгоритмами супротивників досягає високих показників – середній рахунок 806.91, час роботи 2.76 секунд та успішність 68%. При використанні ж команди супротивників, керованої навченою моделлю MADDPG, **A* Search** демонструє зниження

ефективності: середній рахунок падає до 408.03, час роботи залишається подібним (2.41 секунд), а успішність скорочується до 32%.

- **BFS Search** у комбінації з класичними алгоритмами супротивників забезпечує середній рахунок 688.45, час роботи 2.59 секунд та 58% успішних виконань завдань. При використанні проти MADDPG-Trained Model середній рахунок дещо знижується до 393.71, час роботи – 2.51 секунд, а успішність – до 26%.

Таблиця 4.17 – Результати тестування агентів у Small Maze 25w (25% стін)

Агент	**Команда противників**	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random	Rule-Based Algorithms	103.70	2.18	0.00
A* Search	Rule-Based Algorithms	650.12	2.07	46.00
BFS Search	Rule-Based Algorithms	603.70	2.03	41.00
Random	MADDPG-Trained Model	218.19	6.1	0.00
A* Search	MADDPG-Trained Model	200.64	1.57	16.00
BFS Search	MADDPG-Trained Model	633.63	2.15	13.00

У таблиці 4.17 наведено результати тестування роботизованих систем у 2D моделі середовища типу Small Maze 25w з конфігурацією, де стіни займають 25% площі:

- **Random Agent** в парі з класичними алгоритмами супротивників (Rule-Based Algorithms) демонструє середній рахунок 103.70 за 2.18 секунд, при цьому показник успішності завдань становить 0%. При використанні команди супротивників, керованої навченою моделлю MADDPG, показники трохи покращуються – середній рахунок зростає до 218.19, але ефективність залишається недостатньою (час – 6.1 секунд, 0% перемог).

- **A* Search** у комбінації з класичними алгоритмами супротивників досягає середнього рахунку 650.12 за 2.07 секунд, з показником успішності 46%. При взаємодії з MADDPG-Trained Model, ефективність системи знижується – середній рахунок падає до 200.64, час роботи зменшується до 1.57 секунд, а успішність виконання завдань складає лише 16%.
- **BFS Search** демонструє середній рахунок 603.70 за 2.03 секунд з традиційною командою супротивників, що забезпечує 41% успішних виконань завдань. При використанні MADDPG-Trained Model, середній рахунок трохи збільшується до 633.63 за 2.15 секунд, але успішність падає до 13%.

Таблиця 4.18 – Результати тестування агентів у Small Maze 40w (40% стін)

Агент	**Команда противників**	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random	Rule-Based Algorithms	16.76	1.69	0.00
A* Search	Rule-Based Algorithms	767.14	3.86	28.00
BFS Search	Rule-Based Algorithms	932.47	3.49	20.00
Random	MADDPG-Trained Model	118.84	3.38	0.00
A* Search	MADDPG-Trained Model	1027.16	3.27	16.00
BFS Search	MADDPG-Trained Model	984.26	3.2	0.00

У таблиці 4.18 наведено результати тестування роботизованих систем у 2D моделі середовища типу Small Maze 40w з конфігурацією, де стіни займають 40% площі:

- **Random Agent** у парі з класичними алгоритмами супротивників (Rule-Based Algorithms) демонструє дуже низьку ефективність – середній рахунок складає лише 16.76 за 1.69 секунд, з 0% успішних виконань

завдань. При використанні команди супротивників, керованої навченою моделлю MADDPG, показники трохи покращуються (середній рахунок 118.84 за 3.38 секунд), проте успішність залишається 0%.

- **A* Search** у комбінації з традиційними алгоритмами супротивників досягає середнього рахунку 767.14 за 3.86 секунд та демонструє 28% успішних виконань завдань. При взаємодії з MADDPG-Trained Model ефективність дещо знижується – середній рахунок підвищується до 1027.16, час роботи зменшується до 3.27 секунд, але відсоток успішних виконань падає до 16%.
- **BFS Search** показує середній рахунок 932.47 за 3.49 секунд з класичною командою супротивників, що забезпечує 20% успішних виконань завдань. При використанні MADDPG-Trained Model середній рахунок трохи зменшується до 984.26 за 3.2 секунд, проте успішність виконання завдань залишається на 0%.

Таблиця 4.19 – Результати тестування агентів у SM Maze 1 (10% стін)

Агент	**Команда противників**	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random	Rule-Based Algorithms	72.97	3.47	0.00
A* Search	Rule-Based Algorithms	891.81	3.42	63.00
BFS Search	Rule-Based Algorithms	861.50	3.41	63.00
Random	MADDPG-Trained Model	169.34	7.66	0.00
A* Search	MADDPG-Trained Model	763.47	2.78	53.00
BFS Search	MADDPG-Trained Model	853.17	3.06	60.00

У таблиці 4.19 наведено результати тестування роботизованих систем у 2D моделі середовища типу SM Maze 1 з конфігурацією, де стіни займають 10% площі:

- **Random Agent** у парі з класичними алгоритмами супротивників (Rule-Based Algorithms) демонструє середній рахунок 72.97 за 3.47 секунд з 0% успішних виконань завдань. При використанні команди супротивників, керованої навченою моделлю MADDPG, **Random Agent** показує покращення – середній рахунок зростає до 169.34 за 7.66 секунд, проте ефективність залишається невисокою (0% перемог).
- **A* Search** у комбінації з традиційною командою супротивників досягає середнього рахунку 891.81 за 3.42 секунд із показником успішності 63%. При взаємодії з MADDPG-Trained Model, *A Search** демонструє дещо нижчий середній рахунок – 763.47 за 2.78 секунд, а успішність виконання завдань зменшується до 53%.
- **BFS Search** показує схожі результати: при використанні класичних алгоритмів супротивників досягає середнього рахунку 861.50 за 3.41 секунд із 63% успішних виконань завдань. При застосуванні MADDPG-Trained Model середній рахунок підвищується до 853.17 за 3.06 секунд, а відсоток успішних виконань завдань зростає до 60%.

Таблиця 4.20 – Результати тестування агентів у SM Maze 2 (25% стін)

Агент	**Команда противників**	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random	Rule-Based Algorithms	54.37	2.50	0.00
A* Search	Rule-Based Algorithms	738.69	3.64	58.00
BFS Search	Rule-Based Algorithms	851.76	3.49	68.00
Random	MADDPG-Trained Model	127.14	3.72	0.00
A* Search	MADDPG-Trained Model	47.16	1.78	0.00
BFS Search	MADDPG-Trained Model	161.18	2.86	10.00

У таблиці 4.20 наведено результати тестування роботизованих систем у 2D моделі середовища типу SM Maze 2 з конфігурацією, де стіни займають 25% площі:

- **Random Agent** у парі з класичними алгоритмами супротивників (Rule-Based Algorithms) демонструє середній рахунок 54.37 за 2.50 секунд з 0% успішних виконань завдань. При використанні команди супротивників, керованої MADDPG-Trained Model, **Random Agent** показує покращення – середній рахунок зростає до 127.14 за 3.72 секунд, але ефективність залишається низькою (0% перемог).
- **A* Search** у комбінації з класичними алгоритмами супротивників досягає середнього рахунку 738.69 за 3.64 секунд із показником успішності 58%. Проте при взаємодії з MADDPG-Trained Model ефективність різко знижується – середній рахунок падає до 47.16 за 1.78 секунд, а відсоток успішних виконань завдань стає 0%.
- **BFS Search** демонструє середній рахунок 851.76 за 3.49 секунд при традиційній команді супротивників, що забезпечує 68% успішних виконань завдань. При використанні MADDPG-Trained Model середній

рахунок трохи знижується до 161.18 за 2.86 секунд, а відсоток перемог падає до 10%.

Таблиця 4.21 – Результати тестування агентів у SM Maze 3 (40% стін)

Агент	**Команда противників**	**Середній рахунок**	**Час (с)**	**Відсоток перемог (%)**
Random	Rule-Based Algorithms	48.90	3.59	0.00
A* Search	Rule-Based Algorithms	1134.65	4.23	55.00
BFS Search	Rule-Based Algorithms	1003.78	3.91	53.00
Random	MADDPG-Trained Model	77.14	4.43	0.00
A* Search	MADDPG-Trained Model	886.50	3.62	43.00
BFS Search	MADDPG-Trained Model	280.26	2.34	19.00

У таблиці 4.21 наведено результати тестування роботизованих систем у 2D моделі середовища типу SM Maze 3 з конфігурацією, де стіни займають 40% площі. Результати порівнюють ефективність систем, що використовують різні алгоритми управління, при взаємодії з двома типами команд супротивників:

- **Random Agent** у парі з класичними алгоритмами супротивників (Rule-Based Algorithms) демонструє середній рахунок 48.90 за 3.59 секунд із 0% успішних виконань завдань. При використанні команди супротивників, керованої MADDPG-Trained Model, **Random Agent** покращує свій середній рахунок до 77.14 за 4.43 секунд, проте ефективність залишається низькою (0% перемог).
- **A* Search** у комбінації з традиційними алгоритмами супротивників досягає середнього рахунку 1134.65 за 4.23 секунд із показником успішності 55%. При взаємодії з MADDPG-Trained Model, ефективність **A***

Search дещо знижується – середній рахунок падає до 886.50 за 3.62 секунд, а відсоток успішних виконань завдань зменшується до 43%.

- **BFS Search** демонструє середній рахунок 1003.78 за 3.91 секунд при використанні класичних алгоритмів супротивників, що забезпечує 53% успішних виконань завдань. При застосуванні MADDPG-Trained Model, **BFS Search** показує значне зниження ефективності – середній рахунок падає до 280.26 за 2.34 секунд із успішністю лише 19%.

Аналіз отриманих результатів дозволяє сформулювати такі основні висновки:

- Роботизовані системи з алгоритмічним плануванням (**A* Search** та **BFS Search**) демонструють значну перевагу у продуктивності в 2D моделях середовища, особливо при використанні класичних алгоритмів супротивників (Rule-Based Algorithms). Ці системи забезпечують значно вищий середній рахунок та відсоток успішних виконань завдань порівняно з випадковим підходом.
- Використання команди супротивників, керованої навченою нейромережею (MADDPG-Trained Model), призводить до зниження показників ефективності для обох систем пошуку. Для **A* Search** спостерігається зростання середнього рахунку, але зниження успішності виконання завдань, тоді як для **BFS Search** ефект є трохи менш вираженим, але також вагомим. Це свідчить про те, що адаптивні стратегії, здобуті шляхом навчання, створюють додаткові виклики для систем, які покладаються на класичні алгоритмічні методи.
- Загалом, результати досліджень підтверджують, що успіх роботизованих систем у 2D моделях середовища значною мірою залежить від типу використаних алгоритмів управління та стратегії супротивників. Традиційні підходи з класичними алгоритмами забезпечують кращу адаптацію та ефективність, тоді як застосування адаптивних

нейромережевих моделей вимагає подальшої оптимізації для досягнення конкурентних показників у складних умовах мультиагентних середовищ.

Ці висновки підкреслюють важливість ретельного вибору алгоритмічних стратегій при моделюванні керування роботизованими системами, а також вказують на напрямки для подальших досліджень з інтеграції навченої нейромережевої моделі в системи протидії автономним агентам та потенційної комбінації з класичними алгоритмами для розробки гібридних підходів.

Висновки до розділу 4

У четвертому розділі проведено комплексний аналіз ефективності алгоритмів ухвалення рішень у динамічному ігровому середовищі Pac-Man, що використовується як тестовий майданчик для апробації методів автономної навігації та взаємодії мультиагентних систем. Розглянуто як традиційні алгоритми, такі як BFS, A*, Alpha-Beta Pruning, Expectimax і Monte Carlo Tree Search (MCTS), так і навчену нейромережеву модель MADDPG, що дозволяє агентам адаптувати свою поведінку до мінливих умов середовища.

Основні висновки дослідження можна сформулювати таким чином:

1. **Ефективність алгоритмічного планування:** Роботизовані системи, що використовують алгоритми пошуку, такі як **A* Search** та **BFS Search**, демонструють значну перевагу в продуктивності порівняно з випадковим підходом. У невеликих і середніх 2D моделях середовища (Small Maze та Medium Maze) ці методи забезпечують вищі середні рахунки та показники успішності завдань. Результати підтверджують висновки, викладені в роботах [2, 3], де ефективність евристичних алгоритмів у завданнях пошуку шляху була доведена, що дозволяє швидко планувати маршрути за мінімальних обчислювальних витрат.
2. **Переваги підходів, орієнтованих на прогнозування поведінки супротивників:** Алгоритми, що інтегрують механізми прогнозування, такі

як **Alpha-Beta** та **Expectimax**, забезпечують додаткову адаптивність у сценаріях, де взаємодія з роботизованими системами супротивників створює додаткові виклики. Особливо це стосується середовищ середнього масштабу (Medium Maze 1), де застосування цих методів дозволяє досягти вищих рахунків та покращеної успішності виконання завдань. Як зазначають автори роботи [67], Alpha-Beta Pruning є оптимальним для зменшення обчислювальних витрат у детермінованих умовах, однак його продуктивність знижується у великих пошукових просторах. Однак збільшення часу обчислень свідчить про компроміс між точністю прогнозування та швидкістю реакції, що є критичним у реальних застосуваннях. Згідно з дослідженням [68], для ефективного використання Alpha-Beta Pruning у складних середовищах потребується оптимізація пошуку, і як один з варіантів пропонуються транспозиційні таблиці. Крім того, застосування оптимізаційних технік, запропонованих у роботі [69], також може зменшити ці обмеження.

3. **Обмеження та потенціал алгоритму MCTS:** Алгоритм MCTS демонструє високий потенціал завдяки можливості проведення численних симуляцій, що сприяє глибшому аналізу стратегічних сценаріїв. Результати експериментів у великих 2D моделях середовища (XLarge Maze) підтверджують, що при збільшенні кількості симуляцій MCTS може досягати високих рахунків, проте його застосування супроводжується суттєвими обчислювальними витратами та збільшеним часом виконання. Результати експериментів свідчать про високу ефективність MCTS Agent у порівнянні з іншими алгоритмами, особливо у складних середовищах, таких як лабіринти з 20% стін. Ця тенденція відповідає висновкам авторів дослідження [70], де наголошується на гнучкості MCTS у динамічних середовищах. MCTS забезпечує баланс між дослідженням та експлуатацією, що дозволяє йому адаптуватися до мінливих умов. Подібні

результати отримали також у дослідженні [71], де демонстрували, що MCTS може досягати високої продуктивності навіть у реальному часі, завдяки застосуванню стратегій змінної глибини дерева та повторного використання дерева пошуку. Модифіковані версії MCTS, такі як ті, що описані в роботі [72], демонструють подібну ефективність у адаптації до динамічних умов завдяки інтеграції методів машинного навчання, зокрема штучних нейронних мереж. Ці методи дозволяють покращити швидкість конвергенції та оптимізацію, що може стати подальшим перспективним напрямом саме у складних динамічних середовищах і у контексті місій БПЛА або суміжних задач з використанням роботів.

4. **Вплив складності середовища:** Зі збільшенням розміру 2D моделі середовища та кількості перешкод (наприклад, збільшення щільності стін від 10% до 40%) спостерігається загальне зниження ефективності роботизованих систем та зростання часу обчислень. Це свідчить про необхідність подальшої оптимізації алгоритмічних підходів для роботи у складних умовах, де критично важливо швидко приймати рішення.
5. **Використання навченої нейромережевої моделі для протидії автономним агентам:** Результати тестування показують, що застосування адаптивної стратегії за допомогою MADDPG-Trained Model, хоча і має потенціал для координації дій у мультиагентних системах, створює додаткові виклики для традиційних алгоритмів пошуку. Як і зазначалося в роботах [5, 66], інтеграція нейромережевих підходів може покращити координацію та адаптивність, але наші результати демонструють, що без подальшої оптимізації ці моделі знижують ефективність систем, що керуються класичними алгоритмічними методами.
6. **Перспективи подальшого розвитку та інтеграції:** Дослідження підтверджують важливість інтеграції навченої нейромережевої моделі в систему протидії автономним агентам. Зокрема, перспективним напрямом

є розробка гібридних підходів, що комбінують класичні алгоритмічні методи з адаптивними нейромережевими стратегіями. Подібні підходи вже отримали підтвердження у роботах [73, 74], де інтеграція методів навчання з підкріпленням сприяла більш гнучкому реагуванню у складних мультиагентних сценаріях, проте інтеграція таких моделей потребує додаткової оптимізації для досягнення стабільних результатів. Також перспективними є дослідження з розширення масштабів моделювання, включаючи 3D-середовища, а також інтеграція аспектів кібербезпеки для захисту роботизованих систем, як це обговорюється у роботах [75, 76, 77]. Інтеграція міркувань кібербезпеки, як обговорюють у роботі [75], може дати цінну інформацію для оцінки і пом'якшення загроз, пов'язаних зі штучним інтелектом, в системах БПЛА, забезпечуючи надійність стратегій супротивника і їх застосування в безпечних місіях. Моделювання реальних загроз, таких як системи протидії безпілотникам, радіоелектронна боротьба (РЕБ) та інші військові фактори, може ще більше підвищити реалістичність цих симуляцій. Аналіз моделей безпеки операцій БПЛА з урахуванням кібербезпеки, проведений у роботі [77], також підкреслює важливість узгодження методологій на основі ШІ, таких як MADDPG, з надійними системами безпеки і захисту для усунення вразливостей в умовах супротивника. Подальші дослідження можуть бути зосереджені на додаванні відповідних моделей загроз, а також інтеграції алгоритмів для динамічного перегляду маршрутів залежно від зміни параметрів ризику.Впровадження досягнень в гібридних сенсорних мережах, як досліджували автори роботи [76], може підвищити оперативну надійність команд БПЛА, особливо для місій, що вимагають моніторингу навколишнього середовища і надзвичайних ситуацій. Методи надійного зв'язку LiFi в умовах перешкод, запропоновані у роботах [22, 78], можуть

доповнити майбутні експерименти, вирішуючи проблеми багатоагентної координації в умовах обмежених можливостей зв'язку.

Отже, результати експериментів свідчать про те, що для моделювання керування роботизованими системами у складних 2D середовищах найбільш ефективними є алгоритми, засновані на пошуку (A* Search та BFS Search) та прямому прогнозуванні поведінки супротивників (Alpha-Beta, Expectimax). Алгоритм MCTS має значний потенціал для аналізу невизначених сценаріїв, проте його обчислювальні витрати потребують оптимізації для застосування в реальному часі. Крім того, застосування адаптивних нейромережевих моделей, навчених за допомогою MADDPG, демонструє перспективність, але викликає додаткові виклики для класичних алгоритмічних методів, що вказує на необхідність розробки гібридних рішень. Ці результати не лише підтверджують висновки попередніх досліджень, згаданих вище, але й відкривають нові перспективи для подальшого розвитку систем управління роботизованими системами в мультиагентних середовищах, зокрема для завдань керування БПЛА, мобільних роботів та інших систем, де критично важливо забезпечити адаптивність, швидкість реагування та високий рівень координації дій.

РОЗДІЛ 5
АЛЬТЕРНАТИВНІ ПІДХОДИ ДО КЕРУВАННЯ АГЕНТАМИ У СКЛАДНИХ СЕРЕДОВИЩАХ

5.1 Вступ та мотивація

Результати експериментів, представлених у розділі 4, зокрема використання навченої моделі нейромережі за допомогою MADDPG для протидії автономним агентам, надихнули нас дослідити альтернативні нейромережеві підходи для управління роботизованими системами у складних 2D середовищах. У цьому розділі ми пропонуємо теоретичний огляд альтернативних нейронних мережевих архітектур, таких як рекурентні нейронні мережі (RNN) та модифікації у вигляді мережі довготривалої короткочасної пам'яті (LSTM), що потенційно можуть бути імплементовані у подальших дослідженнях і протестовані в розробленій інформаційній технології.

Нинішня реалізація MADDPG базується на використанні багатошарових перцептронів (MLP), які є простими у використанні завдяки своїй прямолінійній, feedforward-архітектурі. Модель MLP складається з вхідного шару, двох прихованих шарів із ReLU-активацією та вихідного шару з лінійною активацією, що дозволяє ефективно здійснювати як політику (actor), так і оцінку Q-функції (critic) у MADDPG [5]. Цей підхід добре підходить для завдань, де вхід має фіксований розмір і не потребує обробки послідовних даних, як-от у поточній реалізації інформаційної системи для прийняття рішень у реальному часі.

Однак, для завдань, пов'язаних із прогнозуванням траєкторії руху та довгостроковим плануванням, класичний MLP може бути обмеженим через відсутність механізму збереження попереднього контексту. На відміну від нього, RNN та LSTM здатні моделювати часові залежності завдяки своїм рекурентним зв'язкам, що дозволяє враховувати попередні стани при прийнятті рішень.

Переваги RNN/LSTM порівняно з MLP:

- **Обробка послідовностей та пам’ять:** RNN/LSTM можуть ефективно працювати з даними, що змінюються з часом, зберігаючи інформацію про попередні стани. Це робить їх корисними для прогнозування траєкторій у динамічних середовищах, де важливо враховувати історію руху.
- **Адаптація до змін:** Завдяки здатності моделювати послідовності, ці мережі можуть краще адаптуватися до непередбачуваних змін у поведінці супротивних систем, що може суттєво покращити координацію дій у мультиагентних системах.

Недоліки RNN/LSTM:

- **Обчислювальні витрати:** Навчання рекурентних мереж є більш складним і повільним через використання backpropagation through time (BPTT) та можливість виникнення проблем зникання або вибуху градієнтів.
- **Потреба у великій кількості даних:** Ефективне навчання RNN/LSTM часто вимагає значних обсягів даних, що може бути викликом у реальному часі.
- **Складність налаштування:** Різноманітність гіперпараметрів та складність архітектур можуть ускладнювати оптимізацію моделі порівняно з простішими MLP.

Аргументація необхідності вивчення альтернативного підходу базується на прагненні забезпечити більш гнучке та точне прогнозування траєкторії руху в умовах динамічних середовищ, що може бути корисним для розробки більш адаптивних стратегій управління роботизованими системами супротивників. Як зазначалося у дослідженні [5], впровадження методів глибинного навчання, що орієнтовані на обробку послідовностей, може значно покращити координацію та адаптивність у мультиагентних системах.

5.2 Сучасні нейронні мережеві архітектури для прогнозування траєкторій

Прогнозування траєкторій у динамічних середовищах є однією з ключових задач у багатьох застосуваннях, зокрема при керуванні роботизованими системами. Сучасні нейронні мережеві архітектури (Див. Рис 5.1) значно розширюють можливості вирішення цієї задачі за рахунок здатності моделювати часові залежності та нелінійні взаємозв'язки в послідовних даних, що продемонстровано у роботах [79, 80].

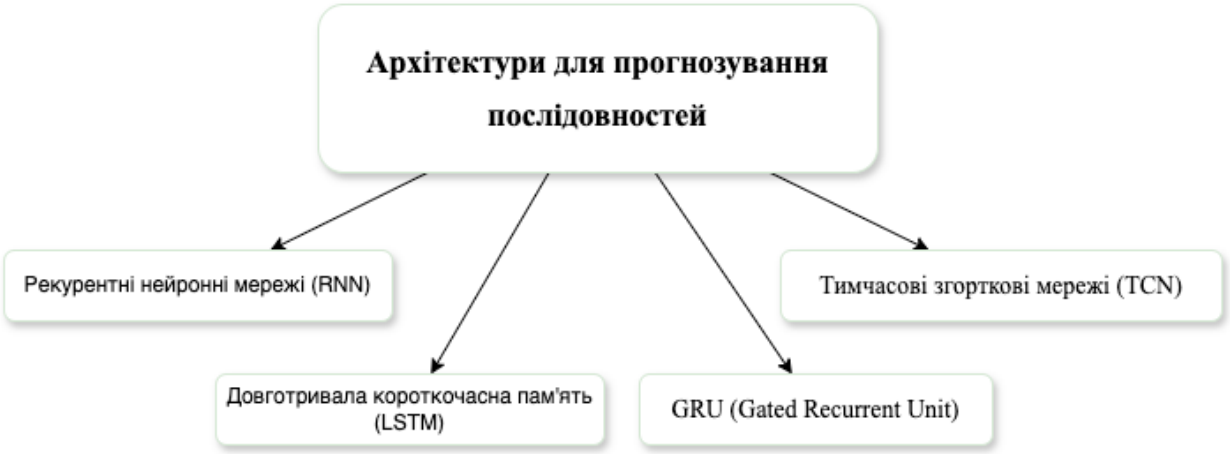


Рис. 5.1 – Архітектури для прогнозування послідовностей

Одним із основних підходів є використання рекурентних нейронних мереж (RNN). Завдяки своїм рекурентним зв'язкам, RNN здатні зберігати інформацію про попередні стани у вигляді прихованих векторів, що дозволяє враховувати контекст при обробці послідовностей. Це робить RNN корисними для прогнозування траєкторій руху, де важливе врахування попередніх кроків агентів та стану середовища, як показано у роботах [81, 82].

Однак класичні RNN мають обмеження, пов'язані з проблемами зникання або вибуху градієнтів, що ускладнює навчання довгих послідовностей. Для подолання цих проблем були запропоновані покращені архітектури, такі як мережі з довготривалою короткочасною пам'яттю (LSTM).

LSTM використовують спеціальні блоки пам’яті, що включають вентилі забування, запису та виходу, які дозволяють ефективно зберігати інформацію на тривалих часових відрізках. Як зазанчаено у роботах [83, 84], завдяки цьому LSTM можуть враховувати як короткострокові, так і довгострокові залежності, що є важливим для точного прогнозування траєкторій руху, наприклад, у грі Pac-Man.

Крім RNN та LSTM, існують інші сучасні підходи. Наприклад, GRU (Gated Recurrent Unit) є спрощеною версією LSTM із меншою кількістю параметрів, що забезпечує більш обчислювальну ефективність без значної втрати якості прогнозування, що розгялянуті у роботах [85, 86]. Інші архітектури, такі як Temporal Convolutional Networks (TCN), використовують згорткові нейронні мережі для обробки послідовних даних, що дозволяє моделювати часові залежності з використанням згорткових операцій, наприклад, у роботі [7]. Такі підходи можуть бути ефективними для задач прогнозування руху в умовах, де важлива швидкість обчислень.

У контексті управління роботизованими системами для виконання місій у динамічних середовищах, прогнозування траєкторії руху агента є критичною задачею. Нейронна мережа аналізує поточне положення агента, його супротивників та інших об’єктів на полі, що дозволяє зробити оптимальний вибір наступного кроку. Використання архітектур RNN та LSTM може суттєво покращити адаптивність систем завдяки здатності враховувати історичний контекст та виявляти повторювані патерни в поведінці агентів, що підтверджується у роботі [80].

Отже, сучасні нейронні мережеві архітектури, такі як RNN, LSTM, GRU та TCN, представляють значний потенціал для задач прогнозування траєкторій. Вони дозволяють враховувати часові залежності та контекст попередніх дій, що є надзвичайно важливим у сценаріях з високою динамікою та непередбачуваністю, характерних для управління роботизованими системами. Ці

архітектури, незважаючи на певні обчислювальні витрати та складність навчання, відкривають нові можливості для розробки адаптивних та ефективних систем прогнозування, що можуть бути інтегровані з іншими алгоритмами для створення гібридних підходів у подальших дослідженнях.

5.3 Гібридні підходи: поєднання нейронних мереж із методами навчання з підкріпленням

Нейронні мережі можуть бути ефективно інтегровані з іншими алгоритмами штучного інтелекту, такими як генетичні алгоритми, розглянуті у роботі [87] або методи навчання з підкріпленням, наприклад, як розглянуті у роботі [82]. Це дозволяє створювати гібридні моделі, які можуть поєднувати в собі найкращі риси різних підходів. Наприклад, нейронні мережі можуть використовуватися для прогнозування траєкторії, а методи навчання з підкріпленням — для коригування поведінки агентів на основі поточних результатів взаємодії та стану середовища. Такі гібридні моделі дозволяють рухомим системам керованими нейромережею навчатися оптимальних стратегій у реальному часі, що робить їхні дії більш складними і непередбачуваними для супротивників.

Гібридні підходи спрямовані на поєднання сильних сторін різних методів штучного інтелекту для досягнення більшої гнучкості та адаптивності у керуванні роботизованими системами, згідно з роботою [82]. Використання нейронних мереж для прогнозування траєкторій у поєднанні з методами навчання з підкріпленням дозволяє створювати системи, здатні як передбачати майбутні стани, так і коригувати свою поведінку на основі зворотного зв’язку з середовища.

Одним із перспективних напрямків є інтеграція можливостей прогнозування нейронних мереж з алгоритмами навчання з підкріпленням, що дозволяє отримати оптимальні стратегії в реальному часі, як це показано у роботі [88]. Наприклад, нейронна мережа може використовуватися для аналізу

послідовностей даних та прогнозування траєкторії руху агента, а алгоритм навчання з підкріпленням – для коригування дій агентів на основі поточних результатів взаємодії зі середовищем, на прикладі робіт [80, 88].

Одним із перспективних підходів для адаптивного керування агентами є застосування навчання з підкріпленням (Reinforcement Learning, RL). У цьому підході агент отримує винагороду або штраф за свої дії та використовує ці дані для поступового вдосконалення своєї стратегії. У контексті мультиагентних систем найбільш ефективними вважаються методи Deep Q-Networks (DQN), згадані у роботі [89], Policy Gradient Methods, розглянуті у дсліджені [90] та Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [5], які дозволяють координувати дії декількох агентів та формувати ефективні командні стратегії.

Такий підхід дозволяє агентам не лише передбачати майбутні позиції супротивника, але й активно коригувати свою стратегію, адаптуючись до змін у середовищі. Гібридна модель може, наприклад, застосовувати нейронну мережу для попереднього аналізу динаміки руху супротивника, а потім використовувати методи навчання з підкріпленням для визначення оптимальних дій на основі отриманих прогнозів, наприклід, як показано у роботах [90, 91]. Цей підхід об’єднує переваги високої точності прогнозування нейронних мереж і здатності навчання з підкріпленням до оптимізації стратегій у невизначених умовах.

Переваги такого гібридного підходу полягають у наступному (Див. Рис 5.2):

- **Покращення адаптивності:** Нейронна мережа може виявляти закономірності в історичних даних, тоді як алгоритми з підкріпленням адаптують стратегії в режимі реального часу, забезпечуючи гнучкість реагування на зміни у середовищі.
- **Зменшення обчислювальних витрат:** Поєднання може дозволити знизити вимоги до ресурсів, оскільки окремо нейронна мережа може бути оптимізована для швидкого прогнозування, а корекційний алгоритм – для ефективного пошуку оптимальних рішень.

- **Універсальність застосувань:** Гібридні моделі можуть бути застосовані у різних широких задачах, де потрібне прогнозування траєкторій і прийняття рішень в режимі реального часу, наприклад, при управлінні безпілотними літальними апаратами чи мобільними роботами.

Рис. 5.2 – Переваги гібридного підходу

Проте, впровадження гібридних підходів має свої виклики. Серед них можна виділити наступні (Див. Рис. 5.3):

- **Складність інтеграції різних методів:** Поєднання нейронних мереж та алгоритмів навчання з підкріпленням вимагає розробки єдиної архітектури, здатної ефективно працювати з обома компонентами. Це включає узгодження процесів навчання, оптимізації та обміну інформацією між модулями, що може бути складним завданням через різну природу цих методів.
- **Оптимізація гіперпараметрів:** Для досягнення стабільного та ефективного навчання необхідно ретельно налаштувати параметри як нейронної мережі, так і алгоритмів навчання з підкріпленням. Невірно підібрані гіперпараметри можуть призвести до надмірного або недостатнього навчання, що негативно впливає на адаптивність системи.
- **Забезпечення стабільного навчання в умовах високої невизначеності:** Дані для навчання часто містять значний рівень шуму та невизначеності, що ускладнює процес оптимізації моделі. Система має бути здатною

адаптуватися до таких умов, зберігаючи точність прогнозів і швидкість прийняття рішень.

- **Збір та обробка великих обсягів даних:** Ефективне навчання гібридних моделей потребує значного обсягу даних, що охоплюють різноманітні сценарії роботи в складних середовищах. Процес збору, очищення, нормалізації та аугментації даних є трудомістким і обчислювально інтенсивним, що потребує значних ресурсів, як зазначають автори роботи [92].

Рис. 5.3 – Виклики гібридного підходу

Методи штучного інтелекту дозволяють підвищити ефективність підконтрольних роботизованих систем. Наприклад, системи, що базуються на правилах (Finite State Machines), використаних у дослідженнях [93, 94], забезпечують гнучкість у налаштуванні поведінки керованих агентів, дозволяючи їм перемикатися між режимами переслідування, втечі чи патрулювання залежно від ситуації. Проте, цей підхід має недоліки через складність розробки та обмежену здатність враховувати комплексні взаємодії.

У підсумку, гібридні підходи, які поєднують можливості нейронних мереж та алгоритмів навчання з підкріпленням, представляють перспективний

напрямок для розробки більш адаптивних та ефективних стратегій управління роботизованими системами. Подальші дослідження в цьому напрямку мають зосередитися на оптимізації інтегрованих моделей, зменшенні обчислювальних витрат та підвищенні стійкості систем у динамічних мультиагентних середовищах.

Висновки до розділу 5

Згідно з обґрунтованими результатами експериментів з використанням моделі MLP нейромережі, навченої за допомогою MADDPG, що описані в розділі 4 виникає припущення, що подальше впровадження альтернативних нейромережевих архітектур має значний потенціал і потребує додаткових експериментальних досліджень із застосуванням розробленої інформаційної системи.

Застосування нейронних мереж, зокрема рекурентних мереж (RNN) та LSTM, дозволяє адаптувати стратегії роботи команди агентів до змін у середовищі та особливостей дій супротивників. Як показує аналіз існуючих досліджень, такі моделі здатні прогнозувати майбутні дії основної мультиагентної системи на основі історичних даних, що сприяє проактивному плануванню та забезпеченню більш стратегічного блокування можливих маршрутів супротивників.

Гібридні підходи, які інтегрують можливості нейронних мереж з методами навчання з підкріпленням, відкривають нові перспективи для підвищення адаптивності систем управління. Поєднання прогнозування траєкторій із корекційними механізмами, заснованими на винагородах і штрафах, дозволяє створити універсальні системи, здатні динамічно реагувати на зміни умов у мультиагентних середовищах.

Отже, подальше впровадження альтернативних нейронних мережевих архітектур, зокрема RNN і LSTM, має перспективи для оптимізації стратегій

управління командами агентів та супротивниками. Розроблена інформаційна система надає можливість інтегрувати ці підходи для моделювання та тестування, що дозволяє перевірити їхню адаптивність, ефективність і масштабованість у реальних сценаріях взаємодії автономних роботизованих систем.

ВИСНОВКИ

По результатам проведених досліджень, завдання, поставлені у роботі, були успішно вирішені, а мету роботи досягнуто. Нижче наведено основні результати, що підтверджує як новизну, так і практичну значимість дослідження:

1. **Вперше запропоновано і розроблено інформаційну систему для моделювання та тестування алгоритмів керування роботизованими пристроями на основі 2D середовища на базі модифікованої гри Pac-Man.** Запропонований фреймворк дозволяє відтворювати сценарії командної взаємодії як з боку протагоніста (агента, що виконує поставлену місію), так і з боку антагоніста (супротивники), що відкриває широкі можливості для дослідження як кооперативних, так і конкурентних стратегій. Крім того, система характеризується високою гнучкістю: вона дозволяє інтегрувати нові 2D моделі середовищ, впроваджувати різноманітні методи керування та гнучко налаштовувати параметри і пріоритезацію завдань агентів. Це відкриває широкі можливості для дослідження як кооперативних, так і конкурентних стратегій.
2. **Інтегровано запропоновані алгоритмічні підходи та модифікації.** Впроваджено агентів з керуванням традиційними пошуковими алгоритмами (A* Search, BFS Search) та пошуковими алгоритмів з модифікацією на прогнозування поведінки супротивників (Alpha-Beta, Expectimax). Експериментальний аналіз показав, що системи з алгоритмічним плануванням забезпечують значну перевагу у продуктивності у порівнянні з випадковими підходами, як це і очікувалось. Зокрема, у 2D моделях середовища невеликого та середнього масштабу запропоновані алгоритми досягають високих середніх рахунків і забезпечують високий відсоток успішних виконань завдань при помірних витратах часу на прийняття.

3. **Запропоновано та досліджено модифікацію алгоритму MCTS.** Розроблено модифікацію алгоритму Monte Carlo Tree Search (MCTS) з використанням специфічної функції оцінки потенційного стану, що дозволяє планувати маршрут і динамічно реагувати на загрози. Експерименти продемонстрували потенціал MCTS для глибшого аналізу стратегічних сценаріїв, саме на складних варіаціях 2D середовищ, проте його застосування супроводжується значними обчислювальними витратами, що вимагає подальшої оптимізації і досліджень для розгляду перспективи реальних застосувань .
4. **Впроваджено можливість інтеграції нейромережевих моделей до інформаційної технології та синтезовано натреновану нейромережеву модель.** Розроблено спеціальні класи для кооперативного та автономного керування агентами за допомогою моделі нейромережі. Проведено дослідження ефективності навченої моделі, заснованої на MADDPG із використанням MLP нейромережі, для керування роботизованими системами супротивників. Отримані результати підтверджують, що інтеграція нейромережевих підходів може значно покращити адаптивність стратегій управління, проте використання адаптивної моделі супротивників створює додаткові виклики щодо необхідної оптимізації та якості прийняття рішень. Це свідчить про необхідність подальшго розгляду щодо розробки гібридних моделей, що поєднують класичні алгоритмічні методи з можливостями нейронних мереж для прогнозування та адаптації в мультиагентних середовищах.
5. **Проведено комп'ютерне моделювання та обґрунтована експериментальна валідація.** Використання роззробленої інформаціної технології на базі 2D середовища гри Pac-Man дозволило провести серію експериментів, спрямованих на порівняння ефективності різних алгоритмів керування роботизованими системами. Збір та аналіз експериментальних

даних засвідчили коректність розробленої інформаційної технології та її здатність відтворювати реалістичні сценарії командної взаємодії агентів, що має потенціал для дослідження і розробки методів керування з метою застосування у різних областях (керування роботизованими системами, БПЛА, автономні транспортні засоби тощо).

Подальші дослідження з використанням розробленої інформаційної технології у сфері керування роботизованими системами можуть зосередитися на ряді напрямків, таких як додавання підтримки 3D-моделей сероедовища з розподілом висот польоту з метою надання можливості використання технології для моделювання процесу керування БПЛА у наближених до реальних умов, інтеграція аспектів реальних загрозу, включно з кібербезпекою, оптимізація та додавання нових алгоритмів в рамках системи, інтеграція програмно-алгоритмічних підходів до формалізації та оптимізації конфігурацій покриття, додавання функціональності для гібридних підходів з участю людини.

3D-реалізації та розподіл висот польоту. Для розширення можливостей тестування розробленої інформаційної технології перспективним є впровадження 3D-моделей середовища. Це передбачає розробку алгоритмів, що забезпечують розподіл висот у лабіринті згідно з обмеженнями польоту безпілотних літальних апаратів (БПЛА). Подібний підхід дозволить враховувати як різні рівні (висоти) у просторі, так і обмеження на можливі висоти для дронів у певній зоні.

Моделювання реальних загроз та інтеграція аспектів кібербезпеки. Подальші дослідження також можуть включати моделювання реальних загроз, таких як системи протидії дронам, електронна боротьба (EW) та інші військові фактори. Аналіз моделей безпеки для операцій БПЛА, як зазначено у роботі [77], підкреслює необхідність інтеграції методів керування агентами, з надійними системами захисту. Наступні кроки можуть зосередитися на додаванні

відповідних моделей загроз і розробці алгоритмів для динамічного коригування маршрутів залежно від змінних ризикових параметрів. Крім того, врахування досягнень у галузі гібридних сенсорних мереж, як це досліджено у роботі [76], може підвищити надійність роботи команд БПЛА або роботизованих систем, особливо для місій з моніторингу навколишнього середовища та надзвичайних ситуацій. Техніки надійного LiFi зв'язку в середовищах з великою кількістю перешкод, представлені дослідженнями [22, 78], можуть стати корисним доповненням до майбутніх експериментів, спрямованих на вирішення проблем координації в мультиагентних системах під обмеженнями комунікації. Не менш важливим кроком може стати інтеграція міркувань кібербезпеки, як зазначають автори роботи [75], це може надати цінну інформацію для оцінки та пом'якшення загроз, пов'язаних зі штучним інтелектом, в системах БПЛА, забезпечуючи надійність стратегій супротивника та їх застосування в безпечних місіях, що потенційно буде актуальним і для інших видів роботизованих систем.

Оптимізація алгоритмів планування шляху. Використання методів оптимізації, наприклад, алгоритмів на основі оптимізації колоній мурах, які пропонуються у роботах [95, 96], дозволить вирішувати комбінаторні задачі оптимізації, що виникають при плануванні маршрутів у складних середовищах. Це сприятиме підвищенню якості прийняття рішень у реальному часі при розподілі ресурсів і плануванні маршрутів.

Інтеграція програмно-алгоритмічних підходів до формалізації та оптимізації конфігурацій покриття. Використання сучасних методів евклідової комбінаторної оптимізації для формалізації, розрахунку та оптимізації конфігурацій максимального покриття територій, які пропонуються у роботі [97] відкриває нові можливості для вирішення складних задач моніторингу. У тому числі, варто розглянути програмно-алгоритмічний підхід до формалізації, розрахунку та оптимізації конфігурацій максимального покриття, що дозволяє ефективно вирішувати складні задачі моніторингу простору та територій,

запропонований у роботі [98]. Ці підходи дозволяють ефективно вирішувати дискретні оптимізаційні задачі, що стосуються розподілу ресурсів та планування в просторі, що має важливе практичне значення для застосувань у сфері автономних систем.

Гібридні підходи з участю людини. Інтеграція людського контролю в алгоритми керування, як показано у дослідженні [74], відкриває перспективи для створення систем, де людина виступає як частина циклу прийняття рішень. Такий підхід може підвищити адаптивність та надійність управління роботизованими системами у складних реальних умовах, забезпечуючи гнучкість та швидку реакцію на непередбачувані ситуації.

Ці напрями подальших досліджень сприятимуть розширенню функціональних можливостей розробленої інформаційної технології та якості рішень для керування агентами, дозволять адаптувати її до більш складних умов, наближючи до реальних і забезпечать ефективність у тестуванні алгоритмів керування роботизованими пристроями у сценаріях командної взаємодії у мультиагентних системах

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Farley, A., Wang, J., & Marshall, J. A. (2022). How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion. *Simulation Modelling Practice and Theory, 120*, 102629. https://doi.org/10.1016/j.simpat.2022.102629
2. Salem, N., Haneya, H., Balbaid, H., & Asrar, M. (2024). Exploring the maze: A comparative study of path finding algorithms for PAC-Man game. In *2024 21st Learning and Technology Conference (L&T)*, 92–97. https://doi.org/10.1109/LT60077.2024.10469459
3. Zou, Y. (2021). General Pacman AI: Game agent with tree search, adversarial search and model-based RL algorithms. In *Proceedings of the 2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, 253–260. https://doi.org/10.1109/ICBASE53849.2021.00053
4. Xu, D., & Chen, G. (2022). Autonomous and cooperative control of UAV cluster with multi-agent reinforcement learning. *The Aeronautical Journal, 126*(1300), 932–951. https://doi.org/10.1017/aer.2021.112
5. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv*. https://arxiv.org/abs/1706.02275
6. Brotee, S., Kabir, F., Razzaque, M. A., Roy, P., Hassan, M. R., & Hassan, M. M. (2024). Optimizing UAV-UGV coalition operations: A hybrid clustering and multi-agent reinforcement learning approach for path planning in obstructed environment. *arXiv*. https://arxiv.org/abs/2401.01481
7. Huang, J., & Ding, W. (2022). Aircraft trajectory prediction based on Bayesian optimised temporal convolutional network–bidirectional gated recurrent unit

hybrid neural network. *International Journal of Aerospace Engineering, 2022*, Article 2086904. https://doi.org/10.1155/2022/2086904

8. Meyer, E., Robinson, H., Rasheed, A., & San, O. (2020). Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning. *IEEE Access, 8*, 41466–41481. https://doi.org/10.1109/ACCESS.2020.2976586
9. Pasumarti, S., Sai, Y. P., Revathi, V., Boopalan, G., & Shanmugasundaram, S. (2024). Comparative analysis of neural network models for autonomous driving: A case study of the NVIDIA model and the DeepDriving model. In *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)* 1–6. https://doi.org/10.1109/AIIoT58432.2024.10574744
10. Austin, J., Corrales-Fatou, R., Wyetzner, S., & Lipson, H. (2020). Titan: A parallel asynchronous library for multi-agent and soft-body robotics using NVIDIA CUDA. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 7754–7760. https://doi.org/10.1109/ICRA40945.2020.9196808
11. UC Barkeley. (2024). CS 188 Spring 2024. (n.d.). https://inst.eecs.berkeley.edu/~cs188/sp24/
12. Mechan, F., Bartonicek, Z., Malone, D., & Lees, R. (2023). Unmanned aerial vehicles for surveillance and control of vectors of malaria and other vector-borne diseases. *Malaria Journal, 22*, Article 23. https://doi.org/10.1186/s12936-022-04414-0
13. Zahínos, R., Abaunza, H., Murillo, J. I., Trujillo, M. A., & Viguria, A. (2022). Cooperative multi-UAV system for surveillance and search & rescue operations over a mobile 5G node. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. 1016–1024. https://doi.org/10.1109/ICUAS54217.2022.9836167

14. Khan, A., Gupta, S., & Gupta, S. K. (2022). Emerging UAV technology for disaster detection, mitigation, response, and preparedness. *Journal of Field Robotics, 39*(6), 905–955. https://doi.org/10.1002/rob.22075
15. Hewawasam, H. S., Ibrahim, M. Y., & Appuhamillage, G. K. (2022). Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments. *IEEE Open Journal of the Industrial Electronics Society, 3*, 353–365. https://doi.org/10.1109/OJIES.2022.3179617
16. Bernardo, R., Sousa, J. M. C., & Gonçalves, P. J. S. (2022). Survey on robotic systems for internal logistics. *Journal of Manufacturing Systems, 65*, 339–350. https://doi.org/10.1016/j.jmsy.2022.09.014
17. Reinisch, A., Liese, J., Padberg, W., & Ulrich, F. (2023). Robotic operations in urgent general surgery: a systematic review. *Journal Robotic Surgery 17,* 275–290. https://doi.org/10.1007/s11701-022-01425-6
18. Causa, F., & Fasano, G. (2020). Navigation-aware path planning for multiple UAVs in urban environment. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. 1–10. https://doi.org/10.1109/DASC50938.2020.9256724
19. Beatty, E., & Dong, E. (2023). Multi-agent reinforcement learning for UAV sensor management. In *Proceedings of SPIE 12544, Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation 2023* (125440H). https://doi.org/10.1117/12.2669269
20. Kouzeghar, M., Song, Y., Meghjani, M., & Bouffanais, R. (2023). Multi-target pursuit by a decentralized heterogeneous UAV swarm using deep multi-agent reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 3289–3295. https://doi.org/10.1109/ICRA48891.2023.10160919
21. Fesenko, H., Illiashenko, O., Kharchenko, V., Leichenko, K., Sachenko, A., & Scislo, L. (2024). Methods and software tools for reliable operation of flying LiFi

networks in destruction conditions. *Sensors, 24*(17), 5707. https://doi.org/10.3390/s24175707

22. Leichenko, K., Fesenko, H., Kharchenko, V., & Illiashenko, O. (2024). Deployment of a UAV swarm-based LiFi network in the obstacle-ridden environment: Algorithms of finding the path for UAV placement. *Radioelectronic and Computer Systems, 2024*(1), 176–195. https://doi.org/10.32620/reks.2024.1.14
23. Tahir, M. A., Mir, I., & Islam, T. U. (2023). A review of UAV platforms for autonomous applications: Comprehensive analysis and future directions. *IEEE Access, 11*, 52540–52554. https://doi.org/10.1109/ACCESS.2023.3273780
24. Javaid, S., Saeed, N., Qadir, Z., Fahim, H., He, B., Song, H., & Bilal, M. (2023). Communication and control in collaborative UAVs: Recent advances and future trends. *IEEE Transactions on Intelligent Transportation Systems, 24*(6), 5719–5739. https://doi.org/10.1109/TITS.2023.3248841
25. Moon, J., Papaioannou, S., Laoudias, C., Kolios, P., & Kim, S. (2021). Deep reinforcement learning multi-UAV trajectory control for target tracking. *IEEE Internet of Things Journal, 8*(20), 15441–15455. https://doi.org/10.1109/JIOT.2021.3073973
26. Wang, L., Wang, K., Pan, C., Xu, W., Aslam, N., & Hanzo, L. (2021). Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking, 7*(1), 73–84. https://doi.org/10.1109/TCCN.2020.3027695
27. Ouahouah, S., Bagaa, M., Prados-Garzon, J., & Taleb, T. (2022). Deep-reinforcement-learning-based collision avoidance in UAV environment. *IEEE Internet of Things Journal, 9*(6), 4015–4030. https://doi.org/10.1109/JIOT.2021.3118949

28. Seid, A. M., Boateng, G. O., Mareri, B., Sun, G., & Jiang, W. (2021). Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network. *IEEE Transactions on Network and Service Management, 18*(4), 4531–4547. https://doi.org/10.1109/TNSM.2021.3096673
29. Bayerlein, H., Theile, M., Caccamo, M., & Gesbert, D. (2021). Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open Journal of the Communications Society, 2*, 1171–1187. https://doi.org/10.1109/OJCOMS.2021.3081996
30. Samvelyan, M., Paglieri, D., Jiang, M., & Rocktäschel, T. (2024). Multi-agent diagnostics for robustness via illuminated diversity. *arXiv*. https://arxiv.org/abs/2401.13460
31. Gulati, A., Soni, S., & Rao, S. (2021). Interleaving fast and slow decision making. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 1535–1541. https://doi.org/10.1109/ICRA48506.2021.9561562
32. Iwatani, T. (2010). Toru Iwatani, 1986 PacMan designer. *Programmers At Work.* Retrieved November 22, 2024, from https://programmersatwork.wordpress.com/toru-iwatani-1986-pacman-designer/
33. Birch, C. (2010). Understanding Pac-Man ghost behavior. *GameInternals.* Retrieved November 22, 2024, from https://gameinternals.com/understanding-pac-man-ghost-behavior
34. Russell, L., Goubran, R., & Kwamena, F. (2019). Emerging urban challenge: RPAS/UAVs in cities. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS).* 546–553. https://doi.org/10.1109/DCOSS.2019.00103
35. Isik, O. K., Hong, J., Petrunin, I., & Tsourdos, A. (2020). Integrity analysis for GPS-based navigation of UAVs in urban environment. *Robotics, 9*(3), 66. https://doi.org/10.3390/robotics9030066

36. Everitt, T., & Hutter, M. (2015). Analytical results on the BFS vs. DFS algorithm selection problem. Part I: Tree search. In *Advances in Artificial Intelligence. AI 2015. Lecture Notes in Computer Science*. 9457, 187–198. Springer, Cham. https://doi.org/10.1007/978-3-319-26350-2_14

37. Panigrahi, P. K., & Tripathy, H. K. (2015). Low complexity graph based navigation and path finding of mobile robot using BFS. In *Proceedings of the 2nd International Conference on Perception and Machine Intelligence (PerMIn '15)*. 189–195. https://doi.org/10.1145/2708463.2709068

38. Navya, P., & Ranjith, R. (2021). Analysis of path planning algorithms for service robots in hospital environment. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 1–6. https://doi.org/10.1109/ICCCNT51525.2021.9579474

39. Holte, R. C. (2010). Common misconceptions concerning heuristic search. *Symposium on Combinatorial Search, 1*(1). https://doi.org/10.1609/SOCS.V1I1.18160

40. Park, J. K., & Kim, J. (2019). Collision avoidance method for UAV using A* search algorithm. In *Advances in Intelligent, Interactive Systems and Applications. IISA 2018. Advances in Intelligent Systems and Computing*. 285–297. Springer, Cham. https://doi.org/10.1007/978-3-030-02804-6_25

41. Esakki, B., Marreddy, G., Ganesh, M. S., & Elangovan, E. (2021). Simulation and experimental approach for optimal path planning of UAV using A* and MEA* algorithms. *International Journal of Simulation, Multi-disciplinary Design and Optimization, 12*, Article 24. https://doi.org/10.1051/smdo/2021024

42. Zammit, C., & Van Kampen, E.-J. (2020). Comparison of A* and RRT in real–time 3D path planning of UAVs. In *AIAA SciTech 2020 Forum, Orlando, FL, USA*. https://doi.org/10.2514/6.2020-0861

43. Liu, C., Yan, J., Ma, Y., Zhao, T., Zhang, Q., & Wei, X. (2020). An adversarial search method based on an iterative optimal strategy. *Mathematics, 8*(9), 1623. https://doi.org/10.3390/math8091623

44. Dam, T., Chalvatzaki, G., Peters, J., & Pajarinen, J. (2022). Monte-Carlo robot path planning. *IEEE Robotics and Automation Letters, 7*(4), 11213–11220. https://doi.org/10.1109/LRA.2022.3199674

45. Lövétei, I. F., Kővári, B., & Bécsi, T. (2021). MCTS based approach for solving real-time railway rescheduling problem. *Periodica Polytechnica Transportation Engineering, 49*(3), 283–291. https://doi.org/10.3311/PPtr.18584

46. Li, W., Liu, Y., Ma, Y., Xu, K., Qiu, J., & Gan, Z. (2023). A self-learning Monte Carlo tree search algorithm for robot path planning. *Frontiers in Neurorobotics, 17*. https://doi.org/10.3389/fnbot.2023.1039644

47. Świechowski, M., Godlewski, K., Sawicki, B., et al. (2023). Monte Carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review, 56*, 2497–2562. https://doi.org/10.1007/s10462-022-10228-y

48. Permatasari, B. D., Haryanto, H., Zuni Astuti, E., & Dolphina, E. (2022). Peningkatan kemenangan non-playable character dalam permainan Triple Triad menggunakan Alpha-Beta Pruning. *Jurnal Komputasi.* https://doi.org/10.23960/komputasi.v10i1.2952

49. Mudda, M. (2022). Tic Tac Toe by minimax alpha-beta pruning using Arduino. *International Journal for Research in Applied Science and Engineering Technology*. https://doi.org/10.22214/ijraset.2022.40115

50. Sharma, N. (2022). Introduction to artificial intelligence Fall 2022. Retrieved from https://inst.eecs.berkeley.edu/~cs188/fa22/assets/notes/cs188-fa22-note06.pdf

51. Buttazzo, G. (2023). Rise of artificial general intelligence: Risks and opportunities. *Frontiers in Artificial Intelligence, 6*. https://doi.org/10.3389/frai.2023.1226990

52. Injadat, M., Moubayed, A., Nassif, A. B., et al. (2021). Machine learning towards intelligent systems: Applications, challenges, and opportunities. *Artificial Intelligence Review, 54*, 3299–3348. https://doi.org/10.1007/s10462-020-09948-w

53. Gasparetto, A., Seriani, S., & Scalera, L. (2021). Modelling and control of mechatronic and robotic systems. *Applied Sciences, 11*(7), 3242. https://doi.org/10.3390/app11073242

54. Armanini, C., Boyer, F., Mathew, A. T., Duriez, C., & Renda, F. (2023). Soft robots modeling: A structured overview. *IEEE Transactions on Robotics, 39*(3), 1728–1748. https://doi.org/10.1109/TRO.2022.3231360

55. Foderaro, G., Swingler, A., & Ferrari, S. (2017). A model-based approach to optimizing Ms. Pac-Man game strategies in real time. *IEEE Transactions on Computational Intelligence and AI in Games, 9*(2), 153–165. https://doi.org/10.1109/TCIAIG.2016.2523508

56. Rohlfshagen, P., Liu, J., Perez-Liebana, D., & Lucas, S. M. (2018). Pac-Man conquers academia: Two decades of research using a classic arcade game. *IEEE Transactions on Games, 10*(3), 233–256. https://doi.org/10.1109/TG.2017.2737145

57. Gallagher, M., & Ryan, A. (2003). Learning to play Pac-Man: An evolutionary, rule-based approach. In *Proceedings of the 2003 Congress on Evolutionary Computation*. 4, 2462–2469. https://doi.org/10.1109/CEC.2003.1299397

58. Foderaro, G., Raju, V., & Ferrari, S. (2011). A cell decomposition approach to online evasive path planning and the video game Ms. Pac-Man. In *2011 IEEE International Symposium on Intelligent Control*. 191–197. IEEE. https://doi.org/10.1109/ISIC.2011.6045414

59. Tucnik, P., Nachazel, T., Cech, P., & Bures, V. (2018). Comparative analysis of selected path-planning approaches in large-scale multi-agent-based

environments. *Expert Systems with Applications, 113*, 415–427. https://doi.org/10.1016/j.eswa.2018.07.001

60. Sajid, Q., Luna, R., & Bekris, K. (2021). Multi-agent pathfinding with simultaneous execution of single-agent primitives. In *Proceedings of the 5th Annual Symposium on Combinatorial Search*. pp. 88–96. http://doi.org/10.1609/socs.v3i1.18243
61. Chatzisavvas, A., Dossis, M., & Dasygenis, M. (2024). Optimizing mobile robot navigation based on A-star algorithm for obstacle avoidance in smart agriculture. *Electronics, 13*(11), 2057. https://doi.org/10.3390/electronics13112057
62. Saffidine, A., Finnsson, H., & Buro, M. (2021). Alpha-beta pruning for games with simultaneous moves. *Proceedings of the AAAI Conference on Artificial Intelligence, 26*(1), 556–562. https://doi.org/10.1609/aaai.v26i1.8148
63. Lee, S. (2019). Exploring to learn winning strategy. In *International Conferences Interfaces and Human Computer Interaction, Game and Entertainment Technologies, and Computer Graphics, Visualization, Computer Vision and Image Processing*. 377–380. http://dx.doi.org/10.33965/g2019_201906C052
64. Scheiermann, J., & Konen, W. (2023). AlphaZero-inspired game learning: Faster training by using MCTS only at test time. *IEEE Transactions on Games, 15*(4), 637–647. https://doi.org/10.1109/TG.2022.3206733
65. Tavakol Aghaei, V., Ağababaoğlu, A., Yıldırım, S., & Onat, A. (2022). A real-world application of Markov chain Monte Carlo method for Bayesian trajectory control of a robotic manipulator. *ISA Transactions, 125*, 580–590. https://doi.org/10.1016/j.isatra.2021.06.010
66. Bachiri, K., Yahyaouy, A., Gualous, H., Malek, M., Bennani, Y., Makany, P., & Rogovschi, N. (2023). Multi-agent DDPG based electric vehicles charging station recommendation. *Energies, 16*(16), 6067. https://doi.org/10.3390/en16166067

67. Singhal, S. P., & Sridevi, M. (2019). Comparative study of performance of parallel alpha-beta pruning for different architectures. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. 115–119. https://doi.org/10.1109/IACC48062.2019.8971591

68. Suancha, C. C., Suarez, M. J., & Besoain, F. A. (2024). Implementation of alpha-beta pruning and transposition tables on checkers game. *IEEE Access, 12*, 46636–46645. https://doi.org/10.1109/ACCESS.2024.3381958

69. Shevtekar, P. S., Malpe, M., & Bhaila, M. (2022). Analysis of game tree search algorithms using minimax algorithm and alpha-beta pruning. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), 8*(6), 328–333. https://doi.org/10.32628/CSEIT1228644

70. Maddipati, H., Kundurthi, A., Raaj, P., Srilatha, K., & Surapaneni, R. (2020). Artificial intelligence based Pacman game. *International Journal of Innovative Technology and Exploring Engineering, 9*, 140–144. https://doi.org/10.35940/ijitee.I6975.079920

71. Pepels, T., Winands, M. H. M., & Lanctot, M. (2014). Real-time Monte Carlo tree search in Ms Pac-Man. *IEEE Transactions on Computational Intelligence and AI in Games, 6*(3), 245–257. https://doi.org/10.1109/TCIAIG.2013.2291577

72. Liu, X., Li, Y., He, S., Fu, Y., Yang, J., Ji, D., & Chen, Y. (2009). To create intelligent adaptive game opponent by using Monte-Carlo for the game of Pac-Man. In *Fifth International Conference on Natural Computation*. https://doi.org/10.1109/ICNC.2009.633

73. Ding, R., Xu, Y., Gao, F., & Shen, X. (2022). Trajectory design and access control for air–ground coordinated communications system with multiagent deep reinforcement learning. *IEEE Internet of Things Journal, 9*(8), 5785–5798. https://doi.org/10.1109/JIOT.2021.3062091

74. Chen, L., Liang, H., Pan, Y., & Li, T. (2023). Human-in-the-loop consensus tracking control for UAV systems via an improved prescribed performance approach. *IEEE Transactions on Aerospace and Electronic Systems, 59*(6), 8380–8391. https://doi.org/10.1109/TAES.2023.3304283

75. Veprytska, O. Y., & Kharchenko, V. (2024). Analysis of AI powered attacks and protection of UAV assets: Quality model-based assessing cybersecurity of mobile system for demining. In *5th International Workshop on Intelligent Information Technologies and Systems of Information Security. IntelITSIS'2024, Khmelnytskyi, Ukraine*.

76. Skorobohatko, S., Fesenko, H., Kharchenko, V., & Yakovlev, S. (2024). Architecture and reliability models of hybrid sensor networks for environmental and emergency monitoring systems. *Cybernetics and Systems Analysis, 60*, 293–304. https://doi.org/10.1007/s10559-024-00670-x

77. Illiashenko, O., Kharchenko, V., Babeshko, I., Fesenko, H., & Di Gian-Domenico, F. (2023). Security-informed safety analysis of autonomous transport systems considering AI-powered cyberattacks and protection. *Entropy, 25*(8), Article 1123. https://doi.org/10.3390/e25081123

78. Leichenko, K., Fesenko, H., Borges, J., & Kharchenko, V. (2023). Search for the shortest route considering physical obstacles: Method of controlled waterfall, tool, and application. In *10th International Conference on Desert Systems and Technologies (DESSERT).* 1–5. https://doi.org/10.1109/DESSERT61349.2023.10416479

79. Kubrak, Y., Plechystyi, D., & Tolstoy, I. (2022). Formation of a comprehensive tracking system for modern UAVs based on artificial intelligence. *Bulletin of KrNU named after Mykhailo Ostrohradskyi, (2/2022) 133*, 41–50. https://doi.org/10.32782/1995-0519.2022.2.5

80. Molina-Leal, A., Gómez-Espinosa, A., Escobedo Cabello, J. A., Cuan-Urquizo, E., & Cruz-Ramírez, S. R. (2021). Trajectory planning for a mobile robot in a

dynamic environment using an LSTM neural network. *Applied Sciences, 11*(22), 10689. https://doi.org/10.3390/app112210689

81. Slaughter, I., Charla, J. L., Siderius, M., & Lipor, J. (2024). Vessel trajectory prediction with recurrent neural networks: An evaluation of datasets, features, and architectures. *Journal of Ocean Engineering and Science*. https://doi.org/10.1016/j.joes.2024.01.002
82. Irio, L., & Oliveira, R. (2021). A comparative evaluation of probabilistic and deep learning approaches for vehicular trajectory prediction. *IEEE Open Journal of Vehicular Technology, 2*, 140–150. http://dx.doi.org/10.1109/OJVT.2021.3063125
83. Lin, L., Gong, S., Peeta, S., & Wu, X. (2021). Long short-term memory-based human-driven vehicle longitudinal trajectory prediction in a connected and autonomous vehicle environment. *Transportation Research Record, 2675*, 380–390. https://doi.org/10.1177/0361198121993471
84. Wang, J., Liu, K., & Li, H. (2024). LSTM-based graph attention network for vehicle trajectory prediction. *Computer Networks, 248*, 110477. https://doi.org/10.1016/j.comnet.2024.110477
85. Shu, W., Cai, K., & Xiong, N. N. (2022). A short-term traffic flow prediction model based on an improved gate recurrent unit neural network. *IEEE Transactions on Intelligent Transportation Systems, 23*(9), 16654–16665. https://doi.org/10.1109/TITS.2021.3094659
86. Mateus, B. C., Mendes, M., Torres Farinha, J., Assis, R., & Marques Cardoso, A. (2021). Comparing LSTM and GRU models to predict the condition of a pulp paper press. *Energies, 14*(21), 6958. https://doi.org/10.3390/en14216958
87. Kazemi, H. R., Khalili-Damghani, K., & Sadi-Nezhad, S. (2021). Tuning structural parameters of neural networks using genetic algorithm: A credit scoring application. *Expert Systems, 38*(7), e12733. https://doi.org/10.1111/exsy.12733

88. Choi, D., Yim, J., Baek, M., & Lee, S. (2021). Machine learning-based vehicle trajectory prediction using V2V communications and on-board sensors. *Electronics, 10*(4), 420. https://doi.org/10.3390/electronics10040420

89. Chen, T., He, Z., & Ciocarlie, M. (2021). Co-designing hardware and control for robot hands. *Science Robotics, 6*(54), eabg2133. https://doi.org/10.1126/scirobotics.abg2133

90. Luo, G., Zhang, H., He, H., Li, J., & Wang, F.-Y. (2021). Multiagent adversarial collaborative learning via mean-field theory. *IEEE Transactions on Cybernetics, 51*(10), 4994–5007. https://doi.org/10.1109/TCYB.2020.3025491

91. Capobianco, S., Millefiori, L. M., Forti, N., Braca, P., & Willett, P. (2021). Deep learning methods for vessel trajectory prediction based on recurrent neural networks. *IEEE Transactions on Aerospace and Electronic Systems, 57*(6), 4329–4346. https://doi.org/10.1109/TAES.2021.3096873

92. Lin, L., Gong, S., Peeta, S., & Wu, X. (2021). Long short-term memory-based human-driven vehicle longitudinal trajectory prediction in a connected and autonomous vehicle environment. *Transportation Research Record, 2675*(6), 380–390. https://doi.org/10.1177/0361198121993471

93. Peng, F., Li, Z., Duan, Z., & Xia, Y. (2022). Multi-objective multi-learner robot trajectory prediction method for IoT mobile robot systems. *Electronics, 11*(13), 2094. https://doi.org/10.3390/electronics11132094

94. Rossander, M., & Lideskog, H. (2023). Design and implementation of a control system for an autonomous reforestation machine using finite state machines. *Forests, 14*(7), 1340. https://doi.org/10.3390/f14071340

95. Horbulin, V. P., Hulianytskyi, L. F., & Sergienko, I. V. (2023). Planning of logistics missions of the "UAV+Vehicle" hybrid systems. *Cybernetics and Systems Analysis, 59*, 733–742. https://doi.org/10.1007/s10559-023-00609-8

96. Hulianytskyi, L., & Rybalchenko, O. (2023). Optimization of decisions when planning a UAV group mission with alternative depots. In *Proceedings of the*

International Scientific Symposium "Intelligent Solutions". 245–256. https://ceur-ws.org/Vol-3538/Paper_22.pdf

97. Stoyan, Y. G., & Yakovlev, S. V. (2020). Theory and methods of Euclidian combinatorial optimization: Current status and prospects. *Cybernetics and Systems Analysis, 56*, 366–379. https://doi.org/10.1007/s10559-020-00253-6
98. Yakovlev, S., Kartashov, O., & Mumrienko, A. (2022). Formalization and solution of the maximum area coverage problem using library Shapely for territory monitoring. *Radioelectronic and Computer Systems, 2*(102). https://doi.org/10.32620/reks.2022.2.03

ДОДАТОК 1

Список наукових публікацій здобувача

Публікації, в яких опубліковано основні наукові результати дисертації

Статті у наукових фахових виданнях, що входять до міжнародних наукометричних баз:

1. Yakovlev S., **Novikov A.,** Gushchin I. Exploring the possibilities of MADDPG for UAV swarm control by simulating in Pac-Man environment. *Radioelectronic and Computer Systems*. 1(113). 2025. P. 327-337. http://doi.org/10.32620/reks.2025.1.21 (Scopus).

Статті у наукових фахових виданнях України:

2. **Новіков А.О.,** Маций О.Б. Потенціал використання нейронних мереж для передачення траекторії руху у мультиагентних системах на прикладі гри Pac-Man. *Вісник Кременчуцького національного університету імені Михайла Остроградського*. 2024. 4. С. 92–104. https://doi.org/10.32782/1995-0519.2024.4.12

(Особистий внесок здобувача: проведення аналізу існуючих методів прогнозування траєкторії руху у динамічних середовищах, розробка запропонованого методу прогнозування траєкторії руху Pac-Man за допомогою нейронних мереж, порівняння традиційних та сучасних підходів до підвищення ефективності стратегій агентів.Особистий внесок Ольги Маций: перевірка наукової достовірності отримуваних результатів, перевірка тексту роботи.)

3. **Novikov A.O.,** Yanovsky V.V. Analysis of Search and Multi-Agent Algorithms in the Pac-Man Game. *Control Systems and Computers*. 2024. 308(4). P. 19–33. https://doi.org/10.15407/csc.2024.04.019

(*Особистий внесок здобувача: програмна розробка середовища та імплементація алгоритмічних підходів, аналіз ефективності класичних пошукових алгоритмів, таких як A* та BFS, а також мультиагентних підходів, таких як Alpha-Beta, Expectimax і MCTS, у контексті модифікованого середовища на основій гри Pac-Man. Проведення експериментів з різними умовами гри, включаючи складність лабіринту, поведінку привидів і динаміку середовища. Оцінка впливу цих факторів на час виконання, оцінку та відсоток успішних заверень завдань агента. Відповідні результати наведені в практичній частині роботи.*

Особистий внесок Володимира Яновського: перевірка наукової достовірності отримуваних результатів, перевірка тексту роботи)

4. **Novikov A.O.,** Yanovsky V.V. Exploring the limits of mcts in pac-man: maze size, simulations, and performance. *Herald of Khmelnytskyi National University. Technical sciences*. 341(5). 2024. P. 351–359. https://doi.org/10.31891/2307-5732-2024-341-5-52

(Особистий внесок здобувача: програмна розробка середовища на основі гри Pac-Man та імплементація алгоритму пошуку дерева Монте-Карло (MCTS), аналіз продуктивності та обмежень алгоритму в різних умовах гри, включаючи змінну складність лабіринтів та кількість симуляцій алгоритму. Проведення експериментів з різними розмірами та конфігураціями лабіринтів, оцінка компромісів між обчислювальною вартістю та результативністю агента. Оцінка впливу середовища на ефективність алгоритму та пропозиція щодо покращень за рахунок адаптивних симуляцій та комбінування MCTS з іншими методами. Відповідні результати наведені в практичній частині роботи. Особистий внесок Володимира Яновського: перевірка наукової достовірності отриманих результатів, рецензування тексту статті, внесення корективів у формулювання та структуру роботи, перевірка правильності теоретичних основ дослідження)

5. **Novikov A.O.,** Yanovsky V.V. Analysis of the decision-making algorithm efficiency in complex game environments on the example of Pac-Man. Information Technologies and Computer Engineering. 21(3), P. 108–118. https://doi.org/10.63341/vitce/3.2024.108

(Особистий внесок здобувача: програмна розробка середовища на основі гри Pac-Man та імплементація алгоритмів Expectimax, MCTS та Alpha-Beta Pruning. Проведення експериментів з різними умовами гри, включаючи складність лабіринтів, кількість перешкод та їх вплив на ефективність алгоритмів. Оцінка ефективності алгоритмів на основі таких показників, як кількість балів, час гри та відсоток виграшів. Аналіз результатів експериментів для визначення найефективнішого підходу до ухвалення рішень у складних середовищах. Результати наведені в практичній частині роботи. Особистий внесок Володимира Яновського: перевірка наукової достовірності отриманих результатів, рецензування тексту статті, перевірка правильності інтерпретації результатів експериментів та оптимізації алгоритмів для складних середовищ.)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. **Novikov A.O.,** Matsyi O.B. Enhancing decision-making in multi-agent systems through neural network-based trajectory prediction. *Матеріали XII Міжнародної науково-практичної конференції «Актуальні проблеми сучасної науки та освіти»*. Львів, Україна. 2024. С. 54-56.
2. **Novikov A.O.,** Matsyi O.B. Comparative analysis of trajectory prediction techniques in multi-agent systems: traditional vs neural network approaches. *Матеріали XIII Міжнародної Науково-Практичної Конференції «Актуальні питання розвитку науки та освіти»*. Львів, Україна. 2024. С. 174-177. URL:
3. **Novikov A.,** Yanovsky V. Identification and analysis of shadow zones in artificial neurons. *Science and society: modern trends in a changing world. Proceedings of the 10th International scientific and practical conference*. MDPC Publishing. Vienna, Austria. 2024. P. 45-50.

Онлайн сервіс створення та перевірки кваліфікованого та удосконаленого електронного підпису

ПРОТОКОЛ
створення та перевірки кваліфікованого та удосконаленого електронного підпису

Дата та час: 12:08:21 15.05.2025

Назва файлу з підписом: Novikov_diss.pdf.asice
Розмір файлу з підписом: 2.0 МБ

Перевірені файли:
Назва файлу без підпису: Novikov_diss.pdf
Розмір файлу без підпису: 2.1 МБ

Результат перевірки підпису: Підпис створено та перевірено успішно. Цілісність даних підтверджено

Підписувач: НОВІКОВ АРТЕМ ОЛЕКСАНДРОВИЧ
П.І.Б.: НОВІКОВ АРТЕМ ОЛЕКСАНДРОВИЧ
Країна: Україна
РНОКПП: 3582801959
Організація (установа): ФІЗИЧНА ОСОБА
Час підпису (підтверджено кваліфікованою позначкою часу для підпису від Надавача): 12:08:20 15.05.2025
Сертифікат виданий: КНЕДП АЦСК АТ КБ "ПРИВАТБАНК"
Серійний номер: 5E984D526F82F38F04000000C02C3501E9F0A805
Алгоритм підпису: ДСТУ 4145
Тип підпису: Удосконалений
Тип контейнера: Підпис та дані в архіві (розширений) (ASiC-E)
Формат підпису: З повними даними для перевірки (XAdES-B-LT)
Сертифікат: Кваліфікований

Версія від: 2025.02.05 13:00