

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПРИЙНЯТТЯ РІШЕНЬ НА ОСНОВІ СТРАТЕГІЙ ТЕОРІЇ ІГОР
RESEARCH AND SOFTWARE IMPLEMENTATION OF DECISION-MAKING
METHODS BASED ON GAME THEORY STRATEGIES

Виконав: студент 2 курсу, групи КНТ-110м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Брагін Я.А.

(прізвище та ініціали)

Керівник Коцур М.І.

(прізвище та ініціали)

Рецензент Гофман Є.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Брагіна Ярослава Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів прийняття рішень на основі стратегій теорії ігор. Research and Software Implementation of Decision-Making Methods Based on Game Theory Strategies

керівник проєкту (роботи) Коцур Михайло Ігорович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “05” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) 9 грудня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблемної області та постановка завдань дослідження. 2. Матеріали та методи. 3. Розробка архітектури програми. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Коцур М.І., доцент		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “ 4 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-13 тиждень	Додатки
8	Нормоконтроль та рецензування.	14–15 тижні	
9	Захист роботи.	16 тиждень	

Студент(ка)

(підпис)

Брагін Я.І.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Коцур М.І.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
92 с., 28 рис., 7 табл., 4 дод., 19 джерел.

ПРИЙНЯТТЯ РІШЕНЬ, ТЕОРІЯ ІГОР, МЕТОД БРАУНА-РОБІНСОНА, ПРОЄКТУВАННЯ, РЕАЛІЗАЦІЯ, РОЗРОБКА, ТЕСТУВАННЯ.

Об'єкт дослідження – процес прийняття рішень на основі стратегій теорії ігор.

Предмет дослідження – методи та стратегії теорії ігор для прийняття рішень.

Мета роботи – дослідити, проаналізувати та реалізувати методи прийняття рішень на основі стратегій теорії ігор.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, теорія прийняття рішень, теорія ігор, мова програмування Java, бібліотеки Gametheory та Algorithms, середовище розробки NetBeans, персональний комп'ютер під управлінням операційної системи Microsoft Windows 11.

Результати. Створено програмний застосунок, що реалізує метод Брауна-Робінсона та запропоновані модифікації до нього.

Висновки. Розроблено програмний застосунок прийняття рішень на основі теорії ігор, а саме за допомогою метода Брауна-Робінсона та запропонованих модифікацій до нього.

Галузь використання – автоматизовані системи контролю та управління.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 92 pages, 28 figures, 7 tables, 4 appendixes, 19 sources.

DECISION MAKING, GAME THEORY, BROWN-ROBINSON METHOD, DESIGN, IMPLEMENTATION, DEVELOPMENT, TESTING

The object of research is the decision – making process based on Game Theory strategies.

The subject of the research is Methods and strategies of Game Theory for decision – making.

The aim of the work is to research, analyze and implement decision – making methods based on Game Theory strategies.

Materials, methods and technical tools: structural and Object-Oriented Programming, decision theory, Game Theory, Java programming language, Gametheory and Algorithms libraries, NetBeans development environment, personal computer running the Microsoft Windows 11 operating system.

Results. A software application has been created that implements the Brown-Robinson method and proposed modifications to it.

Conclusions. A software solution for decision-making based on Game Theory has been developed, namely using the Brown-Robinson method and proposed modifications to it.

The field of use is automated control and management systems.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблемної області та постановка завдань дослідження	11
1.1 Критерії прийняття рішення	11
1.2 Предмет і завдання теорії ігор	13
1.3 Основні поняття та визначення	15
1.4 Прийняття рішень в умовах часткової невизначеності.	22
1.5 Порівняння методів ТІ.....	27
1.6 Постановка завдання	28
1.7 Висновки за розділом 1	28
2 Матеріали та методи	29
2.1 Ітеративний метод Брауна-Робінсона	29
2.2 Метод Брауна для симетричної гри	31
2.3 Модифікація методу Брауна	32
2.4 Висновки за розділом 2	34
3 Розробка архітектури програми	35
3.1 Опис основних вимог до програми	35
3.2 Вибір мови програмування	36
3.3 Бібліотека Gametheory	39
3.4 Бібліотека Algorithms	40
3.5 Вибір середовища розробки.....	41
3.6 Вимоги до технічних засобів	44
3.7 Проєктування розробки.....	44
3.8 Висновки за розділом 3	46
4 Основні рішення щодо реалізації компонентів системи	47
4.1 Програмне забезпечення	47
4.2 Висновки за розділом 4	53
5 Експлуатація, тестування та експериментальне дослідження програми ..	54

5.1 Опис застосунку	54
5.2 Тестовий набір даних	56
5.3 Результати тестування та їх аналіз.....	57
5.4 Тест кейси для розробленого ПЗ.....	59
5.5 Висновки за результатами тестування.....	59
Висновки	60
Перелік джерел посилання	61
Додаток А Технічне завдання	64
Додаток Б Опис програми	69
Додаток В Текст програми	73
Додаток Д Слайди презентації.....	84

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JVM – віртуальна машина Java;

ОПР – особа, яка приймає рішення;

ОС – операційна система.

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ТІ – теорія ігор;

ВСТУП

Світова спільнота була спрямована на розробку інноваційних продуктів, нових технологій, зберігання знань та інформації, експорт високоякісних продуктів [1]-[4]. З цією метою країни Європейського союзу та Скандинавії, США, Китай, Сінгапур та Японія інвестують значні фінансові та людські ресурси в інноваційну діяльність [1]-[4].

Погіршення економічної кон'юнктури, викликане впливом світової кризи, вимагає пошуку нових рішень для оптимізації економічної діяльності [1]-[4]. Стратегічні дії, такі як вдосконалення процесу управління на підприємстві, виробництво інноваційних товарів і послуг, вжиття заходів щодо розробки стратегії оптимального виробництва продукції, сприяють ефективному функціонуванню підприємців та подоланню фінансово-економічної кризи [1]-[4].

Теорія ігор, що вивчає процес прийняття стратегічних рішень, об'єднує розрізнені дисципліни, такі як математика, психологія і філософія [1]-[4]. Теорія ігор була винайдена Джоном фон Нейманом і Оскаром Моргенштерном в 1944 році і з тих пір пройшла довгий шлях [1]-[4]. Важливість теорії ігор для сучасного аналізу та прийняття рішень можна оцінити за тим фактом, що з 1970 року цілих 12 провідних економістів і вчених були удостоєні Нобелівської премії з економічних наук за їх внесок у теорію ігор [1]-[4].

Теорія ігор застосовується в ряді областей, включаючи бізнес, фінанси, економіку, політологію і психологію [1]-[4]. Розуміння стратегій теорії ігор - як популярних, так і деяких відносно менш відомих стратегій - важливо для вдосконалення навичок міркування та прийняття рішень у складному світі [1]-[4].

У дослідженнях конкурентної діяльності вчені розглядають учасників і підкреслюють необхідність розробки стратегій, в яких унікальність

продукту або низький рівень витрат формують основу стратегічної переваги [1]-[4]. Для досягнення цієї мети необхідно забезпечити спадкоємність ідей розвитку і створення ринків майбутнього [1]-[4].

Вчені, які досліджували стратегію та конкурентні переваги в рамках теорії ігор, були зосереджені на прогнозуванні конкурентних дій та пошуку ринкових сигналів, які впливали на їх поведінку [1]-[4]. Такий підхід до вивчення конкурентної поведінки призвів до підвищення репутації, інвестицій у виробничі потужності [1]-[4].

Розробка стратегій для вирішення завдань оптимізації економічної діяльності підприємницьких структур, управління виробничими структурами та їх окремими підрозділами в сучасних мінливих ринкових умовах являє собою складний процес, пов'язаний з різними ризиками і залежить від багатьох факторів [1]-[4]. Беручи до уваги перераховані вище аргументи, управління без використання сучасних математичних інструментів, інформаційних технологій і комп'ютерних ресурсів неможливо [1]-[4].

Мета дипломної кваліфікаційної роботи магістра полягає в тому, щоб навчитися застосовувати теорію ігор в житті, тобто вибирати найбільш вигідні для себе стратегії або хоча б безпрограшні. А для цього розглядається розділ теорії ігор «біматричні ігри» і виконується огляд стратегій їх розв'язання.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Метою першого розділу є перегляд можливості використання одного з класів ігрових моделей – так званих стратегічних ігор – для вирішення завдань прийняття рішень в умовах невизначеності. Для цього надається загальний опис можливості використання теорії ігор для прийняття рішень. Після чого розглядаються термінологічні одиниці проблемної області, а саме: стратегічна гра, класи стратегічних ігор двох осіб з нульовою сумою, а також з постійною ненульовою сумою. Визначається поняття рівноваги в чистих і змішаних стратегіях. Наведено приклади прийняття рішень для різних областей [1]-[4].

1.1 Критерії прийняття рішення

Людині постійно доводиться приймати рішення. Так, перш ніж вийти з дому, ви замислюєтеся над тим, чи взяти парасольку. Звичайно, не хотілося б носити його з собою в гарну погоду [1]-[4]. Але в дощовий день парасолька буде вельми до речі. Постає питання: чи брати, чи не брати парасольку [1]-[4].

Відповісти на це питання нескладно, якщо точно знати, яка буде погода. Якщо з ранку йде дощ, більшість людей візьме парасольки. Хоча, можливо, не всі [1]-[4]. Деякі загартовані люди швидше віддадуть перевагу промокнути, аби не носити весь день з собою парасольку. Наслідки відсутності парасольки в погану погоду оцінюються кожною людиною по-різному. Ці оцінки впливають на рішення [1]-[4].

Складніше прийняти рішення, якщо відсутня достовірна інформація про те, яка очікується погода [1]-[4]. Не можна повністю довіряти прогнозу. Він ніколи не буває абсолютно точним. Ще складніше прийняти рішення, якщо ви не знаєте [1]-[4].

Теорія прийняття рішень-це аналітичний підхід до вибору найкращої альтернативи або послідовності дій [1]-[4]. У теорії прийняття рішень використовуються розумні процедури вибору найкращої з декількох можливих альтернатив [1]-[4].

Наскільки правильним буде вибір, залежить від якості даних, використовуваних при описі ситуації, в якій приймається рішення [1]-[4].

Відповідно існують три типи моделей, які залежать від ступеня визначеності можливих результатів або наслідків, з якими стикається особа, яка приймає рішення (ОПР) [1]-[4].

1. Прийняття рішень в умовах визначеності, коли дані відомі точно і ОПР точно знає наслідки і результати будь-якої альтернативи або вибору рішення. Наприклад, ОПР з повною визначеністю знає, що внесок 100 тис. грн. на поточний рахунок призведе до збільшення балансу цього рахунку на 100 тис. грн. [1]-[4].

2. Методи прийняття рішень в умовах відсутності достовірної інформації про можливі наслідки, коли дані можна описати за допомогою імовірнісних розподілів і ймовірностей настання результатів або наслідків для кожного рішення, вивчаються теорією ризику. Ми можемо не знати того, що завтра буде дощ, але ми можемо знати, що ймовірність дощу 0,3.

Ця теорія має широку сферу застосування в економіці. Одне з найбільш важливих-вибір інвестиційних проектів.

3. Прийняття рішень в умовах невизначеності, коли даним не можна приписати відносні ваги (вагові коефіцієнти), які представляли б ступінь їх значущості в процесі прийняття рішень.

По суті, в умовах визначеності, дані надійно визначені, в умовах невизначеності вони не визначені. Прийняття рішень в умовах ризику, отже, представляє проміжний випадок [1]-[4].

1.2 Предмет і завдання теорії ігор

Переважає більшість соціально-економічних рішень доводиться приймати з урахуванням суперечливих інтересів, що відносяться або до різних осіб або організацій, або до різних аспектів розглянутого явища, або до того й іншого. У таких випадках неможливо застосувати традиційні методи оптимізації. У звичайних екстремальних завданнях мова йде про вибір рішення однією особою, і результат рішення залежить від цього вибору, тобто визначається діями тільки однієї особи. У таку схему не вкладаються ситуації, де рішення, оптимальні для однієї сторони, зовсім не оптимальні для іншої і результат рішення залежить від усіх конфліктуючих сторін [5]-[9].

Конфліктний характер таких завдань не передбачає ворожнечі між учасниками, а свідчить про різні інтереси. Необхідність аналізувати подібні ситуації викликала до життя спеціальний математичний апарат – теорію ігор (ТІ) [5]-[9].

ТІ являє собою частину великої теорії, що вивчає процеси прийняття оптимальних рішень. Вона дає формальну мову для опису процесів прийняття свідомих, цілеспрямованих рішень за участю однієї або декількох осіб в умовах невизначеності і конфлікту, що викликається зіткненням інтересів конфліктуючих сторін [5]-[9].

Метою ТІ є вироблення рекомендацій щодо раціонального способу дій учасників у конфліктних ситуаціях, тобто визначення оптимальної стратегії кожного з них [5]-[9].

Перші роботи по ТІ (Цермело, Борель, фон Нейман) відносяться до початку ХХ століття. Але тільки поява і широке поширення електронних обчислювальних машин привернуло до ТІ увагу широкого кола фахівців.

Теорія стратегічних ігор, у своїй математичній формі, виникла в 30-х роках нашого століття. Її творцем вважається Джон фон Нейман [5]-[9].

Практичне значення ТІ полягає в тому, що вона служить основою моделювання ігрових експериментів, зокрема, ділових ігор, що дозволяють визначати оптимальну поведінку в складних ситуаціях.

Приклади практичного і в тому числі економічного змісту покликані, швидше за все, змістовно інтерпретувати математичні положення теорії ігор, ніж вказувати на фактичні або можливі їх застосування. Від реальної конфліктної ситуації гра відрізняється тим, що ведеться за цілком певними правилами. Реальні конфлікти зазвичай важко піддаються формальному опису, тому будь-яка гра є спрощенням вихідної задачі, в ній відображаються лише основні, першорядні фактори, що відображають суть процесу або явища [5]-[9].

Залежно від того, якими даними володіє дослідник, і яке завдання перед собою ставить, можуть бути сформульовані різні теоретико-ігрові моделі. Розрізняють три основних типи завдань [5]-[9].

1. Знаходження оптимального результату. Як результат в загальному випадку може розглядатися соціально-економічна ситуація. Залежно від змісту завдання ситуацію можна описати наборами благ, одержуваних кожним гравцем (виграшами), або результатом може бути обрання того чи іншого кандидата, прийняття того чи іншого проекту, договору і т.д. При цьому в загальному випадку треба знайти коаліційну структуру і коаліційні стратегії, при яких оптимальний результат реалізується [5]-[9].

2. Знаходження оптимального результату при фіксованій коаліційній структурі, тобто коли нам завідомо відомо, що, наприклад, утворення коаліцій, заборонено, неможливо або наявна коаліційна структура не повинна змінюватися з яких-небудь політичних або економічних міркувань. У цьому випадку спільним завданням є знаходження правил прийняття рішень в коаліціях (порядок винагороди її членів), при яких дана коаліційна структура не розпадеться, і, значить, система буде функціонувати відповідно до інтересів і можливостей її учасників [5]-[9].

3. Знаходження стійкої коаліційної структури при заданих правилах прийняття рішень (Конституції, нормативних актах, статуті підприємства та ін.) в коаліціях. Такі завдання часто зустрічаються при вирішенні економічних і соціальних проблем [5]-[9].

Формалізовані моделі конфліктів відомі з давніх пір: це ігри в буквальному сенсі слова – шахи, карти, кістки і т.п. ці ігри носять характер змагання, що протікає за відомими правилами. Термінологія, запозичена з практики таких ігор, може бути застосована і для інших конфліктних ситуацій, які розглядає ТІ.

1.3 Основні поняття та визначення

Задовго до появи ті широко використовували подібні спрощені моделі конфліктів – гри в буквальному сенсі слова: шашки, шахи, доміно і т.д. звідси і назва самої теорії ігор і різні терміни [5]-[9].

Грою називається всяка конфліктна ситуація, що вивчається в теорії ігор і представляє собою спрощену, схематизовану модель ситуації. Від реальної конфліктної ситуації гра відрізняється тим, що не включає другорядні, несуттєві для ситуації фактори і ведеться за певними правилами, які в реальній ситуації можуть порушуватися

Будь-яка гра включає в себе три елементи: учасників гри – гравців, правила гри, оцінку результатів дій гравців.

Гравцем (особою, стороною, або коаліцією) називається окрема сукупність інтересів, що відстоюється в грі. Якщо дану сукупність інтересів відстоює кілька учасників гри, то вони розглядаються як один гравець. Гравці, які мають протилежні по відношенню один до одного інтереси, називаються противниками. У грі можуть стикатися інтереси двох або більше супротивників. Одна реалізація гри називається партією; вибір дії – в межах правил) – ходом. Ходи бувають Особисті і випадкові. Особистий хід передбачає свідомий вибір тієї чи іншої дії, дозволеного правилами гри, а

випадковий – не залежить від волі гравця (наприклад, він може бути визначений підкиданням монети або гральної кістки і т.п.). Ігри, в яких є особисті ходи, називаються стратегічними. Ігри, що складаються тільки з випадкових ходів, називають азартними. Характерний приклад: гра в лото [5]-[9].

Стратегією гравця називається сукупність правил, що визначають вибір варіанту дій при кожному особистому ході в залежності від ситуації, що склалася [5]-[9].

Залежно від числа стратегій гри діляться на кінцеві і нескінченні. Гра називається кінцевою, якщо у кожного гравця є в розпорядженні тільки кінцеве число стратегій. В іншому випадку гра називається нескінченною.

Оптимальною стратегією гравця називається така, яка забезпечує йому найкраще положення в даній грі, тобто максимальний виграш. Якщо гра повторюється неодноразово і містить, крім особистих, ще й випадкові ходи, оптимальна стратегія забезпечує максимальний середній виграш [5]-[9].

Гра називається грою з нульовою сумою, якщо сума виграшів всіх гравців дорівнює нулю, тобто кожен гравець виграв тільки за рахунок інших. Найпростіший випадок-парна гра з нульовою сумою-називається антагоністичною. Теорія антагоністичних ігор - найбільш розвинений розділ теорії ігор, з чіткими рекомендаціями [5]-[9].

1.3.1 Антагоністичні ігри

Класифікація ігор, що використовуються в ігрових завданнях, представлена на рис. 1.1 [5]-[9].

За ступенем часової залежності безлічі стратегій гравців і елементів платіжної матриці розрізняють динамічні (диференціальні або багатокрокові) і статичні ігри. За кількістю можливих стратегій гри їх поділяють на кінцеві і нескінченні. За кількістю наявної апіорної інформації визначають: гру з

повною інформацією про стан гри, про стратегії дій сторін і про платіжну матрицю; гру з неповною (імовірнісною) інформацією про функції виграшу, про стратегії сторін, про обізнаність сторін (канали спостереження); ігри в умовах повної невизначеності [5]-[9].

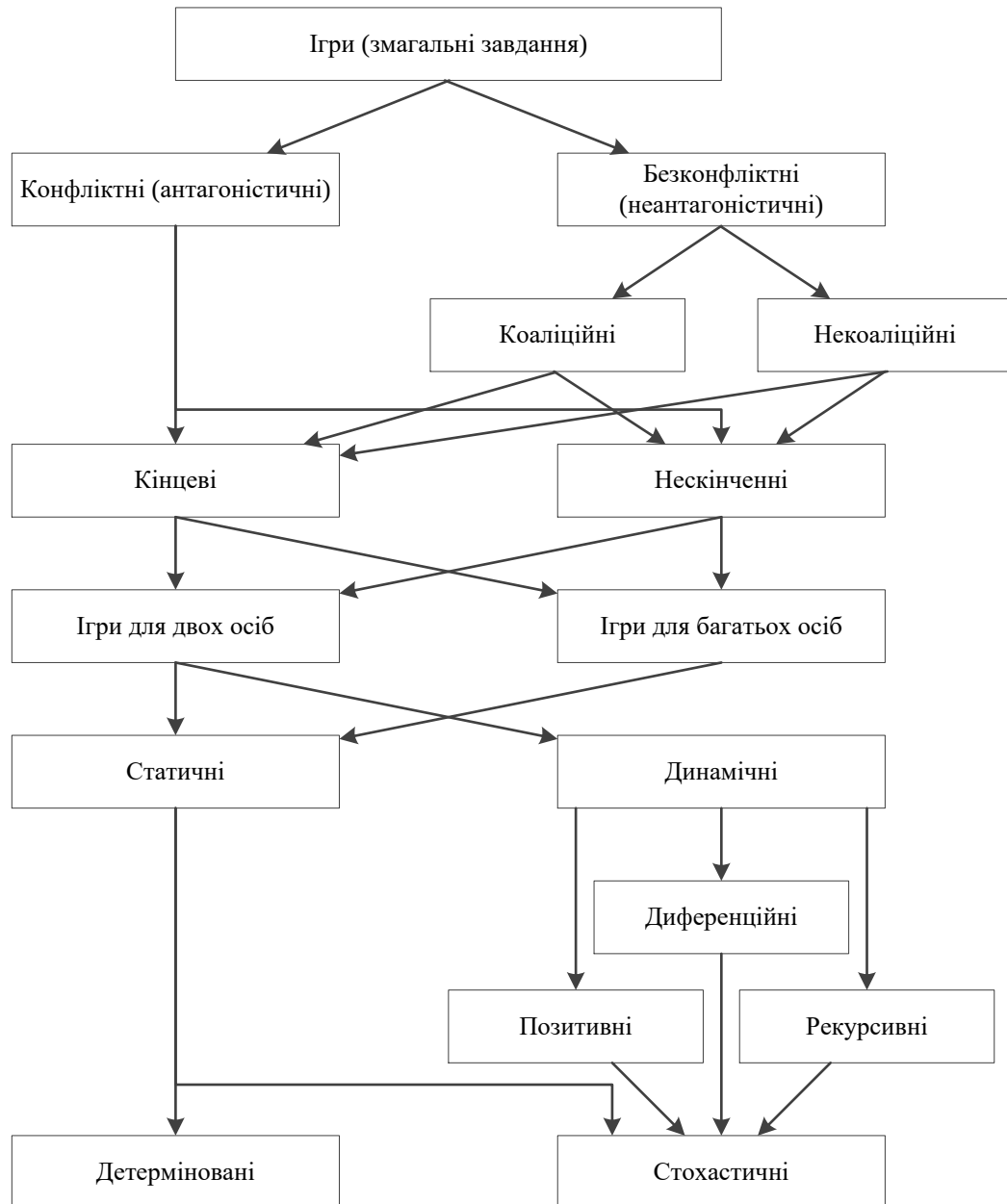


Рисунок 1.1 – Класифікація ігрових завдань

Розглянемо антагоністичні ігри більш детально [5]-[9].

Гру будемо позначати буквою G . У цій грі беруть участь два гравці a і B , що мають протилежні інтереси: виграш одного дорівнює програві іншого.

Так як виграш гравця A дорівнює виграшу гравця B зі зворотним знаком, ми можемо цікавитися тільки виграшем a гравця A . Природно, A хоче максимізувати, а B – мінімізувати a . Для простоти ототожнимо себе з гравцем A і будемо його називати – ми, а гравця B – противник (зрозуміло, ніяких реальних переваг для гравця a з цього не випливає). Нехай у нас є m можливих стратегій $A_1, A_2, \dots, A_m$, а у противника – n -можливих стратегій $B_1, B_2, \dots, B_n$ (така гра називається грою $m \times n$). Позначимо a_{ij} наш виграш у разі, якщо ми користуємося стратегією A_i , а противник – стратегією B_j . Припустимо, що для кожної пари стратегій A_i, B_j виграш (або середній виграш) a_{ij} нам відомий [5]-[9]. Тоді в принципі можна скласти прямокутну таблицю (матрицю), в якій перераховані стратегії гравців і відповідні виграші (див. табл. 1.1).

Таблиця 1.1 – Матриця, в якій перераховані стратегії гравців і відповідні виграші

	B_1	B_2		B_n
A_1	a_{11}	a_{12}		a_{1n}
A_2	a_{21}	a_{22}		a_{2n}
A_m	a_{m1}	a_{m2}		a_{mn}

Якщо така таблиця складена, то кажуть, що гра G приведена до матричної форми. Відзначимо, що саме по собі приведення гри до такої форми вже може скласти важке завдання, а іноді і практично не здійсненне, через незорного безлічі стратегій. Зауважимо, що якщо гра приведена до матричної форми, то багатоходова гра фактично зведена до одноходової – від гравця потрібно зробити тільки один хід: вибрати стратегію. Будемо коротко позначати матрицю гри (a_{ij}) [5]-[9].

Принцип максиміна. Будемо аналізувати гру, використовуючи платіжну матрицю, показану на рис. 1.2. Припустимо, що ми (гравець A)

вибирає стратегію A_1 . Тоді в залежності від того, яку стратегію обере противник, наш виграш буде дорівнює або 8, або 3, або 10. Отже, вибираючи стратегію A_1 , ми в гіршому випадку отримуємо виграш 3. Якщо ж виберемо стратегію A_2 або A_3 , то будемо мати в гіршому випадку виграш 1. Запишемо мінімальні можливі виграші для різних стратегій A_i у вигляді додаткового стовпця платіжної матриці (рис. 1.3). Ясно, що слід вибирати ту стратегію, де мінімальний можливий виграш виявляється найбільшим (в порівнянні з іншими стратегіями). В даному випадку це стратегія A_1 . Виграш 3 є максимальним у трійці мінімальних виграшів (у трійці 3, 1, 1). Його називають максимінним виграшем або, простіше, максиміном. Є у нього ще одна назва-нижня ціна гри [5]-[9].

	B_1	B_2	B_3
A_1	8	3	10
A_2	3	10	1
A_3	10	1	12

Рисунок 1.2 – Платіжна матриця

	B_1	B_2	B_3	
A_1	8	3	10	3
A_2	3	10	1	1
A_3	10	1	12	1
	10	10	12	

Рисунок 1.3 – Мінімальні можливі виграші для різних стратегій у вигляді додаткового стовпця платіжної матриці

Матриці називають сідловою точкою матриці, а про гру говорять, що вона має сідлову точку. Аналогічним чином міркує противник. Якщо він вибере стратегію B_1 , то в гіршому для себе випадку дозволить нам отримати виграш 10. Те ж можна сказати і про стратегію B_2 . При виборі стратегії B_3 найгірший (для противника) випадок відповідає нашому виграшу, Рівному 12. Числа 10, 10, 12-максимальні значення наших виграшів, що відповідають стратегіям противника B_1, B_2, B_3 відповідно. Випишемо ці значення у вигляді додаткової строї платіжної матриці (див. рис. 1.3). Ясно, що противник повинен вибрати ту стратегію, де наш виграш виявляється найменшим. Це є або стратегія B_1 , або B_2 . Обидві стратегії є мінімаксними., обидві вони дають противнику гарантію, що наш виграш не перевищить мінімакса, або, інакше, верхньої ціни гри, рівної в даному випадку 10 [5]-[9].

Наша максимінна стратегія, так само як і мінімаксна стратегія противника, є найбільш обережною, перестраховальною стратегією. Принцип обережності, що диктує гравцям вибір таких стратегій, називають принципом мінімакса.

Підбиваючи підсумок. Антагоністичні ігри, в яких кожен гравець має кінцеве безліч стратегій, називаються матричними іграми. Для завдання такої гри досить виписати так звану платіжну матрицю, в якій рядки відповідають стратегіям першого гравця, а стовпці – стратегіям другого гравця. Елементами матриці служать виграші першого гравця.

Та чи означає все це, що теорія ігор рекомендує дотримуватися тільки мінімаксних (максимінних) стратегій . Відповідь на це питання залежить від того, чи має платіжна матриця гри сідлову точку.

Гра з сідловою точкою [5]-[9].Розглянемо деяку гру 3×3 , платіжна матриця якої дана на рис. 1.4. Тут як максимінний, так і мінімаксний виграші рівні 4. Іншими словами, в даній грі нижня і верхня ціна гри збігаються, обидві рівні 4. Виграш 4 є одночасно і максимальним з мінімальних виграшів для стратегій A_1, A_2, A_3 і мінімальним з максимальних виграшів для стратегій B_1, B_2, B_3 . В геометрії точку на поверхні, що є одночасно мінімумом по одній

осі координат і максимумом по іншій, називають сідловою точкою. За аналогією з геометрією елемент $a_{22}=4$ розглянутої тут платіжної.

Досить подивитися уважно на матрицю, показану на рис. 1.4, щоб зрозуміти, що кожен з гравців повинен дотримуватися максимінної (мінімаксної) стратегії. Ці стратегії є оптимальними в грі з сідловою точкою. Будь-яке відхилення від них буде невигідно для гравця, який допустив відхилення [5]-[9].

	B_1	B_2	B_3	
A_1	2	3	7	2
A_2	5	4	6	4
A_3	6	2	1	1
	6	4	7	

Рисунок 1.4 – Платіжна матриця з сідловою точкою

Якщо ж гра не має сідлової точки (див. рис. 1.3), то жодна зі стратегій A_i або B_i не є оптимальною.

Як бути, якщо гра не має сідлової точки. Якщо кожен гравець змушений вибирати одну-єдину чисту стратегію, то робити нічого: треба керуватися принципом мінімакса. Інша справа, якщо можна свої стратегії змішувати, чергувати випадковим чином з якимись ймовірностями. Застосування змішаних стратегій мислиться таким чином: гра повторюється багато разів; перед кожною партією гри, коли гравцеві надається особистий хід, він передоручує свій вибір випадковості, кидає жереб, і бере ту стратегію, яка випала [5]-[9].

Змішані стратегії в теорії ігор представляють модель мінливої, гнучкої тактики, коли жоден з гравців не знає, як поведе себе противник в даній

партії. Така тактика (правда, зазвичай без жодних математичних обґрунтувань) часто застосовується в карткових іграх.

1.4 Прийняття рішень в умовах часткової невизначеності

Предметом розгляду даного розділу служать статистичні моделі прийняття рішень, що трактуються як статистичні ігри або ігри з природою при використанні додаткової статистичної інформації про її стратегії. Характерна риса статистичної гри-можливість отримання інформації в результаті деякого статистичного експерименту для оцінки розподілу ймовірностей стратегій природи. Дослідження механізму випадкового вибору стратегії природою дозволяє прийняти оптимальне рішення, яке буде найкращою стратегією в грі з неантагоністичним противником людини – природою [5]-[9].

У розглянутих розділах теорії ігор передбачалося, що обидва противника (або більше двох) активно протидіють один одному, що обидва вони досить розумні, щоб шукати і знайти свою оптимальну стратегію, і обережні, щоб не відступати від неї. Такий стан дає можливість передбачати поведінку гравців. Невизначеність була лише у виборі противником конкретної чистої стратегії в кожній окремій партії [5]-[9].

Але можливий випадок, коли невизначеність в грі викликана не свідомим протидією противника, а незнанням умов, в яких буде прийматися рішення, випадкових обставин. Такі ігри називаються іграми з природою.

Гра людини з природою теж відображає конфліктну ситуацію, що виникає при зіткненні інтересів у виборі рішення. Але стихійним силам природи не можна приписати розумні дії, спрямовані проти людини і тим більше будь-якої злий умисел. Таким чином, коректніше говорити про конфліктну ситуацію, викликану зіткненням інтересів людини і невизначеністю дій природи [5]-[9].

Дії природи можуть, як завдавати шкоди, так і приносити прибуток. Поведінку природи можна оцінити статистичними методами, визначити властиві їй закономірності. Залежно від ступеня знання цих закономірностей, що визначають поведінку природи, розрізняються гри з природою в умовах визначеності та гри з природою в умовах невизначеності.

По-перше, поведінка природи відомо повністю (задані ймовірностями). По-друге-дії природи не відомі, або вивчені частково.

До явищ природи, що впливає на результат рішення, відносять не тільки погодні та сезонні явища (дощ, посуху, урожай, неврожай), але і прояв будь-яких, не залежних від нас обставин: наприклад, затримки на транспорті.

Пошуком рішень в таких ситуаціях і займається теорія статистичних рішень [10]-[13].

Людина, граючи з природою, прагнути максимізувати свій виграш, тому, якщо він обережний гравець (а теорія ігор розглядає саме таких гравців), він повинен при виборі своєї стратегії керуватися тим, що невідомі або відомі йому закономірні дії природи приведуть до найменш сприятливих наслідків. Саме тому такі ігри можна розглядати як ігри двох осіб з нульовою сумою, які були вже нами розглянуті [10]-[13].

Формалізація завдання відбувається наступним чином: у активного гравця (людини) можливі дії як і раніше називаються стратегіями, а можливі дії пасивного гравця (природи) – станами або умовами природи.

В якості першого гравця завжди виступає людина, тому в матриці записується його виграш. Так як нас цікавить оптимальна стратегія людини і його гарантований виграш, то в гру досить визначити максимінну стратегію першого гравця і нижню ціну гри. Визначення верхньої ціни гри має сенс, якщо дана гра повторюється багаторазово і оптимальна стратегія може бути змішаною [10]-[13].

1.4.1 Критерій песимізму-оптимізму Гурвіца

Видається логічним, що при виборі рішення замість двох крайнощів в оцінці ситуації дотримуватися деякої проміжної позиції, що враховує можливість як найгіршого, так і найкращого, сприятливого поведінки природи. Такий компромісний варіант і був запропонований Гурвіцем. Відповідно до цього підходу для кожного рішення необхідно визначити лінійну комбінацію $\min$ і $\max$ виграшу і взяти ту стратегію, для якої ця величина виявиться найбільшою, що представлено на формулі (1.1) [10]-[13]:

$$v^H = \max_i [a \max_j a_{ij} + (1-a) \min_j a_{ij}], \quad (1.1)$$

де a – ступінь оптимізму.

При $a=0$ критерій Гурвіца тотожний критерію Вальда, а при $a=1$ збігається з максимінним рішенням.

На вибір значення ступеня оптимізму впливає міра відповідальності: чим серйозніше наслідки помилкових рішень, тим більше бажання приймає рішення застрахуватися, тобто ступінь оптимізму a ближче до нуля [10]-[13].

1.4.2 Критерій Севіджа (критерій мінімакса ризику)

На практиці, вибираючи одне з можливих рішень, часто зупиняються на тому, здійснення якого призведе до найменш тяжких наслідків, якщо вибір виявиться помилковим. Цей підхід до вибору рішення математично був сформульований американським статистиком Севіджем в 1954 році і отримав назву принципу Севіджа. Він особливо зручний для економічних завдань і часто застосовується для вибору рішень в іграх людини з природою [10]-[13].

За принципом Севіджа кожне рішення характеризується величиною додаткових втрат, які виникають при реалізації цього рішення, в порівнянні з реалізацією рішення, правильного при даному стані природи. Природно, що

правильне рішення не тягне за собою ніяких додаткових втрат, і їх величина дорівнює нулю.

При виборі рішення, найкращим чином відповідного різним станам природи, слід брати до уваги тільки ці додаткові втрати, які по суті, будуть наслідком помилок вибору [10]-[13].

Для вирішення завдання будується так звана матриця ризиків, елементи якої показують, який збиток понесе гравець (ОПР) в результаті вибору неоптимального варіанту рішення.

Ризиком гравця r_{ij} при виборі стратегії i в умовах (станах) природи j називається різниця між максимальним виграшем, який можна отримати в цих умовах і виграшем, який отримає гравець в тих же умовах, застосовуючи стратегію i [10]-[13].

Якби гравець знав заздалегідь майбутнє стан природи j , він вибрав би стратегію, якій відповідає $\max_i$ елемент в даному стовпці – $\max_i a_{ij}$, що представлено на формулі (1.2):

$$r_{ij} = \max_i a_{ij} - a_{ij} . \quad (1.2)$$

Критерій Севіджа рекомендує в умовах невизначеності вибирати рішення, що забезпечує мінімальне значення максимального ризику, що представлено на формулі (1.3):

$$v^S = \min_i \max_j r_{ij} = \min_i \max_j (\max_i a_{ij} - a_{ij}) . \quad (1.3)$$

1.4.3 Критерій Лапласа

У ряді випадків представляється правдоподібним наступне міркування: оскільки невідомі майбутні стану природи, остільки можна

вважати їх рівноймовірними. Цей підхід до вирішення використовується в Критерії недостатнього підстави Лапласа [10]-[13].

Для вирішення задачі для кожного рішення підраховується математичне очікування виграшу (ймовірності станів природи покладаються рівними $q_j = \frac{1}{n}, j = 1:n$), і вибирається те рішення, при якому величина цього виграшу максимальна, що представлено на формулі (1.4) [10]-[13].

$$v^L = \min_i \frac{a_i}{n \cdot a_{ij}} \max_j r_{ij} = \frac{1}{n} \min_i a \cdot a_{ij}. \quad (1.4)$$

Гіпотеза про равноймовірності станів природи є досить штучною, тому принципом Лапласа можна користуватися лише в обмежених випадках. У більш загальному випадку слід вважати, що стану природи не рівноймовірні і використовувати для вирішення критерій Байєса-Лапласа [10]-[13].

1.4.4 Критерій Байєса-Лапласа

Цей критерій відступає від умов повної невизначеності - він передбачає, що можливим станам природи можна приписати певну ймовірність їх настання і, визначивши математичне очікування виграшу для кожного рішення, вибрати те, яке забезпечує найбільше значення виграшу, що представлено на формулі (1.5):

$$v^{III} = \max_i a \cdot a_{ij} \cdot q_i. \quad (1.5)$$

Цей метод передбачає можливість використання будь-якої попередньої інформації про стани природи. При цьому передбачається як повторюваність станів природи, так і повторюваність рішень, і перш за все,

наявність досить достовірних даних про минулі стани природи. Тобто, ґрунтуючись на попередніх спостереженнях прогнозувати майбутній стан природи (статистичний принцип) [10]-[13].

1.5 Порівняння методів ТІ

Для вирішення завдання прийняття рішення з використанням методів ТІ слід чітко проаналізувати їх переваги та недоліки, більше дізнатися про залежність їх ефективності від умов задачі та рівня невизначеності [10]-[13].

Завдання теорії ігор-вироблення рекомендацій щодо раціонального способу дій учасників конфлікту. Це означає, що можна виробити оптимальні правила поведінки кожної сторони, яка бере участь у вирішенні конфліктної ситуації. При цьому будують спрощену модель конфліктної ситуації, звану грою.

Порівнюємо різні методи ТІ, а результати представимо у вигляді таблиці – 1.2.

Таблиця 1.2 – Порівняння методів

	Метод Брауна-Робінсона	Метод зведення гри до задачі лінійного програмування	Матрична гра
Залежність від розмірності гри (задачі)	Низька	Висока	Висока
Швидкість збіжності	Повільна	Середня	Середня
Поведінка в умовах більшої невизначеності	Не впливає на роботу	Середнє впливає на роботу	Середнє впливає на роботу
Перегляд позитивних стратегій гравців	Основний результат роботи	Загальне представлення стратегій	Загальне представлення стратегій

Як видно з таблиці порівняння, більшість методів ТІ схожі, адже переслідують однакові цілі та працюють за схожим сценарієм. Проте саме метод Брауна-Робінсона з порівня є найбільш перспективним для подальшого дослідження та запропонування модифікацій з метою покращення результатів. Одним з головних недоліків методу є повільна швидкість збіжності методу проте внесення певних модифікацій в подальшому має вирішити цю проблему.

1.6 Постановка завдання

Дослідити та реалізувати один з методів ТІ, а саме метод Брауна-Робінсона для автоматизації прийняття рішень. Додатково розглянути можливі модифікації методу для вирішення проблем методу, а саме пришвидшення швидкості збіжності.

1.7 Висновки за розділом 1

Під час роботи над першим розділом було розглянуто проблемну ТІ та область використання методів ТІ. Було детально проаналізовано переваги та недоліки існуючих методів, було обрано один з найбільш перспективних методів для подальшої роботи та внесення модифікацій з метою усунення існуючих недоліків та розширення кола практичних завдань.

2 МАТЕРІАЛИ ТА МЕТОДИ

В першому розділі було проаналізовано сучасне становище проблемної прийняття рішень та використання стратегій ТІ для автоматизації прийняття рішень. Було проаналізовано сучасні методи.

За результатами аналізу найбільш перспективним було визнано метод Брауна-Робінсона. Проте, як і усі методи він має свої недоліки, а саме повільну збіжність.

У другому розділі планується більш детально проаналізувати класичний ітеративний метод Брауна-Робінсона. Виявити його переваги та недоліки для того, щоб запропонувати ряд модифікацій які в подальшому допоможуть уникнути їх та підвищити якісні характеристики методу.

2.1 Ітеративний метод Брауна-Робінсона

Одним з найбільш відомих ітеративних методів відшукування сідлової точки в матричних іграх є метод Брауна-Робінсона. Сутність методу полягає в багаторазовому фіктивному розігруванні гри і певному правилі вибору чистих стратегій в повтореннях гри. У кожній партії гравці вибирають такі чисті стратегії, які забезпечують першому максимальний середній виграш, а другому – мінімальний середній програш з урахуванням частот вибору чистих стратегій партнером, що представлено на формулі (2.1) [10], [14]:

$$\begin{aligned} i(k+1) &= \underset{0 \leq i \leq m}{\operatorname{Arg} \max} (E(i, Q(k))), \\ j(k+1) &= \underset{0 \leq j \leq n}{\operatorname{Arg} \min} (E(P(k), j)), \end{aligned} \quad (2.1)$$

де i, j – стратегії першого та другого гравця відповідно;

m, n – лічильники кількості можливих стратегій для кожного гравця;

(P, Q) – сідлова точка в матричних іграх, за умови $P = (p_1, \dots, p_m)$ та $Q = (q_1, \dots, q_n)$;

$E(i, j)$ – математичне сподівання виграшу (програшу) одного (другого) гравця.

При цьому справедливі верхня і нижня оцінки для ціни гри на кроці k , що представлено на формулі (2.2) [10], [14]:

$$\begin{aligned} \max_{0 \leq i \leq m} (E(i, Q(k))) &= v_1(k), \\ \min_{0 \leq j \leq n} (E(P(k), j)) &= v_2(k). \end{aligned} \quad (2.2)$$

З використанням (2.2) в якості наближеного значення ціни гри можна прийняти середнє арифметичне верхньої і нижньої цін гри на даному кроці, що представлено на формулі (2.3) [10], [14]:

$$v(k) = \frac{v_1(k) + v_2(k)}{2}. \quad (2.3)$$

Змішані стратегії гравців у методі Брауна на наступній ітерації мають вигляд, що представлено на формулі (2.4):

$$\begin{aligned} P(k+1) &= \frac{P(k)k + P_{i(k+1)}}{k+1}, \\ Q(k+1) &= \frac{Q(k)k + Q_{j(k+1)}}{k+1}, \\ k &= 1, 2, 3, \dots \end{aligned} \quad (2.4)$$

де P_i, Q_j – змішані стратегії, що відповідають чистим стратегіям i, j із (2.1).

Ітерації зазвичай припиняються при досягненні точності, з якою знайдена ціна гри. тобто при виконанні умови, що представлено на формулі (2.5) [10], [14]:

$$v_1(k) - v_2(k) \leq \varepsilon. \quad (2.5)$$

Вибір такого правила обумовлений тим, що в методі Брауна-Робінсона $\lim_{k \rightarrow \infty} v_1(k) = \lim_{k \rightarrow \infty} v_2(k) = v$, а будь-які граничні точки p^0 і q^0 послідовностей $\{P(k)\}$, $\{Q(k)\}$ $\{()(\kappa)\}$ є оптимальними змішаними стратегіями гравців [10], [14].

2.2 Метод Брауна для симетричної гри

Антагоністичні ігри, в яких гравці рівноправно беруть участь в грі, називаються симетричними [10], [14].

Розглянемо динаміку зміни оцінок ціни гри в методі Брауна на прикладі 5×5 симетричної матричної гри, згенерованої випадковим чином, що представлено на формулі (2.6):

$$A = \begin{pmatrix} 0 & 10 & 0 & 9 & -7 \\ -10 & 0 & -5 & 7 & 3 \\ 0 & 5 & 0 & -2 & -1 \\ -9 & -7 & 2 & 0 & -3 \\ 7 & -3 & 1 & 3 & 0 \end{pmatrix}. \quad (2.6)$$

Провівши 2000 кроків стандартним методом Брауна, отримали ціну гри $v = 0,0375$ з точністю $\varepsilon = 0,154$: за 4000 кроків – ціна гри $v = 0,0115$ з точністю $\varepsilon = 0,108$. Експерименти показали, що метод Брауна повільно сходиться до ціни гри.

2.3 Модифікація методу Брауна

У симетричній матричній грі оптимальні стратегії сторін збігаються і ціна гри $v = 0$. Для таких ігор в роботі запропонована наступна модифікація ітеративного методу. Після проведення фіксованого числа кроків (параметр методу) стандартним методом, з'ясовується, яка з оцінок ціни гри, верхня або нижня, ближче до ціни гри (рис. 2.1) [14].

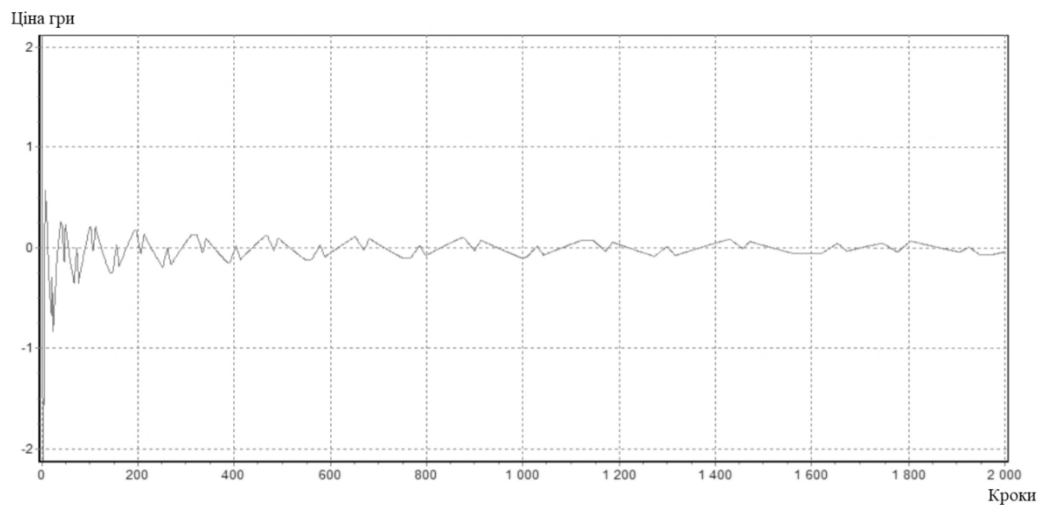


Рисунок 2.1 – Збіжність послідовності оцінок ціни гри за методом Брауна

Якщо $|v_1(k)| < |v_2(k)|$, то стратегія першого краще стратегії другого, і другому гравцеві доцільно взяти стратегію першого: $Q(k) = P(k)$.

Якщо $|v_1(k)| > |v_2(k)|$, то аналогічно $P(k) = Q(k)$.

Момент часу, в який порівнюються оцінки ціни гри, назовемо моментом перемикання [14].

На рис. 2.2 наведені графіки, що ілюструють роботу стандартного і модифікованого методів на розглянутому прикладі (2).

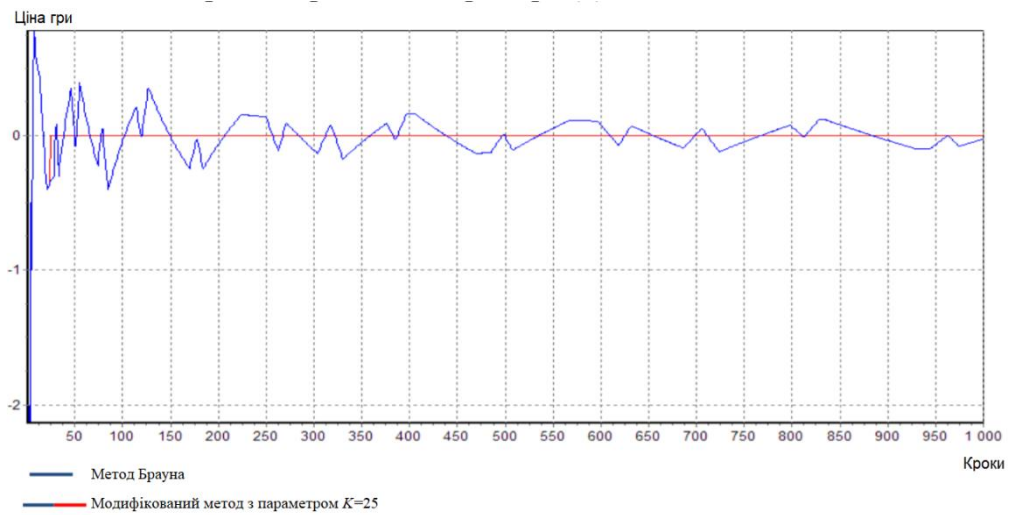


Рисунок 2.2 – Графік зміни оцінок ціни гри (1000 кроків)

Обчислення похибки відбуватиметься за наступною формулою, що представлено на формулі (2.7):

$$\varepsilon(k) = |v_1(k) - v_2(k)|. \quad (2.7)$$

2.3.1 Модифікація за верхньою ціною гри

Однак підібрати оптимальне значення параметра перемикання методу в загальному випадку не вдається. Навіть незначна зміна параметра K може призводити до збільшення похибки [14].

Щоб позбутися від проблеми вибору значення параметра, введемо динамічне правило перемикання:

- якщо $|v_1(k+1)| \leq |v_1(k)|$, то продовжуємо метод Брауна;
- якщо $|v_1(k+1)| > |v_1(k)|$ – то виконуємо процедуру порівняння верхньої і нижньої оцінок гри.

Таку модифікацію назовемо модифікацією за верхньою ціною гри. Аналогічно можна розглядати модифікацію за нижньою ціною гри, і за ціною гри, перевіряючи нерівність що представлено на формулі (2.8) :

$$|v_1(k+1)+v_2(k+1)| < |v_1(k)+v_2(k)|. \quad (2.8)$$

2.4 Висновки за розділом 2

У другому розділі проаналізовано запропонований метод Брауна-Робінсона. Було виявлено його переваги та недоліки.

З метою усунення існуючих недоліків було запропоновано ряд модифікацій, які в подальшому мають покращити практичні результати використання методу.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Другий розділ було присвячено дослідженню запропонованого методу та можливих модифікацій для покращення його якісних характеристик.

Третій розділ буде присвячено висвітленню питань з проєктування архітектури та реалізації програмного комплексу для дослідження запропонованих методів та можливих модифікацій. Особливу увагу буде присвячено вибору технічних засобів: мови програмування, середовища розробки, тощо.

3.1 Опис основних вимог до програми

Програмне забезпечення (ПЗ) для автоматизації прийняття рішень на основі стратегій ТІ використовується для прийняття рішень та подальшого керування та контролю об'єктами, системами та процесами (технічними, виробничими).

Таким чином серед функціональних вимог виділяються:

- завантаження набору даних;
- генерація набору вхідних даних (у випадку простого тестування роботи);
- реалізація звичайного методу Брауна-Робінсона для прийняття рішень та визначення позитивних стратегій;
- реалізація запропонованих модифікацій методу Брауна-Робінсона для дослідження нових можливостей методу та його чутливості до вносимих модифікацій.

Виконання наведених функцій в повній мірі розкрис практичний потенціал запропонованих методів та розроблюваного ПЗ.

3.2 Вибір мови програмування

В області прийняття рішень та ТІ використовується безліч мов програмування. Але Java, незважаючи на свою значну історію (більше 25 років), все ще залишається одним з найпопулярніших в цій сфері [15], [16].

Мова програмування Java відрізняється [15], [16]:

- універсальністю: вона призначена для використання, для написання програмного забезпечення в самих різних областях додатків і не має спеціалізованих функцій для будь-якого конкретного домену;
- class-based: її структура об'єктів визначена в класах. У екземплярів класів завжди є ті поля і методи, які вказані в їх визначеннях класів;
- статичною типізацією: компілятор перевіряє під час компіляції, що типи змінних дотримуються. Наприклад, якщо метод очікує аргумент типу String, цей аргумент насправді повинен бути рядком при виклику методу;
- об'єктно-орієнтовністю: більшість речей у програмі Java є екземплярами класу, тобто пучками станів (полів) і поведінки (методи, які працюють з даними і утворюють інтерфейс об'єкта для зовнішнього світу);
- portable: він може бути скомпільований на будь-якій платформі за допомогою javac і отримані файли класів можуть запускатися на будь-якій платформі з JVM [15], [16].

Java призначений для того, щоб дозволити розробникам додатків «писати один раз, працювати де завгодно», що означає, що скомпільований Java-код може працювати на всіх платформах, що підтримують Java, без необхідності перекомпіляції [15], [16].

Java-код скомпільований в байт-код («.class»), які, в свою чергу, інтерпретуються віртуальною машиною Java (JVM). Теоретично байт-код, створений одним компілятором Java, повинен працювати однаково на будь-якому JVM, навіть на іншому комп'ютері. JVM може (і в реальних

програмах) вибирати для компіляції у власні машинні команди частини байт-коду, які виконуються часто. Це називається компіляцією «точно в строк» [15], [16].

Java-версії та стандарти. Існує три видання Java, визначені Sun/Oracle:

- Java Standard Edition – це видання, призначене для загального використання;
- Java Enterprise Edition – додає ряд можливостей для створення сервісів рівня підприємства на Java. Java EE розглядається окремо;
- Java Micro Edition – заснована на підмножині Java SE і призначена для використання на невеликих пристроях з обмеженими ресурсами.

Java володіє значними перевагами в порівнянні з іншими мовами і середовищами, які роблять його відповідним практично для будь-якого завдання програмування [15], [16].

Переваги Java полягають в наступному [15], [16]:

- Java легко освоїти. Java була розроблена, щоб бути простою у використанні, і тому її легше писати, компілювати, налагоджувати і вивчати, ніж інші мови програмування;
- Java є об'єктно-орієнтованою. Це дозволяє створювати модульні програми і повторно використовуваний код;
- Java не залежить від платформи. Одним з найбільш істотних переваг Java є її здатність легко переміщатися з однієї комп'ютерної системи на іншу. Можливість запускати одну і ту ж програму на багатьох різних системах має вирішальне значення для програмного забезпечення Всесвітньої павутини, і Java досягає успіху в цьому, будучи незалежною від платформи як на вихідному, так і на двійковому рівнях [15], [16];
- завдяки надійності Java, простоті використання, крос-платформних можливостей і функцій безпеки, він став мовою вибору для надання глобальних інтернет-рішень.

Висока продуктивність мови. Хоча продуктивність інтерпретованого байт-коду зазвичай більш ніж достатня, бувають ситуації, коли потрібна більш висока продуктивність. Байт-коди можуть бути переведені «на льоту» (під час виконання) в машинний код для конкретного процесора, на якому працює додаток. Для тих, хто звик до звичайного дизайну компілятора і динамічного завантажувача, це чимось схоже на установку кінцевого генератора машинного коду в динамічний завантажувач [15], [16].

Формат байт-коду був розроблений з урахуванням генерації машинних кодів, тому фактичний процес генерації машинного коду, як правило, простий. Створюється досить хороший код: він автоматично розподіляє регістри, і компілятор виконує деяку оптимізацію при створенні байт-кодів [15], [16].

У інтерпретованому коді ми отримуємо близько 300 000 викликів методів в секунду на Sun Microsystems SPARCstation 10. Продуктивність байт-коду, перетвореного в машинний код, майже не відрізняється від мови C або C++ [15], [16].

Мова Java забезпечує потужне доповнення до інструментів, якими володіють програмісти. Java спрощує програмування, оскільки вона об'єктно-орієнтована і має автоматичну збірку сміття. Крім того, оскільки скомпільований код Java не залежить від архітектури, застосунки Java ідеально підходять для такого різноманітного середовища, як Інтернет [15], [16].

У таблиці 3.1 наведено порівняння обраної мови програмування із іншими найпопулярнішими мовами програмування.

Як видно з таблиці не дивлячись на те, що мова програмування Java виникла вже достатньо давно, вона і зараз достатньо популярна та активно продовжує розвиватися. Більш того саме вона найбільш вдалий вибір, якщо майбутнє ПЗ повинне бути кроссплаформним.

За результатами можна зробити висновок, що велике практичне коло використання Java для розробки застосунків значно відрізняє її спроміж

інших аналогів. А не дивлячись на те, що і для Python, і для Matlab на сьогоднішній день вже існує велика кількість бібліотек, Java залишається одним з лідерів за швидкістю роботи застосунків, що розробляються на ній.

Саме тому для розробки ПЗ було обрано саме Java.

Таблиця 3.1 – Порівняння мов програмування

Критерії порівняння	Мови програмування		
	Java	Matlab	Python
Наявність матеріалів для вивчення та підтримки розробників	++	++	++
Парадигми програмування (об'єктно-орієнтовне, функціональне, тощо)	+	—	++
Бібліотеки шаблонів структур даних	++	++	++
Кроссплатформеність	++	—	++
Швидкість роботи та загальна продуктивність застосунків	++	—	+
Бібліотеки для розширення функціоналу	++	+	++
Розробка графічного інтерфейсу користувача	++	+	+
Паралельне виконання високопродуктивних обчислень	++	±	++

3.3 Бібліотека Gametheory

Для роботи з параметрами моделі не потрібно ніяких зовнішніх залежностей. Код Java використовує базові бібліотеки, знайдені в будь-якому сучасному java JDK. Однак для того, щоб відобразити результати, необхідний інструмент побудови графіків, який може поєднуватися з кодом Java [17].

Нічого не потрібно встановлювати. Бібліотека просто додається Java до існуючого проекту [17].

Автори: August von Nacht та Karl Bäckström.

3.4 Бібліотека Algorithms

Неформально алгоритм – це будь-яка чітко визначена обчислювальна процедура, яка приймає деяке значення або набір значень в якості вхідних даних і видає деяке значення або набір значень в якості вихідних. Таким чином, алгоритм являє собою послідовність обчислювальних кроків, які перетворюють вхідні дані у вихідні [18].

Алгоритм повинен володіти трьома важливими характеристиками, щоб вважатися дійсним [18]:

- він повинен бути кінцевим: якщо ваш алгоритм ніколи не припиняє спроби вирішити проблему, для вирішення якої він був розроблений, то він марний;
- у ньому повинні бути чітко визначені інструкції: кожен крок алгоритму повинен бути точно визначений; інструкції повинні бути однозначно вказані для кожного випадку;
- він повинен бути ефективним: алгоритм повинен вирішувати задачу, для вирішення якої він був розроблений. І повинна бути можливість продемонструвати, що алгоритм сходиться тільки з папером і олівцем.

Категорії структура проекту в бібліотеці Algorithms [18]:

- штучний інтелект;
- маніпуляція з бітами;
- клітинний автомат;
- шифрування;
- обчислювальна геометрія;
- криптографія;
- структури даних;
- динамічне програмування;
- теорія ігор;
- графіка;

- жадібні алгоритми;
- математика;
- чисельний аналіз;
- рандомізовані алгоритми;
- алгоритми вибору;
- сортування.

Серед можливостей алгоритмів штучного інтелекту виділяються [18]:

- лінійна регресія;
- логістична регресія;
- штучна нейронна мережа;
- згорткова нейронна мережа;
- дерево рішень ті інші.

Для продуктивної роботи бібліотека надає шаблони певних спеціалізованих структур даних для реалізації моделей з різними топологіями [18]:

- торба
- хеші
- пов'язаний список
- список
- черга
- стек
- дерево.

3.5 Вибір середовища розробки

У червні 2000 року компанія Sun Microsystems зробила NetBeans відкритим вихідним кодом, яка залишалася спонсором проекту до січня 2010 року, коли Sun Microsystems стала дочірньою компанією Oracle [19].

Протягом всієї своєї історії в Sun Microsystems і Oracle NetBeans була безкоштовною і з відкритим вихідним кодом, і її спонсор використовував її в якості механізму для просування екосистеми Java вперед.

У 2016 році Oracle передала вихідний код NetBeans до Фонду програмного забезпечення Apache. У квітні 2019 року Apache NetBeans став проектом Apache вищого рівня. Будь ласка, ознайомтеся з нашим Розділом історії для отримання додаткової інформації [19].

Apache NetBeans – це проект Apache вищого рівня, присвячений надання надійних продуктів для розробки програмного забезпечення (IDE Apache NetBeans і платформа Apache NetBeans), які задовольняють потреби розробників, користувачів та підприємств, які покладаються на NetBeans в якості основи для своїх продуктів; зокрема, щоб вони могли швидко, ефективно і легко розробляти ці продукти, використовуючи сильні сторони платформи Java та інших відповідних галузевих стандартів [19].

Два основні продукти, IDE Apache NetBeans та платформа Apache NetBeans, є безкоштовними для комерційного та некомерційного використання відповідно до ліцензії Apache. Вихідний код для обох доступний будь-якому користувачеві для повторного використання на свій розсуд відповідно до Умов використання.

Проект Apache NetBeans також є активним співтовариством, в якому люди з усього світу можуть задавати питання, давати поради, вносити свій внесок і в кінцевому підсумку брати участь в успіху наших продуктів. У списках розсилки та форумах NetBeans ви знайдете повідомлення від студентів, розробників з провідних компаній та приватних осіб, які бажають розширити свої навички [19].

Завдяки більш ніж 18 мільйонам завантажень IDE NetBeans на сьогоднішній день і більш ніж 800 000 розробників, що беруть участь, проект NetBeans процвітає і продовжує зростати завдяки приватним особам і компаніям-партнерам. Щоб стати частиною спільноти, просто зареєструйтеся безкоштовно.

Більшість розробників визнають IDE NetBeans оригінальним безкоштовним середовищем розробки Java. Саме це і багато іншого! Середовище IDE Apache NetBeans забезпечує підтримку декількох мов (Java, PHP, JavaFX, JavaScript і т.д.) і фреймворків, і найближчим часом будуть включені інші (C/C++) [19].

Платформа Apache NetBeans – це універсальна платформа для застосунків Swing. Вона забезпечує техніку, яку раніше кожен розробник повинен був писати самостійно – збереження стану, підключення дій до пунктів меню, елементів панелі інструментів і сполученням клавіш; управління вікнами і так далі [19].

У таблиці 3.2 наведено порівняння обраного середовища розробки для мови програмування Java з іншими найпопулярнішими рішеннями.

Таблиця 3.2 – Порівняння середовищ розробки

Критерії	NetBeans	IntelliJ IDEA
Кроссплатформність	$\pm$	$\pm$
Оновлення та підтримка	+	++
Системи контролю версій	++	++
Серверна підтримка	++	++
Інструментарій для розробки графічного інтерфейсу	++	++
Ліцензія (безкоштовність)	++	$\pm$

Як видно із таблиці порівняння більша кількість балів у NetBeans, але IntelliJ IDEA розповсюджується за платною ліцензією, у той час, як NetBeans офіційно безкоштовний. З іншого боку NetBeans працює із більш старшими версіями Java та JDK, у той час, як IntelliJ IDEA підтримує завжди найактуальніші версії Java та JDK. Тож можна зробити висновок, що NetBeans – найбільш оптимальний варіант для подальшої розробки.

3.6 Вимоги до технічних засобів

Вимоги до складу і параметрів технічних засобів

Технічна характеристика:

- персональний комп'ютер (ПК) з операційною системою (ОС) Microsoft Windows 8.1 та старше;
- Java Virtual Machine (JVM) 9.0 та вище;
- процесор сімейства Intel (лінійка Core) або AMD (лінійка Ryzen) з тактовою частотою 2.0 МГц та вище;
- вільне місце на накопичувачі від 1 GB;
- оперативна пам'ять від 4 GB.

3.7 Проектування розробки

Для зручності реалізації, підтримки та подальшого оновлення ПЗ пропонується реалізувати його в модульному форматі, що представлено на рисунку (3.1).

Так при проектуванні було надано перевагу розробленню 2 основних модулів:

- програмний модуль;
- модуль графічного інтерфейсу користувача.

Так програмний модуль повинен відповідати за 2 основних пакети: підтримка роботи із даними та підтримка обчислень.

Модуль графічного інтерфейсу відповідає за взаємодію ПЗ із користувачем, адже планується виведення особливих графічних результатів роботи.

Окремо планується реалізувати базу даних для збереження проміжкових даних про роботу застосунку.

Така організація архітектури застосунку спростить розробку, забезпечить розподіл функціоналу між окремими пакетами та структурними

одинацями (класами), що гарантує легкість підтримки та оновлення функціоналу застосунку, більш того буде доречним при використанні сторонніх бібліотек та плагінів для розширення функціоналу.

Подальша реалізація буде відбуватися з урахуванням розробленої структури та розподілу функціоналу за модулями.

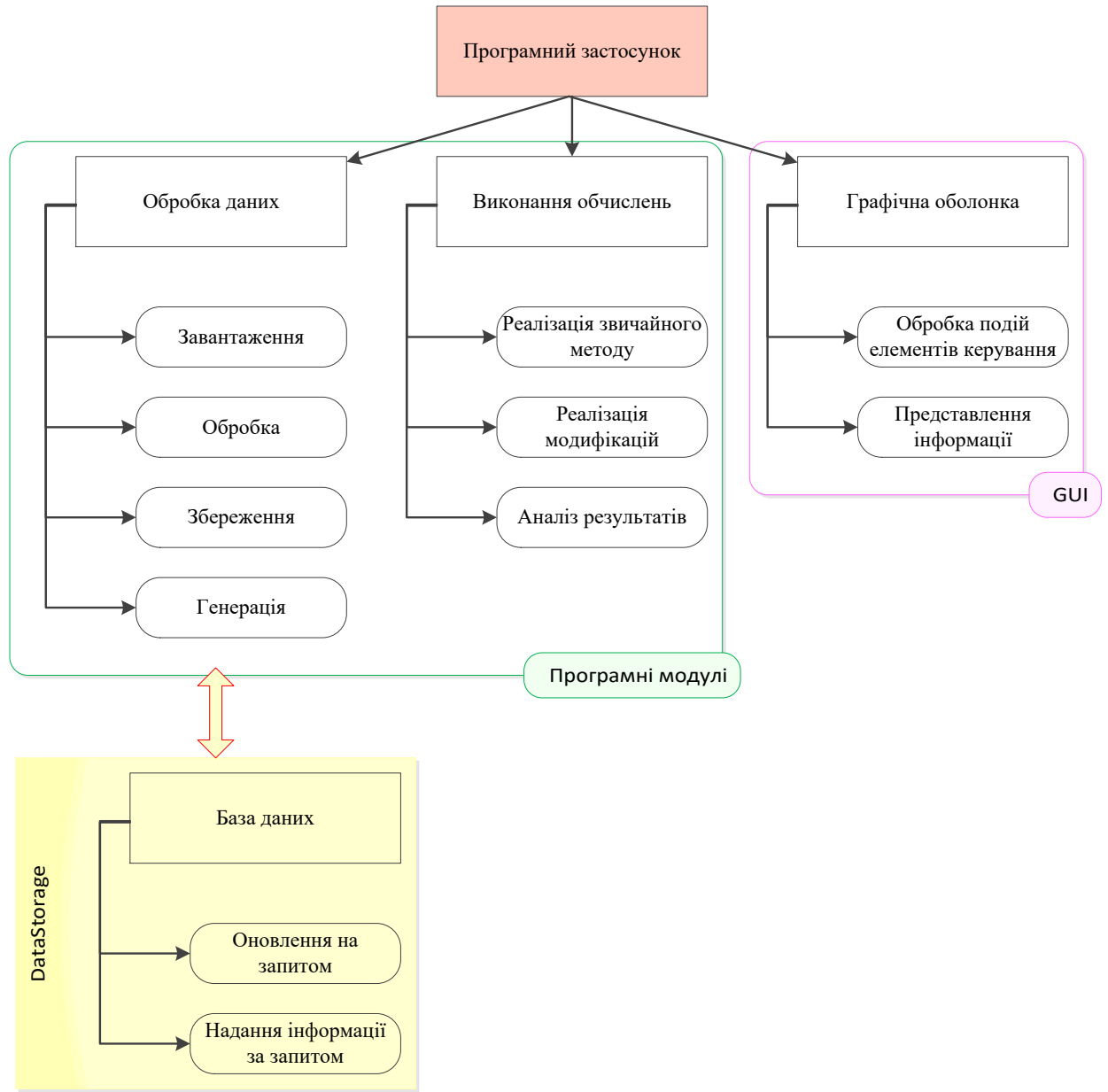


Рисунок 3.1 – Модульне представлення ПЗ

3.8 Висновки за розділом 3

Під час роботи над розділом було спроектовано архітектуру майбутнього ПЗ. Були спроектовані модулі майбутньої програми. Було проаналізовано найбільш популярні мови програмування та після ретельного порівняння перевага була віддана мові Java. Також проаналізувавши найбільш популярні середовища розробки, було обрано середовище NetBeans.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

У третьому розділі було досліджено та вирішено ряд питань з проєктування архітектури та реалізації ПЗ для дослідження запропонованого методу та можливих модифікацій. Особливу увагу було присвячено вибору технічних засобів: мови програмування та додаткових бібліотек.

Метою четвертого розділу є висвітлення рішень, щодо розробки ПЗ для аналізу та дослідження методу Брауна та його модифікацій.

У розділі описаний поетапний процес розробки програми в середовищі NetBeans. Описана технологія розробки з використанням Java. Наведено опис розроблених програмних пакетів та класів, досліджено їх методи та властивості.

4.1 Програмне забезпечення

Для повноцінної роботи із системою були розроблені пакети (Java package), які за функціоналом повністю повторюють раніше спроектовані модулі ПЗ, проте розширюють їх. Серед розроблених модулів можна виділити:

- WorkData;
- BrownRobinson;
- ModBrowRob;
- UserGUI;
- BDmod.

Як видно, пакет WorkData призначений для обробки вхідних даних і реалізує функції модулю завантаження даних. Пакет BrownRobinson призначений для синтезу виконання класичного методу Брауна-Робінсона, а ModBrowRob – для реалізації запропонованих модифікацій. Пакет BDmod призначений для зв'язку із базою даних і реалізує функції модулю зв'язку із

БД. Пакет UserGUI призначений для виведення графічного інтерфейсу користувача та обробки подій елементів інтерфейсу і реалізує функції модулю графічного інтерфейсу користувача.

Розглянемо розроблені модулі детальніше. Так основна взаємодія модулів представлена на рис. 4.1, де відображається взаємодія потоків модулів.

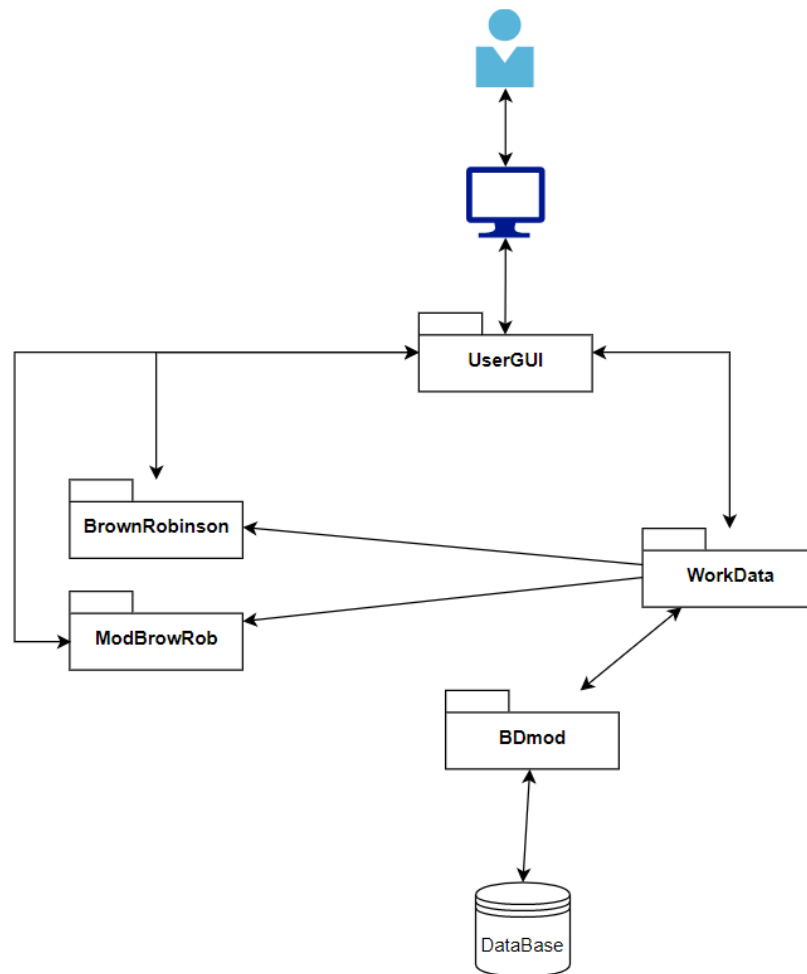


Рисунок 4.1 – Взаємодія розроблених модулів

4.1.1 Пакет призначений для роботи із даними

Пакет WorkData складається з одного класу: клас WorkData – оброблює вибірки даних: завантажує їх, зчитує та зберігає. До властивостей класу відносяться:

– double[] inputs – масив вхідних даних;

- double output – вихідні дані.

Серед методів класу можна виділити:

- getInput() – повертає вхідні дані із вибірки;
- getOutput() – повертає вихідні дані.

4.1.2 Пакет BrownRobinson для виконання класичного методу Брауна-Робінсона

Пакет BrownRobinson складається з чотирьох класів: BrownRobinsonController, BrownRobinsonElements та BrownRobinsonStart.

Клас BrownRobinsonStart – запускає роботу методу з вказаними параметрами. До властивостей класу відносяться:

- Scene – масив для збереження сцени гри;
- nodes – вихідні дані.

Серед методів класу можна виділити:

- set Scene() – ініціалізація сцени;
- set Nodes() – ініціалізація вхідних даних.

Клас BrownRobinsonController – регулює роботу операторів методу.

До властивостей класу відносяться:

- gameMod – матриця моделі гри.

Серед методів класу можна виділити:

- Analysis() – збирання та аналіз проміжної інформації під час виконання методу;

- playOptim() – підбір сценарію гри із оптимальними стратегіями.

Клас BrownRobinsonElements – спеціальний клас для збереження та контролю стану елементів методу – мета-параметрів методу. До властивостей класу відносяться:

- amountCountA – стратегії першого гравця;
- amountCountB – стратегії другого гравця;
- amountCountWinsA – перемоги першого гравця;

- amountCountWinsA – перемоги другого гравця;
- CountSumG – підбиття підсумків ігор;
- CountAverG – середні штрафи та перемоги ігор.

Серед методів класу можна виділити:

- GetamountCountA – підрахунок стратегії першого гравця;
- GetamountCountB – підрахунок стратегії другого гравця;
- GetamountCountWinsA – підрахунок перемог першого гравця;
- GetamountCountWinsA – підрахунок перемог другого гравця;
- GetCountSumG – експорт підсумків ігор;
- GetCountAverG – експорт середніх штрафів та перемог ігор.

Діаграму класів наведено на рис. 4.2.

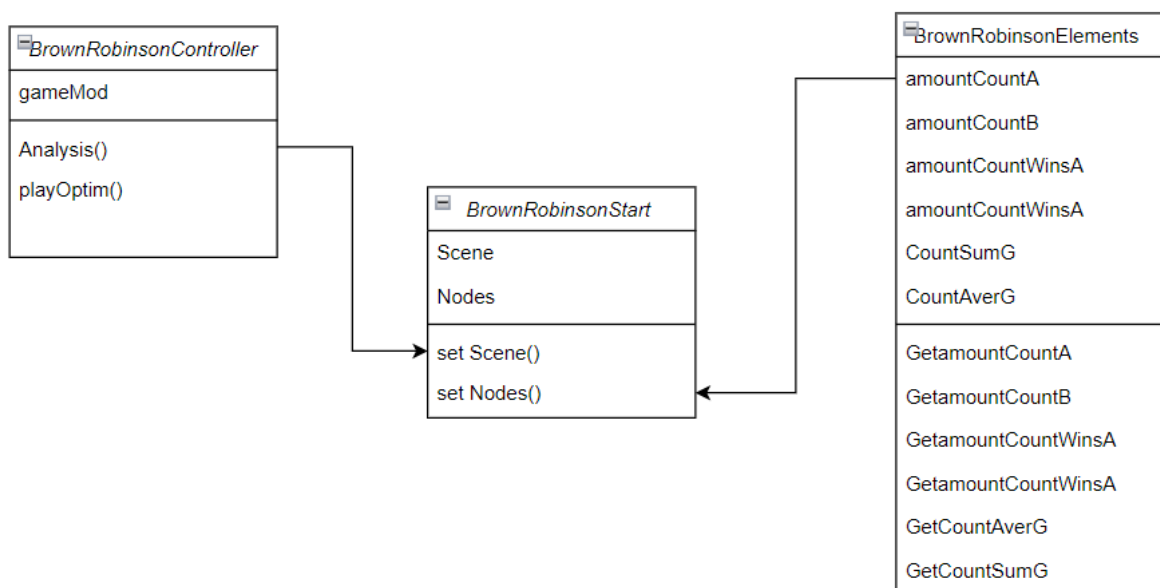


Рисунок 4.2 – Діаграма класів пакету BrownRobinson в нотації UML 2.0

4.1.3 Пакет ModBrowRob для реалізації запропонованих модифікацій методу

Пакет ModBrowRob складається з чотирьох класів: ModBrowRobController, ModBrowRobElements та ModBrowRobStart.

Клас `ModBrowRobStart` – запускає роботу методу з вказаними параметрами. До властивостей класу відносяться:

- `Scene` – масив для збереження сцени гри;
- `nodes` – вихідні дані.

Серед методів класу можна виділити:

- `set Scene()` – ініціалізація сцени;
- `set Nodes()` – ініціалізація вхідних даних.

Клас `ModBrowRobController` – регулює роботу операторів методу. До властивостей класу відносяться:

- `gameMod` – матриця моделі гри.

Серед методів класу можна виділити:

– `Analysis()` – збирання та аналіз проміжної інформації під час виконання методу;

- `playOptim()` – підбір сценарію гри із оптимальними стратегіями.

Клас `ModBrowRobElements` – спеціальний клас для збереження та контролю стану елементів методу – мета-параметрів методу. До властивостей класу відносяться:

- `critRes` – збереження стану критерію перезапуску гри;
- `critPriceLine` – збереження стану критерію верхньої границі ціни гри;
- `amountCountA` – стратегії першого гравця;
- `amountCountB` – стратегії другого гравця;
- `amountCountWinsA` – перемоги першого гравця;
- `amountCountWinsB` – перемоги другого гравця;
- `CountSumG` – підбиття підсумків ігор;
- `CountAverG` – середні штрафи та перемоги ігор.

Серед методів класу можна виділити:

- `calcCritRes` – обрахунок критерію перезапуску гри
- `calcCritPriceLine` – обрахунок критерію верхньої границі ціни гри;
- `GetamountCountA` – підрахунок стратегії першого гравця;
- `GetamountCountB` – підрахунок стратегії другого гравця;

- GetamountCountWinsA – підрахунок перемог першого гравця;
- GetamountCountWinsB – підрахунок перемог другого гравця;
- GetCountSumG – експорт підсумків ігор;
- GetCountAverG – експорт середніх штрафів та перемог ігор.

Діаграму класів наведено на рис. 4.3.

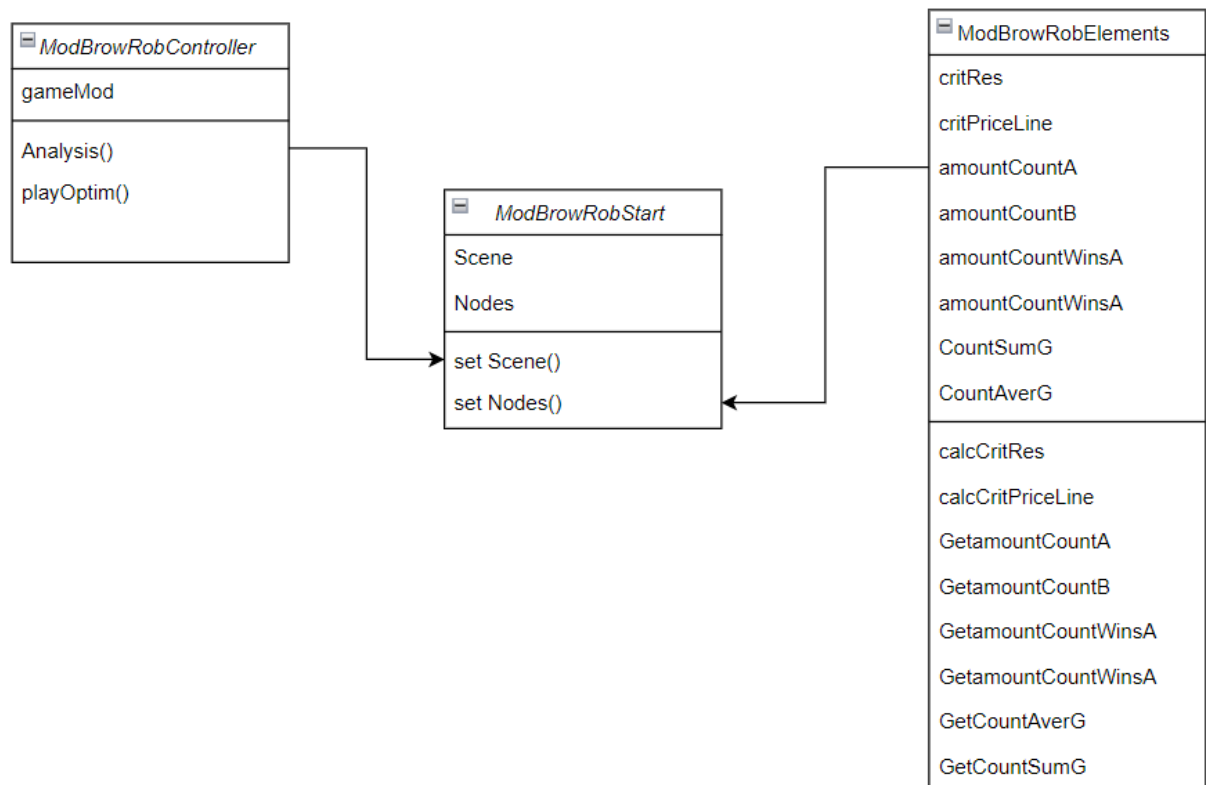


Рисунок 4.3 – Діаграма класів пакету ModBrowRob в нотації UML 2.0

4.1.4 Пакет призначений для зв'язку із базою даних – BDmod

Пакет BDmod складається з двох класів: BDConnect та DBOperation.

Клас BDConnect – призначений для підключення до бази даних. До властивостей класу відносяться:

- hostname – тип хосту;
- dbName – ім'я БД;
- userName – ім'я користувача (обліковий запис у системі керування базами даних);

– password – пароль, для доступу до бази даних.

Серед методів класу можна виділити:

– Connection getMySQLConnection() – створення підключення до бази даних.

Клас DBOperation – призначений для виконання основних транзакцій з БД. До властивостей класу відносяться:

– login – збереження логіну користувача;

– password – збереження паролю користувача;

– date – збереження дати та часу;

– setname – збереження ім'я вибірки даних.

Серед методів класу можна виділити:

– Authorization() – створення підключення до БД;

– UpdateSession() – запис результатів сесії до БД.

4.1.5 Пакет призначений для виведення графічного інтерфейсу користувача

Склад пакету UserGUI відповідає кількості робочих форм, що використовуються під час роботи із програмою. Основних форм для комфортної роботи користувача із програмою три: AuthorizationForm, MainForm, ResultForm.

4.2 Висновки за розділом 4

Розроблено ПЗ для реалізації та дослідження експериментальних результатів запропонованого методу Брауна та його модифікацій. Обрано механізм реалізації відповідних методів та алгоритмів, а також елементи керування, з якими користувачу буде зручно використовувати розроблений програмний продукт.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Метою п'ятого розділу є перевірка роботи розробленої програми реалізації запропонованого методу та його модифікацій..

У цьому розділі буде надано опис стратегії тестування та результати тестування ефективності розробленого ПЗ.

5.1 Опис застосунку

Грою називається математична модель конфліктної ситуації між двома (або більше) зацікавленими сторонами - гравцями. Гравці приймають рішення з деякого безлічі можливих рішень. Це безліч називають безліччю стратегій, а обрані гравцями рішення називають стратегіями. Дії гравців призводять до деякого результату гри, який залежить від обраних стратегій.

У разі кінцевого числа стратегій двох учасників конфлікту і протилежності їх інтересів, гра описується матрицею, що представлено на формулі (5.1) (матрична гра):

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \quad (5.1)$$

в якій рядки ($1 \leq i \leq m$) - стратегії першого гравця, стовпці ($1 \leq j \leq n$) – стратегії другого гравця. а елементи матриці a_{ij} – виграші першого гравця (програші другого) при відповідних стратегіях.

Рішенням матричної гри називається сідлова точка матриці, тобто пара стратегій (i^0, j^0) , що задовольняє нерівностей, що представлено на формулі (5.2).

$$(\forall i = \overline{1, m}, \forall j = \overline{1, n}) a_{ij^0} \leq a_{i^0 j^0} \leq a_{i^0 j}. \quad (5.2)$$

Якщо в матриці є сідлова точка, то величина $v = a_{i^0 j^0}$ називається ціною гри, і кажуть, що матрична гра має рішення в чистих стратегіях.

У загальному випадку сідлова точка матричної гри існує в змішаних стратегіях-розподілах ймовірностей на множинах вихідних (чистих) стратегій.

Пара стратегій (P^0, Q^0) , де $P^0 = (p_1^0, p_2^0, \dots, p_m^0)$, $Q^0 = (q_1^0, q_2^0, \dots, q_n^0)$, називається сідловою точкою змішаного розширення матричної гри, що представлено на формулі (5.3), якщо:

$$(\forall P, Q) E(P, Q^0) \leq E(P^0, Q^0) \leq E(P^0, Q),$$

$$E(P, Q) = \sum_{i=1}^m \sum_{j=1}^n a_{ij} p_i q_j. \quad (5.3)$$

Метод Брауна являє собою ітераційний метод наближеного рішення матричних ігор. Він полягає в багаторазовому фіктивному повторенні гри, при якому кожен з гравців в поточний момент часу вибирає чисту стратегію, що є найкращою відповіддю на змішану стратегію партнера, яка визначається частотами використання чистих стратегій в минулих повтореннях.

Розроблений ПЗ реалізує метод Брауна-Робінсона та певний набір модифікацій для пришвидшення роботи та збільшення якісних показників.

5.2 Тестовий набір даних

Антагоністичні ігри, в яких гравці рівноправно беруть участь в грі, називаються симетричними.

Під час тестування буде розглядатися динаміка зміни оцінок ціни гри в методі Брауна на прикладі 5×5 симетричної матричної гри, згенерованої випадковим чином, що представлено на формулі (5.4):

$$A = \begin{pmatrix} 0 & 10 & 0 & 9 & -7 \\ -10 & 0 & -5 & 7 & 3 \\ 0 & 5 & 0 & -2 & -1 \\ -9 & -7 & 2 & 0 & -3 \\ 7 & -3 & 1 & 3 & 0 \end{pmatrix}. \quad (5.4)$$

Додатково буде виконуватися тестування на заданому наборі даних – матричної гри, що представлено на формулі (5.5):

$$A = \begin{pmatrix} 7 & 8 & 4 & 6 & 10 & 3 & 1 & 7 & 0 & 6 \\ 3 & 10 & 4 & 3 & 8 & 9 & 2 & 12 & 7 & 5 \\ 12 & 6 & 5 & 10 & 5 & 7 & 3 & 9 & 4 & 7 \\ 8 & 7 & 11 & 9 & 8 & 7 & 5 & 7 & 6 & 5 \\ 7 & 8 & 8 & 8 & 5 & 2 & 6 & 13 & 6 & 9 \\ 9 & 8 & 10 & 9 & 7 & 4 & 3 & 7 & 10 & 9 \\ 3 & 2 & 12 & 6 & 10 & 11 & 10 & 11 & 11 & 11 \\ 11 & 7 & 8 & 0 & 13 & 3 & 11 & 9 & 9 & 6 \\ 13 & 6 & 3 & 7 & 8 & 8 & 9 & 6 & 8 & 13 \\ 7 & 11 & 10 & 5 & 9 & 10 & 6 & 10 & 6 & 11 \end{pmatrix}. \quad (5.5)$$

Провівши 2000 кроків стандартним методом Брауна, отримали ціну гри $v = 0,0375$ з точністю $\varepsilon = 0,154$: за 4000 кроків – ціна гри $v = 0,0115$ з точністю $\varepsilon = 0,108$. Експерименти показали, що метод Брауна повільно сходиться до ціни гри.

5.3 Результати тестування та їх аналіз

Обчислені похибки методів за (2.7) зведені в табл. 5.1.

Таблиця 5.1 – Похибки методів

Випадково згенерований набір даних			
Метод Брауна-Робінсона	Кількість кроків		
	1000	2000	4000
Стандарт.	0,317	0,165	0,117
$K=25$	0,151	0,112	0,0694
$K=32$	0,163	0,117	0,081
$K=33$	0,18	0,123	0,0912
Задана матриця			
Метод Брауна-Робінсона	Кількість кроків		
	1000	2000	4000
Стандарт.	0,1335	0,0844	0,0675
$K=25$	0,1237	0,0747	0,0607
$K=32$	0,1139	0,065	0,0539
$K=33$	0,1041	0,0553	0,0471

Експерименти, проведені для різних матричних ігор, показали, що модифікація методу Брауна за допомогою перемикачів дає істотне прискорення при знаходженні ціни гри в порівнянні зі стандартним методом.

Однак підібрати оптимальне значення параметра перемикачів методу в загальному випадку не вдається. З табл. 5.1 видно, що навіть незначна зміна параметра K може призводити до збільшення похибки. На рис. 5.1 і в табл. 5.2 наведені результати порівняння модифікацій для прикладу 5×5 гри.

Таблиця 5.2 – Похибки методів

Метод	Кількість кроків					
	100	1000	2000	4000	10000	100000
Стандартний	0,466	0,298	0,277	0,253	0,229	0,207
Модифікація із параметром $K=25$	0,388	0,276	0,262	0,243	0,221	0,205
Модифікація за нижньою ціною гри	0,310	0,253	0,248	0,232	0,213	0,203
Модифікація за ціною гри	0,232	0,230	0,233	0,222	0,204	0,201

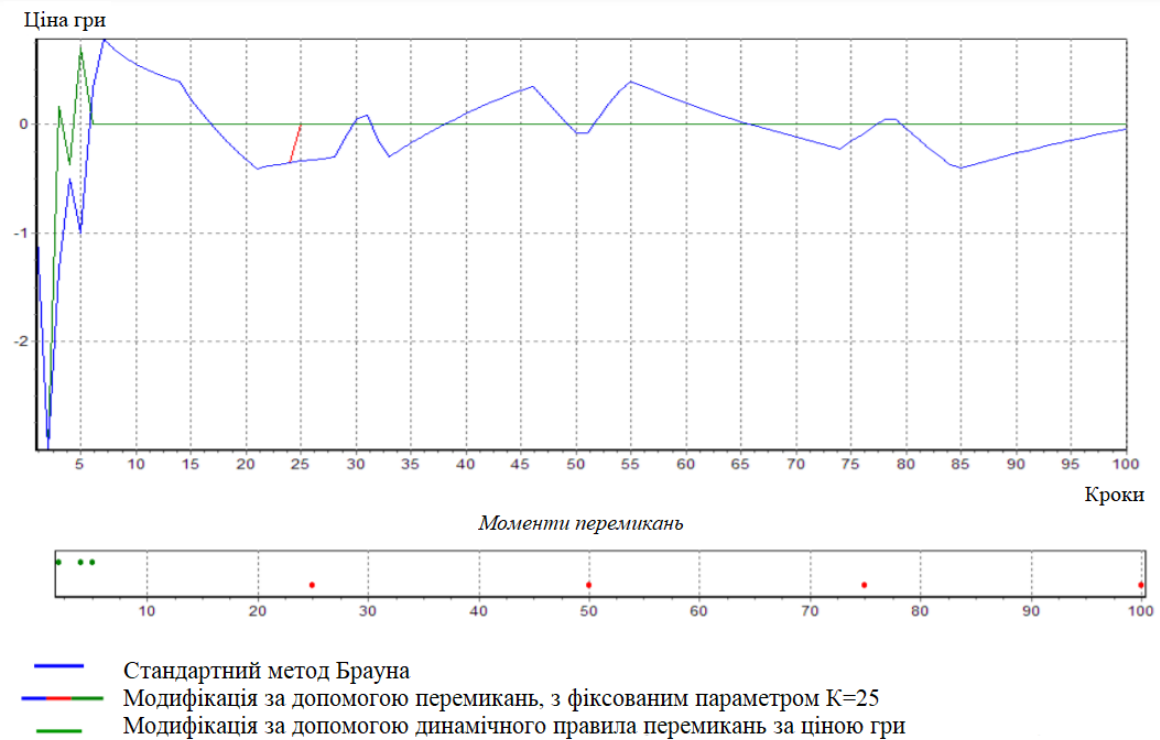


Рисунок 5.1 – Графік зміни оцінок ціни гри (100 кроків)

З рис. 5.1 видно, що модифікація, що використовує динамічне правило перемикання, призводить до істотно меншого числа перемикань. Такого ж типу динаміка спостерігається і в інших експериментах незалежно від розміру матриці. Модифікації з динамічним правилом перемикання забезпечують швидкість збіжності до ціни гри, що не поступається швидкості збіжності методу.

Обчислювальний експеримент показав що всі запропоновані динамічні модифікації дають приблизно однакову похибку. Однак модифікація за допомогою вирішального правила за ціною гри сходиться швидше і використовує мінімальну кількість перемикань. При цьому не потрібно завдання значення параметра K , що робить дану модифікацію найбільш кращою.

5.4 Тест кейси для розробленого ПЗ

У табл. 5.3 наведено чек-листу із кейсами тестування саме для розробленого ПЗ.

Таблиця 5.3 – Чек-лист із кейсами для тестування ПЗ

Опис кейсу	Очікувані результати	Тестування на платформі AMD Athlon X4 950 (3.5–3.8 ГГц), RAM 8ГБ, Ubuntu 21.10	Тестування на платформі Intel Core i5-10400F (2.9–4.3 ГГц) / RAM 16 ГБ, Windows 11
Запуск ПЗ, авторизація	Запуск застосунку із виконуваного файлу (або виклик через командний рядок) та авторизація за існуючим профілем користувача	Пройдено (через консоль)	Пройдено
Завантаження задачі	Завантаження зовнішньої вибірки даних із характеристикою задач	Пройдено	Пройдено
Генерація задачі	Генерація випадкової вибірки даних для задачі	Пройдено	Пройдено
Виконання методу або вибір модифікацій	Виконання обчислення стратегій гри та можливість вибору виконуваних модифікацій	Пройдено	Пройдено

5.5 Висновки за результатами тестування

Проведено експериментальну оцінку ефективності методів. Метод визначення полярності показав результати на рівні сучасних робіт. Розроблений ПЗ в повній мірі реалізував запропонований метод.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра, було розглянуто основні можливості використання методів ТІ для автоматизації процесу прийняття рішень. Для цього було вирішено задачі:

- дослідження області теорії прийняття рішень;
- дослідження поля використання ТІ;
- дослідження методів та критеріїв ТІ;
- аналіз найбільш популярних рішень;
- розбір існуючого методу та формування набору модифікацій для посилення якісних характеристик;
- проектування ПЗ для реалізації та дослідження обраного методу та запропонованих модифікацій для нього;
- реалізація спроектованого ПЗ;
- тестування та дослідження результатів тестування ПЗ та методу і модифікацій.

Таким чином, в процесі підготовки дипломної кваліфікаційної роботи магістра була досягнута основна мета роботи і вирішені поставлені завдання

Під час розробки програми та аналізу розробленого програмного комплексу з'явилися ідеї з удосконалення, такі як: використання штучних нейронних мереж для збільшення рівня самонавчання систем із прийняття рішень.

Нові можливості зможуть розширити коло вирішення практичних задач, адже останні роботи засвідчили високу точність роботи таких систем автоматизованого прийняття рішень та контролю стану систем на основі мереж із самонавчанням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Творошенко, І. С. Технології прийняття рішень в інформаційних системах [Текст] : [навч. посіб.] / І. С. Творошенко ; Харків. нац. ун-т радіоелектроніки. - Харків : ХНУРЕ, 2021. - 118 с.
2. Григорків, В. С. Моделі прийняття рішень в економіці [Текст] : навч. посіб. / В. С. Григорків, М. В. Григорків ; Чернівець. нац. ун-т ім. Юрія Федьковича. - Чернівці : ЧНУ : Рута, 2021. - 255 с.
3. Обґрунтування концепції вдосконалення об'єктів залізничної техніки на підставі теорії прийняття рішень [Текст] : монографія / Горбунов М. І. [та ін.] ; Східноукр. нац. ун-т ім. Володимира Даля. - Одеса : Купрієнко С. В., 2020. - 97 с.
4. Теоретичні основи забезпечення якості прийняття управлінських рішень в умовах європейської інтеграції [Текст] : монографія / [М. М. Новікова та ін.] ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. - Харків : Друкарня Мадрид, 2020. - 335 с.
5. Останкова Л. А. Теорія ігор [Текст] : довідковий метод. посіб. для студ. екон. спец. / Л. А. Останкова, Н. Ю. Шевченко ; Донбаська держ. машинобудівна академія. - Краматорськ : ДДМА, 2007. - 103 с. -
6. Дослідження операцій та математичні методи [Текст] : навч. посіб. для студ. спец. 7.080401 "Інформаційні управляючі системи" / упоряд. Ю. І. Ларіонов [та ін.] ; Ін - т змісту і методів навчання, Харків. держ. техн. ун-т радіоелектроніки. - Х. : ХДТУР, 1999 . Ч. 3 : Теорія ігор. - 2000. - 76 с.
7. Шиян А.А. Управління розвитком соціально-економічних систем. Теорія ігор: основи та застосування в економіці та менеджменті [Текст] : навч. посіб. / А. А. Шиян ; Вінниць. нац. техн. ун-т. - Вінниця : ВНТУ, 2010. - 161 с.
8. Плахотнік О. В. Теорія ігор у кримінальному процесі України [Електронний ресурс] / О. В. Плахотнік // Криміналістика і судова експертиза. - 2019. - Вип. 64. - С. 294-305.

9. Данова К. В. Теорія ігор у процесі прийняття управлінських рішень на робочих місцях працівників із інвалідністю [Електронний ресурс] / К. В. Данова, В. В. Малишева, Н. М. Попович // Вісті Донецького гірничого інституту. - 2020. - № 1. - С. 165-171.

10. Козачок Л. М. Аналіз та прогнозування часових рядів динаміки значень досліджуваної величини за допомогою експоненціального згладжування методом Брауна [Електронний ресурс] / Л. М. Козачок, А. Є. Козачок // Автомобіль і електроніка. Сучасні технології. - 2017. - Вип. 12. - С. 146-150.

11. Запєвалов А. С. Искажение оценок асимметрии возвышений морской поверхности, полученных по данным альтиметрических измерений / А. С. Запєвалов, В. В. Пустовойтенко // Екол. безпека прибереж. та шельф. зон та комплекс. використ. ресурсів шельфу : зб. наук. пр. - 2012. - Вип. 26, т. 1. - С. 367-374.

12. Корниль Т. Л. Математическое программирование : Учеб. пособие для студ. экон. спец. / Т. Л. Корниль, Н. А. Кириллова, Л. С. Тимченко; Нац. техн. ун-т "Харьк. политехн. ин-т". - Х., 2002. - 127 с.

13. Бурий В. В. Математичне програмування : навч. посіб. Модуль. 2. Спеціальні методи математичного програмування / В. В. Бурий, О. С. Давидов, І. В. Шевченко; Нац. авіац. ун-т. - К., 2008. - 124 с.

14. Баркалов А. В. Модифицированный итеративный метод для решения симметричных матричных игр [Текст] / А. В. Баркалов, Е. С. Гвоздева // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы XI Всероссийской конференции. – Н. Новгород: ННГУ. – 2012. – С. 22-26.

15. Заковоротный А. Ю. Модель накопления уязвимостей в исходном коде Java-приложений, а также бинарная система оценки качества кода на ее основе [Текст] / А. Ю. Заковоротный, М. А. Зорян // Вісник Національного технічного університету "ХПІ". Серія : Інформатика та моделювання. – 2018. – № 24. – С. 37-46.

16. Яшина О. М. Дослідження мікросервісної архітектури, архітектурний стиль rest та їх сучасна реалізація на java [Текст] / О. М. Яшина, О. А. Кравчук // Вісник Хмельницького національного університету. Технічні науки. – 2020. – № 5. – С. 106-114.

17. Java lib Gametheory [Electronic resource]. – Access mode: <https://github.com/vonhachtaugust/GameTheory>.

18. Java lib Algorithms [Electronic resource]. – Access mode: <https://github.com/TheAlgorithms/Java>.

19. Apache NetBeans [Electronic resource]. – Access mode: <https://netbeans.apache.org/>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.2 Призначення та область застосування

Метод Брауна являє собою ітераційний метод наближеного рішення матричних ігор. Він полягає в багаторазовому фіктивному повторенні гри, при якому кожен з гравців в поточний момент часу вибирає чисту стратегію, що є найкращою відповіддю на змішану стратегію партнера, яка визначається частотами використання чистих стратегій в минулих повтореннях.

Розроблений ПЗ реалізує метод Брауна-Робінсона та певний набір модифікацій для пришвидшення роботи та збільшення якісних показників.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, яке було визначене за темою «Дослідження та програмна реалізація методів прийняття рішень на основі стратегій теорії ігор» і затверджене наказом № 435 від 05 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.3 Призначення розробки

Розроблений ПЗ реалізує метод Брауна-Робінсона та певний набір модифікацій для пришвидшення роботи та збільшення якісних показників.

А.4 Вимоги до програми

А.4.1 Вимоги до функціональних характеристик

До програми пред'являються такі функціональні вимоги:

- завантаження набору даних;
- генерація набору вхідних даних (у випадку простого тестування роботи);
- реалізація звичайного методу Брауна-Робінсона для прийняття рішень та визначення позитивних стратегій;
- реалізація запропонованих модифікацій методу Брауна-Робінсона для дослідження нових можливостей методу та його чутливості до вносимих модифікацій.

Виконання наведених функцій в повній мірі розкриє практичний потенціал запропонованих методів та розроблюваного ПЗ.

А.4.2 Вимоги до надійності

Для сталого функціонування програмного забезпечення необхідно:

- наявність безперебійних блоків живлення персонального комп'ютера.

Програма повинна видавати відмову доступу до комп'ютера при спробі несанкціонованого доступу.

А.4.3 Вимоги до технічних та програмних засобів

Технічна характеристика:

- ПК з ОС Microsoft Windows 8.1 та старше;
- Java Virtual Machine (JVM) 9.0 та вище;

- процесор сімейства Intel (лінійка Core) або AMD (лінійка Ryzen) з тактовою частотою 2.0 МГц та вище;
- вільне місце на накопичувачі від 1 GB;
- оперативна пам'ять від 4 GB.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- опис проєктування;
- опис тестування;
- опис програми.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Розробка повинна бути проведена в три стадії:

- розробка технічного завдання;
- робоче проєктування;
- впровадження.

A.6.2 Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, погодження та затвердження справжнього технічного завдання.

На стадії робочого проєктування повинні бути виконані перераховані нижче етапи робіт:

- розробка програми;
- розробка програмної документації;

- випробування програми.

На стадії впровадження повинен бути виконаний етап розробки підготовка і передача програми.

A.6.3 Зміст робіт по етапах

На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи:

- постановка задачі;
- визначення і уточнення вимог до технічних засобів;
- визначення вимог до програми;
- визначення стадій, етапів і термінів розробки програми та документації на неї;
- узгодження і затвердження технічного завдання.

На етапі розробки програми повинна бути виконана робота з програмування (кодування) і налагодженні програми.

На етапі розробки програмної документації повинна бути виконана розробка програмних документів відповідно до вимог до складу документації.

На етапі випробувань програми повинні бути виконані перераховані нижче види робіт:

- розробка, узгодження та затвердження та методики випробувань;
- проведення приймально-здавальних випробувань;
- коректування програми і програмної документації за результатами випробувань.

ДОДАТОК Б
Опис програми

Б.1 Загальні свідчення

Метод Брауна являє собою ітераційний метод наближеного рішення матричних ігор. Він полягає в багаторазовому фіктивному повторенні гри, при якому кожен з гравців в поточний момент часу вибирає чисту стратегію, що є найкращою відповіддю на змішану стратегію партнера, яка визначається частотами використання чистих стратегій в минулих повтореннях.

Розроблений ПЗ реалізує метод Брауна-Робінсона та певний набір модифікацій для пришвидшення роботи та збільшення якісних показників.

Б.2 Функціональне призначення

Таким чином серед функціональних вимог виділяються:

- завантаження набору даних;
- генерація набору вхідних даних (у випадку простого тестування роботи);
- реалізація звичайного методу Брауна-Робінсона для прийняття рішень та визначення позитивних стратегій;
- реалізація запропонованих модифікацій методу Брауна-Робінсона для дослідження нових можливостей методу та його чутливості до вносимих модифікацій.

Виконання наведених функцій в повній мірі розкриє практичний потенціал запропонованих методів та розроблюваного ПЗ.

Б.3 Опис логічної структури

Для зручності реалізації, підтримки та подальшого оновлення ПЗ пропонується реалізувати його в модульному форматі.

Так при проектуванні було надано перевагу розробленню 2 основних модулів:

- програмний модуль;
- модуль графічного інтерфейсу користувача.

Так програмний модуль повинен відповідати за 2 основних пакети: підтримка роботи із даними та підтримка обчислень.

Модуль графічного інтерфейсу відповідає за взаємодію ПЗ із користувачем, адже планується виведення особливих графічних результатів роботи.

Окремо планується реалізувати базу даних для збереження проміжкових даних про роботу застосунку.

Така організація архітектури застосунку спростить розробку, забезпечить розподіл функціоналу між окремими пакетами та структурними одиницями (класами), що гарантує легкість підтримки та оновлення функціоналу застосунку, більш того буде доречним при використанні сторонніх бібліотек та плагінів для розширення функціоналу.

Подальша реалізація буде відбуватися з урахуванням розробленої структури та розподілу функціоналу за модулями.

Б.4 Використані технічні засоби

Технічна характеристика:

- ПК з ОС Microsoft Windows 8.1 та старше;
- Java Virtual Machine (JVM) 9.0 та вище;
- процесор сімейства Intel (лінійка Core) або AMD (лінійка Ryzen) з тактовою частотою 2.0 МГц та вище;
- вільне місце на накопичувачі від 1 GB;
- оперативна пам'ять від 4 GB.

На рисунку Б.1 зображено модульне представлення ПЗ.

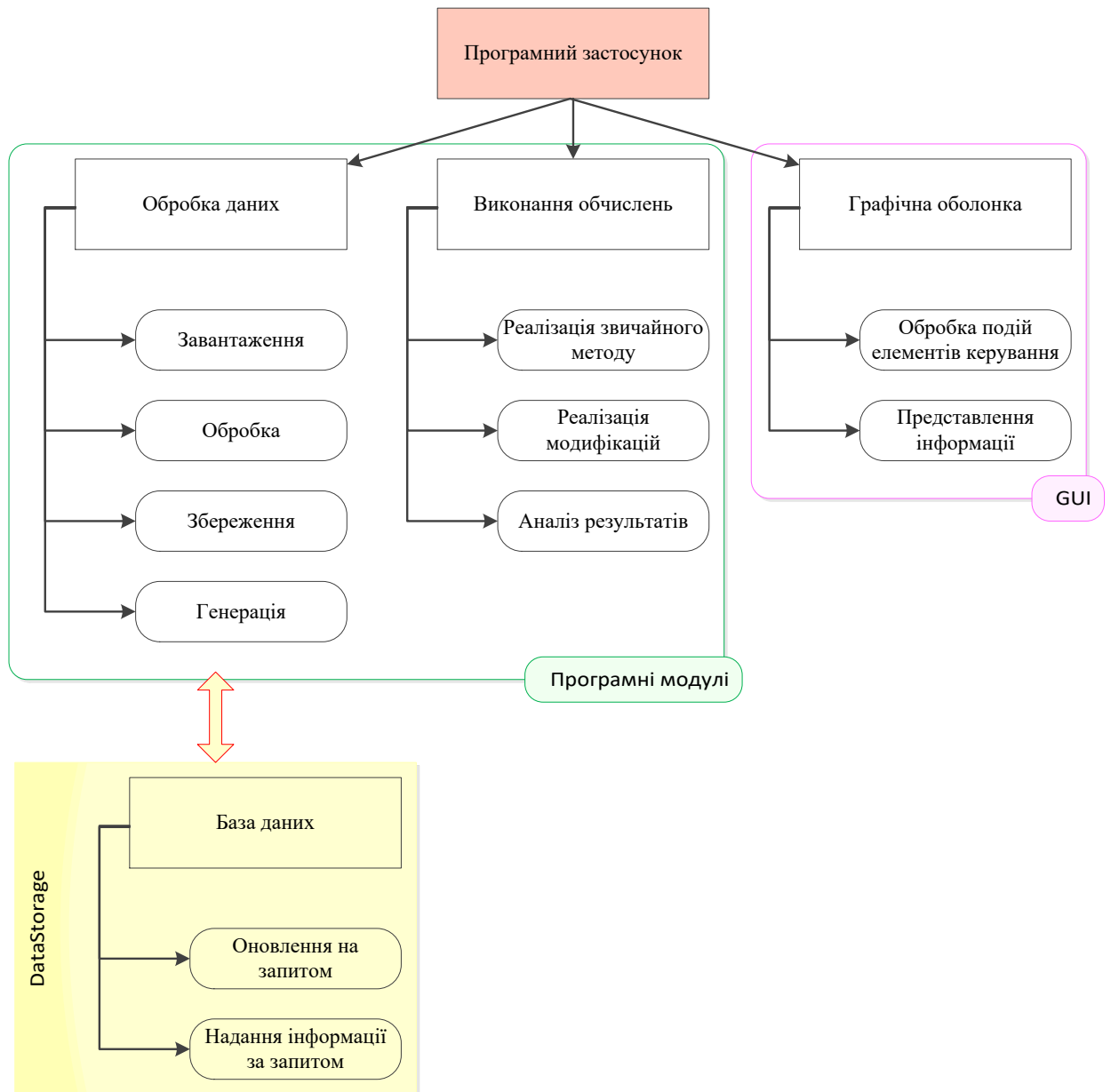


Рисунок Б.1 – Модульне представлення ПЗ

ДОДАТОК В
Текст програми

Сирцевий код розробленого застосунку:

```
public class gameMod {
    private      ObservableList<enterLogG>      LogG      =
FXCollections.observableArrayList();
    private double[][] GScene;
    private int NumCountG;

    private Double[] amountCountA ;
    private Double[] amountCountB ;

    private double[] statAnalitSumG;
    private double[] statAverageValueOfGame;

    private double[] amountCountWinsA ;
    private double[] amountCountWinsB ;
    private double anGameValue;

    private int winsAmountA;
    private int amountCountWinsB;

    public int getWinsAmountA() {
        return winsAmountA;
    }

    public int getamountCountWinsB() {
        return amountCountWinsB;
    }

    gameMod (){
```

```

GScene = new double[2][2];
//LogG = new ArrayList<>();

amountCountA = new Double[2];
amountCountB = new Double[2];

for(int i = 0; i < 2; ++i) {
    amountCountA [i] = 0.0;
    amountCountB [i] = 0.0;
}

winsAmountA = 0;
amountCountWinsB = 0;

}

public void playOptim(){
    for(int iter = 0; iter < NumCountG; ++iter){
        enterLogG ent = new enterLogG();
        ent.setGameNumber(iter + 1);
        int statGRA;

        if(iter < 1){
            Random rnd = new Random();
            statGRA = rnd.nextBoolean() ? 2 : 1;
        }else{
            statGRA = LogG.get(iter-1).getMaxIndex() + 1;
        }
        ent.setChosenstatGRA(statGRA);

        Double[] scoresByB = new Double[2];

```

```

        for(int i = 0; i < 2; ++i){
            if(iter == 0){
                scoresByB[i] = GScene[statGRA-1][i];
            }
            else{
                scoresByB[i] = LogG.get(iter-1).getScoresByB()[i] +
GScene[statGRA-1][i];
            }
        }
        ent.setScoresByB(scoresByB);

        int minIndex = IntStream.range(0,
Arrays.asList(scoresByB).size()).boxed().min(Comparator.comparingDouble(Arra
ys.asList(scoresByB)::get)).get();
        Double lowerScore = scoresByB[minIndex] /
ent.getGameNumber();

        int statGRB = minIndex + 1;
        ent.setChosenstatGRB(statGRB);
        ent.setLowerPrice(lowerScore);

        Double[] scoresByA = new Double[2];
        for(int i = 0; i < 2; ++i){
            if(iter == 0){
                scoresByA[i] = GScene[i][statGRB-1];
            }
            else{
                scoresByA[i] = LogG.get(iter-1).getScoresByA()[i] +
GScene[i][statGRB-1];

```

```

        }
    }
    ent.setScoresByA(scoresByA);

    int          maxIndex          =          IntStream.range(0,
Arrays.asList(scoresByA).size()).boxed().max(Comparator.comparingDouble(Arra
ys.asList(scoresByA)::get)).get();
    Double       higherScore       =          scoresByA[maxIndex]          /
ent.getGameNumber();
    ent.setHigherPrice(higherScore);

    Double       averageScore      =          (ent.getHigherPrice()      +
ent.getLowerPrice()) / 2;
    ent.setAveragePrice(averageScore);

    ent.setBz(scoresByB[0]);
    ent.setBo(scoresByB[1]);
    ent.setAz(scoresByA[0]);
    ent.setAo(scoresByA[1]);

    ent.setMaxIndex(maxIndex);

    LogG.add(ent);

    }
}

public void setGScene(double[][] GScene) {
    this.GScene = GScene;
}

```

```

public void setLogG(ObservableList<enterLogG> LogG) {
    this.LogG = LogG;
}

```

```

public void setNumCountG(int NumCountG) {
    this.NumCountG = NumCountG;
}

```

```

public int getNumCountG(){
    return NumCountG;
}

```

```

public ObservableList<enterLogG> getLogG() {
    return LogG;
}

```

```

public double[][] getGScene() {
    return GScene;
}

```

```

public Double[] getamountCountA () {
    return amountCountA ;
}

```

```

public Double[] getamountCountB () {
    return amountCountB ;
}

```

```

public void findProbs(){

```

```

for(int i = 0; i < NumCountG; ++i){
    int stratNum = LogG.get(i).getChosenstatGRA();
    amountCountA [stratNum-1]++;
    stratNum = LogG.get(i).getChosenstatGRB();
    amountCountB [stratNum-1]++;
}

for(int i = 0; i < 2; ++i){
    amountCountA [i] /= NumCountG;
    amountCountB [i] /= NumCountG;
}

}

public void printOptStrats(){
    System.out.printf("Optimal strategy for A: (");
    for(int i = 0; i < amountCountA .length; ++i){
        System.out.printf("%f", amountCountA [i]);
        if(i != amountCountA .length - 1){
            System.out.printf("; ");
        }
    }
    System.out.printf(")\n");

    System.out.printf("Optimal strategy for B: (");
    for(int i = 0; i < amountCountB .length; ++i){
        System.out.printf("%f", amountCountB [i]);
        if(i != amountCountB .length - 1){
            System.out.printf("; ");
        }
    }

```

```

    }
}

```

```

System.out.printf("\n");

```

```

}

```

```

public void debugDummy(){

```

```

}

```

```

public void initStatLogArrays(){

```

```

    statAnalitSumG = new double[NumCountG];

```

```

    statAverageValueOfGame = new double[NumCountG];

```

```

}

```

```

private int pickStratIndexRandomly (Double[] weights) {

```

```

    // calculate the total weight

```

```

    double cumulativeSum = 0;

```

```

    for(int i = 0; i < weights.length; i++){

```

```

        cumulativeSum += weights[i];

```

```

    }

```

```

        double randChoice

```

```

        =

```

```

ThreadLocalRandom.current().nextDouble((double) 0, cumulativeSum);

```

```

//System.out.println(randChoice);

```

```

    for(int i = 0; i < weights.length; i++){

```

```

        randChoice -= weights[i];

```

```

        if(randChoice < 0){

```

```

            return i;

```

```

    }
}
return weights.length - 1;
}

```

```

public double[] getStatAverageValueOfGame() {
    return statAverageValueOfGame;
}

```

```

public double[] getstatAnalitSumG() {
    return statAnalitSumG;
}

```

```

public void calcAnalytically(){

    amountCountWinsA = new double[2];
    amountCountWinsA = new double[2];
    amountCountWinsA [0] = (GScene[1][1] - GScene[1][0]) /
(GScene[0][0] + GScene[1][1] - GScene[1][0] - GScene[0][1]) ;
    amountCountWinsA [1] = 1 - amountCountWinsA [0];
    amountCountWinsA [0] = (GScene[1][1] - GScene[0][1]) /
(GScene[0][0] + GScene[1][1] - GScene[1][0] - GScene[0][1]) ;
    amountCountWinsA [1] = 1 - amountCountWinsA [0];
    anGameValue = (GScene[0][0] * GScene[1][1] - GScene[0][1] *
GScene[1][0]) / (GScene[0][0] + GScene[1][1] - GScene[1][0] - GScene[0][1]);
}

public double getAnGameValue() {
    return anGameValue;
}

```

```

public double[] getamountCountWinsA () {
    return amountCountWinsA ;
}

```

```

public double[] getamountCountWinsA () {
    return amountCountWinsA ;
}

```

```

public void playWithOptStrats(){
    int statGRA = pickStratIndexRandomly(amountCountA );
    int statGRB = pickStratIndexRandomly(amountCountB );

    statAnalitSumG[0] = GScene[statGRA][statGRB];
    statAverageValueOfGame[0] = statAnalitSumG[0];
    if(GScene[statGRA][statGRB] > 0){
        winsAmountA++;
    }else if(GScene[statGRA][statGRB] < 0){
        amountCountWinsB++;
    }
}

```

```

for(int i = 1; i < NumCountG; i++){
    statGRA = pickStratIndexRandomly(amountCountA );
    statGRB = pickStratIndexRandomly(amountCountB );

```

```

    statAnalitSumG[i] = statAnalitSumG[i - 1] +
    GScene[statGRA][statGRB];
    statAverageValueOfGame[i] = statAnalitSumG[i] / ( i + 1 );
    if(GScene[statGRA][statGRB] > 0){
        winsAmountA++;

```

```
    }else if(GScene[statGRA][statGRB] < 0){  
        amountCountWinsB++;  
    }  
  
}  
  
    debugDummy();  
}  
  
}
```

ДОДАТОК Д
Слайди презентації

Дослідження та програмна реалізація методів прийняття рішень на основі стратегій теорії ігор

1

Студент: Брагін Ярослав Андрійович
КНТ-110М

Керівник: Коцур Михайло Ігорович
к.т.н., доцент

Рисунок Д.1 – Слайд №1

2

Об'єкт, предмет та мета роботи

- **Об'єкт дослідження** – процес прийняття рішень на основі стратегій теорії ігор.
- **Предмет дослідження** – методи та стратегії теорії ігор для прийняття рішень.
- **Мета роботи** – дослідити, проаналізувати та реалізувати методи прийняття рішень на основі стратегій теорії ігор.

Рисунок Д.2 – Слайд №2

Постановка завдання

Дослідити та реалізувати один з методів теорії ігор, а саме метод Брауна-Робінсона для автоматизації прийняття рішень. Додатково розглянути можливі модифікації методу для вирішення проблем методу, а саме пришвидшення швидкості збіжності.

Рисунок Д.3 – Слайд №3

Класифікація ігрових завдань



Рисунок Д.4 – Слайд №4

Порівняння методів теорії ігор

	Метод Брауна-Робінсона	Метод зведення гри до задачі лінійного програмування	Матрична гра
Залежність від розмірності гри (задачі)	Низька	Висока	Висока
Швидкість збіжності	Повільна	Середня	Середня
Поведінка в умовах більшої невизначеності	Не впливає на роботу	Середнє впливає на роботу	Середнє впливає на роботу
Перегляд позитивних стратегій гравців	Основний результат роботи	Загальне представлення стратегій	Загальне представлення стратегій

Рисунок Д.5 – Слайд №5

Запропоновані модифікації

$$\begin{aligned} |v_1(k)| < |v_2(k)| & \quad Q(k) = P(k) \\ |v_1(k)| > |v_2(k)| & \quad P(k) = Q(k) \end{aligned}$$

$$|v_1(k+1) + v_2(k+1)| < |v_1(k) + v_2(k)|$$

Рисунок Д.6 – Слайд №6

Порівняння мов програмування

Критерії порівняння	Мови програмування		
	Java	Matlab	Python
Наявність матеріалів для вивчення та підтримки розробників	++	++	++
Парадигми програмування (об'єктно-орієнтоване, функціональне, тощо)	+	–	++
Бібліотеки шаблонів структур даних	++	++	++
Кроссплатформність	++	–	++
Швидкість роботи та загальна продуктивність застосунків	++	–	+
Бібліотеки для розширення функціоналу	++	+	++
Розробка графічного інтерфейсу користувача	++	+	+
Паралельне виконання високопродуктивних обчислень	++	±	++

Рисунок Д.7 – Слайд №7

Порівняння середовищ розробки

Критерії	NetBeans	IntelliJ IDEA
Кроссплатформність	±	±
Оновлення та підтримка	+	++
Системи контролю версій	++	++
Серверна підтримка	++	++
Інструментарій для розробки графічного інтерфейсу	++	++
Ліцензія (безкоштовність)	++	±

Рисунок Д.8 – Слайд №8

Діаграми класів розроблених пакетів

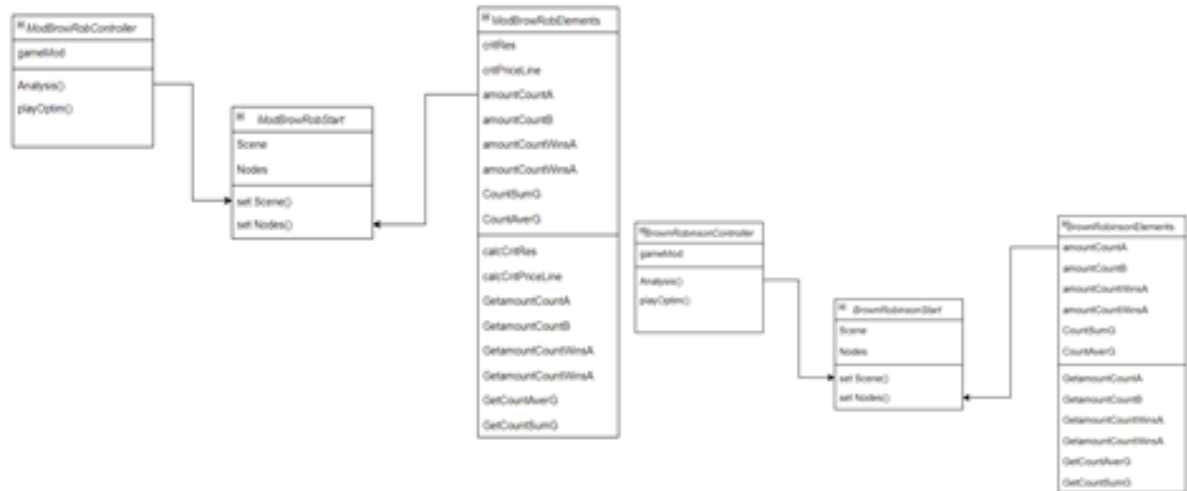


Рисунок Д.11 – Слайд №11

Результати тестування та їх аналіз

Випадково згенерований набір даних			
Метод Брауна-Робінсона	Кількість кроків		
	1000	2000	4000
Стандарт.	0,317	0,165	0,117
K=25	0,151	0,112	0,0694
K=32	0,163	0,117	0,081
K=33	0,18	0,123	0,0912
Задана матриця			
Метод Брауна-Робінсона	Кількість кроків		
	1000	2000	4000
Стандарт.	0,1335	0,0844	0,0675
K=25	0,1237	0,0747	0,0607
K=32	0,1139	0,065	0,0539
K=33	0,1041	0,0553	0,0471

Рисунок Д.12 – Слайд №12

Результати тестування та їх аналіз

Метод	Кількість кроків					
	100	1000	2000	4000	10000	100000
Стандартний	0,466	0,298	0,277	0,253	0,229	0,207
Модифікація із параметром K=25	0,388	0,276	0,262	0,243	0,221	0,205
Модифікація за нижньою ціною гри	0,310	0,253	0,248	0,232	0,213	0,203
Модифікація за ціною гри	0,232	0,230	0,233	0,222	0,204	0,201

Рисунок Д.13 – Слайд №13

Чек-лист тестування застосунку

Опис кейсу	Очікувані результати	Тестування на платформі AMD Athlon X4 950 (3.5–3.8 GHz), RAM 8ГБ, Ubuntu 21.10	Тестування на платформі Intel Core i5-10400F (2.9–4.3 GHz) / RAM 16 ГБ, Windows 11
Запуск ПЗ, авторизація	Запуск застосунку із виконаного файлу (або виклик через командний рядок) та авторизація за існуючим профілем користувача	Пройдено (через консоль)	Пройдено
Завантаження задачі	Завантаження зовнішньої півірки даних із характеристикою задачі	Пройдено	Пройдено
Генерація задачі	Генерація випадкової півірки даних для задачі	Пройдено	Пройдено
Виконання методу або вибір модифікацій	Виконання обчислення стратегій гри та можливість виконуваних модифікацій	Пройдено	Пройдено

Рисунок Д.14 – Слайд №14

Висновки

Під час виконання дипломної кваліфікаційної роботи магістра, було розглянуто основні можливості використання методів ТІ для автоматизації процесу прийняття рішень. Для цього було вирішено задачі:

- дослідження області теорії прийняття рішень;
- дослідження поля використання ТІ;
- дослідження методів та критеріїв ТІ;
- аналіз найбільш популярних рішень;
- розбір існуючого методу та формування набору модифікацій для посилення якісних характеристик;
- проектування ПЗ для реалізації та дослідження обраного методу та запропонованих модифікацій для нього;
- реалізація спроектованого ПЗ;
- тестування та дослідження результатів тестування ПЗ та методу і модифікацій.

Рисунок Д.15 – Слайд №15

Дякую за увагу!

Рисунок Д.16 – Слайд №16