

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

До захисту допущено:

В.о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«__» _____ 20__ р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Агент для гри в некласичні шахи з використанням навчання з
підкріпленням»

Виконав:

студент IV курсу, групи КІ-21

Лаптії Назар Сергійович _____

Керівник:

к.ф-м.н, доцент кафедри ШІ

Пишнограєв Іван Олександрович _____

Консультант з нормоконтролю:

фахівець першої категорії кафедри ШІ

к.т.н., доцент,

Комариста Богдана Миколаївна _____

Рецензент:

доцент кафедри економічної кібернетики, к.ф-м.н., доцент

Лазаренко Ірина Сергіївна _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В.о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«15» січня 2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Лаптію Назару Сергійовичу

1. Тема роботи «Агент для гри в неklasичні шахи з використанням навчання з підкріпленням», керівник роботи Пишнограєв Іван Олександрович, затверджені наказом по НН ІПСА від «27» травня 2026 р. № 1944-с.
2. Термін подання студентом роботи «05» червня 2026 року.
3. Вихідні дані до роботи: теоретичні відомості про предметну область неklasичних шахів Chess960; наукові публікації та навчальні матеріали з тематики штучного інтелекту, машинного навчання та навчання з підкріпленням; програмні засоби для розроблення, навчання і тестування інтелектуального ігрового агента; результати експериментальної перевірки роботи створеної системи.
4. Зміст роботи: характеристика предметної області неklasичних шахів Chess960; огляд літератури та існуючих підходів до створення інтелектуальних ігрових агентів; постановка задачі дослідження; розроблення структури та компонентів програмної системи; реалізація ігрового середовища та агента на основі методів навчання з підкріпленням; проведення експериментів та аналіз отриманих результатів.

5. Перелік ілюстративного матеріалу: використані метрики, результати роботи програми, схеми роботи алгоритмів, електронна презентація.

6. Дата видачі завдання «02» лютого 2026 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	20.04.2026	Виконано
2	Підготовка першого розділу	27.04.2026	Виконано
3	Підготовка другого розділу	04.05.2026	Виконано
4	Розробка програмного продукту	11.05.2026	Виконано
5	Підготовка третього розділу	18.05.2026	Виконано
6	Оформлення розділів відповідно до нормоконтролю	25.05.2026	Виконано
7	Оформлення дипломної роботи	01.06.2026	Виконано
8	Підготовка презентації доповіді	05.06.2026	Виконано

Студент

Назар ЛАПТІЙ

Керівник

Іван ПИШНОГРАЄВ

РЕФЕРАТ

Дипломна робота: 73 с., 10 рис., 2 табл., 31 посилання, додаток.

НЕКЛАСИЧНІ ШАХИ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ШАХОВИЙ АГЕНТ, НЕЙРОННА МЕРЕЖА, ШТУЧНИЙ ІНТЕЛЕКТ, САМОСТІЙНА ГРА, ШАХОВИЙ АЛГОРИТМ, ІГРОВЕ СЕРЕДОВИЩЕ, ПРОГРАМНЕ МОДЕЛЮВАННЯ.

Об'єктом дослідження є процес функціонування ігрового агента у середовищі некласичних шахів.

Предметом дослідження є методи та алгоритми навчання з підкріпленням для побудови ігрового агента в середовищі некласичних шахів.

Метою роботи є розроблення програмної системи для моделювання гри в некласичні шахи Chess960 та дослідження можливостей застосування методів навчання з підкріпленням для створення інтелектуального ігрового агента, який навчається на основі партій, зіграних у режимі самостійної гри.

У роботі оглянуто особливості некласичних шахів Chess960, проаналізовано основні підходи до створення інтелектуальних ігрових агентів та розглянуто можливості застосування навчання з підкріпленням у шахових середовищах. Розроблено програмну систему, яка забезпечує моделювання шахових партій, виконання ходів агентом, візуалізацію ігрового процесу, збереження результатів експериментів та проведення навчання на основі зіграних партій.

Реалізовано інтелектуального агента для гри в Chess960, який використовує модель оцінювання шахових позицій і навчається в режимі самостійної гри. У процесі навчання агент багаторазово взаємодіє з ігровим середовищем, формує досвід на основі зіграних партій, аналізує їх результати та поступово коригує власну поведінку. Для перевірки якості роботи агента проведено експериментальне оцінювання з використанням базових стратегій, зокрема випадкового та жадібного алгоритмів. Отримані результати дозволили визначити переваги й обмеження запропонованого підходу, а розроблена система може бути використана як основа для подальших досліджень у сфері інтелектуальних ігрових систем і навчання агентів у середовищах із варіативними початковими умовами.

ABSTRACT

Bachelor's thesis: 73 pages, 10 figures, 2 tables, 31 references, 1 appendix.

NON-CLASSICAL CHESS, REINFORCEMENT LEARNING, CHESS AGENT, NEURAL NETWORK, ARTIFICIAL INTELLIGENCE, SELF-PLAY, CHESS ALGORITHM, GAME ENVIRONMENT, SOFTWARE MODELING.

The object of the study is the process of functioning of a game-playing agent in a non-classical chess environment.

The subject of the study is reinforcement learning methods and algorithms for building a game-playing agent in a non-classical chess environment.

The aim of the work is to develop a software system for modeling the game of non-classical chess Chess960 and to study the possibilities of applying reinforcement learning methods to create an intelligent game-playing agent that learns based on games played in self-play mode.

The work reviews the features of non-classical chess Chess960, analyzes the main approaches to creating intelligent game-playing agents, and considers the possibilities of applying reinforcement learning in chess environments. A software system was developed that provides chess game simulation, execution of moves by the agent, visualization of the gameplay, saving of experimental results, and training based on played games.

An intelligent agent for playing Chess960 was implemented. The agent uses a chess position evaluation model and is trained in self-play mode. During training, the agent repeatedly interacts with the game environment, forms experience based on played games, analyzes their results, and gradually adjusts its own behavior. To assess the quality of the agent's performance, an experimental evaluation was conducted using baseline strategies, in particular random and greedy algorithms. The obtained results made it possible to determine the advantages and limitations of the proposed approach, while the developed system can be used as a basis for further research in the field of intelligent game systems and agent learning in environments with variable initial conditions.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 НЕКЛАСИЧНІ ШАХИ ТА ПІДХОДИ ДО ПОБУДОВИ ІГРОВИХ АГЕНТІВ.....	10
1.1 Некласичні шахи як середовище для дослідження ігрових агентів	10
1.2 Підходи до створення агентів у шахах	15
1.3 Постановка задачі.....	18
Висновки до розділу 1	21
РОЗДІЛ 2 НАВЧАННЯ ІГРОВОГО АГЕНТА	24
2.1 Представлення ігрового процесу як задачі навчання з підкріпленням....	24
2.2 Формування стану, дій та винагороди для агента.....	27
2.3 Навчання агента через самостійну гру	33
Висновки до розділу 2.....	38
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ІГРОВОГО АГЕНТА ДЛЯ CHESS960	41
3.1 Розробка програмного середовища Chess960	41
3.2 Реалізація агента та процедури навчання в режимі самостійної гри...	47
3.3. Оцінювання результатів навчання агента.....	57
Висновки до розділу 3.....	62
ВИСНОВКИ	65
ПЕРЕЛІК ПОСИЛАНЬ	69
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ.....	73

ВСТУП

Шахи є однією з класичних інтелектуальних ігор, яка протягом тривалого часу використовується як середовище для дослідження алгоритмів прийняття рішень, пошуку оптимальних стратегій та методів штучного інтелекту. Завдяки чітко визначеним правилам, дискретному простору станів і можливості формального опису ігрового процесу шахи стали важливою платформою для розвитку комп'ютерних ігрових агентів. Водночас класичні шахи вже мають значну кількість добре досліджених підходів, зокрема на основі мінімаксної логіки, евристичних оцінювальних функцій та глибокого пошуку в дереві можливих ходів.

Саме тому для дослідження адаптивності ігрових агентів доцільно розглядати не лише класичні шахи, а й їхні некласичні варіанти, у яких зменшується роль заздалегідь підготовлених дебютних схем. Особливий інтерес у цьому контексті становлять шахи Фішера, або Chess960. На відміну від класичних шахів, у Chess960 початкове розташування фігур на першій та восьмій горизонталях змінюється відповідно до спеціальних правил. Це суттєво зменшує роль заздалегідь підготовлених дебютних схем і підвищує значення здатності агента самостійно оцінювати позицію, адаптуватися до нових умов і формувати стратегію без опори на фіксовані шаблони гри. Саме тому Chess960 є доцільним середовищем для дослідження агентів, які навчаються на основі власного ігрового досвіду.

Актуальність теми зумовлена стрімким розвитком методів штучного інтелекту, зокрема навчання з підкріпленням, а також необхідністю створення агентів, здатних приймати рішення у складних дискретних середовищах із великою кількістю можливих станів. У задачах ігрового типу такий підхід є особливо важливим, оскільки агент може поступово накопичувати досвід, оцінювати наслідки власних дій і покращувати якість прийняття рішень. Для шахових середовищ важливим є також підхід самостійної гри, за якого агент

отримує навчальний досвід не з готового набору партій, а через багаторазове проведення ігор із самим собою. Застосування такого підходу до Chess960 дає змогу дослідити, наскільки ефективно агент може адаптуватися до змінних початкових умов і приймати рішення в середовищі з великою кількістю можливих станів.

Метою дипломної роботи є розроблення агента для гри в некласичні шахи Chess960 з використанням методів навчання з підкріпленням, організація його навчання в режимі самостійної гри та оцінювання ефективності отриманої моделі у порівнянні з базовими ігровими стратегіями.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати особливості некласичних шахів Chess960 як середовища для дослідження ігрових агентів;
- розглянути основні підходи до створення агентів для шахових ігор та визначити доцільність використання навчання з підкріпленням;
- формально описати ігровий процес як задачу навчання з підкріпленням;
- визначити способи представлення станів, дій і функції винагороди для ігрового агента;
- обґрунтувати використання режиму самостійної гри для формування навчального досвіду агента;
- розробити програмне середовище для моделювання гри в Chess960;
- реалізувати агента, здатного навчатися на основі багаторазової взаємодії з ігровим середовищем;
- провести навчання агента в режимі гри з самим собою;
- оцінити результати навченого агента у партіях проти базових стратегій;
- проаналізувати отримані результати та визначити можливі напрями подальшого вдосконалення розробленого підходу.

Об'єктом дослідження є процес функціонування ігрового агента у середовищі некласичних шахів.

Предметом дослідження є методи та алгоритми навчання з підкріпленням для побудови ігрового агента в середовищі некласичних шахів.

Методи дослідження включають аналіз наукових джерел, методи теорії ігор, методи штучного інтелекту, навчання з підкріпленням, самостійну гру, програмне моделювання, експериментальне тестування та порівняльний аналіз результатів роботи ігрового агента.

Практичне значення роботи полягає у створенні програмної реалізації агента для гри в Chess960, який навчається на основі партій, зіграних у режимі самостійної гри. Розроблена система може бути використана як основа для подальших досліджень у сфері навчання з підкріпленням, ігрового штучного інтелекту та моделювання поведінки агентів у складних дискретних середовищах із варіативними початковими умовами.

Робота складається зі вступу, трьох розділів, висновків, переліку використаних джерел та додатка. У першому розділі розглянуто предметну область неklasичних шахів та основні підходи до створення ігрових агентів. У другому розділі описано математичне подання ігрового процесу, навчання з підкріпленням та підхід самостійної гри. У третьому розділі наведено програмну реалізацію середовища Chess960 та ігрового агента, описано процедуру його навчання в режимі самостійної гри та аналіз результатів експериментального оцінювання ефективності розробленого підходу¹.

¹Тут і нижче використано такий інструмент штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського (протокол №11 Вченої ради КПІ ім. Ігоря Сікорського від 11 грудня 2023 р.)

РОЗДІЛ 1 НЕКЛАСИЧНІ ШАХИ ТА ПІДХОДИ ДО ПОБУДОВИ ІГРОВИХ АГЕНТІВ

1.1 Некласичні шахи як середовище для дослідження ігрових агентів

Шахи традиційно розглядаються як одна з базових предметних областей для дослідження методів штучного інтелекту, оскільки ця гра має формалізовані правила, скінченну множину допустимих дій у кожній позиції, чітко визначені умови завершення партії та можливість однозначного оцінювання результату. Завдяки цьому шахи можуть бути подані не лише як настільна гра, а і як формальне середовище прийняття рішень, у якому агент послідовно аналізує поточний стан, обирає дію та переходить до нового ігрового стану [21, 23].

У класичних шахах початкове розташування фігур завжди є однаковим. З одного боку, це забезпечує стабільність правил і зручність аналізу, однак з іншого – призводить до значної ролі дебютної теорії та попередньо підготовлених варіантів. Для людини це означає необхідність вивчення великої кількості дебютних схем, а для програмного агента – ризик надмірної орієнтації на типові початкові структури гри.

Приклад традиційного розташування фігур у класичних шахах наведено на рисунку 1.1.

У таких умовах дослідження здатності агента до узагальнення, адаптації та самостійного пошуку рішень може бути частково обмеженим.

З цієї причини окремий інтерес становлять неklasичні варіанти шахів, у яких змінюються окремі елементи стандартної гри: початкове розташування фігур, допустимі правила ходів, ціль гри або загальна структура ігрового процесу. Такі варіанти дозволяють зберегти основну логіку шахового мислення, але водночас створюють менш передбачуване середовище для агента. Для задач штучного інтелекту це є важливим, оскільки агент має не лише відтворювати заздалегідь відомі шаблони, а й будувати стратегію на основі поточної позиції.

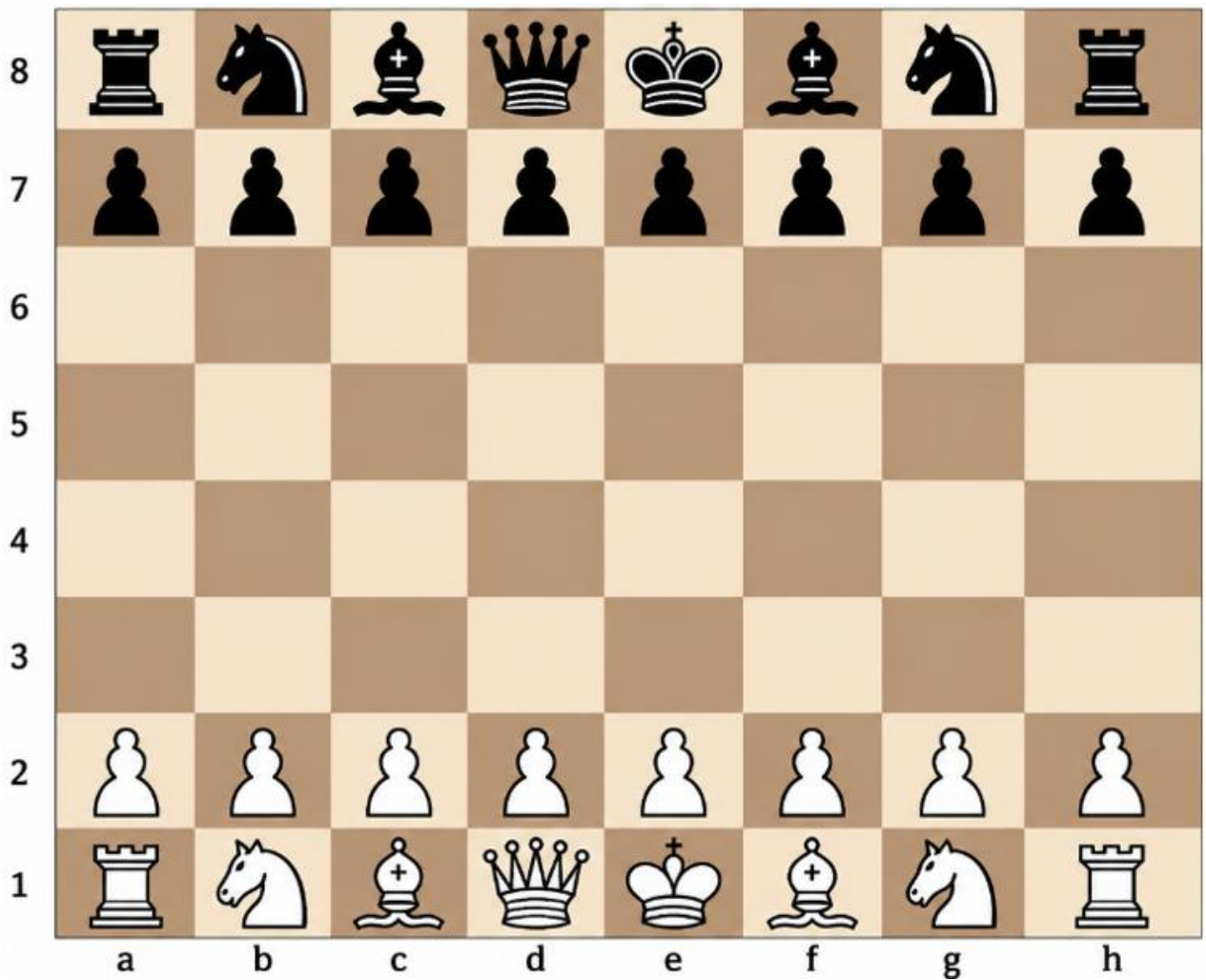


Рисунок 1.1 – Початкове розташування фігур у класичних шахах

Формально шахову гру можна подати як дискретну динамічну систему:

$$G = \langle S, A, T, s_0, Z, R \rangle,$$

де S – множина можливих станів дошки,

$A(s)$ – множина допустимих дій у стані s ,

$T(s,a)$ – функція переходу до нового стану після виконання дії a ,

s_0 – початковий стан гри,

Z – множина термінальних станів,

R – функція результату партії.

У межах цієї роботи основний інтерес становлять такі варіанти шахів, у яких зберігаються базові правила гри, набір фігур і мета партії, але змінюється початкова конфігурація фігур. Такий підхід дозволяє залишити шахове середовище формально строгим і водночас зменшити залежність агента від стандартних дебютних схем. Завдяки цьому агент має орієнтуватися не лише на наперед відомі послідовності ходів, а й на аналіз поточної позиції.

Одним із найбільш відомих і практично придатних варіантів некласичних шахів є шахи Фішера, або Chess960. У цьому варіанті зберігаються стандартна шахова дошка, набір фігур, основні правила ходів і ціль гри, але змінюється початкове розташування фігур на першій та восьмій горизонталях. Відповідно до правил Chess960, пішаки розташовуються так само, як у класичних шахах, слони мають стояти на полях різного кольору, король повинен бути розміщений між двома турами, а позиція чорних фігур дзеркально відповідає позиції білих фігур. Ці вимоги закріплені у правилах FIDE для Chess960 [5].

Приклад однієї з можливих стартових позицій у шахах Фішера (Chess960) наведено на рисунку 1.2.

Кількість можливих початкових позицій у шахах Фішера дорівнює 960. Це можна пояснити комбінаторно. Для першого слона існує 4 можливі поля одного кольору, для другого слона – 4 поля іншого кольору. Після цього для ферзя залишається 6 можливих полів. Далі два коні розміщуються на двох із п'яти вільних полів, що дає $\binom{5}{2} = 10$ варіантів. Після розміщення слонів, ферзя та коней три вільні поля займають король і дві тури, причому король обов'язково має бути між турами, тому для них залишається лише один допустимий порядок. Отже:

$$4 * 4 * 6 * \binom{5}{2} = 4 * 4 * 6 * 10 = 960.$$

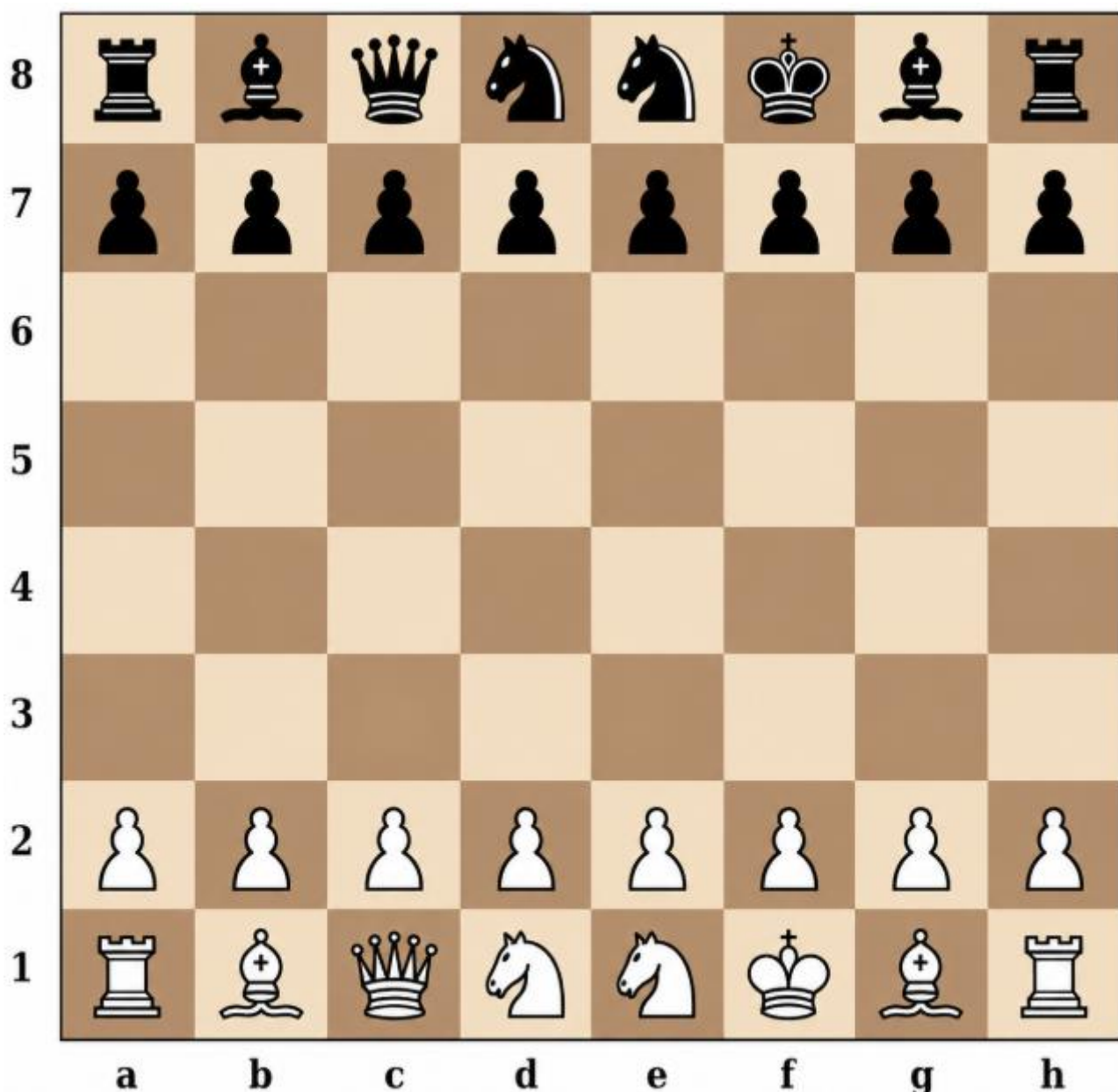


Рисунок 1.2 – Приклад початкового розташування фігур у Chess960

Таким чином, у Chess960 множина початкових станів має вигляд:

$$S_0^{(Chess960)} = \{s_0^{(1)}, s_0^{(2)}, \dots, s_0^{(960)}\}.$$

Це означає, що агент повинен бути здатним грати не з однієї сталої початкової позиції, а з множини різних початкових конфігурацій. Для дослідження ігрового агента така особливість є суттєвою, оскільки вона

підвищує вимоги до здатності моделі аналізувати позицію, оцінювати взаємодію фігур і приймати рішення без прямої залежності від стандартної дебютної бази.

Окрему увагу в Chess960 необхідно приділяти рокіровці. На відміну від класичних шахів, початкові позиції короля і тура можуть бути різними, тому сама дія рокіровки потребує спеціальної інтерпретації. Водночас кінцеві позиції після рокіровки залишаються аналогічними класичним шахам: після рокіровки в бік королівського флангу король опиняється на вертикалі g, а тура – на вертикалі f; після рокіровки в бік ферзевого флангу король опиняється на вертикалі c, а тура – на вертикалі d. FIDE окремо описує правила рокіровки для Chess960, оскільки стандартні правила передбачають фіксоване початкове розташування короля і тура, що не завжди виконується у шахах Фішера [5].

Для наочного пояснення цієї особливості на рисунку 1.3 подано схему рокіровки у шахах Фішера.

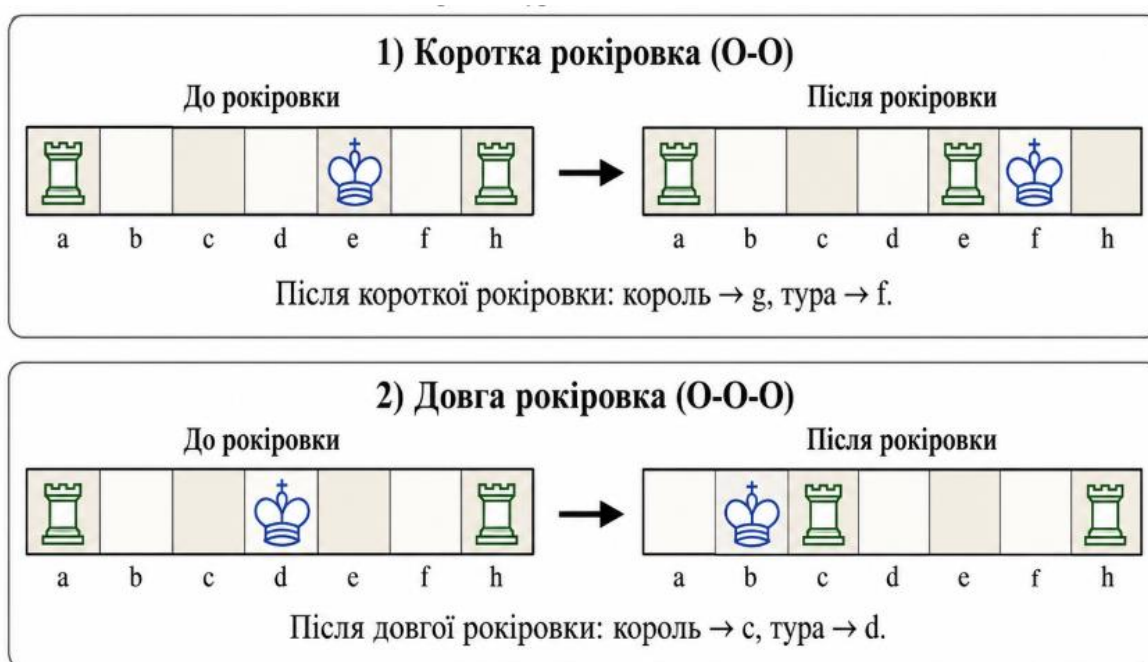


Рисунок 1.3 – Схема рокіровки у шахах Фішера

Для програмної реалізації це означає, що Chess960 є складнішим середовищем, ніж класичні шахи, але водночас не потребує повної зміни шахової

логіки. Основна відмінність полягає у генерації початкової позиції, коректній обробці рокіровки та здатності агента працювати з різними стартовими конфігураціями.

Саме тому шахи Фішера є доцільним вибором для дослідження ігрового агента: вони зберігають формальну строгість шахів, але водночас створюють більш варіативне середовище, у якому зменшується роль шаблонних дебютних структур і зростає значення аналізу поточної позиції.

1.2 Підходи до створення агентів у шахах

У шахових іграх агент виконує роль системи прийняття рішень, яка на кожному ході аналізує поточну позицію та обирає одну з допустимих дій. У загальному вигляді це можна подати як відображення:

$$\pi: S \rightarrow A,$$

де S – множина можливих станів шахової дошки,

A – множина допустимих ходів,

π – стратегія або політика агента, яка визначає вибір ходу в поточній позиції.

Класичним підходом до побудови шахових агентів є використання алгоритмів пошуку в дереві гри. У такому випадку кожна позиція розглядається як вузол дерева, а можливі ходи – як переходи до наступних вузлів. Агент аналізує варіанти на певну глибину та обирає хід, який приводить до найкращої оцінки позиції [3, 21, 23].

Основою такого підходу є мінімаксний алгоритм, який передбачає максимізацію оцінки позиції для одного гравця та її мінімізацію для суперника.

Для гри двох суперників оцінка ходу може бути подана у вигляді:

$$a^* = \arg \max_{a \in A(s)} V(T(s, a)),$$

де s – поточний стан дошки,

$A(s)$ – множина допустимих ходів у цьому стані,

$T(s, a)$ – новий стан після виконання ходу a ,

V – функція оцінювання позиції.

Перевагою такого підходу є його формальна зрозумілість і передбачуваність. Агент не потребує навчання у повному сенсі, оскільки його поведінка визначається правилами пошуку та функцією оцінювання. Однак ефективність такого агента значною мірою залежить від якості евристичної оцінки позиції, глибини пошуку та оптимізацій, які дозволяють скоротити кількість варіантів для аналізу.

У сучасних шахових рушіях пошуковий підхід часто поєднується зі складними функціями оцінювання. Наприклад, Stockfish використовує оцінювання позиції, результат якого застосовується в межах пошуку найкращого ходу; у сучасних версіях рушія важливу роль відіграє нейромережева оцінка NNUE, оптимізована для швидкого обчислення на процесорі. Такий підхід є дуже ефективним для практичної гри, однак він спирається на потужний пошук, велику кількість інженерних оптимізацій і багаторічне вдосконалення рушія [16].

Іншим напрямом є використання методів машинного навчання, коли агент не лише виконує пошук за наперед заданими правилами, а й поступово формує власну стратегію на основі досвіду. У цьому випадку поведінка агента визначається не тільки ручною функцією оцінювання, а й параметризованою моделлю:

$$\pi_{\theta}(a | s),$$

де θ – параметри моделі,

s – поточний стан,

$\pi_{\theta}(a|s)$ – ймовірність вибору ходу a у цьому стані.

Особливе місце серед таких підходів займає навчання з підкріпленням. У цьому випадку агент взаємодіє із середовищем, виконує ходи, отримує винагороду за результат гри або проміжні зміни позиції та поступово коригує свою поведінку. Такий підхід добре відповідає природі шахової гри, оскільки рішення агента мають довгострокові наслідки: окремий хід може не давати негайної переваги, але впливати на результат партії через багато наступних ходів.

Окремим важливим способом організації навчання з підкріпленням в ігрових задачах є самотійна гра. У такому режимі агент не навчається на заздалегідь підготовленому наборі партій і не залежить від постійного зовнішнього суперника, а формує навчальний досвід через багаторазове проведення партій із самим собою. Це дозволяє поступово накопичувати приклади ігрових ситуацій, оцінювати наслідки власних рішень і змінювати поведінку на основі результатів зіграних партій.

Показовим прикладом цього напряму є AlphaZero, який використовував загальний алгоритм навчання з підкріпленням у поєднанні з методом Монте-Карло для пошуку в дереві гри. На відміну від класичних шахових рушіїв, AlphaZero не спирався на традиційний альфа-бета пошук із великою кількістю спеціалізованих шахових евристик, а навчався через самотійну гру. Це демонструє, що для складних ігрових середовищ можливим є підхід, за якого агент формує ігрову поведінку на основі власного досвіду, отриманого в процесі багаторазової взаємодії з грою [24].

Для задачі Chess960 підхід на основі навчання з підкріпленням і самотійної гри має окрему перевагу. Оскільки початкова позиція не є сталою, агент повинен орієнтуватися не на фіксовані дебютні схеми, а на загальні принципи оцінювання позиції та вибору ходу. У режимі самотійної гри агент

може отримувати досвід у різних стартових конфігураціях Chess960, що є важливим для формування більш узагальненої поведінки.

У межах цієї роботи шаховий агент розглядається не як повноцінний конкурент сучасним шаховим рушіям, а як досліджувана модель, здатна взаємодіяти з варіативним середовищем Chess960. Тому головний акцент робиться не на досягненні максимальної ігрової сили, а на побудові працездатного агента, який отримує стан дошки, обирає допустимий хід і покращує свою поведінку на основі результатів партій, зіграних у режимі самотійної гри. Випадкові та евристичні стратегії в такій постановці доцільно розглядати не як основний спосіб навчання, а як базові підходи для подальшого порівняльного оцінювання результатів агента.

1.3 Постановка задачі

Після визначення Chess960 як предметного середовища та розгляду основних підходів до створення шахових агентів задача роботи може бути сформульована як розроблення програмного агента, здатного приймати рішення у варіативному шаховому середовищі та покращувати власну поведінку на основі результатів зіграних партій. На відміну від класичних шахів, де початкова позиція є сталою, у шахах Фішера агент повинен працювати з однією з 960 допустимих стартових конфігурацій. Це ускладнює задачу, оскільки якість гри не може базуватися лише на заздалегідь відомих дебютних схемах.

У межах цієї роботи шаховий агент розглядається як програмна система, яка отримує поточний стан шахової дошки, визначає множину допустимих ходів і обирає дію відповідно до сформованої стратегії. Основною метою є не створення рушія, що конкурує із сучасними шаховими програмами, а побудова працездатної моделі агента, здатної навчатися через багаторазову взаємодію з ігровим середовищем. При цьому основним способом формування навчального

досвіду є самостійна гра, у межах якої агент проводить партії із самим собою та використовує їх результати для подальшого оновлення моделі.

Формально процес гри можна подати як послідовність станів і дій:

$$s_0, a_0, s_1, a_1, \dots, s_t, a_t, s_{t+1},$$

де s_t – стан шахової дошки на кроці t ,

a_t – хід, обраний агентом у цьому стані,

s_{t+1} – новий стан після виконання ходу.

Для середовища Chess960 початковий стан належить множині допустимих стартових позицій:

$$s_0 \in S_0^{Chess960}, |S_0^{Chess960}| = 960.$$

На кожному кроці агент не може обирати довільну дію, оскільки в шахах множина можливих ходів обмежується правилами гри. Тому множина дій залежить від поточного стану:

$$a_t \in A(s_t),$$

де $A(s_t)$ – множина всіх легальних ходів у позиції s_t .

Після вибору ходу середовище переходить до нового стану:

$$s_{t+1} = T(s_t, a_t),$$

де T – функція переходу, що визначається правилами шахів Фішера. У цій функції мають бути враховані стандартні правила шахів, а також специфічні особливості Chess960, зокрема генерація початкової позиції та коректна обробка рокіровки.

Поведінка агента задається політикою:

$$\pi_{\theta}(a_t | s_t),$$

де θ – параметри моделі,

s – поточний стан,

$\pi_{\theta}(a_t | s_t)$ – правило вибору ходу агентом у заданій позиції.

У процесі навчання параметри моделі мають змінюватися так, щоб агент поступово покращував якість прийнятих рішень. Оскільки навчання відбувається в режимі самотійної гри, одна й та сама модель може використовуватися для прийняття рішень за обидві сторони. Це дозволяє агенту накопичувати досвід гри як за білих, так і за чорних, а також навчатися на позиціях, що виникають у різних стартових конфігураціях Chess960.

Для оцінювання результату гри використовується функція винагорода:

$$R = \begin{cases} 1, & \text{якщо результат партії є виграшним для агента,} \\ 0, & \text{якщо партія завершилась нічиєю,} \\ -1, & \text{якщо результат партії є програшним для агента.} \end{cases}$$

У найпростішому випадку винагорода пов'язується з кінцевим результатом партії: перемогою, поразкою або нічиєю. Такий підхід відповідає природі шахів, оскільки головною метою агента є не локальне покращення окремої позиції, а досягнення кращого підсумкового результату партії.

Загальна задача навчання агента полягає у пошуку такої політики, яка забезпечує найкращий очікуваний результат у партіях, що починаються з різних стартових позицій Chess960:

$$\pi^* = \arg \max_{\pi \in \Pi} G,$$

де G – підсумкова винагорода за партію,

π^* – оптимальна або наближено оптимальна політика агента.

З практичної точки зору задача розроблення агента включає кілька взаємопов'язаних складових. Необхідно реалізувати ігрове середовище Chess960, забезпечити генерацію допустимих початкових позицій, сформуванню подання стану дошки для моделі, визначити механізм вибору легального ходу, реалізувати процедуру навчання в режимі самостійної гри та провести оцінювання якості гри агента.

Окреме значення має порівняння результатів навченого агента з базовими стратегіями. У межах цієї роботи випадковий і жадібний агенти не розглядаються як основне джерело навчання, а використовуються для експериментальної перевірки якості отриманої моделі. Таке порівняння дозволяє визначити, чи демонструє агент кращу поведінку порівняно з найпростішими способами вибору ходу.

Якість роботи агента доцільно оцінювати за такими показниками: частка перемог, частка нічий і поразок, середня тривалість партії, здатність завершувати партії коректним результатом, а також стабільність гри проти різних типів базових стратегій. Для середовища Chess960 важливо також, щоб агент демонстрував працездатність не лише в одній фіксованій позиції, а в різних початкових конфігураціях.

Таким чином, поставлена задача поєднує дві взаємопов'язані складові: побудову шахового середовища з підтримкою правил Chess960 та реалізацію агента, здатного навчатися в цьому середовищі через самостійну гру. Це створює основу для подальшого розгляду теоретичних підходів до навчання ігрового агента та практичної реалізації відповідної програмної системи.

Висновки до розділу 1

У першому розділі дипломної роботи було розглянуто шахи Фішера, або Chess960, як предметне середовище для дослідження ігрового агента. Було

показано, що цей різновид шахів зберігає основні правила класичної гри, однак змінює початкове розташування фігур на першій та восьмій горизонталях. Завдяки цьому зменшується залежність гри від заздалегідь підготовлених дебютних схем і зростає значення аналізу поточної позиції.

Особливу увагу було приділено правилам формування початкових позицій у Chess960. У роботі зазначено, що пішаки залишаються на стандартних місцях, слони мають розташовуватися на полях різного кольору, король повинен стояти між двома турами, а позиція чорних фігур дзеркально відповідає позиції білих. Це дозволяє отримати 960 допустимих стартових конфігурацій, що робить середовище більш варіативним порівняно з класичними шахами.

Окремо було розглянуто специфіку рокіровки у шахах Фішера. На відміну від класичних шахів, початкові позиції короля і тур можуть відрізнятися, тому сама дія рокіровки потребує спеціальної інтерпретації. Водночас кінцеві позиції після рокіровки залишаються аналогічними класичним шахам, що дає змогу зберегти загальну шахову логіку, але потребує додаткової уваги під час програмної реалізації.

У межах розділу також було проаналізовано основні підходи до створення шахових агентів. Було розглянуто класичні пошукові алгоритми, евристичні підходи, методи машинного навчання та навчання з підкріпленням. Окремо було визначено роль самотійної гри як способу формування навчального досвіду агента без використання готового набору партій або постійного зовнішнього суперника.

Також було сформульовано постановку задачі роботи. Основна задача полягає в розробці програмного агента, здатного приймати рішення в середовищі Chess960, працювати з різними початковими конфігураціями, обирати легальні ходи та покращувати власну поведінку на основі партій, зіграних у режимі самотійної гри. Було визначено, що для цього необхідно реалізувати ігрове середовище, подання стану дошки, механізм вибору дії, процедуру навчання та оцінювання результатів роботи агента.

Таким чином, перший розділ сформував основу для подальшого дослідження: було обґрунтовано вибір Chess960 як більш варіативного шахового середовища, визначено його ключові особливості та сформульовано задачу розробки агента, здатного навчатися через самостійну гру. Випадкові та евристичні стратегії в межах роботи доцільно розглядати як базові підходи для подальшого порівняльного оцінювання якості отриманої моделі.

РОЗДІЛ 2 НАВЧАННЯ ІГРОВОГО АГЕНТА

2.1 Представлення ігрового процесу як задачі навчання з підкріпленням

Після постановки задачі розробки агента для гри в шахи Фішера необхідно визначити теоретичну модель, у межах якої буде описано процес прийняття рішень. Оскільки шахова партія складається з послідовності ходів, а кожне рішення впливає не лише на поточну позицію, а й на подальший перебіг гри, така задача природно узгоджується з підходом навчання з підкріпленням.

Навчання з підкріпленням розглядає агента як систему, що взаємодіє із середовищем у дискретні моменти часу. На кожному кроці агент отримує інформацію про поточний стан середовища, обирає дію, після чого середовище переходить у новий стан і повертає агенту певну винагороду. Метою агента є не просто вибір найкращої дії в окремий момент, а формування такої стратегії поведінки, яка забезпечує найкращий результат на всій послідовності взаємодії [13, 27].

У контексті Chess960 середовищем є шахова гра з правилами шахів Фішера, а агентом – програмна модель, що приймає рішення щодо вибору ходу. Поточний стан відповідає позиції на дошці, дія – одному з легальних ходів, перехід — зміні позиції після виконання ходу, а винагорода – числовій оцінці результату партії або проміжного стану гри.

Формально таку взаємодію можна подати як марковський процес прийняття рішень:

$$M = \langle S, A, P, R, \gamma \rangle,$$

де S – множина можливих станів середовища,

A – множина дій,

P – функція переходу між станами,

R – функція винагороди,

γ – коефіцієнт дисконтування майбутніх винагород.

Для шахової гри множина дій залежить від поточного стану, оскільки в кожній позиції доступні лише ті ходи, які не порушують правил гри:

$$a_t \in A(s_t).$$

Після виконання ходу середовище переходить у новий стан:

$$s_{t+1} = T(s_t, a_t).$$

У шахах цей перехід є детермінованим з погляду правил гри: якщо відомі поточна позиція та виконаний легальний хід, наступна позиція визначається однозначно. Водночас для агента повна траєкторія партії не є наперед відомою, оскільки вона залежить від подальших рішень обох сторін.

Послідовність взаємодії агента із середовищем можна подати у вигляді траєкторії:

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_t),$$

де s_0 – початковий стан,

a_t – дія агента на кроці t ,

r_{t+1} – винагорода після виконання дії,

s_T – кінцевий стан партії.

Для Chess960 важливо, що початковий стан не є фіксованим. Він обирається з множини допустимих стартових позицій:

$$s_0 \in S_0^{Chess960}, |S_0^{Chess960}| = 960.$$

Ця особливість впливає на навчання агента, оскільки модель має навчатися не для однієї стандартної початкової позиції, а для ширшої множини можливих початкових конфігурацій. Тому агент повинен орієнтуватися на структуру поточної позиції, а не на заздалегідь відомі дебютні послідовності.

У межах цієї роботи навчання агента організовується в режимі самостійної гри. Це означає, що агент отримує навчальний досвід не з готового набору партій і не через постійну гру проти зовнішнього суперника, а шляхом багаторазового проведення партій із самим собою. У такій постановці одна й та сама модель або її поточна версія може використовуватися для прийняття рішень за обидві сторони, а результат партії використовується для подальшого оновлення поведінки агента.

Метою агента в навчанні з підкріпленням є максимізація сумарної винагороди, отриманої протягом партії. Для моменту часу t така сумарна винагорода може бути записана як [1, 27, 31]:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1},$$

де G_t – накопичена винагорода від моменту t до завершення партії,

γ – коефіцієнт дисконтування,

r_{t+k+1} – винагорода, отримана на наступних кроках.

У шахах результат багатьох дій проявляється не відразу. Наприклад, хід може не давати миттєвої матеріальної переваги, але створювати позиційні передумови для виграшу в майбутньому. Саме тому навчання з підкріпленням є придатним для шахового середовища: воно дозволяє враховувати довгострокові наслідки послідовності рішень.

Політика агента визначає, яку дію він обирає у заданому стані:

$$\pi_{\theta}(a_t | s_t),$$

де θ – параметри моделі.

У процесі навчання ці параметри змінюються так, щоб підвищити очікувану сумарну винагороду за партію:

$$\pi^* = \arg \max_{\pi \in \Pi} E_{\pi}[G_0],$$

де Π – множина допустимих політик,

$E_{\pi}[G_0]$ – очікувана сумарна винагорода при використанні політики π

У режимі самотійної гри дії іншої сторони також формуються агентом або його поточною моделлю. З формального погляду це дозволяє розглядати партію як послідовну взаємодію агента із середовищем, у якому обидві сторони використовують спільний механізм прийняття рішень. Такий підхід дає змогу агенту накопичувати досвід гри в різних позиціях, отримувати результати завершених партій і поступово коригувати свою поведінку.

Представлення Chess960 як задачі навчання з підкріпленням дозволяє перейти від загального опису шахової гри до конкретних компонентів навчального процесу. Для подальшої реалізації необхідно окремо визначити спосіб подання стану дошки, множину дій, доступних агенту, принцип формування винагороди та механізм організації самотійної гри.

2.2 Формування стану, дій та винагороди для агента

Для того щоб шахову партію можна було використати в навчанні з підкріпленням, необхідно подати гру у вигляді набору формальних компонентів, з якими може працювати програмний агент. У загальному вигляді агент не сприймає шахову позицію так, як її бачить людина. Для моделі позиція має бути перетворена у структурований числовий опис, а можливі ходи – у множину

допустимих дій, серед яких агент здійснює вибір. Окремо необхідно визначити функцію винагороди, яка задає напрям навчання та дозволяє оцінювати наслідки прийнятих рішень.

Першим елементом є стан середовища. У шахах стан відповідає поточній позиції на дошці, але для коректного опису гри недостатньо враховувати лише розташування фігур. Необхідно також зберігати інформацію про сторону, яка має виконати хід, права на рокіровку, можливість взяття на проході та факт завершення партії. Для Chess960 особливе значення мають права на рокіровку, оскільки початкове розташування короля і тур може відрізнитися від класичного.

Формально стан на кроці t можна подати як:

$$s_t = B_t, c_t, q_t,$$

де B_t – розташування фігур на дошці в момент часу t ,

c_t – сторона, яка має виконати хід,

q_t – додаткова службова інформація про стан гри, зокрема права на рокіровку та можливість спеціальних ходів.

Для використання в моделі стан може бути перетворений у тензорне подання:

$$X_t \in \mathbb{R}^{8 \times 8 \times n},$$

де 8×8 відповідає розміру шахової дошки,

n – кількість інформаційних каналів, які описують фігури та додаткові ознаки позиції. Наприклад, окремі канали можуть відповідати різним типам фігур білих і чорних, стороні ходу, правам на рокіровку та іншим параметрам, необхідним для повного опису позиції.

Таке подання є зручним для навчання, оскільки воно зберігає просторову структуру дошки. Для шахового агента важливо не лише знати, які фігури

присутні на дошці, а й враховувати їхнє взаємне розташування, контроль полів і можливі напрямки руху. Саме тому подання стану у вигляді структури, пов'язаної з дошкою, є природним для задачі побудови агента.

Другим елементом є простір дій. У шахах дія агента відповідає ходу. Однак агент не може обирати будь-яке переміщення фігури, оскільки кожна позиція має власну множину легальних ходів. Тому множина дій залежить від поточного стану:

$$A_t = A(s_t),$$

де $A(s_t)$ – множина всіх ходів, допустимих правилами гри у стані s_t .

Вибір дії агентом можна подати як:

$$a_t = \pi_\theta(s_t),$$

або, якщо політика має ймовірнісний характер:

$$a_t \sim \pi_\theta(a|s_t),$$

де π_θ – політика агента з параметрами, θ – хід, обраний у поточному стані.

У практичній реалізації важливо, щоб агент обирав дію лише з множини легальних ходів. Це дозволяє уникнути ситуацій, коли модель пропонує хід, який не відповідає правилам шахів. Для Chess960 це особливо важливо через специфічну обробку початкового розташування фігур і рокіровки, оскільки деякі спеціальні ходи відрізняються від звичного вигляду в класичних шахах.

Після виконання дії середовище переходить до нового стану:

$$s_{t+1} = T(s_t, a_t),$$

де T – функція переходу, що визначається правилами шахів. Після виконання легального ходу положення фігур оновлюється, змінюється сторона ходу, а також можуть змінюватися додаткові параметри гри, наприклад права на рокіровку або можливість взяття на проході.

Третім елементом є функція винагороди. Саме вона визначає, які результати для агента є бажаними. У шахах природним джерелом винагороди є підсумковий результат партії. Наприклад, перемога може відповідати додатній винагороді, поразка - від'ємній, а нічия – нейтральному або проміжному значенню:

$$R = \begin{cases} 1, & \text{якщо агент виграв партію,} \\ 0, & \text{якщо партія завершилась нічиєю} \\ -1, & \text{якщо агент програв партію.} \end{cases}$$

Такий підхід є простим і добре відповідає кінцевій меті шахової гри. Водночас він створює певну складність для навчання, оскільки агент отримує основний сигнал лише після завершення партії. Через це окремі ходи в середині гри складно напряму пов'язати з кінцевим результатом. У шахах це особливо важливо, оскільки наслідки стратегічного рішення можуть проявитися лише через багато ходів.

У режимі самотійної гри ця особливість має додаткове значення. Оскільки агент бере участь у прийнятті рішень за обидві сторони, навчальні приклади можуть формуватися як для позицій білих, так і для позицій чорних. Після завершення партії результат використовується для оцінювання збережених позицій: позиції сторони, яка перемогла, мають отримувати позитивний навчальний сигнал, а позиції сторони, яка програла, - негативний. У разі нічиєї навчальний сигнал може бути нейтральним або близьким до нуля.

Для часткового вирішення проблеми відкладеної винагороди може використовуватися проміжна оцінка позиції, пов'язана зі зміною матеріального балансу, мобільності фігур, контролю центральних полів або іншими

характеристиками гри. У загальному вигляді винагороду можна подати як функцію переходу між станами:

$$r_{t+1} = R(s_t, a_t, s_{t+1}).$$

У межах цієї роботи доцільно розглядати винагороду як інструмент, що поєднує кінцевий результат партії з оцінкою корисності окремих переходів. Такий підхід дозволяє агенту отримувати навчальний сигнал не лише наприкінці гри, а й у процесі взаємодії із середовищем. При цьому підсумковий результат партії залишається головним критерієм оцінювання, оскільки саме він визначає, чи була послідовність рішень успішною.

Для Chess960 вибір стану, простору дій і функції винагороди має додаткове значення через варіативність початкових позицій. Агент повинен коректно працювати з різними стартовими конфігураціями, не порушувати правила рокіровки та приймати рішення на основі поточної структури дошки. Тому формування стану не може обмежуватися лише розташуванням фігур, а множина дій має формуватися відповідно до правил саме шахів Фішера.

Практично це означає, що на кожному кроці навчання виконується певна послідовність дій: поточна позиція перетворюється у числове подання, агент обирає один із легальних ходів, середовище оновлює стан гри, після чого зберігається інформація про виконаний перехід. У режимі самотійної гри цей цикл повторюється для обох сторін до завершення партії, після чого результат гри використовується для формування навчальних цільових значень.

Для наочного подання цієї послідовності на рисунку 2.1 наведено цикл взаємодії агента із середовищем у процесі навчання.



Рисунок 2.1 – Цикл взаємодії агента із середовищем у процесі навчання

Визначення стану, простору дій і функції винагороди формує основу навчального середовища агента. Після цього можна перейти до вибору моделі та алгоритму, які будуть використовуватися для наближеного оцінювання позицій і покращення поведінки агента в процесі самостійної гри.

2.3 Навчання агента через самостійну гру

Для подальшої реалізації агента необхідно обрати підхід, за допомогою якого він буде покращувати свою поведінку в процесі взаємодії з шаховим середовищем. Для задачі Chess960 важливо, щоб модель могла працювати з різними початковими позиціями, аналізувати структуру поточної дошки та оцінювати наслідки можливих ходів. Тому доцільним є використання підходу, у якому агент навчається оцінювати шахові позиції та на основі цієї оцінки обирати подальшу дію.

У межах навчання з підкріпленням одним із базових підходів є використання функції цінності стану. Вона показує, наскільки перспективним є певний стан для агента з погляду очікуваного результату партії. Формально функцію цінності можна подати так:

$$V^{\pi}(s) = E_{\pi}[G_t | s_t = s],$$

де $V^{\pi}(s)$ – очікувана цінність стану s за умови використання політики π ,

G_t – сумарна винагорода від моменту часу t до завершення партії.

Для шахової гри така інтерпретація є природною. Якщо стан має високу оцінку, це означає, що з нього агент має більші шанси отримати позитивний результат. Якщо оцінка низька, позиція є менш бажаною. Таким чином, замість прямого запам'ятовування окремих дебютних схем агент може навчатися оцінювати загальну якість позиції.

У практичній задачі точну функцію $V(s)$ неможливо задати вручну для всіх можливих шахових позицій. Тому її доцільно наближати за допомогою параметризованої моделі:

$$V(s) \approx V_{\theta}(s),$$

де $V_\theta(s)$ – оцінка позиції, яку повертає модель з параметрами θ . У ролі такої моделі може використовуватися нейронна мережа, яка отримує числове подання шахової дошки та повертає оцінку поточного стану [6].

Оскільки стан дошки подається у вигляді багатоканального тензора розмірності $18 \times 8 \times 8$, для оцінювання позицій доцільним є використання згорткової нейронної мережі. На відміну від повнозв'язної моделі, згорткова архітектура не зводить одразу шахову дошку до плоского вектора, а дозволяє зберегти її просторову структуру. Це важливо для шахового середовища, оскільки якість позиції залежить не лише від наявності певних фігур, а й від їхнього взаємного розташування, контролю полів, розміщення фігур відносно центра та безпеки короля.

Тоді вибір ходу можна подати через оцінювання станів, що виникають після виконання допустимих дій. Для кожного легального ходу $a \in A(s_t)$ середовище може бути умовно переведене в наступний стан $T(s_t, a)$, після чого модель оцінює його цінність. Агент обирає хід, який веде до найбільш перспективного стану:

$$a_t^* = \arg \max_{a \in A(s_t)} V_\theta(T(s_t, a)).$$

Для ходу чорних та сама модель може використовуватися з протилежною інтерпретацією: агент обирає хід який приводить до стану з найменшим значенням оцінки:

$$a_t^* = \arg \min_{a \in A(s_t)} V_\theta(T(s_t, a)).$$

Таким чином, одна й та сама модель оцінювання може використовуватися для прийняття рішень за обидві сторони. Різниця полягає лише в тому, що під час ходу білих агент максимізує оцінку позиції, а під час ходу чорних – мінімізує її.

Основною особливістю цієї роботи є організація навчання агента в режимі самотійної гри. Це означає, що агент не навчається на готовому наборі шахових партій і не використовує зовнішнього суперника як основне джерело навчального досвіду. Натомість він багаторазово проводить партії із самим собою, накопичує позиції, ходи та результати цих партій, після чого використовує отриману інформацію для оновлення параметрів моделі.

У режимі самотійної гри поточна версія агента приймає рішення за обидві сторони. На кожному ході формується поточний стан дошки, визначається множина легальних ходів, після чого агент обирає дію відповідно до поточної політики. Партія триває до досягнення одного з термінальних станів: перемоги однієї зі сторін, нічиєї або завершення за встановленим обмеженням кількості ходів. Після завершення партії її результат використовується для формування навчального сигналу.

Загальну ідею самотійної гри можна подати у вигляді послідовності:

$$\pi_{\theta}^{(k)} \rightarrow \tau^{(k)} \rightarrow D^{(k)} \rightarrow \pi_{\theta}^{(k+1)},$$

де $\pi_{\theta}^{(k)}$ – поточна політика агента на k – й ітерації навчання,

τ^k – партії, зіграні агентом із самим собою,

$D^{(k)}$ – набір навчальних прикладів, сформований на основі цих партій,

$\pi_{\theta}^{(k+1)}$ – оновлена політика після навчання моделі.

Навчальний приклад у такій постановці може містити стан дошки та цільове значення, сформоване на основі результату партії або оцінки позиції. Якщо партія завершується перемогою білих, позиції, вигідні для білих, мають отримувати позитивнішу оцінку, а позиції, вигідні для чорних, – нижчу. Якщо перемагають чорні, інтерпретація результату змінюється у протилежний бік. У разі нічиєї цільове значення може бути нейтральним або близьким до нуля.

Для підвищення стабільності навчання в роботі використовується буфер навчальних прикладів. У ньому зберігаються позиції, отримані під час

попередніх партій самостійної гри, разом із відповідними цільовими значеннями. Завдяки цьому оновлення параметрів моделі виконується не лише на основі останньої зіграної партії, а на вибірках із ширшої множини позицій. Такий підхід дозволяє зменшити залежність навчання від окремих епізодів, підвищити різноманітність навчальних даних і зробити процес оновлення моделі стабільнішим.

Навчання моделі полягає в поступовому коригуванні параметрів θ так, щоб оцінка позиції краще відповідала фактичному результату гри. У спрощеному вигляді ціль навчання можна подати як мінімізацію помилки між прогнозованою оцінкою позиції та цільовим значенням:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (V_{\theta}(s_i) - y_i)^2,$$

Де $L(\theta)$ – функція втрат,

N – кількість навчальних прикладів,

$V_{\theta}(s_i)$ – прогноз моделі для стану s_i ,

y_i – цільове значення, сформоване на основі результату партії або додаткової оцінки позиції.

У випадку навчання з підкріпленням цільове значення може враховувати не лише поточну винагороду, а й оцінку наступного стану [1, 27]:

$$y_t = r_{(t+1)} + \gamma V_{\theta}(s_{t+1}),$$

де r_{t+1} – винагорода після виконання ходу,

γ – коефіцієнт дисконтування майбутніх винагород,

$V_{\theta}(s_{t+1})$ – оцінка наступного стану.

Ця формула відображає ключову ідею навчання з підкріпленням: цінність поточного стану залежить не лише від негайної винагороди, а й від майбутніх можливостей, які відкриваються після виконання дії. Для шахів це має особливе

значення, оскільки більшість ходів не дають миттєвого результату, але можуть впливати на підсумок партії через багато наступних рішень.

Окремо необхідно врахувати баланс між використанням уже відомої стратегії та дослідженням нових ходів. Якщо агент завжди обирає лише той хід, який на поточний момент має найвищу оцінку, він може швидко зафіксуватися на обмеженій поведінці. Тому під час самостійної гри доцільно використовувати механізм дослідження, за якого агент іноді обирає випадковий легальний хід. Один із простих варіантів такого підходу – ϵ -жадібна стратегія:

$$a_t = \begin{cases} \text{випадковий хід з } A(s_t), \text{ з імовірністю } \epsilon, \\ \arg \max_{a \in A(s_t)} V_\theta(T(s_t, a)), \text{ з імовірністю } 1 - \epsilon. \end{cases}$$

де ϵ – параметр, що визначає частку випадкових дій. На початкових етапах навчання його значення може бути більшим, щоб агент активніше досліджував середовище. Надалі ϵ може поступово зменшуватися, що дозволяє агенту частіше використовувати вже сформовану оцінку позицій.

Для Chess960 самостійна гра має окрему перевагу, оскільки кожна нова партія може починатися з іншої стартової конфігурації. Це дозволяє агенту накопичувати досвід у різних початкових позиціях і зменшує ризик прив'язки до однієї стандартної структури дошки. У результаті модель має навчатися не конкретним дебютним послідовностям, а більш загальним закономірностям оцінювання позицій і вибору ходів.

Випадковий та евристичний агенти в межах цієї роботи не розглядаються як основний спосіб навчання моделі. Їх доцільно використовувати як базові стратегії для подальшого експериментального оцінювання. Найпростішим орієнтиром є випадковий агент, який обирає хід випадково з множини легальних ходів. Більш змістовним базовим суперником є евристичний або жадібний агент, який використовує прості правила оцінювання, наприклад матеріальну перевагу. Порівняння з такими суперниками дозволяє перевірити, чи здатна модель,

навчена через самотійну гру, демонструвати кращу якість гри порівняно з найпростішими стратегіями.

У межах цієї роботи доцільно використовувати підхід на основі функції цінності, за якого модель оцінює шахову позицію, а вибір ходу здійснюється через порівняння станів, що виникають після легальних дій. Навчальний досвід формується в режимі самотійної гри, де агент багаторазово взаємодіє з середовищем Chess960, граючи партії за обидві сторони. Такий підхід є достатньо зрозумілим для реалізації, напрямку пов'язаний із формалізацією стану, дії та винагороди, а також дозволяє дослідити навчання агента в умовах варіативних початкових позицій.

Сформована теоретична основа визначає подальшу логіку практичної частини роботи: необхідно реалізувати середовище Chess960, перетворення шахової позиції у числове подання, модель оцінювання стану, механізм вибору легального ходу та процедуру навчання агента в режимі самотійної гри.

Висновки до розділу 2

У другому розділі дипломної роботи було розглянуто представлення гри в Chess960 як задачі навчання з підкріпленням. Було показано, що шахова партія природно описується як послідовність станів, дій і винагород, де кожен хід впливає не лише на поточну позицію, а й на подальший перебіг гри. У такій постановці агент взаємодіє із середовищем, отримує стан дошки, обирає одну з допустимих дій, після чого середовище оновлює позицію та формує відповідний навчальний сигнал.

У межах розділу було визначено основні компоненти навчального процесу: стан середовища, простір допустимих дій, функцію винагороди, політику агента та сумарну винагороду. Особливу увагу було приділено тому, що в Chess960 початковий стан не є фіксованим, а належить множині

допустимих стартових конфігурацій. Це підвищує вимоги до агента, оскільки він має приймати рішення не на основі заздалегідь відомих дебютних структур, а на основі поточного розташування фігур і множини легальних ходів.

Також було розглянуто подання шахової позиції у вигляді числового стану, придатного для обробки моделлю. Для середовища Chess960 важливо враховувати не лише розташування фігур, а й додаткові ознаки, зокрема сторону ходу, права на рокіровку та можливість спеціальних ходів. Було показано, що вибір дії агентом має здійснюватися лише з множини легальних ходів, оскільки це забезпечує відповідність поведінки агента правилам шахів Фішера.

Окремо було обґрунтовано використання самостійної гри як способу формування навчального досвіду агента. У такому режимі агент не залежить від готового набору партій або постійного зовнішнього суперника, а накопичує досвід через багаторазове проведення партій із самим собою. Це дозволяє формувати навчальні приклади для різних стартових позицій Chess960 і поступово покращувати поведінку моделі на основі результатів зіграних партій.

У розділі також було обґрунтовано використання підходу на основі функції цінності. Такий підхід передбачає, що модель оцінює перспективність шахової позиції, а агент обирає хід через порівняння станів, які виникають після виконання можливих легальних дій. Для підтримки балансу між використанням уже сформованої оцінки та дослідженням нових варіантів гри було розглянуто ε-жадібну стратегію, яка дозволяє агенту іноді обирати випадковий легальний хід.

Таким чином, другий розділ сформував теоретичну основу для практичної реалізації агента. Було визначено, як шахи Фішера можуть бути представлені як середовище навчання з підкріпленням, які компоненти необхідні для побудови навчального процесу, чому підхід на основі функції цінності є доцільним для оцінювання позицій у Chess960 та яку роль відіграє самостійна гра у формуванні навчального досвіду агента. Окремо було обґрунтовано доцільність використання згорткової нейронної мережі для обробки багатоканального подання шахової дошки, а також використання буфера навчальних прикладів для

підвищення стабільності навчання. Це створило основу для подальшої програмної реалізації середовища, агента та процедури його навчання.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ІГРОВОГО АГЕНТА ДЛЯ CHESS960

3.1 Розробка програмного середовища Chess960

Практична частина роботи передбачає створення програмного середовища, у межах якого ігровий агент може взаємодіяти з шаховою партією у форматі Chess960. Основним завданням цього етапу є не лише реалізація шахової дошки та правил гри, а й підготовка середовища до подальшого використання в процесі навчання агента в режимі самостійної гри. Для цього необхідно забезпечити генерацію допустимих початкових позицій шахів Фішера, формування множини легальних ходів, оновлення стану після виконання дії, визначення результату партії та подання позиції у вигляді, придатному для обробки моделлю.

Розроблене середовище виконує роль проміжного шару між шаховою логікою та агентом. Воно відповідає за коректне функціонування гри, тоді як агент використовує його для отримання стану, вибору дії та аналізу результатів виконаних ходів. У режимі самостійної гри це середовище також забезпечує проведення повних партій, у яких рішення за обидві сторони можуть прийматися однією й тією самою моделлю або її поточною версією. Саме тому реалізація середовища є базовим етапом практичної частини роботи.

Для реалізації було використано мову програмування Python. Основна шахова логіка реалізована з використанням бібліотеки `python-chess`, яка забезпечує роботу з шаховою дошкою, генерацію легальних ходів, перевірку завершення партії, обробку спеціальних правил та підтримку режиму Chess960. Для візуального відображення дошки й ручної перевірки роботи середовища було використано бібліотеку `Pygame` [4, 18].

Програмну реалізацію можна умовно поділити на дві частини. Перша частина відповідає за навчальне середовище, у якому агент взаємодіє з шаховою партією під час навчання та оцінювання. Друга частина відповідає за

візуалізацію гри, тобто відображення дошки, фігур, легальних ходів, статусу партії та спеціальних дій, зокрема перетворення пішака і рокіровки. Такий поділ дозволяє окремо використовувати середовище для автоматичного навчання та для візуальної перевірки коректності реалізації.

Для наочного подання загальної структури реалізованого середовища доцільно використати схему (рис. 3.1), на якій буде показано зв'язок між основними програмними модулями.

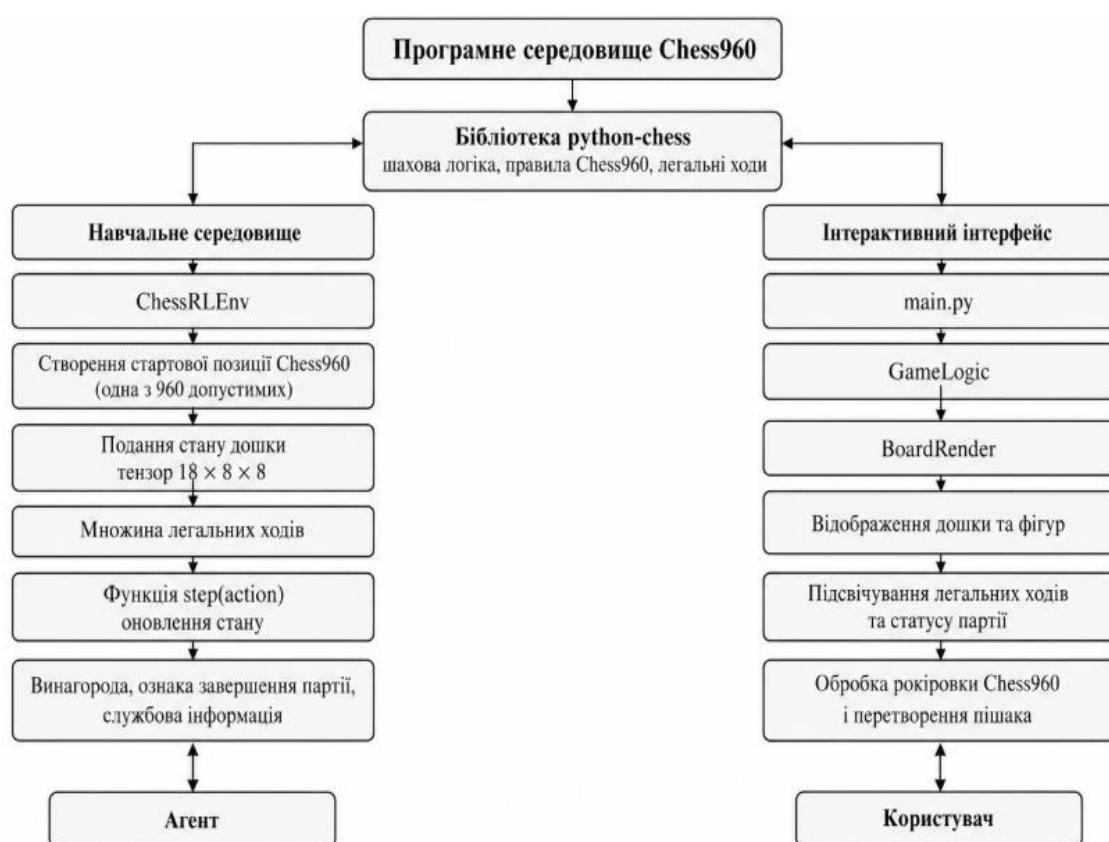


Рисунок 3.1 – Схема структури програмного середовища Chess960

Основним класом навчального середовища є `ChessREnv`. Він відповідає за створення шахової партії, збереження поточного стану дошки, виконання ходів агента та повернення даних, необхідних для навчання. Під час створення середовища можна вказати колір агента, режим використання Chess960 та

конкретний індекс початкової позиції. Якщо індекс не задається явно, він обирається випадково з діапазону від 0 до 959. Після цього створюється відповідна стартова позиція Chess960, а для дошки активується режим шахів Фішера.

Практично це означає, що кожен новий епізод навчання або тестування може починатися з іншого початкового розташування фігур. Такий підхід відповідає постановці задачі, оскільки агент не повинен орієнтуватися лише на одну стандартну стартову позицію. Навпаки, він має працювати з різними конфігураціями дошки та приймати рішення на основі поточного стану гри.

У середовищі передбачено метод скидання партії, який створює нову дошку та повертає початковий стан для агента. Це дозволяє використовувати середовище в циклі навчання, де кожна партія розглядається як окремий епізод. Для режиму самостійної гри це особливо важливо, оскільки після завершення однієї партії середовище має швидко створити нову партію з іншою або заданою стартовою конфігурацією Chess960.

Окремо реалізовано подання стану шахової дошки у числовому вигляді. Стан формується як тривимірний масив розмірності $18 \times 8 \times 8$. Перші 12 каналів використовуються для позначення фігур обох кольорів: окремо для білих і чорних пішаків, коней, слонів, тур, ферзів і королів. Ще один канал відображає сторону, яка має право ходу. Чотири канали використовуються для позначення прав на рокіровку для білих і чорних, а останній канал відповідає за можливість взяття на прохід. Завдяки такому поданню агент отримує не лише розташування фігур, а й важливі додаткові ознаки поточної позиції.

Цей спосіб подання стану є важливим саме для Chess960, оскільки права на рокіровку мають суттєве значення. У класичних шахах початкові позиції короля і тур є фіксованими, тоді як у шахах Фішера вони можуть відрізнитися. Тому для коректного аналізу позиції агенту необхідно знати не лише місце розташування фігур, а й те, чи зберігається можливість рокіровки для кожної зі сторін.

Множина доступних дій формується на основі легальних ходів поточної позиції. Середовище не дозволяє виконувати довільні або некоректні дії: кожен хід перевіряється на належність до множини легальних ходів. Якщо хід є допустимим, він виконується на дошці, після чого середовище повертає оновлений стан, винагороду, ознаку завершення партії та службову інформацію про поточний стан гри. Якщо ж хід є недопустимим, середовище може повернути штрафну винагороду та позначити такий хід як некоректний. У нормальному режимі роботи агент обирає дії лише з множини легальних ходів, тому такі ситуації мають допоміжний характер.

Загальний цикл взаємодії із середовищем у межах одного ходу реалізовано через послідовне виконання кількох дій. Спочатку формується поточний стан дошки у числовому вигляді. Після цього середовище визначає множину легальних ходів для поточної позиції. Далі агент або інший механізм вибору дії обирає один із допустимих ходів. Обраний хід передається до середовища, де перевіряється його коректність і виконується оновлення шахової дошки. Після виконання ходу середовище визначає, чи завершилася партія, формує винагороду та повертає оновлений стан разом зі службовою інформацією.

Якщо партія не завершена, середовище переходить до наступного кроку. У режимі самотійної гри цей цикл повторюється для обох сторін до завершення партії або досягнення встановленого обмеження на кількість півходів. Якщо партія завершена, середовище визначає результат гри: перемогу білих, перемогу чорних або нічию. Саме цей результат надалі використовується під час формування навчальних прикладів і цільових значень для моделі.

Функція винагороди реалізована у простому вигляді. Якщо партія ще не завершена, агент не отримує підсумкової винагороди за результат гри. Якщо партія завершується перемогою певної сторони, для цієї сторони формується позитивний результат, а для протилежної – негативний. Нічия оцінюється як нейтральний результат. Такий підхід відповідає загальній логіці шахової задачі, у якій головним критерієм є кінцевий результат партії.

Важливою особливістю реалізації є обробка завершення партії. Середовище враховує не лише мат, а й інші можливі завершення гри: пат, недостатність матеріалу, повторення позиції та правило ходів. Завдяки цьому партія може завершуватися не тільки перемогою однієї зі сторін, а й нічиєю. Це важливо для подальшого оцінювання агента, оскільки в шахах нічийний результат є нормальним можливим результатом гри.

Окрім навчального середовища, було реалізовано окрему логіку для ручної гри та візуальної перевірки роботи програми. За це відповідає клас GameLogic, який створює нову дошку Chess960, обробляє вибір клітинки користувачем, виконання ходів, перетворення пішака та визначення статусу партії. Візуальна частина реалізована в класі BoardRender, який відповідає за побудову шахової дошки, відображення фігур, координат, підсвічування вибраної клітинки, можливих ходів і статусної панелі.

Окрему увагу в реалізації було приділено рокіровці в Chess960. У шахах Фішера початкові позиції короля і тур можуть бути різними, тому візуальне відображення рокіровки потребує додаткової обробки. У програмі передбачено перетворення внутрішнього подання ходу рокіровки у зручне для користувача відображення. Якщо користувач вибирає короля, програма показує кінцеву клітинку рокіровки, а також дозволяє виконати рокіровку через вибір відповідної тури. Це робить ручну перевірку Chess960-партії зрозумілішою і дозволяє переконатися, що спеціальні правила цього різновиду шахів обробляються коректно.

Також у візуальному інтерфейсі передбачено відображення поточного ходу, статусу партії та можливість почати нову гру (рис. 3.2). Під час вибору фігури програма підсвічує доступні ходи: звичайні ходи позначаються окремим маркером, а ходи зі взяттям – іншим візуальним елементом. У разі досягнення пішаком останньої горизонталі відкривається меню вибору фігури для перетворення. Це дозволяє використовувати програму не лише для автоматичного навчання агента, а й для перевірки правильності реалізації шахової логіки в інтерактивному режимі.



Рисунок 3.2 – Приклад інтерфейсу програми з дошкою Chess960

У межах розробленого середовища також передбачено тестові сценарії. Зокрема, окремий тестовий файл дозволяє створити середовище Chess960, вивести індекс початкової позиції, перевірити форму стану, отримати список перших легальних ходів і виконати один тестовий хід. Це дає змогу швидко переконатися, що середовище створюється коректно, повертає стан потрібної розмірності та правильно оновлюється після виконання дії.

Розроблене програмне середовище забезпечує повний цикл роботи з партією Chess960: створення допустимої початкової позиції, формування множини легальних ходів, виконання дій, оновлення стану дошки, визначення результату партії та формування винагороди. Окремо реалізовано числове подання стану, яке містить інформацію про розташування фігур, сторону ходу, права на рокіровку та можливість взяття на проході. Наявність візуального інтерфейсу дає змогу перевіряти коректність роботи середовища в

інтерактивному режимі, зокрема для специфічних правил Chess960, пов'язаних із рокіровкою.

Таким чином, створене середовище є основою для подальшої реалізації агента та процедури його навчання. Воно дозволяє проводити повні партії Chess960, отримувати числові стани для моделі, формувати легальні дії та визначати результат гри. Це робить середовище придатним як для навчання агента в режимі самотійної гри, так і для подальшого експериментального оцінювання отриманої моделі.

3.2 Реалізація агента та процедури навчання в режимі самотійної гри

Після розробки програмного середовища Chess960 наступним етапом є реалізація агента, який взаємодіє з цим середовищем, отримує поточний стан шахової дошки, оцінює можливі продовження та обирає хід із множини легальних дій. У межах цієї роботи агент не намагається безпосередньо запам'ятовувати конкретні послідовності ходів або готові дебютні схеми. Натомість використано підхід, за якого модель навчається оцінювати якість шахової позиції, що виникає після виконання певного ходу. Такий підхід є доцільним для Chess960, оскільки початкове розташування фігур змінюється, а отже, агент має орієнтуватися не на заздалегідь підготовлену дебютну базу, а на структуру поточної дошки.

Архітектуру реалізованого агента доцільно розглядати не лише як нейронну мережу, а як повну систему прийняття рішень, що поєднує кілька взаємопов'язаних компонентів. У межах цієї системи середовище Chess960 відповідає за створення початкової позиції, збереження поточного стану дошки, генерацію множини легальних ходів, виконання обраної дії та визначення

результату партії. Агент, у свою чергу, використовує інформацію, отриману від середовища, для оцінювання можливих продовжень і вибору ходу.

Першим етапом роботи агента є отримання поточної шахової позиції. Оскільки нейронна модель не може безпосередньо працювати з шаховою дошкою у вигляді фігур і клітинок, позиція перетворюється у числове подання. У реалізованій системі для цього використовується тензор розмірності $18 \times 8 \times 8$. Перші дванадцять каналів відповідають розташуванню фігур обох сторін, окремий канал містить інформацію про сторону ходу, чотири канали відповідають правам на рокіровку для білих і чорних, а останній канал відображає можливість взяття на проході. Таке подання дозволяє зберегти просторову структуру шахової дошки та водночас передати службову інформацію, необхідну для коректної гри в Chess960.

Другим важливим компонентом є механізм роботи з легальними ходами. Агент не формує хід довільно і не передбачає його напряму як один із можливих класів. Замість цього він отримує від середовища множину всіх допустимих ходів у поточній позиції. Це гарантує, що вибір дії завжди відбувається в межах правил шахів Фішера, включаючи спеціальні випадки, пов'язані з рокіркою, взяттям на проході та перетворенням пішака.

Третім компонентом є моделювання можливих наступних станів. Для кожного легального ходу створюється копія поточної дошки, у якій виконується відповідний хід. Після цього нова позиція знову перетворюється у числове подання розмірності $18 \times 8 \times 8$ і передається до моделі оцінювання. Таким чином, агент оцінює не сам хід ізольовано, а позицію, яка виникає в результаті його виконання.

Центральним елементом архітектури агента є CNN-модель оцінювання шахових позицій. Вона отримує на вхід тензорне подання дошки та повертає одне числове значення в діапазоні від -1 до 1. Це значення інтерпретується як оцінка позиції з погляду білих фігур: додатні значення відповідають перевазі білих, від'ємні – перевазі чорних, а значення поблизу нуля – приблизно рівній позиції. Використання згорткової архітектури є доцільним, оскільки шахова

дошка має просторову структуру, а якість позиції залежить не лише від наявності фігур, а й від їхнього взаємного розташування, контролю полів і локальних тактичних взаємодій.

Після оцінювання всіх можливих наступних станів агент порівнює отримані значення. Якщо хід виконують білі, обирається продовження з найбільшою оцінкою, оскільки модель оцінює позицію з погляду білих. Якщо хід виконують чорні, обирається продовження з найменшою оцінкою, тобто позиція, яка є найменш вигідною для білих і, відповідно, більш вигідною для чорних. Завдяки цьому одна й та сама модель може використовуватися для прийняття рішень за обидві сторони, без створення окремої моделі для гри білими та чорними фігурами.

Узагальнену архітектуру реалізованого агента наведено на рисунку 3.3.

Окремо від загальної архітектури агента доцільно розглянути внутрішню структуру нейронної моделі оцінювання позиції. У реалізованій CNN-моделі спочатку використовується послідовність згорткових шарів. Перший згортковий шар приймає на вхід 18-канальне подання шахової дошки та формує 64 карти ознак. Далі використовуються ще два згорткові шари з 64 каналами та один згортковий шар, який зменшує кількість каналів до 32. Після кожного згорткового шару застосовується нормалізація проміжних ознак і функція активації ReLU. Це дозволяє моделі поступово формувати внутрішнє представлення позиції, зберігаючи просторові взаємозв'язки між клітинками дошки.

Після проходження згорткових шарів отримане представлення має розмірність $32 \times 8 \times 8$. Далі воно перетворюється у вектор розмірності 2048 і передається до повнозв'язного блоку оцінювання. У цьому блоці використовується повнозв'язний шар, який формує проміжне представлення розмірності 256, після чого застосовується функція активації ReLU. Завершальний вихідний шар формує одне числове значення, а функція Tanh обмежує його діапазоном від -1 до 1.

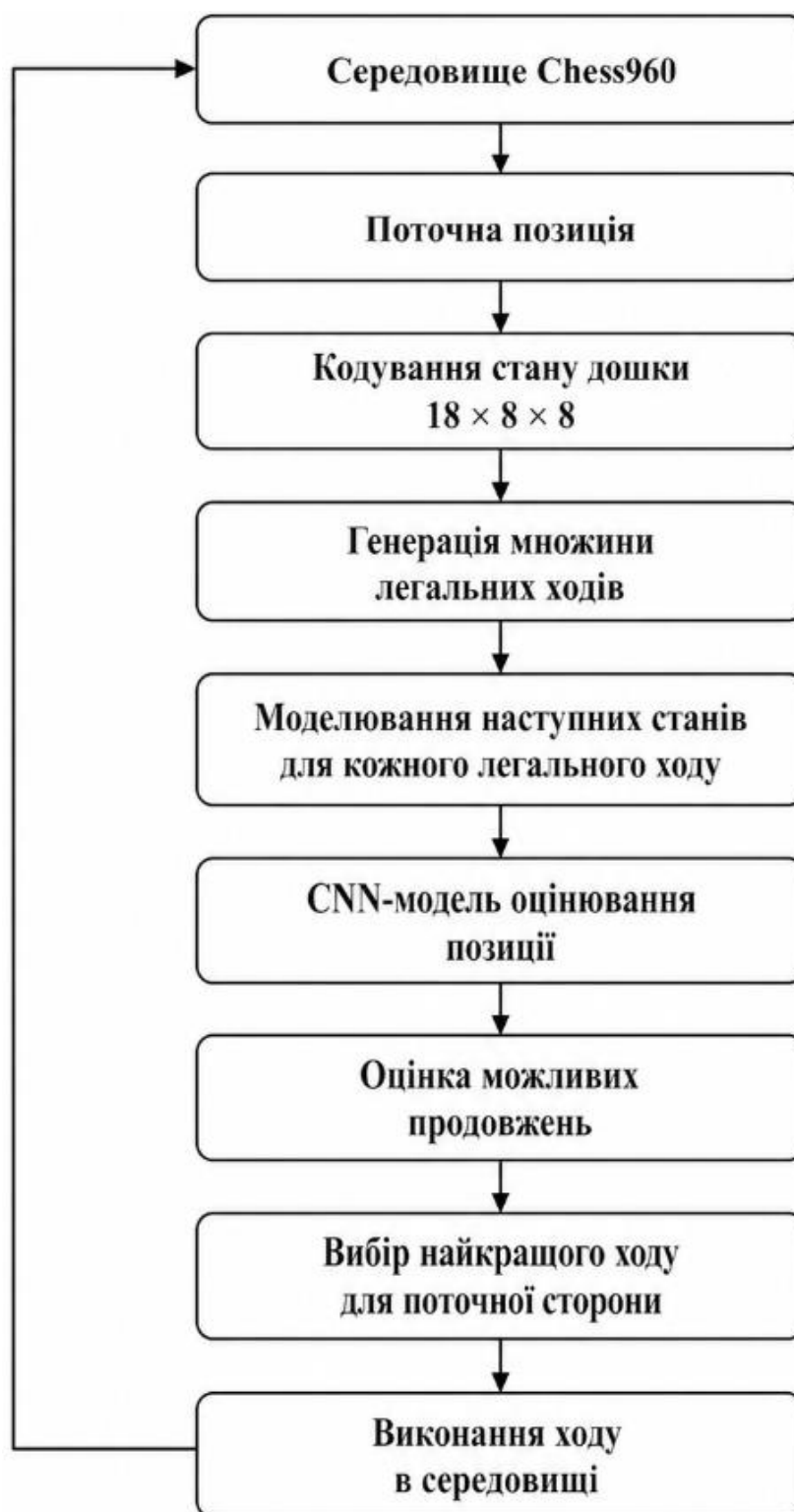


Рисунок 3.3 – Узагальнена архітектура агента для гри в Chess960

Узагальнену архітектуру CNN-моделі оцінювання позиції представлено на рисунку 3.4.

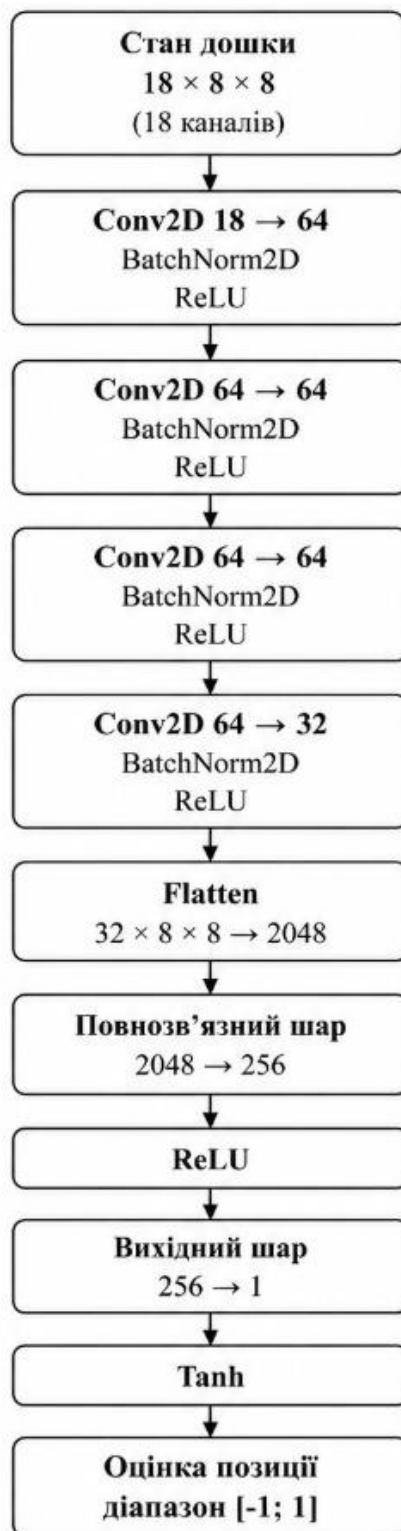


Рисунок 3.4 – Архітектура CNN-моделі оцінювання позиції

Таким чином, архітектура агента складається з двох рівнів. Перший рівень – це загальна логіка прийняття рішення, яка включає отримання стану від середовища, формування множини легальних ходів, моделювання наступних позицій, їх оцінювання та вибір найкращого ходу. Другий рівень – це внутрішня архітектура CNN-моделі, яка безпосередньо виконує оцінювання шахової позиції. Такий поділ дозволяє чітко відокремити логіку взаємодії агента із середовищем від структури нейронної мережі, що використовується для оцінювання позицій.

Під час навчання додатково використовується ϵ -жадібна стратегія. Її призначення полягає в тому, щоб забезпечити баланс між використанням уже сформованої оцінки моделі та дослідженням нових варіантів гри. З певною ймовірністю агент обирає випадковий легальний хід, що дозволяє отримувати різноманітніші позиції та уникати передчасного закріплення неефективної поведінки. В інших випадках агент обирає хід, який є найкращим відповідно до поточної оцінки моделі. На початкових етапах навчання значення параметра ϵ є більшим, оскільки агент має активніше досліджувати середовище. Надалі це значення поступово зменшується, що дозволяє агенту частіше використовувати вже сформовану оцінку позицій.

Основною особливістю реалізованої процедури навчання є використання режиму самотійної гри. Це означає, що агент не навчається на готовому наборі шахових партій і не використовує випадкового або евристичного суперника як основне джерело навчального досвіду. Натомість поточна версія агента проводить партії сама із собою, приймаючи рішення як за білі, так і за чорні фігури. Завдяки цьому навчальні приклади формуються безпосередньо в процесі взаємодії з середовищем Chess960.

Процедура навчання організована як послідовність епізодів, де один епізод відповідає одній зіграній партії. На початку кожного епізоду створюється нова стартова позиція Chess960. Якщо конкретний індекс стартової позиції не задається, він обирається випадково з множини допустимих конфігурацій. Це дає

змогу моделі накопичувати досвід у різних початкових позиціях і зменшує залежність агента від однієї фіксованої структури дошки.

Під час партії агент по черзі приймає рішення за обидві сторони. Для кожної позиції формується числове подання стану, визначається множина легальних ходів, після чого агент обирає дію відповідно до поточної моделі оцінювання та ϵ -жадібної стратегії. Після виконання ходу середовище оновлює стан дошки й перевіряє, чи завершилася партія. Цей процес повторюється до досягнення одного з термінальних станів, зокрема мату, нічиєї або встановленого обмеження на кількість півходів.

Під час кожної партії зберігаються навчальні приклади. Кожен приклад містить числове подання стану дошки, інформацію про сторону, яка виконувала хід, а також допоміжну оцінку позиції. Евристична оцінка використовується як додатковий орієнтир для навчання моделі. Вона враховує матеріальний баланс, мобільність сторін, можливість ходів у центральні поля та ситуації шаху. Після завершення партії визначається її результат: перемога білих, перемога чорних, нічия або незавершена партія після досягнення встановленого обмеження.

Для підвищення стабільності навчання було використано буфер навчальних прикладів. Після завершення кожної партії сформовані приклади додаються до буфера, у якому зберігаються позиції з багатьох попередніх епізодів самотійної гри. Під час оновлення параметрів моделі навчальна вибірка формується не лише з останньої партії, а випадковим чином із цього буфера. Це дозволяє моделі навчатися на більш різноманітному наборі позицій і зменшує ризик того, що оновлення параметрів буде надто сильно залежати від окремої партії.

Використання буфера є важливим саме для режиму самотійної гри, оскільки агент поступово змінює власну поведінку, а отже, послідовні партії можуть мати різний характер. Збереження позицій із попередніх епізодів дозволяє частково стабілізувати навчання та підтримувати різноманітність навчального досвіду.

Цільове значення для навчання формується на основі двох складових: підсумкового результату партії та евристичної оцінки конкретної позиції. Якщо партія завершується перемогою білих, позиції, сприятливі для білих, отримують додатний навчальний сигнал, а позиції, сприятливі для чорних, – від’ємний. Якщо перемагають чорні, інтерпретація змінюється у протилежний бік. У разі нічиєї цільове значення є нейтральним або близьким до нуля.

Для проміжних позицій навчальної партії цільові значення формуються з урахуванням як фінального результату, так і евристичної оцінки позиції. При цьому для позицій, ближчих до завершення партії, більшу вагу має підсумковий результат, а для ранніх позицій частково зберігається вплив евристичної оцінки. Такий підхід дозволяє моделі отримувати навчальний сигнал не лише після завершення гри, а й через оцінку якості проміжних позицій.

Навчання моделі виконується через мінімізацію помилки між прогнозованою оцінкою позиції та сформованим цільовим значенням. У реалізації використовується функція втрат Smooth L1 Loss, яка є стійкішою до окремих великих відхилень, ніж звичайна квадратична помилка. Для оновлення параметрів моделі використовується оптимізатор AdamW. Також застосовано обмеження норми градієнта, що зменшує ризик нестабільних оновлень параметрів під час навчання [19].

Основні параметри, використані під час навчання агента, наведено в таблиці 3.1.

Загальний алгоритм навчання агента складається з кількох послідовних етапів. Спочатку створюється середовище Chess960 з випадковою стартовою позицією. Далі формується початковий стан дошки, після чого агент починає партію в режимі самостійної гри. На кожному ході агент отримує числовий стан, визначає множину легальних ходів, оцінює можливі наступні позиції та обирає дію. Після завершення партії визначається фінальний результат, формуються цільові значення, обчислюється функція втрат і виконується оновлення параметрів моделі.

Таблиця 3.1 – Основні параметри процедури навчання агента

<i>Параметр</i>	<i>Значення</i>
Кількість епізодів навчання	8000
Максимальна кількість півходів у партії	350
Максимальна кількість навчальних прикладів з однієї партії	180
Розмір буфера навчальних прикладів	80000
Розмір навчальної вибірки	256
Швидкість навчання	0.0001
Оптимізатор	AdamW
Функція втрат	Smooth L1 Loss
Коефіцієнт регуляризації	0.00001
Обмеження норми градієнта	1
Періодичність проміжного оцінювання	Кожні 500 епізодів
Кількість каналів подання стану	18

Для наочного подання процедури навчання на рисунку 3.5 показано загальний алгоритм роботи навчального циклу.

Окремо передбачено періодичне збереження моделі. Після визначеної кількості епізодів формується checkpoint, який містить параметри моделі, стан оптимізатора, номер поточного епізоду, кількість каналів стану, режим навчання та інформацію про використання Chess960. Це дозволяє продовжувати навчання з попереднього стану, а також використовувати збережену модель для окремого оцінювання.

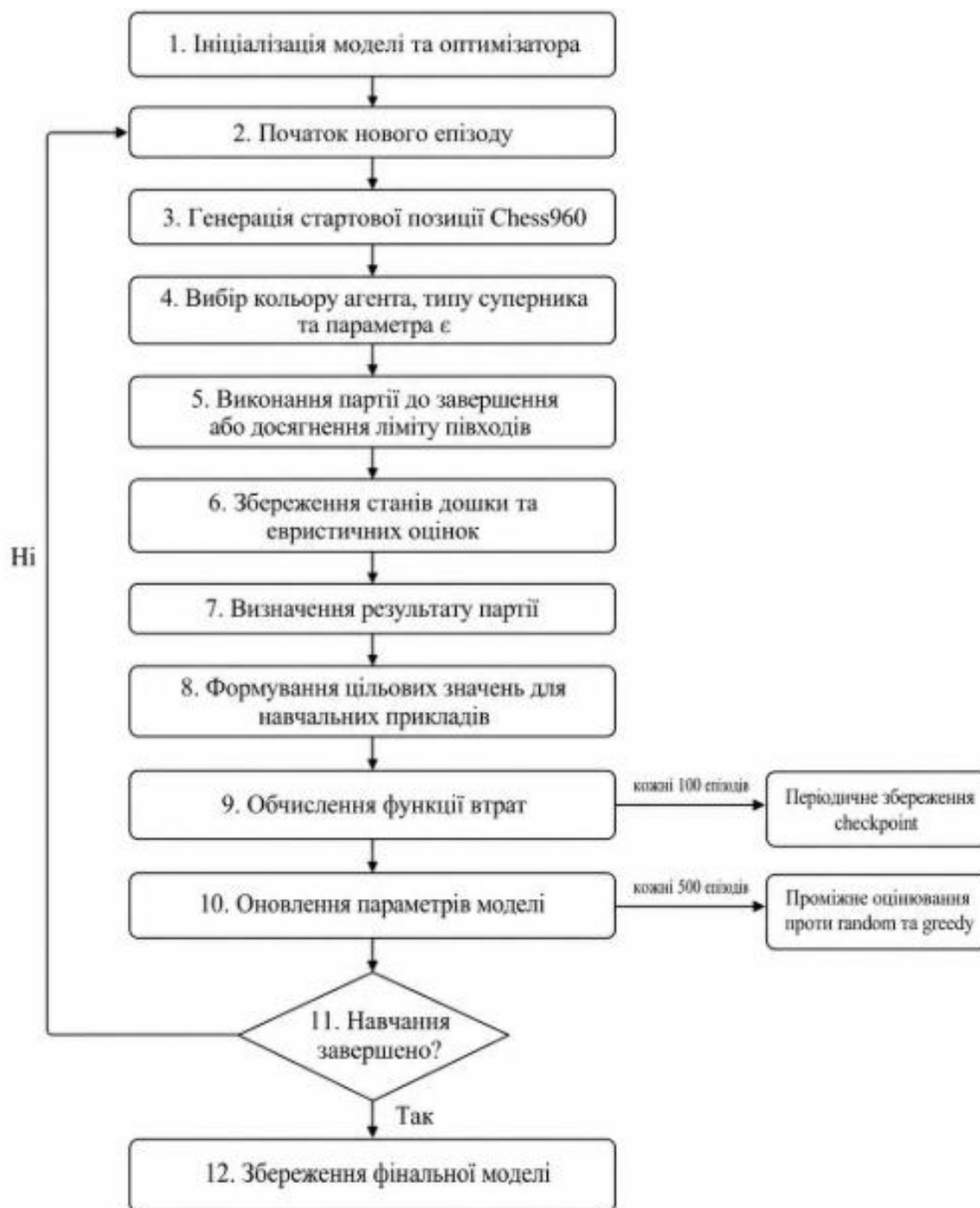


Рисунок 3.5 – Алгоритм навчання агента

Крім збереження моделі, у процесі навчання передбачено проміжну перевірку якості агента. Проте випадковий і жадібний агенти в цій роботі не розглядаються як основне джерело навчального досвіду. Вони використовуються як базові стратегії для порівняльного оцінювання вже

навченої моделі. Такий підхід дозволяє окремо розділити процес навчання агента через самостійну гру та процес перевірки його якості проти простіших способів вибору ходів.

3.3. Оцінювання результатів навчання агента

Після реалізації середовища Chess960, моделі оцінювання позицій та процедури навчання в режимі самостійної гри необхідно перевірити працездатність розробленого агента. Основною метою оцінювання є визначення того, чи здатний агент приймати осмислені рішення в умовах варіативних початкових позицій та чи демонструє він кращу якість гри порівняно з базовими стратегіями.

Оцінювання проводилося шляхом запуску серії партій у середовищі Chess960. Для перевірки використовувалися різні стартові позиції, що дозволяло оцінити здатність агента працювати не лише з однією фіксованою конфігурацією, а з різними допустимими початковими розташуваннями фігур. Такий підхід є важливим саме для шахів Фішера, оскільки якість гри агента не повинна залежати від запам'ятовування конкретної стартової позиції.

Для експериментального оцінювання було використано модель, отриману після 8000 епізодів навчання в режимі самостійної гри. Модель використовує 18-канальне подання стану шахової дошки та працює в режимі Chess960. Максимальна кількість півходів під час оцінювання становила 350. Якщо партія не завершувалася в межах цього обмеження, вона мала позначатися як незавершена, однак у проведених експериментах таких партій не було.

Для порівняння результатів було використано два типи базових суперників. Першим є випадковий агент, який на кожному ході обирає один із легальних ходів випадковим чином. Такий суперник задає найпростіший базовий рівень гри. Другим є жадібний агент, який використовує прості евристичні

правила вибору ходу, зокрема надає перевагу ходам із виграшем матеріалу або очевидним тактичним можливостям. Порівняння з таким суперником є змістовнішим, оскільки він уже використовує елементарну шахову логіку.

Якість роботи агента оцінювалася за такими показниками: кількість перемог, поразок і нічиїх, відсоток перемог, середня тривалість партії в півходах, а також кількість партій, які не були завершені в межах заданого обмеження. Окремо агент перевірявся під час гри білими та чорними фігурами, оскільки в реалізованій моделі оцінка позиції подається з погляду білих, а вибір ходу для чорних виконується через мінімізацію цієї оцінки.

Результати оцінювання наведено в таблиці 3.2.

За результатами оцінювання видно, що агент, навчений у режимі самотійної гри, демонструє високі результати проти випадкового суперника. У партіях за білі фігури агент здобув 49 перемог із 50, що відповідає частці перемог 98,00%. Одна партія завершилася нічиєю, а поразок у цьому режимі не було. Під час гри чорними фігурами агент здобув 47 перемог із 50, 2 партії завершилися нічиєю, а 1 партія була програна. Отже, загалом проти випадкового агента було зіграно 100 партій, з яких 96 завершилися перемогою навченої моделі, 3 – нічиєю і 1 – поразкою.

Такі результати свідчать про те, що агент не виконує ходи випадково, а здатний використовувати певні закономірності шахової позиції. Висока частка перемог проти випадкового суперника показує, що модель після навчання в режимі самотійної гри навчилася обирати ходи, які в середньому приводять до значно кращого результату, ніж випадковий вибір із множини легальних ходів. Особливо важливо, що агент демонструє стабільну перевагу як під час гри білими, так і під час гри чорними фігурами. Наявність лише однієї поразки у 100 партіях проти випадкового суперника не змінює загального висновку про те, що модель упевнено перевищує випадковий рівень гри.

Таблиця 3.2 – Результати оцінювання агента після навчання в режимі самотійної гри

<i>Показник</i>	<i>Значення</i>
Режим оцінювання	Агент білими проти випадкового
Кількість партій	50
Перемоги агента	49
Нічий	1
Поразки агента	0
Незавершені партії	0
Частка перемог	98,00%
Частка нічий	2,00%
Частка поразок	0,00%
Середня тривалість, півходів	93,86
Режим оцінювання	Агент чорними проти випадкового
Кількість партій	50
Перемоги агента	47
Нічий	2
Поразки агента	1
Незавершені партії	0
Частка перемог	94,00%
Частка нічий	4,00%
Частка поразок	2,00%
Середня тривалість, півходів	90,62
Режим оцінювання	Агент білими проти жадібного
Кількість партій	50
Перемоги агента	1
Нічий	10
Поразки агента	39
Незавершені партії	0
Частка перемог	2,00%
Частка нічий	20,00%
Частка поразок	78,00%
Середня тривалість, півходів	83,28
Режим оцінювання	Агент чорним проти жадібного
Кількість партій	50
Перемоги агента	1
Нічий	22
Поразки агента	27
Незавершені партії	0
Частка перемог	2,00%
Частка нічий	44,00%
Частка поразок	54,00%
Середня тривалість, півходів	88,82

Складнішим етапом оцінювання була гра проти жадібного агента, який використовує прості евристичні правила вибору ходу. У цьому випадку результати виявилися значно слабшими. За білих фігур навчений агент здобув 1 перемогу, 10 партій завершилися нічиєю, а 39 партій були програні. За чорних фігур агент також здобув 1 перемогу, 22 партії завершилися нічиєю, а 27 партій програв. Загалом проти жадібного агента було зіграно 100 партій, із яких агент виграв 2, зіграв унічию 32 і програв 66.

Для наочного порівняння результатів агента проти базових стратегій було побудовано діаграму, наведену на рисунку 3.6.

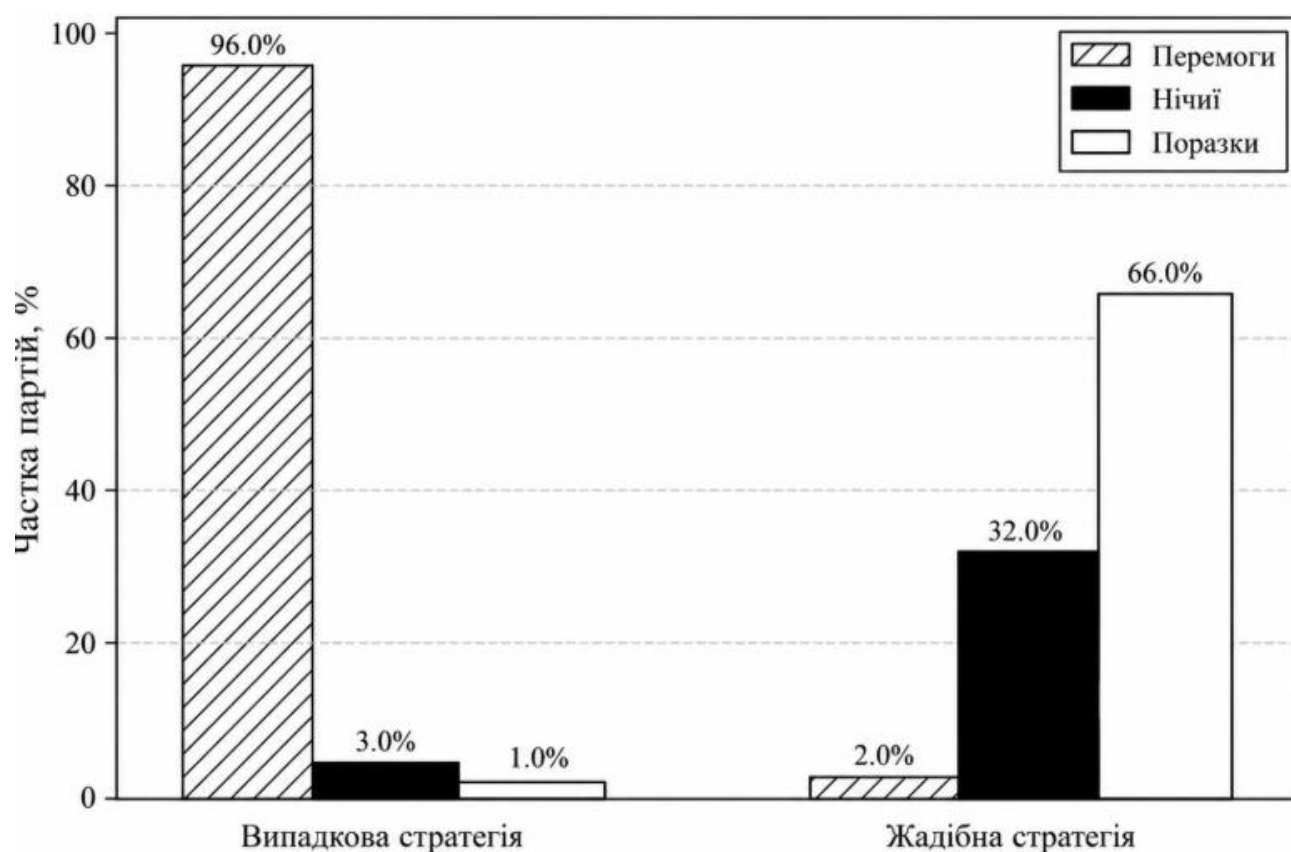


Рисунок 3.6 – Порівняння результатів агента проти базових стратегій

Окремо слід зазначити, що в усіх режимах оцінювання не було незавершених партій. Це є позитивним результатом, оскільки агент здатний доводити партії до коректного завершення в межах встановленого обмеження на

кількість півходів. Середня тривалість партій проти випадкового агента становила 93,86 півходів під час гри білими та 90,62 півходів під час гри чорними. Проти жадібного агента середня тривалість партій становила 83,28 півходів за білих і 88,82 півходів за чорних. Це свідчить про те, що жадібний агент не лише створює складніші позиції для навченої моделі, а й часто швидше реалізує перевагу, що призводить до більшої кількості поразок агента.

Отримані результати також показують різницю між грою агента за білі та чорні фігури. Проти випадкового суперника агент демонструє високу ефективність за обидва кольори, хоча результат за білих є дещо кращим: 98,00% перемог проти 94,00% під час гри чорними. Проти жадібного агента частка перемог є однаковою для обох кольорів і становить 2,00%, однак за чорних фігур агент частіше досягає нічиєї: 44,00% проти 20,00% за білих. Водночас частка поразок проти жадібного агента залишається високою в обох режимах, що вказує на обмежену здатність моделі протидіяти навіть простій евристичній стратегії.

За підсумками експериментів можна зробити висновок, що реалізований агент здатний працювати в середовищі Chess960, обробляти різні початкові конфігурації та приймати рішення на основі оцінки поточної позиції. Навчання в режимі самотійної гри дозволило моделі сформувати базову ігрову поведінку без використання готового набору партій. Це підтверджується високими результатами проти випадкового агента, де модель здобула 96 перемог у 100 партіях.

Водночас результати проти жадібного агента показують, що поточна модель має обмежену стратегічну та тактичну глибину. Значна кількість поразок свідчить про те, що модель часто не здатна ефективно протидіяти супернику, який використовує просту матеріальну або позиційну евристику. Водночас наявність 32 нічиїх у 100 партіях проти жадібного агента показує, що агент у частині випадків здатний уникати програшу, однак цього недостатньо для стабільного перегравання евристичної стратегії.

Висновки до розділу 3

У третьому розділі дипломної роботи було розглянуто практичну реалізацію програмного середовища Chess960, ігрового агента та процедури його навчання в режимі самотійної гри. Було створено середовище, яке забезпечує генерацію допустимих початкових позицій шахів Фішера, формування множини легальних ходів, виконання дій, оновлення стану дошки, визначення результату партії та підготовку даних для навчання агента.

У межах програмної реалізації було використано мову програмування Python та бібліотеки, необхідні для роботи із шаховою логікою, візуалізацією гри й побудовою нейронної моделі. Реалізоване середовище підтримує режим Chess960, враховує особливості формування початкової позиції, специфіку рокіровки та дозволяє подавати стан шахової дошки у вигляді числового тензора розмірності $18 \times 8 \times 8$. Таке подання містить інформацію про розташування фігур, сторону ходу, права на рокіровку та можливість взяття на проході, що є необхідним для коректної роботи агента в середовищі шахів Фішера.

Окремо було реалізовано агента, який використовує згорткову нейронну мережу для оцінювання шахових позицій. Модель отримує на вхід числове подання стану дошки та повертає одне значення, яке інтерпретується як оцінка позиції з погляду білих фігур. Вибір ходу здійснюється шляхом аналізу всіх легальних продовжень: для кожного допустимого ходу формується наступний стан, після чого він оцінюється моделлю. Під час гри білими агент обирає продовження з найбільшою оцінкою, а під час гри чорними – з найменшою.

Процедура навчання була організована як послідовність партій у режимі самотійної гри. У такому підході агент не використовує готовий набір шахових партій і не навчається через гру проти постійного зовнішнього суперника. Натомість поточна модель проводить партії сама із собою, приймаючи рішення як за білі, так і за чорні фігури. Під час навчання зберігаються позиції з партій, формується цільове значення на основі результату гри та допоміжної

евристичної оцінки, після чого параметри CNN-моделі оновлюються за допомогою функції втрат Smooth L1 Loss та оптимізатора AdamW.

Для підвищення стабільності навчання було використано буфер навчальних прикладів. Завдяки цьому модель оновлювалася не лише на основі останньої зіграної партії, а на вибірках із ширшої множини позицій, отриманих у попередніх епізодах самотійної гри. Такий підхід дозволив зменшити залежність навчання від окремих партій і забезпечити більшу різноманітність навчальних прикладів.

Після завершення навчання було проведено експериментальне оцінювання агента. Для перевірки використовувалася модель, отримана після 8000 епізодів самотійної гри. Оцінювання проводилося проти двох базових стратегій: випадкового агента та жадібного агента. Для кожного типу суперника було проведено по 50 партій за білі фігури та по 50 партій за чорні фігури, тобто загалом 100 партій проти випадкового агента і 100 партій проти жадібного агента.

Результати оцінювання показали, що агент упевнено перевищує випадковий рівень гри. Проти випадкового агента було зіграно 100 партій, із яких 96 завершилися перемогою навченої моделі, 3 – нічиєю і лише 1 – поразкою. Загальна частка перемог становила 96,00%, частка нічиїх – 3,00%, а частка поразок – 1,00%. Це свідчить про те, що модель після навчання в режимі самотійної гри здатна використовувати певні закономірності шахової позиції та обирати ходи значно ефективніше, ніж випадкова стратегія.

Водночас результати проти жадібного агента виявилися значно слабшими. Із 100 партій проти жадібної стратегії агент здобув 2 перемоги, 32 партії завершив унічию та 66 програв. Загальна частка перемог становила 2,00%, частка нічиїх – 32,00%, а частка поразок 66,00%. Це показує, що поточна модель має обмежену здатність протидіяти супернику, який використовує навіть просту евристичну оцінку позицій.

Окремо слід зазначити, що в усіх режимах оцінювання не було незавершених партій. Це підтверджує коректність роботи програмного

середовища та здатність агента доводити партії до завершення в межах установленого обмеження на кількість півходів. Також результати показали, що агент демонструє стабільну перевагу проти випадкової стратегії як за білих, так і за чорних фігур, однак проти жадібного агента часто не може ефективно реалізувати позиційні можливості або уникнути матеріальних втрат.

Таким чином, у третьому розділі було реалізовано практичну частину роботи: створено середовище Chess960, побудовано CNN-модель оцінювання позицій, реалізовано механізм вибору легальних ходів, організовано навчання в режимі самостійної гри та проведено експериментальне оцінювання отриманої моделі. Отримані результати підтверджують працездатність розробленої системи та демонструють можливість застосування навчання з підкріпленням для побудови агента в середовищі Chess960.

Разом із тим результати проти жадібного агента вказують на наявність обмежень реалізованого підходу. Основними напрямками подальшого вдосконалення можуть бути збільшення кількості навчальних партій, покращення функції винагороди, розширення евристичної складової навчального сигналу, ускладнення архітектури моделі або поєднання оцінювання позицій із глибшим аналізом можливих ходів. Це дозволило б підвищити не лише базову ігрову стабільність агента, а й його здатність протидіяти сильнішим евристичним суперникам.

ВИСНОВКИ

У дипломній роботі було досліджено задачу розробки ігрового агента для шахів Фішера, або Chess960, з використанням методів навчання з підкріпленням. Обрана предметна область є доцільною для дослідження, оскільки Chess960 зберігає формальну строгість класичних шахів, але водночас усуває залежність від однієї сталої початкової позиції та зменшує роль заздалегідь підготовлених дебютних схем. Це створює умови, у яких агент має орієнтуватися не на фіксовані шаблони гри, а на аналіз поточного стану дошки.

У процесі аналізу було визначено основні особливості шахів Фішера, які впливають на побудову ігрового агента. До них належать наявність 960 допустимих стартових конфігурацій, спеціальні правила формування початкового розташування фігур, а також особлива інтерпретація рокіровки. Саме ці властивості роблять середовище Chess960 більш варіативним порівняно з класичними шахами та вимагають від агента здатності адаптуватися до різних початкових умов.

Було сформульовано задачу розробки агента, здатного працювати в середовищі Chess960, обирати легальні ходи, враховувати результат партії та покращувати власну поведінку на основі ігрового досвіду. Для цього було розглянуто представлення шахової партії як задачі навчання з підкріпленням, у якій агент взаємодіє із середовищем через послідовність станів, дій і винагород. Така постановка дозволяє описати шахову гру як процес прийняття рішень, де кожен хід може впливати не лише на поточну позицію, а й на подальший перебіг партії.

У роботі було визначено ключові компоненти навчального процесу: стан дошки, множину доступних дій, функцію винагороди, політику агента та підхід до оцінювання позицій. Було обґрунтовано використання підходу на основі функції цінності, за якого модель навчається оцінювати перспективність шахової позиції, а вибір ходу здійснюється через порівняння станів, що виникають після

виконання легальних дій. Такий підхід добре узгоджується з особливостями Chess960, оскільки агент має орієнтуватися на поточну структуру дошки, а не на фіксовані дебютні варіанти.

Окрему увагу було приділено самотійній грі як способу формування навчального досвіду агента. У такому режимі агент не навчається на готовому наборі шахових партій і не залежить від постійного зовнішнього суперника, а багаторазово проводить партії із самим собою. Це дозволяє формувати навчальні приклади безпосередньо в середовищі Chess960, накопичувати досвід гри за обидві сторони та поступово коригувати поведінку моделі на основі результатів зіграних партій.

Практична частина роботи була присвячена створенню програмного середовища Chess960. Реалізоване середовище забезпечує генерацію допустимих початкових позицій, формування множини легальних ходів, виконання дій, оновлення стану дошки, визначення результату партії та формування даних, необхідних для навчання агента. Окремо було реалізовано числове подання шахової позиції розмірності $18 \times 8 \times 8$, яке містить інформацію про розташування фігур, сторону ходу, права на рокіровку та можливість взяття на проході.

Для перевірки роботи середовища було створено візуальний інтерфейс, який дозволяє відображати шахову дошку, фігури, доступні ходи, статус партії та спеціальні дії. Особливу увагу було приділено коректній обробці рокіровки у Chess960, оскільки в цьому різновиді шахів початкові позиції короля і тур можуть відрізнятися від класичного розташування. Наявність інтерактивної перевірки дала змогу контролювати правильність реалізації шахової логіки та спеціальних правил гри.

Також було реалізовано агента, який використовує згорткову нейронну мережу для оцінювання шахових позицій. Модель отримує на вхід числове подання стану дошки розмірності $18 \times 8 \times 8$ та повертає оцінку позиції в діапазоні від -1 до 1 . На основі цієї оцінки агент порівнює можливі легальні ходи та обирає той, що веде до найбільш перспективного стану для поточної сторони. Під час

гри білими агент максимізує оцінку позиції, а під час гри чорними — мінімізує її, оскільки модель інтерпретує значення з погляду білих фігур.

Процедура навчання була побудована як послідовність партій у режимі самотійної гри. У процесі навчання агент приймав рішення за обидві сторони, зберігав навчальні приклади, формував цільові значення на основі результату партії та допоміжної евристичної оцінки позиції, після чого оновлював параметри CNN-моделі. Для підвищення стабільності навчання було використано буфер навчальних прикладів, функцію втрат Smooth L1 Loss, оптимізатор AdamW та обмеження норми градієнта. Навчання проводилося протягом 8000 епізодів.

Експериментальне оцінювання було проведено для моделі, отриманої після 8000 епізодів навчання в режимі самотійної гри. Для перевірки якості роботи агента використовувалися дві базові стратегії: випадковий агент і жадібний агент. Випадковий агент обирає хід випадково з множини легальних ходів, тоді як жадібний агент використовує прості евристичні правила, зокрема орієнтацію на матеріальну вигоду або очевидні тактичні можливості.

Результати оцінювання показали, що агент упевнено перевищує випадковий рівень гри. Проти випадкового агента було зіграно 100 партій, із яких 96 завершилися перемогою навченої моделі, 3 – нічиєю і лише 1 – поразкою. Загальна частка перемог проти випадкового суперника становила 96,00%, частка нічий – 3,00%, а частка поразок – 1,00%. Це свідчить про те, що агент після навчання в режимі самотійної гри здатний використовувати певні закономірності шахової позиції та обирати ходи значно ефективніше, ніж випадкова стратегія.

Водночас результати проти жадібного агента виявилися значно слабшими. Із 100 партій проти жадібної стратегії агент здобув 2 перемоги, 32 партії завершив унічию та 66 програв. Загальна частка перемог становила 2,00%, частка нічий – 32,00%, а частка поразок – 66,00%. Це показує, що модель здатна в окремих випадках уникати поразки, однак має обмежену здатність стабільно протидіяти супернику, який використовує навіть просту шахову евристику.

У результаті роботи було створено програмну основу для навчання і тестування ігрового агента в середовищі Chess960. Реалізована система поєднує шахове середовище, числове подання стану, CNN-модель оцінювання позицій, механізм вибору легальних ходів і процедуру навчання в режимі самостійної гри. Отримані результати підтверджують можливість застосування навчання з підкріпленням для побудови агента, здатного працювати не з однією фіксованою позицією, а з множиною допустимих початкових конфігурацій шахів Фішера.

Разом із тим результати експериментів показали наявність певних обмежень реалізованого підходу. Зокрема, модель демонструє високу ефективність проти випадкового суперника, але має труднощі під час гри проти жадібного агента, який використовує просту евристичну оцінку позицій. Це може бути пов'язано з обмеженою глибиною аналізу можливих продовжень, відсутністю пошуку на кілька ходів уперед, особливостями формування навчального сигналу та недостатньою кількістю навчальних епізодів для стабільного опанування складніших позиційних закономірностей.

Подальше вдосконалення системи може передбачати збільшення кількості навчальних партій, покращення функції винагороди, розширення евристичної складової навчального сигналу, використання складнішої нейромережевої архітектури або поєднання моделі оцінювання позицій із глибшим аналізом можливих ходів. Це дозволило б підвищити не лише базову ігрову стабільність агента, а й його здатність протидіяти сильнішим евристичним або пошуковим суперникам.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bellman, Richard. "TERMINAL CONTROL, TIME LAGS, AND DYNAMIC PROGRAMMING." *Proceedings of the National Academy of Sciences*, vol. 43, no. 10, Oct. 1957, pp. 927-930. <https://doi.org/10.1073/pnas.43.10.927>.
2. Bhalekar, Aishwarya. "Design and Development of AI Powered Chess Engine." *International Journal for Research in Applied Science and Engineering Technology*, 2025, <https://doi.org/10.22214/ijraset.2025.73063>.
3. Campbell, Murray, et al. "Deep Blue." *Artificial Intelligence*, vol. 134, no. 1-2, Jan. 2002, pp. 57-83. [https://doi.org/10.1016/S0004-3702\(01\)00129-1](https://doi.org/10.1016/S0004-3702(01)00129-1).
4. Fiekas, Niklas. "Python-chess: A Chess Library for Python." 2012. <https://python-chess.readthedocs.io/en/stable/core.html> (дата звернення: 15.05.2026).
5. FIDE Rules Commission. "Guidelines II. Chess960 Rules." <https://rcc.fide.com/guidelinesii> (дата звернення: 11.05.2026).
6. Goodfellow, Ian, et al. *Deep Learning*. MIT Press, 2016. <https://www.deeplearningbook.org/> (дата звернення: 14.05.2026).
7. Hafez, A. H. Abdul, et al. "A Comprehensive Review of Reinforcement Learning Algorithms for Control Systems: Applications, Challenges, and Future Directions." *IEEE Access*, vol. 14, 2026, pp. 42809-42837. <https://doi.org/10.1109/ACCESS.2026.3674539>.
8. Hastuti, Dwi, and W. Wijaya. "Development of a Machine Learning Based Chess Game in Python." *BEST Journal of Applied Electrical Science & Technology*, vol. 7, no. 1, Mar. 2025, pp. 21-28. <https://doi.org/10.36456/best.vol7.no1.10343>.
9. Holly, Michael, et al. "An Interactive Chess-Puzzle-Simulation for Computer Science Education." 2022 8th International Conference of the Immersive Learning Research Network (iLRN), May 2022, pp. 1-5. <https://doi.org/10.23919/ILRN55037.2022.9815923>.

10. Liu, Zhihao, et al. "Task-Level Decision-Making for Dynamic and Stochastic Human-Robot Collaboration Based on Dual Agents Deep Reinforcement Learning." *The International Journal of Advanced Manufacturing Technology*, vol. 115, nos. 11-12, June 2021, pp. 3533-3552. <https://doi.org/10.1007/s00170-021-07265-2>.
11. Madake, Jyoti, et al. CHESS AI: Machine Learning and Minimax Based Chess Engine. Jan. 2023, pp. 1-6. <https://doi.org/10.1109/ICONAT57137.2023.10080746>.
12. Memarian, Bahar, and Tenzin Doleck. "A Scoping Review of Reinforcement Learning in Education." *Computers and Education Open*, vol. 6, Mar. 2024, p. 100175. <https://doi.org/10.1016/j.caeo.2024.100175>.
13. Mnih, Volodymyr, et al. "Human-Level Control Through Deep Reinforcement Learning." *Nature*, vol. 518, no. 7540, Feb. 2015, pp. 529-533. <https://doi.org/10.1038/nature14236>.
14. Morozov-Leonov, Oleksandr Sviatoslavovych. Sposib Hry v Ho z Vykorystanniam Shtuchnoho Intelektu. 2023. <https://ela.kpi.ua/items/33a06572-bef4-4675-a11b-93924aa810ec> (дата звернення: 12.05.2026).
15. Nayeve, Serhiy, et al. "Scenario Modelling in the Context of Foresight Studies." *Studies in Computational Intelligence*, Jan. 2023, pp. 397-418. https://doi.org/10.1007/978-3-031-37450-0_23.
16. NNUE. Stockfish Docs, <https://official-stockfish.github.io/docs/nnue-pytorch-wiki/docs/nnue.html> (дата звернення: 14.05.2026).
17. Panchal, Hitanshu, et al. Chess Moves Prediction Using Deep Learning Neural Networks. Oct. 2021, pp. 1-6. <https://doi.org/10.1109/ICACC-202152719.2021.9708405>.
18. Pygame Community. Pygame v2.6.0 Documentation. <https://www.pygame.org/docs/> (дата звернення: 16.05.2026).
19. PyTorch Foundation. PyTorch Documentation. <https://docs.pytorch.org/docs/stable/index.html> (дата звернення: 15.05.2026).

20. Riady, Muhammad Alvin, et al. Agent Performance Comparison of the Q-Learning Algorithm and SARSA Algorithm in Javanese Chess Game. Oct. 2024, pp. 1-6. <https://doi.org/10.1109/ICIC64337.2024.10957223>.
21. Russell, Stuart, and Peter Norvig. Intelligence Artificielle: Une approche moderne. Pearson, 2021. <https://hal.science/hal-04245057/>.
22. Rybalko, Anastasiia Anatoliivna. Hlyboke Navchannia z Pidkriplenniam dlia Keruvannia Dronom v Zadachi Peresliduvannia. 2024. <https://ela.kpi.ua/handle/123456789/64987> (дата звернення: 12.05.2026).
23. Shannon, Claude E. "Programming a Computer for Playing Chess." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 41, no. 314, 1950, pp. 256-275. <https://doi.org/10.1080/14786445008521796>.
24. Silver, David, et al. "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go Through Self-Play." Science, vol. 362, no. 6419, Dec. 2018, pp. 1140-1144. <https://doi.org/10.1126/science.aar6404>.
25. Souchleris, Konstantinos, et al. "Reinforcement Learning in Game Industry – Review, Prospects and Challenges." Applied Sciences, vol. 13, no. 4, 2023, p. 2443. <https://doi.org/10.3390/app13042443>.
26. Srivastava, Kushagra, et al. Advance Chess Engine: An Use of ML Approach. Mar. 2024, pp. 1-6. <https://doi.org/10.1109/INOCN60754.2024.10511742>.
27. Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. 2nd ed., MIT Press, 2018. <https://incompleteideas.net/book/the-book-2nd.html> (дата звернення: 11.05.2026).
28. Upasani, Nilam, et al. Dev-Zero: A Chess Engine. June 2021, pp. 1-6. <https://doi.org/10.1109/ICCICT50803.2021.9510148>.
29. Vikulov, Vladyslav, and Ivan Pyshnograiev. "Automated Detection of SQL Injection Vulnerabilities in Chatbots Using Reinforcement Learning." Cybersecurity: Education, Science, Technique, Sept. 2025, p. 223. <https://doi.org/10.28925/2663-4023.2025.29.873>.

30. Vo, Nghia. Implementation of a Robotic Arm with Machine Vision in Chess: Integrating Computer Vision and Robotics for Intelligent Gameplay. 2025. <https://www.theseus.fi/handle/10024/908109> (дата звернення: 17.05.2026).
31. Watkins, Christopher J. C. H., and Peter Dayan. “Q-Learning.” Machine Learning, vol. 8, nos. 3-4, May 1992, pp. 279-292. <https://doi.org/10.1007/BF00992698>.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Лістинг програми розміщено за посиланням

URL: <https://github.com/SolekLgabi/chess960-rl-agent>