НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

на тему: «Порівняльний аналіз архітектур на основі великих мовних моделей та детермінованих алгоритмів у багатоагентних системах»

Виконав:
студент IV курсу, групи КІ-21
Макаренко Владислав Юрійович _____

Керівник:
доцент кафедри кафедри штучного інтелекту,
к.ф-м.н, доцент Пишнограєв Іван Олександрович _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри штучного інтелекту,
к.т.н., доцент Комариста Богдана Миколаївна _____

Рецензент:
доцент кафедри економічної кібернетики ФММ,
к.ф-м.н., доцент Лазаренко Ірина Сергіївна _____

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

Київ – 2026 року

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

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«15» січня 2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Макаренку Владиславу Юрійовичу

1. Тема роботи «Порівняльний аналіз архітектур на основі великих мовних моделей та детермінованих алгоритмів у багатоагентних системах», керівник роботи Пишнограєв Іван Олександрович, затверджені наказом по НН ІПСА від «27» травня 2026 р. № 1944-с.

2. Термін подання студентом роботи «05» червня 2026 року.

3. Вихідні дані до роботи: теоретичні відомості про агентно-орієнтоване моделювання та класичні багатоагентні сценарії; наукові публікації з тематики великих мовних моделей, методів конструювання запитів та ланцюгових міркувань; програмні засоби для реалізації та порівняльного тестування агентних стратегій; результати експериментальних прогонів симуляцій із кількісним вимірюванням колективної поведінки агентів.

4. Зміст роботи: огляд агентно-орієнтованого моделювання та класичних сценаріїв колективної поведінки; аналіз архітектури великих мовних моделей та існуючих підходів до їх інтеграції у мультиагентні системи; постановка задачі дослідження; проєктування архітектури програмної системи на основі патерну «Стратегія»; реалізація евристичних та LLM-стратегій для п'яти класичних симуляцій; проведення порівняльних експериментів та аналіз отриманих результатів.

5. Перелік ілюстративного матеріалу: порівняльні таблиці метрик по симуляціях, графіки часових рядів колективної поведінки, схема архітектури програмної системи, електронна презентація.

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

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	20.04.2026	Виконано
2	Підготовка першого розділу	27.04.2026	Виконано
3	Підготовка другого розділу	04.05.2026	Виконано
4	Розробка програмного продукту	11.05.2026	Виконано
5	Підготовка третього розділу	18.05.2026	Виконано
6	Підготовка четвертого розділу	21.05.2026	Виконано
8	Оформлення розділів відповідно до нормоконтролю	01.05.2026	Виконано
8	Оформлення дипломної роботи	02.06.2026	Виконано
9	Підготовка презентації доповіді	05.06.2026	Виконано

Студент Владислав МАКАРЕНКО

Керівник Іван ПИШНОГРАЄВ

РЕФЕРАТ

Дипломна робота: 65 с., 11 рис., 14 табл., 20 посилань, додаток.

АГЕНТНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ, ВЕЛИКІ МОВНІ МОДЕЛІ, МУЛЬТИАГЕНТНІ СИМУЛЯЦІЇ, ЕМЕРДЖЕНТНА ПОВЕДІНКА, УПЕРЕДЖЕНІСТЬ САМОЗБЕРЕЖЕННЯ, ПОРІВНЯЛЬНИЙ АНАЛІЗ.

У роботі досліджується здатність агентів на основі великих мовних моделей відтворювати емерджентну колективну поведінку класичних мультиагентних симуляцій. Основну увагу приділено кількісному порівнянню LLM-агентів та евристичних агентів у п'яти сценаріях агентно-орієнтованого моделювання. Мета дослідження – виявити умови, за яких заміна детермінованих правил на LLM-стратегію є ефективною або принципово неможливою.

Об'єктом дослідження є п'ять класичних ABM-симуляцій: Sugarscape, модель розподілу багатства Больцмана, модель сегрегації Шеллінга, SIR-модель поширення вірусу та модель хижак-жертва (Вовк-Вівця). Предметом дослідження є якісні та кількісні відмінності між евристичними агентами з детермінованими правилами та LLM-агентами на основі GPT-4o-mini за метриками виживаності, розподілу ресурсів, сегрегації та епідемічної динаміки.

Для кожної симуляції реалізовано евристичну стратегію як базову лінію та LLM-стратегію у двох варіантах – без ланцюгових міркувань і з ними. Проведено 21 незалежний прогін на кожну симуляцію за трьома рівнями температури та трьома генераторами псевдовипадкових чисел.

Встановлено, що ефективність LLM-агентів визначається відповідністю між індивідуальною раціональністю агента та механізмом колективного ефекту. LLM повністю відтворила евристичний результат у моделі Шеллінга та наблизилась до нього у Sugarscape з ланцюговими міркуваннями. У моделях Больцмана, SIR та Вовк-Вівця виявлено системну упередженість самозбереження: LLM відмовляється від дій, що підвищують індивідуальний ризик, але є необхідними для виникнення колективних ефектів. Результати рекомендовано використовувати як практичне керівництво при виборі між евристичними та LLM-агентами для конкретного класу ABM-задач.

ABSTRACT

Bachelor's thesis: 65 pages, 11 figures, 14 tables, 20 references, 1 appendix.

AGENT-BASED MODELING, LARGE LANGUAGE MODELS, MULTI-AGENT SIMULATIONS, EMERGENT BEHAVIOR, SELF-PRESERVATION BIAS, COMPARATIVE ANALYSIS.

This work investigates whether agents powered by large language models can reproduce the emergent collective behavior of classical multi-agent simulations. The study focuses on a quantitative comparison of LLM-driven and heuristic agents across five agent-based modeling scenarios. The research goal is to identify conditions under which replacing deterministic rules with an LLM strategy is effective or fundamentally impossible.

The object of study is five classical ABM simulations: Sugarscape, the Boltzmann wealth distribution model, the Schelling segregation model, the SIR virus spread model, and the Wolf-Sheep predator-prey model. The subject of study is the qualitative and quantitative differences between heuristic agents with deterministic rules and LLM agents based on GPT-4o-mini, measured through metrics of survival, resource distribution, segregation, and epidemic dynamics.

For each simulation, a heuristic strategy was implemented as a baseline and an LLM strategy in two variants – with and without chain-of-thought reasoning. A total of 21 independent runs per simulation were conducted across three temperature levels and three random seeds.

Results show that LLM agent effectiveness depends on the alignment between individual agent rationality and the mechanism of collective emergence. LLM agents perfectly reproduced the heuristic result in the Schelling model and approached it in Sugarscape with chain-of-thought reasoning. In the Boltzmann, SIR, and Wolf-Sheep models, a systematic self-preservation bias was identified: LLM agents refuse actions that increase individual risk but are necessary for collective emergent dynamics to arise. The findings are recommended as practical guidance for choosing between heuristic and LLM-based agents for specific classes of ABM tasks.

ЗМІСТ

ВСТУП

Агентно-орієнтоване[1] моделювання є одним із найпотужніших інструментів дослідження складних систем у соціальних, економічних та екологічних науках. Починаючи з класичних робіт Томаса Шеллінга [2] та Джошуа Епштейна й Роберта Екселла [1], вчені використовують ABM (Agent-Based Model – агентно-орієнтована модель) для того, щоб знаходити емерджентні колективні ефекти – розподіл багатства, просторову сегрегацію, динаміку хижак-жертва, епідеміологічні хвилі – спостерігаючи за тим, як складна поведінка виникає із простих локальних правил. Програмні інструменти імітаційного моделювання складних дискретно-подієвих систем активно розвиваються і у вітчизняній науковій школі [20]. Проте традиційні ABM покладаються на жорстко закодовані алгоритмічні правила, які розробник задає вручну. Це обмежує гнучкість агентів і унеможливлює адаптивну поведінку без переписування коду.

Великі мовні моделі (LLM – Large Language Model), такі як GPT-4 компанії OpenAI [3], відкривають принципово нову парадигму для прийняття рішень автономними агентами. Замість детермінованого алгоритму агент отримує опис свого стану та оточення і генерує дію за допомогою нейронної мережі, навченої на мільярдах людських текстів. Такий підхід теоретично дозволяє агентам демонструвати гнучку поведінку без явного програмування кожного сценарію. У 2023–2025 роках з'явилась ціла низка робіт, що досліджують LLM-агентів у соціальних симуляціях – зокрема відома робота «Generative Agents» [4], де 25 агентів на основі LLM демонстрували правдоподібну людиноподібну поведінку у симульованому містечку.

---

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

Проте ключове питання залишається відкритим: чи здатні LLM-агенти відтворювати саме емерджентні ефекти класичних ABM-сценаріїв – ті колективні закономірності, які нерідко потребують від агента ірраціональної з індивідуальної точки зору поведінки? Безумовний обмін ресурсами у моделі Больцмана, розмноження незалежно від рівня енергії у моделі хижак-жертва, вільний контакт із потенційно інфікованими агентами в SIR-моделі (Susceptible–Infected–Resistant – сприйнятливі–інфіковані–стійкі) – це приклади поведінки, що суперечить принципу самозбереження, але є необхідною умовою для виникнення спостережуваних колективних ефектів. LLM, навчена на людських текстах, де самозбереження є нормою, може систематично відмовлятись від такої поведінки. Систематичного кількісного дослідження цього питання в літературі досі не представлено.

Метою цієї роботи є дослідження того, чи можуть агенти на основі LLM відтворювати емерджентні колективні поведінки класичних мультиагентних симуляцій, та виявлення умов, за яких така заміна є ефективною або принципово неможливою. Для досягнення мети у роботі реалізовано п'ять класичних ABM-симуляцій – Sugarscape, модель Больцмана, модель Шеллінга, SIR модель поширення вірусу та модель Вовк-Вівця – із двома типами агентів кожна: евристичним (детерміновані правила) та LLM-агентом на основі GPT-4o-mini. Порівняльні експерименти проводились за трьома рівнями температури та трьома незалежними генераторами псевдовипадкових чисел, що забезпечило статистичну надійність результатів.

Результати роботи можуть бути корисними для розуміння фундаментальних обмежень LLM-агентів у задачах ABM-моделювання та для розробки практичних рекомендацій щодо застосування LLM у гібридних мультиагентних системах. Зокрема, виявлена у дослідженні системна закономірність між структурою емерджентного ефекту і придатністю LLM-заміни дає змогу наперед оцінювати, чи доцільне використання LLM для конкретного класу ABM-задач.

РОЗДІЛ 1 АГЕНТНО-ОРІЄНТОВАНІ СИСТЕМИ ТА ВИКОРИСТАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ У ЇХ ПОБУДОВІ

1.1 Агентно-орієнтоване моделювання та класичні сценарії

Агентно-орієнтоване моделювання є обчислювальним методом дослідження складних систем, у яких поведінка цілого не зводиться до суми поведінок частин. Ключова ідея ABM полягає в тому, що глобальні закономірності – сегрегація, розподіл ресурсів, динаміка популяцій – моделюються знизу вгору: через взаємодію множини автономних агентів, кожен із яких дотримується відносно простого набору локальних правил. На відміну від рівняннєвих методів (систем диференціальних рівнянь), ABM дозволяє явно представляти гетерогенність агентів, просторову структуру взаємодій і стохастичні процеси, що робить його особливо придатним для моделювання соціальних та екологічних систем [18].

Витоки ABM сягають 1970-х років. У 1970 році Джон Конвей запропонував «Гру Життя» [5] – клітинний автомат, у якому три прості правила породжують надзвичайно різноманітну і непередбачувану поведінку, що наочно продемонструвало феномен емерджентності. Знаковою роботою у становленні ABM стала модель сегрегації Томаса Шеллінга [2], у якій агенти двох типів переміщуються по сітці: якщо частка сусідів того самого типу нижча за певний поріг, агент переміщується в інше місце. Ця модель показала, що навіть незначна перевага до сусідів свого типу призводить до яскраво вираженої просторової сегрегації – результат із глибокими соціальними імплікаціями. У 1987 році Крейг Рейнольдс розробив модель Boids, що відтворювала реалістичну поведінку зграї птахів всього з трьох локальних правил: розділення, вирівнювання та зближення [6]. Вирішальний крок у формалізації ABM як наукової методології зробили Джошуа Епштейн і Роберт Екселл у книзі «Growing Artificial Societies» [1], де вони представили Sugarscape – модель штучного суспільства на основі просторового розподілу ресурсів – і

сформулювали ключовий принцип: складні соціальні явища можна відтворювати у комп'ютерних середовищах.

У дослідженні використано п'ять класичних ABM-сценаріїв, які охоплюють різні класи колективної поведінки та відтворюють добре вивчені кількісні закономірності, що робить їх придатними для об'єктивного порівняння стратегій агентів.

1. *Sugarscape.* Агенти переміщуються по двовимірній сітці з нерівномірним розподілом двох ресурсів – цукру та спецій. На кожному кроці агент витрачає фіксовану кількість кожного ресурсу і переміщується до найбагатшої вільної клітини у межах свого радіусу зору. Гетерогенність агентів – різні значення метаболізму та радіусу зору – у поєднанні з нерівномірним розподілом ресурсів породжує стійку майнову нерівність: коефіцієнт Джині стабілізується у діапазоні 0.1–0.35 залежно від параметрів середовища. Епштейн і Екселл [1] показали, що цей розподіл якісно відтворює реальну нерівномірність розподілу доходів без будь-яких апріорних припущень про жадібність чи раціональність агентів. Ключова властивість для цього дослідження: агент, що максимізує свій добробут, діє раціонально з індивідуальної точки зору, а значить, LLM-агент із правильно сформульованим завданням потенційно здатен відтворити цю поведінку.
2. *Модель розподілу багатства Больцмана.* Драгулеску та Яковенко [9] показали, що якщо агенти випадково обмінюються одиницею багатства з випадково обраним сусідом, система у граничному стані стабілізується у розподілі Больцмана-Гіббса – тому самому, що описує розподіл енергій молекул в ідеальному газі. Коефіцієнт Джині такого розподілу теоретично дорівнює 0.5 і не залежить від початкового розподілу багатства чи кількості агентів. Принциповою особливістю моделі є те, що обмін є безумовним і випадковим: агент передає одиницю багатства незалежно від свого поточного

стану, від того, хто є партнером і чи вигідна ця угода. Саме ця індивідуально «ірраціональна» поведінка є необхідною умовою для виникнення розподілу Больцмана-Гіббса: будь-яке селективне або умовне виконання обміну руйнує еквівалентність між моделлю та фізичним газом і призводить до іншого розподілу.

3. *Модель сегрегації Шеллінга.* Агенти двох типів розміщені на сітці; кожен є задоволеним, якщо частка сусідів того самого типу перевищує поріг однорідності (homophily). Незадоволені агенти переміщуються у випадкову вільну клітину. Шеллінг [2] продемонстрував разючий результат: навіть якщо поріг однорідності дорівнює лише 0.3 – тобто агент погоджується бути у меншості – система все одно сходиться до майже повної просторової сегрегації з індексом сегрегації ~0.84. Це є класичним прикладом мікро-макро зв'язку: незначна індивідуальна перевага, що сама по собі далека від нетерпимості, породжує колективний ефект, який виглядає як абсолютна нетерпимість. Вирішення агента переміститись є раціональним з його власної точки зору, що робить цю симуляцію потенційно придатною для LLM-агентів – на відміну від моделей, де вимагається ірраціональна поведінка.

4. *SIR-модель поширення вірусу.* Класична епідеміологічна модель Кермака та Маккендріка (1927) [10] розрізняє три стани агентів: сприйнятливі (S – Susceptible), інфіковані (I – Infected) та стійкі (R – Resistant). Агенти переміщуються по сітці, і при контакті інфікованого зі сприйнятливим відбувається зараження з ймовірністю β; інфікований одужує із ймовірністю γ за крок. Характеристичним результатом моделі є дзвоноподібна крива захворюваності: кількість інфікованих зростає, досягає піку і спадає – за умови, що базове репродуктивне число $R_0 = \beta/\gamma > 1$. Критично важливою умовою для відтворення цієї кривої є вільне переміщення агентів без уникання контактів з інфікованими: якщо

агенти починають практикувати соціальну дистанцію, епідемічна хвиля пригнічується і характеристична крива не формується. Саме ця вимога до «безтурботної» поведінки є потенційним джерелом конфлікту з LLM-агентом, навченим на текстах, де самозбереження є нормою.

5. *Модель хижак-жертва (Вовк-Вівця).* Заснована на рівняннях Лотки-Вольтерри [11, 12], модель відтворює класичні осциляції чисельності хижаків і жертв: популяції вовків і овець коливаються навколо стійкої рівноважної точки з постійним зсувом фаз між ними. Вівці споживають траву та розмножуються; вовки полюють на овець, отримуючи енергію, та також розмножуються; обидва види гинуть при виснаженні енергії. Ключовою особливістю еталонної реалізації, що забезпечує стійкість осциляцій, є ймовірнісне безумовне розмноження: агент відтворюється з фіксованою ймовірністю на кожному кроці незалежно від поточного рівня енергії. З індивідуальної точки зору це нераціонально – витрачати енергію на розмноження, коли її і так мало. Проте саме ця «безумовність» підтримує популяцію на рівні, необхідному для осциляцій: вовки, що відмовляються розмножуватись задля «збереження енергії», неминуче вимирають, оскільки природна смертність не компенсується приростом.

Наразі роботи, присвячені систематичному кількісному порівнянню LLM-агентів і традиційних евристик у зазначених класичних сценаріях, практично відсутні. Переважна більшість досліджень або тестує LLM-агентів у нових, спеціально розроблених середовищах, або розглядає лише один сценарій без порівняння з детермінованою базою. Саме цю прогалину заповнює дане дослідження.

1.2 Великі мовні моделі та методи конструювання запитів

Великі мовні моделі являють собою нейронні мережі, навчені на масштабних текстових корпусах для генерації статистично правдоподібних послідовностей токенів. Сучасні LLM базуються на архітектурі Трансформер, запропонованій у роботі «Attention Is All You Need» [13]. Ключовим механізмом є само-увага, що дозволяє моделі враховувати взаємозв'язки між довільними позиціями у послідовності незалежно від відстані між ними. На відміну від попередніх рекурентних архітектур, Трансформер обробляє всю послідовність паралельно, що уможливлює ефективне масштабування.

На базі архітектури Трансформера компанія OpenAI розробила серію GPT-моделей (GPT – Generative Pre-trained Transformer, генеративний попередньо навчений трансформер). GPT-3 [14], що містить 175 мільярдів параметрів, вперше продемонстрував здатність до виконання нових завдань без додаткового навчання лише на основі прикладів у контексті запиту. GPT-4 [3] значно розширив ці можливості, демонструючи рівень виконання тестів, порівнянний із людським у багатьох предметних галузях. GPT-4o-mini є полегшеною та оптимізованою за вартістю версією GPT-4o: вона зберігає більшість функціональних можливостей при суттєво нижчій вартості API-запитів (Application Programming Interface – програмний інтерфейс застосунків). Саме ці характеристики роблять GPT-4o-mini оптимальним вибором для масових LLM-викликів у мультиагентних симуляціях [19].

Prompt engineering (конструювання запитів) – це дисципліна розробки текстових шаблонів, що спрямовують поведінку LLM до бажаного результату. У контексті ABM-симуляцій виділяють два рівні запитів. Системний запит (system prompt) визначає роль агента в цілому: задає контекст симуляції, правила виживання та формат відповіді. Користувацький запит (user prompt) передає поточний стан агента – його позицію, ресурси, інформацію про сусідів – і запитує конкретне рішення.

Ланцюгові міркування [15] (CoT – Chain-of-Thought), є технікою, за якої модель отримує вказівку явно формулювати проміжні кроки обґрунтування перед тим, як дати остаточну відповідь. CoT значно покращує якість рішень у задачах, що вимагають логічного виведення та багатокрокового планування. У контексті АВМ-агентів CoT реалізується через додаткове поле reasoning у структурованій відповіді, яке агент заповнює обґрунтуванням свого рішення перед вказівкою конкретної дії.

Температура (temperature) є гіперпараметром LLM, що контролює випадковість генерації: при значенні 0.0 модель завжди обирає найбільш імовірний токен, при вищих значеннях вноситься стохастичність. У мультиагентних симуляціях температура відіграє роль аналога «шуму» у поведінці агентів. Структуроване виведення (structured output) є технікою, за якої LLM генерує відповідь у заздалегідь визначеному форматі JSON (JavaScript Object Notation).

1.3 Аналіз існуючих підходів до інтеграції LLM у мультиагентні системи

Ідея використання LLM як «мозку» агентів у симуляціях отримала значну увагу дослідницької спільноти в 2023–2025 роках. У роботі «Generative Agents: Interactive Simulacra of Human Behavior» [4] представили симуляцію невеликого міста з 25 NPC-агентами (NPC – Non-Player Character, персонаж, що не керується гравцем), кожен із яких мав пам'ять, планування та рефлексію на основі LLM. Агенти демонстрували реалістичну соціальну поведінку: ініціювали розмови, планували свій день, поширювали інформацію. Ця робота стала першим масштабним доказом того, що LLM-агенти здатні породжувати правдоподібну поведінку у симульованому середовищі. Однак у ній не

проводилось кількісного порівняння з детермінованими правилами і не досліджувались класичні ABM-феномени з відомими еталонними значеннями.

Розгорнутий огляд підходів до побудови LLM-керованих мультиагентних систем [16] дозволив систематизувати архітектурні рішення та виділити три основних підходи до інтеграції LLM в агентні системи:

- реактивна архітектура: LLM отримує поточний стан і повертає дію без збереження пам'яті попередніх кроків;
- архітектура з короткотерміновою пам'яттю: у запит включається «вікно» останніх кроків симуляції;
- архітектура з довготерміновою пам'яттю: агент підтримує зовнішнє сховище значущих подій.

Попри зростаючу кількість робіт у цій галузі, жодна з них не проводить систематичного кількісного порівняння LLM-агентів із детермінованими евристиками одночасно у кількох класичних ABM-сценаріях із відомими еталонними метриками. Більшість досліджень або вводять нові середовища без «правильної відповіді», або зосереджуються на якісному опису поведінки. Саме ця прогалина визначає наукову новизну та задачу дослідження.

1.4 Постановка задачі

На основі проведеного огляду предметної галузі сформульовано задачу дослідження.

Об'єктом дослідження є п'ять класичних ABM-симуляцій, кожна з яких характеризується відомою емерджентною поведінкою та набором кількісних метрик для її вимірювання:

- Sugarscape: виживаність агентів (%), середній добробут, коефіцієнт Джині;

- модель Больцмана: коефіцієнт Джині, стандартне відхилення багатства;
- модель Шеллінга: відсоток задоволених агентів (% Happy), індекс сегрегації;
- SIR-модель: рівень атаки (Attack Rate), часовий ряд кількості інфікованих;
- модель Вовк-Вівця: часові ряди чисельності вовків та овець.

Задача дослідження включає виконання таких кроків:

1) реалізувати евристичну стратегію для кожної симуляції та зафіксувати базові значення метрик за трьома незалежними генераторами псевдовипадкових чисел;
2) реалізувати LLM-стратегію у двох варіантах: без ланцюгових міркувань та з ними;
3) провести порівняльні експерименти: 3 профілі × 3 значення температури × 3 параметри середовища – 27 прогонів на симуляцію;
4) порівняти метрики LLM-профілів із евристичною базою та відповісти на питання: чи відтворюють LLM-агенти характеристичну емерджентну поведінку кожної симуляції?;
5) виявити системні закономірності, що визначають клас симуляцій, для яких LLM-заміна є ефективною, та теоретично обґрунтувати ці закономірності.

Дослідження проводиться з урахуванням таких обмежень:

- використовується виключно GPT-4o-mini як LLM-провайдер;
- симуляції проводяться за фіксованих параметрів середовища, ідентичних для всіх трьох профілів;
- кожна симуляція виконується протягом 50 кроків.

Висновки до розділу 1

Розглянуто розвиток агентно-орієнтованого моделювання від ранніх клітинних автоматів до сучасних фреймворків, описано п'ять класичних ABM-сценаріїв та їхні ключові емерджентні ефекти. Проаналізовано архітектуру сучасних LLM, методи конструювання запитів і ланцюгових міркувань як інструменти управління поведінкою агентів. Розглянуто існуючі підходи до інтеграції LLM у мультиагентні системи та основні програмні засоби реалізації. Сформульовано задачу дослідження: кількісно порівняти евристичних та LLM-агентів у п'яти класичних симуляціях і виявити умови ефективності LLM-заміни.

РОЗДІЛ 2 ФОРМАЛІЗАЦІЯ ТА МАТЕМАТИЧНІ МОДЕЛІ МУЛЬТИАГЕНТНОГО МОДЕЛЮВАННЯ

2.1 Формальна модель агента та середовища

У формальному сенсі агент агентно-орієнтованої моделі визначається як кортеж:

$$A = (S,\ Act,\ \pi,\ \tau),$$

де S – множина можливих станів агента; Act – множина допустимих дій; $\pi : S \to Act$ – функція прийняття рішень (стратегія); $\tau : S \times Act \times E \to S$ – функція переходу, що визначає новий стан агента після виконання дії в середовищі E.

Середовище симуляції моделюється як двовимірна дискретна сітка:

$$G = \{(x,\, y) \mid x \in \{0, \ldots, W-1\},\ y \in \{0, \ldots, H-1\}\},$$

де W та H – ширина і висота сітки відповідно.

Стан агента в момент часу t описується вектором:

$$s_i(t) = \big(\mathrm{pos}_i(t),\ \mathrm{res}_i(t),\ \mathrm{state}_i(t)\big),$$

де $\mathrm{pos}_i(t) \in G$ – поточна позиція; $\mathrm{res}_i(t) \in \mathbb{R}$ – кількість ресурсів агента; $\mathrm{state}_i(t)$ – додаткові дискретні атрибути (тип, стан здоров'я тощо).

Система з N агентів еволюціонує дискретними кроками. На кожному кроці t агент i: спостерігає локальний контекст $c_i(t)$ у межах свого радіусу зору; застосовує стратегію $a_i(t) = \pi(s_i(t),\, c_i(t))$; виконує дію, що змінює власний стан та стан середовища. При евристичній стратегії π є

детермінованою функцією; при LLM-стратегії π є стохастичною функцією, де рішення генерується нейронною мережею.

2.2 Математичні моделі класичних ABM-сценаріїв

2.2.1 Sugarscape

Кожна клітина (x, y) містить поточний рівень ресурсу $r(x, y) \in [0,\ r_{\max}]$, який зростає на ρ одиниць за крок і не перевищує максимум:

$$r(x, y) \leftarrow \min\big(r(x, y) + \rho,\ r_{\max}\big).$$

Коли агент відвідує клітину, він збирає весь наявний ресурс, після чого $r(x, y) = 0$. Завдяки цьому у будь-який момент клітини мають різні рівні r, і вибір напряму руху є змістовним.

Агент i з метаболізмом m_i та радіусом зору v_i переміщується у клітину з найбільшим ресурсом у межах зору:

$$\mathrm{pos}_i^*(t) = (x^*, y^*) : r(x^*, y^*) \geq r(x, y)\ \forall\,(x, y) \in V\big(\mathrm{pos}_i(t),\ v_i\big),$$

де $V(p, v)$ – множина клітин на відстані Чебишева не більше v від позиції p.

Рівень добробуту оновлюється за правилом:

$$w_i(t + 1) = w_i(t) + r\big(\mathrm{pos}_i^*(t)\big) - m_i.$$

Агент гине, якщо $w_i(t) \leq 0$.

2.2.2 Модель розподілу багатства Больцмана

На кожному кроці агент i випадково обирає сусіда j і безумовно передає йому одну одиницю багатства:

$$w_i(t+1) = w_i(t) - 1, \qquad w_j(t+1) = w_j(t) + 1$$

за умови $w_i(t) > 0$. Драгулеску та Яковенко [9] показали, що у граничному стані розподіл багатства збігається до розподілу Больцмана-Гіббса:

$$P(w) = \frac{1}{\langle w \rangle} \exp\left(-\frac{w}{\langle w \rangle}\right),$$

де $\langle w \rangle$ – середнє багатство в системі. Цей розподіл є аналогом канонічного розподілу в статистичній механіці, де роль температури відіграє $\langle w \rangle$.

2.2.3 Модель сегрегації Шеллінга

Кожен агент i належить до типу $t_i \in \{0, 1\}$ та має поріг однорідності $\theta_i \in [0, 1]$. Агент вважається задоволеним, якщо

$$f_i^{\text{same}} = \frac{\left|\{j \in N(i) : t_j = t_i\}\right|}{|N(i)|} \geq \theta_i, \quad (2.4)$$

де $N(i)$ – множина зайнятих сусідніх клітин. Незадоволений агент переміщується у випадкову вільну клітину.

2.2.4 SIR-модель поширення вірусу

Динаміка захворюваності описується системою диференціальних рівнянь:

$$\frac{dS}{dt} = -\frac{\beta \cdot S \cdot I}{N},$$
$$\frac{dI}{dt} = \frac{\beta \cdot S \cdot I}{N} - \gamma \cdot I,$$
$$\frac{dR}{dt} = \gamma \cdot I,$$

де S, I, R – кількість сприйнятливих, інфікованих та стійких агентів; $N = S + I + R$; β – коефіцієнт передачі; γ – коефіцієнт одужання. Ключовим параметром є базове репродуктивне число:

$$R_0 = \frac{\beta}{\gamma}.$$

При $R_0 > 1$ епідемія поширюється; при $R_0 < 1$ – згасає.

2.2.5 Модель хижак-жертва (Вовк-Вівця)

Безперервна динаміка описується рівняннями Лотки-Вольтерри:

$$\frac{dx}{dt} = \alpha x - \beta xy,$$
$$\frac{dy}{dt} = \delta xy - \gamma y,$$

де x – чисельність жертв (вівці), y – чисельність хижаків (вовки); α – коефіцієнт відтворення жертв; β – коефіцієнт виїдання; δ – коефіцієнт приросту хижаків; γ – коефіцієнт природної смертності хижаків. Вищепредставлені рівняння мають ненульову рівноважну точку:

$$x^* = \frac{\gamma}{\delta}, \qquad y^* = \frac{\alpha}{\beta}.$$

Навколо неї система здійснює стійкі осциляції.

2.3 LLM як стохастична функція прийняття рішень

Основу сучасних LLM становить механізм само-уваги [13]. Для матриць запитів Q, ключів K і значень V розмірності d_k вихід блоку уваги обчислюється як:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

Множник $1/\sqrt{d_k}$ стабілізує градієнти при великих розмірностях. Багатоголова увага виконує рівняння паралельно для h різних проекцій. Генерація тексту LLM – це послідовне семплування токенів. На кожному кроці модель обчислює розподіл ймовірностей над словниковим запасом:

$$P(x_t \mid x_1, \ldots, x_{t-1}) = \mathrm{softmax}\left(\frac{z_t}{\tau}\right),$$

де $z_t \in \mathbb{R}^{|V|}$ – вектор логітів; $\tau > 0$ – параметр температури. При $\tau \to 0$

розподіл вироджується у детерміновану функцію вибору найімовірнішого токена. При $\tau > 1$ розподіл стає більш рівномірним, збільшуючи різноманітність рішень.

LLM-стратегія формально відрізняється від евристичної: замість аналітичної функції маємо

$$\pi_{\mathrm{LLM}}(s,\, c,\, \tau) = \mathrm{Decode}\big(\mathrm{LLM}(\mathrm{Encode}(s,\, c)),\, \tau\big),$$

де Encode перетворює стан агента на природномовний запит, LLM генерує текстову відповідь, а Decode розбирає її у структуровану дію. Ланцюгові міркування (CoT) модифікують: модель спочатку генерує проміжне обґрунтування r, а потім на його основі – кінцеву дію, що підвищує якість рішень у задачах із неочевидними критеріями.

2.4 Метрики оцінювання колективної поведінки

2.4.1 Коефіцієнт Джині

Вимірює нерівномірність розподілу ресурсів між агентами. Для вектора добробуту $(w_1, w_2, \ldots, w_n)$, впорядкованого за зростанням:

$$G = \frac{2\sum_{i=1}^{n} i \cdot w_i}{n\sum_{i=1}^{n} w_i} - \frac{n+1}{n}.$$

Значення $G \in [0, 1]$: при $G = 0$ усі агенти мають однаковий добробут; при $G = 1$ весь добробут зосереджений у одного агента. Рівноважний розподіл

Больцмана-Гіббса теоретично дає $G \approx 0{,}5$.

2.4.2 Індекс сегрегації

Вимірює середній рівень локальної однорідності сусідства:

$$SI = \frac{1}{N} \sum_{i=1}^{N} f_i^{\text{same}},$$

де f_i^{same} визначено у формулі (2.4). Значення $SI \in [0, 1]$: $SI \approx 0$ – повне змішання типів; $SI \approx 1$ – повна сегрегація.

2.4.3 Рівень атаки

У SIR-моделі визначає частку первісно сприйнятливих агентів, що зазнали зараження:

$$AR = \frac{R(T)}{S(0)},$$

де $R(T)$ – кількість стійких агентів на останньому кроці T; $S(0)$ – початкова кількість сприйнятливих.

2.4.4 Виживаність

У моделі Sugarscape визначається як частка агентів, що залишились живими після T кроків

$$SR = \frac{N_{\text{surv}}(T)}{N(0)},$$

де $N_{\text{surv}}(T)$ – кількість живих агентів у кроці T; $N(0)$ – початкова кількість агентів.

Висновки до розділу 2

Розглянуто формальну модель агента як кортежу (S, Act, π, τ), що є математичним підґрунтям патерну «Стратегія». Виведено ключові рівняння п'яти класичних ABM-сценаріїв – правило оновлення ресурсів Sugarscape , розподіл Больцмана-Гіббса, умову задоволеності Шеллінга, систему рівнянь SIR та рівняння Лотки-Вольтерри. Описано механізм само-уваги та вплив параметра температури τ на стохастичність LLM-рішень. Визначено чотири кількісні метрики – коефіцієнт Джині , індекс сегрегації , рівень атаки та виживаність, що використовуються для порівняння агентів у наступному розділі.

РОЗДІЛ 3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

3.1 Загальна архітектура системи

Реалізована система є програмним засобом для проведення порівняльних ABM-симуляцій із двома типами агентів: евристичними та LLM-агентами. Головна архітектурна вимога, що випливає з наукової задачі, – можливість змінювати механізм прийняття рішень агента без модифікації симуляції, що дозволяє ізолювати вплив саме стратегії на колективну поведінку.

Система реалізована мовою Python 3.12 і побудована із трьох основних модулів:

- модуль $agents/$ – абстрактний інтерфейс стратегії прийняття рішень і базовий клас агента;
- модуль $simulations/$ – реалізації п'яти симуляцій, кожна з яких містить визначення моделі, евристичну стратегію, LLM-стратегію, скрипт запуску та застосунок для візуалізації;
- модуль $utils/$ – спільні інструменти: логер рішень, функції обчислення метрик та скрипти порівняльного аналізу.

Структуру системи представлено в таблиці 3.1.

Таблиця 3.1 – Структура модулів системи

№ п/п	*Модуль*	*Файли*	*Призначення*
1	$agents/$	$strategy.py$, $base.py$	Інтерфейс $DecisionStrategy$, $BaseAgent$
2	$simulations/$	$X.py$, $X\_strategy.py$, $X\_llm.py$, $run\_X.py$, $X\_app.py$	Реалізація 5 симуляцій
3	$utils/$	$decision\_logger.py$, $metrics.py$, $analyze\_results.py$, $summarize\_decisions.py$	Збір і аналіз даних

У кожній симуляції

$X \in \{sugarscape,\ boltzmann,\ schelling,\ virus,\ wolf\_sheep\}$

виділено однакову структуру файлів. Файл `X.py` містить клас моделі та клас агента. Файл $X\_strategy.py$ реалізує евристичну стратегію, специфічну для цієї симуляції. Файл $X\_llm.py$ реалізує LLM-стратегію, Pydantic-моделі структурованого виведення та конфігурацію профілів. Файл $run\_X.py$ є точкою входу для командного рядка, що приймає аргументи $mode$, $llm-config$, $temperature$, $seed$. Файл $X\_app.py$ запускає інтерактивну візуалізацію.

Такий поділ забезпечує єдину точку зміни: для підключення нової стратегії достатньо реалізувати новий клас, що успадковує $DecisionStrategy$, і передати екземпляр у конструктор агента – жодних змін у логіці симуляції не потрібно.

3.2 Патерн «Стратегія» для прийняття рішень

Центральним архітектурним рішенням є застосування патерну «Стратегія», що належить до групи поведінкових патернів проєктування. Патерн передбачає відокремлення алгоритму від об'єкта, що його використовує: алгоритм (стратегія) інкапсулюється в окремий клас, а об'єкт зберігає посилання на інтерфейс стратегії, не знаючи конкретної реалізації.

У системі патерн реалізується через ієрархію:

- $DecisionStrategy$ – абстрактний базовий клас інтерфейсу;
- $HeuristicStrategy$ – базова реалізація для евристичних правил;
- $LLMStrategy$ – базова реалізація для LLM-рішень;
- $XHeuristicStrategy$ та $XLLMStrategy$ – конкретні реалізації для кожної симуляції X.

Завдяки цій ієрархії один і той самий клас агента може отримувати рішення від будь-якої стратегії, що гарантує коректне порівняння: агент починає крок в ідентичному стані незалежно від типу стратегії.

3.2.1 Методологія та умови проведення експериментів

Інтерфейс $DecisionStrategy$ визначено в модулі $agents/strategy.py$ і є абстрактним базовим класом. Клас оголошує єдиний абстрактний метод $decide(context\colon\ dict[str,\ Any])\ ->\ dict[str,\ Any]$, який отримує словник контексту агента, що описує його поточний стан та оточення, і повертає словник дій для виконання.

Використання словника $dict[str,\ Any]$ як уніфікованого контракту між агентом і стратегією є свідомим рішенням: він забезпечує гнучкість – різні симуляції передають різні ключі, і водночас зберігає єдиний інтерфейс стратегії. Стратегія відповідає лише за читання контексту і формування словника дій; всю взаємодію з середовищем виконує агент.

Базовий клас агента $BaseAgent$ визначає загальний потік кроку симуляції у методі $step()$. На кожному кроці агент виконує такі дії:

1. Викликає метод $get\_context()$, щоб зібрати поточний стан: позицію, ресурси, інформацію про сусідів тощо.
2. Передає зібраний контекст у метод $get\_context()$, отримуючи словник рекомендованих дій.
3. Виконує повернуті дії через метод $execute\_action(action)$.

Такий трирівневий поділ – збір контексту, прийняття рішення, виконання дії – дозволяє кожній із частин змінюватись незалежно. Наприклад, у режимі асинхронного LLM-виклику кроки 1 і 3 залишаються незмінними, а лише крок 2 виконується паралельно для всіх агентів.

3.2.2 Детерміновані правила

$HeuristicStrategy$ є базовим класом для всіх евристичних стратегій. Конкретна реалізація $XHeuristicStrategy$ перевизначає метод $decide()$ і реалізує детерміновану логіку, характерну для еталонної поведінки кожної симуляції.

Евристичні стратегії реалізовані таким чином.

1. $SugarscapeHeuristicStrategy$. Стратегія перебирає всі клітини у межах радіусу зору агента, обчислює для кожної значення добробуту як суму цукру та спецій і обирає клітину з найвищим добробутом. При рівних значеннях перевагу отримує найближча клітина за манхеттенською відстанню. Дія повертається у форматі $\{"move": (x, y)\}$.
2. $BoltzmannHeuristicStrategy$. Стратегія безумовно передає одну одиницю багатства випадковому сусіду без будь-яких умов, реалізуючи саме той «ірраціональний» обмін, що породжує розподіл Больцмана-Гіббса [9]. Дія: $\{"give": True, "partner\_id": id\}$.
3. $SchellingHeuristicStrategy$. Стратегія обчислює частку сусідів того самого типу серед усіх зайнятих сусідніх клітин. Якщо частка нижча за поріг `homophily`, агент переміщується; інакше залишається. Дія: $\{"give": True, "partner\_id": id\}$.
4. $VirusHeuristicStrategy$. Стратегія реалізує вільне переміщення без уникання контактів та ймовірнісне зараження при контакті інфікованого агента зі сприйнятливим. Дія: $\{"move": (x, y), "interact": True\}$.
5. $WolfSheepHeuristicStrategy$. Вівця переміщується до клітини з травою; якщо трава відсутня – випадково. Вовк переміщується до найближчої вівці; за відсутності їжі – випадково. Розмноження – ймовірнісне й безумовне: незалежно від рівня енергії агент відтворюється з фіксованою ймовірністю.

3.2.3 LLM-стратегія та механізм структурованого виведення

$LLMStrategy$ є базовим класом для LLM-стратегій і успадковує $DecisionStrategy$. Конкретна реалізація $XLLMStrategy$ перевизначає метод $decide()$, який у синхронному режимі викликає $asyncio.run(decide\_async(context))$, а в асинхронному режимі – $decide\_async(context)$ очікується ззовні.

Ключовий технічний компонент – бібліотека $instructor$ [17], що розширює клієнт OpenAI API можливістю структурованого виведення: замість довільного рядка модель повертає валідний об'єкт Pydantic-моделі. Для кожної симуляції визначено дві Pydantic-моделі:

- $XAction$ – мінімальна модель без полів міркувань, що містить лише поля рішення;
- $XActionWithReasoning$ – розширена модель з додатковим полем `reasoning: str`, що містить текстове обґрунтування рішення перед конкретною дією.
- $LLMStrategy$ – базова реалізація для LLM-рішень;

Вибір моделі виведення визначається конфігурацією профілю: при $include\_reasoning = True$ використовується $XActionWithReasoning$, що реалізує ланцюгові міркування [15].

Структурований виклик API здійснюється через метод $client.chat.completions.create()$ із додатковим параметром $response\_model$, який instructor автоматично транслює у JSON Schema та валідує відповідь. Якщо LLM повертає неправильно відформатовану відповідь, $instructor$ виконує повторний запит до досягнення коректного формату або сигналізує про помилку.

Для ілюстрації структури промпту наведемо приклад для Sugarscape-агента у профілі $optimized$. Системний промпт визначає роль та ціль агента: "Sugarscape agent. Survive by collecting sugar+spice. You use

2S+1Sp/turn. Die if either reaches 0. Pick best move." Користувацький промпт передає поточний стан у мінімальному форматі: "You:(3,7) S4,Sp2 Moves:(3,8):S3,Sp1,(4,7):S0,Sp4,(3,6):S2,Sp3". Такий компактний формат скорочує кількість токенів у запиті і, відповідно, вартість та затримку. У профілі $baseline$ той самий запит оформлюється розгорнутим природномовним описом з переліком усіх сусідніх клітин, відстанями та явним запитом на обґрунтування – що підвищує читабельність reasoning, але збільшує розмір промпту у 3–4 рази.

У разі будь-якого збою (мережева помилка, таймаут, невалідна відповідь після повторних спроб) стратегія перехоплює виключення та повертає безпечну резервну дію, фіксуючи помилку в логері. Це гарантує, що один збій API не зупиняє всю симуляцію.

3.2.4 Конфігурація профілів

Для управління параметрами LLM-стратегії введено клас конфігурації $LLMConfig$. Клас містить такі поля:

- $prompt\_style$: str – стиль промпту ("$short$" – мінімальний опис стану, "$long$" – повний контекстуальний опис);
- $include\_reasoning$: $bool$ – використовувати Pydantic-модель із полем $reasoning$.
- use_async: $bool$ – використовувати асинхронний режим виклику API;
- max_tokens: int – максимальна кількість токенів у відповіді;
- $temperature$: $float$ – параметр стохастичності LLM;
- $model$: str – ідентифікатор моделі;

Для зручності клас надає три фабричних методи класу:

- $LLMConfig.optimized(temperature)$ – профіль без ланцюгових міркувань: короткий промпт, асинхронний режим, $max\_tokens = 30$;
- $LLMConfig.optimized\_with\_reasoning(temperature)$ – профіль із ланцюговими міркуваннями: короткий промпт, $max\_tokens = 200$;
- $LLMConfig.baseline(temperature)$ – профіль із довгим промптом та reasoning, $use\_async = False$. Використовується для налагодження та якісного аналізу поведінки.

Конфігурація профілю передається в конструктор $XLLMStrategy$ і зберігається як атрибут $self.config$. Під час кожного виклику $decide\_async()$ стратегія звертається до $self.config$ для вибору промпту, Pydantic-моделі та параметрів API.

3.3 Проєктування модулів симуляцій

3.3.1 Контекстний об'єкт агента

Контекст агента – це словник $dict[str,\ Any]$, що формується методом $get\_context()$ кожного конкретного класу агента перед передачею у стратегію. Контекст виконує роль снапшоту стану агента та його локального оточення на момент прийняття рішення.

Склад контексту залежить від симуляції, але у всіх випадках дотримується уніфікований мінімум:

- $agent\_id:\ int$ – унікальний ідентифікатор агента;
- $position:\ tuple[int,\ int]$ – координати поточної клітини.

Специфічні поля контексту по симуляціях наведено в таблиці 3.2.

Таблиця 3.2 – Склад контекстного об'єкта по симуляціях

№ п/п	*Симуляція*	*Специфічні ключі контексту*
1	Sugarscape	$sugar$, $spice$, $metabolism\_sugar$, $metabolism\_spice$, $vision$, $neighbors$, $grid$
2	Boltzmann	$wealth$, $neighbors\_wealth$, $neighbor\_ids$
3	Schelling	$agent\_type$, $similar\_fraction$, $homophily$,$total\_neighbors$
4	Virus SIR	$state$, $infected\_neighbors$, $step$
5	Вовк-Вівця	$energy$, $agent\_kind$, $nearby\_prey$,$nearby\_wolves$, $grass\_available$

Важливим аспектом є серіалізація контексту перед логуванням. Об'єкти не можуть бути серіалізовані у JSON безпосередньо. Тому у модулі $utils/decision\_logger.py$ визначено функцію $make\_serializable\_context()$, яка рекурсивно обходить словник контексту, пропускає ключі $cell$, $grid$, $neighbors$ і замінює складні об'єкти на рядкові представлення. Це дозволяє зберігати повний контекст кожного рішення у файл для подальшого аналізу без втрати читабельності.

Для LLM-стратегії контекст відіграє ще одну роль: з нього формуються системний та користувацький промпти. Метод $decide\_async()$ перетворює числові поля контексту на природномовний текст, що подається у LLM як опис ситуації. Таким чином, якість рішення LLM безпосередньо залежить від повноти та чіткості контексту.

3.3.2 Відокремлення рішення від виконання

Після отримання словника дій від стратегії агент передає його в метод $execute\_action(action)$, який відповідає за фізичне виконання рішення у середовищі. Відокремлення виконання від прийняття рішення є принциповим: стратегія не має прямого доступу до API і не може самостійно змінити стан симуляції – вона лише декларує намір.

Метод $execute\_action(action)$ виконує такі дії:

- інтерпретує ключі словника дій;
- здійснює валідацію рішення: наприклад, якщо LLM повернув координати поза межами сітки або зайнятої клітини, агент залишається на поточній позиції; оновлює атрибути агента (позицію, рівень ресурсів, стан здоров'я) відповідно до правил симуляції;
- оновлює стан середовища: споживає ресурс клітини, передає багатство сусіду, змінює стан зараження.

Валідація на рівні $execute\_action(action)$ є критично важливою для стійкості системи: LLM може згенерувати технічно коректну відповідь за Pydantic-схемою, але семантично некоректну (наприклад, рух до зайнятої клітини). Резервна поведінка у таких випадках – залишитись на місці – забезпечує безперервність симуляції.

У моделі Sugarscape після виконання переміщення метод $eat()$ автоматично збирає ресурс поточної клітини та нараховує метаболічні витрати. Якщо після цього $sugar <= 0$ або $spice <= 0$, агент вважається загиблим і видаляється з моделі.

3.4 Система збору та аналізу даних

3.4.1 Логування рішень

Клас $DecisionLogger$, реалізований у модулі $utils/decision\_logger.py$, призначений для детального запису кожного рішення агента в перебігу симуляції. Logger приймає шлях до вихідного JSON-файлу і зберігає список записів у пам'яті до явного виклику методу $save()$.

Для кожного рішення фіксуються такі поля:

- $step$ – поточний крок симуляції;
- $agent\_id$ – унікальний ідентифікатор агента;
- $agent\_type$ – тип агента;
- $strategy$ – тип стратегії;
- $context$ – серіалізований словник контексту агента;
- $action$ – прийнята дія.

Для LLM-рішень додатково фіксуються:

- $system\_prompt$ – системний промпт, переданий у LLM;
- $user\_prompt$ – користувацький промпт із поточним станом агента;
- $reasoning$ – текст поля $reasoning$;
- $latency\_ms$ – час відповіді API у мілісекундах.

Функція $measure\_latency()$ повертає поточний UNIX-час у мілісекундах. Затримка обчислюється як різниця між значеннями до та після виклику API.

Метод $measure\_latency()$ фіксує збої LLM-виклику: тип і текст виключення та резервну дію, що була застосована. Це дозволяє постфактум відрізнити справжні рішення LLM від резервної поведінки при аналізі.

Метод $measure\_latency()$ формує агреговану статистику: загальна кількість записів, кількість помилок, розподіл прийнятих дій. Цей метод використовується скриптом $summarize\_decisions.py$ для швидкого огляду поведінки агентів без завантаження повного JSON-файлу.

3.4.2 Збирачі даних та метрики симуляції

Для збору кількісних метрик симуляції на кожному кроці використовується Mesa DataCollector [8] – вбудований компонент фреймворку, що дозволяє реєструвати функції-обчислювачі для рівня моделі та рівня агента.

У кожній симуляції DataCollector реєструється у конструкторі моделі і прив'язується до двох типів збирачів:

- збирачі рівня моделі (model reporters) – функції, що приймають об'єкт моделі і повертають скалярне значення метрики на поточному кроці;
- збирачі рівня агента (agent reporters) – функції, що приймають об'єкт агента і повертають значення атрибуту.

Модельні метрики по симуляціях наведено в таблиці 3.3.

Таблиця 3.3 – Метрики DataCollector по симуляціях

№ п/п	*Симуляція*	*Метрики рівня моделі*
1	Sugarscape	$AgentCount$, $MeanWealth$, $GiniCoefficient$, $SurvivalRate$
2	Boltzmann	$GiniCoefficient$, $MeanWealth$, $WealthStd$
3	Schelling	$HappyAgents$, $SegregationIndex$, $UnhappyCount$
4	Virus SIR	$Susceptible$, $Infected$, $Resistant$, $AttackRate$
5	Вовк-Вівця	$WolfCount$, $SheepCount$, $GrassCount$

Функція $calculate\_gini()$ у модулі $utils/metrics.py$ реалізує формулу (2.15): сортує вектор добробутів, обчислює зважену суму і нормалізує на загальний добробут. Функція використовується DataCollector безпосередньо як лямбда-обгортка, що отримує список значень атрибута $wealth$ від усіх живих агентів.

Після завершення симуляції накопичені дані DataCollector витягуються методом $model.datacollector.get\_model\_vars\_dataframe()$, що повертає об'єкт $pandas.DataFrame$, де рядки – кроки, стовпці – метрики. Цей DataFrame зберігається у CSV-файл (Comma-Separated Values – значення, розділені комами) для подальшої обробки.

3.4.3 Скрипти порівняльного аналізу

Для автоматизації масових експериментів і збору результатів розроблено систему bash-скриптів та Python-аналізаторів.

Оскільки Mesa виконує крок симуляції синхронно через $model.step()$, а LLM-виклик є мережевою операцією з затримкою 1–5 с, послідовне виконання агентів у моделі з 20–30 агентами займало б 20–150 секунд на крок. Для подолання цього обмеження введено механізм асинхронного паралельного виконання. Замість виклику $model.step()$ скрипт запуску викликає власну корутину $run\_agents\_async()$, яка:

- збирає всі LLM-агенти поточного кроку;
- для кожного викликає $agent.get\_context()$ і передає контекст у корутину $strategy.decide\_async()$;
- запускає всі корутини паралельно через $asyncio.gather()$;
- після завершення всіх корутин послідовно виконує $execute\_action()$ для кожного агента.

Паралельне виконання знижує час одного кроку до медіани найдовшого окремого запиту (а не суми всіх), що на практиці дає 5–8-кратне прискорення при 20 LLM-агентах. Через те що Mesa не підтримує асинхронний $step()$, вік агента та виконання дій інкрементуються вручну у корутині, а не через стандартний метод $step()$.

Bash-скрипти організовані за єдиним шаблоном для кожної симуляції:

- $run\_X\_multiseed.sh\ MODEL\ TEMP$ – запускає три незалежні прогони з seed 456, 789 та 1337 для заданого профілю та температури;
- $run\_X\_temperature.sh\ MODEL$ – запускає прогони для трьох значень температури (0.0, 0.3, 0.7);
- $compare\_X\_results.sh\ MODEL$ – формує зведені таблиці порівняння метрик.

Модуль $utils/analyze\_results.py$ агрегує результати по seed, обчислює середні значення та стандартні відхилення метрик і формує підсумковий CSV. Модуль $utils/summarize\_decisions.py$ завантажує JSON-файл DecisionLogger і обчислює розподіл рішень, відсоток помилок та статистику затримок.

Результати зберігаються у директорії $results\_X\_profile\_tempT/$, де X – назва симуляції, profile – ім'я профілю, T – значення температури. Така структура директорій дозволяє скриптам аналізу автоматично знаходити файли за шаблоном без ручного вказування шляхів.

3.5 Інтерактивна візуалізація симуляцій

Для якісного аналізу поведінки агентів та демонстрації відмінностей між стратегіями розроблено інтерактивний веб-застосунок на основі бібліотеки Solara [8]. Застосунок реалізовано для кожної з п'яти симуляцій у файлах $X\_app.py$ і запускається локально командою

$solara\ run\ src/simulations/X\_app.py$, відкриваючи браузер на адресі $http://localhost:8765$.

Застосунок побудовано на компонентах Mesa візуалізації: $SolaraViz$ забезпечує інтерактивний цикл симуляції з кнопками запуску, паузи та скидання; $make\_space\_component$ відображає двовимірну сітку агентів; $make\_plot\_component$ будує часовий ряд обраної метрики в реальному часі.

Ключовою особливістю є підтримка кількох конфігурацій у межах одного застосунку.

Перемикання між конфігураціями здійснюється через випадаючий список без перезавантаження сторінки. При виборі LLM-конфігурації застосунок перевіряє наявність змінної середовища $OPENAI\_API\_KEY$ і відображає попередження, якщо ключ відсутній.

Візуальне кодування агентів дозволяє миттєво оцінити стан популяції. Колір відображає рівень ресурсів агента: зелений – здоровий стан (запаси >80), жовтий – помірний (30–80), помаранчевий – низький (15–30), червоний – критичний (<15). Форма маркера розрізняє тип стратегії: коло – евристичний агент, квадрат – LLM-агент. Фонові шари сітки відображають поточний розподіл ресурсів (цукор і спеції) у вигляді напівпрозорих теплових карт.

Скриншот застосунку для симуляції Sugarscape у режимі еврестичної популяції представлено на рисунку 3.1.

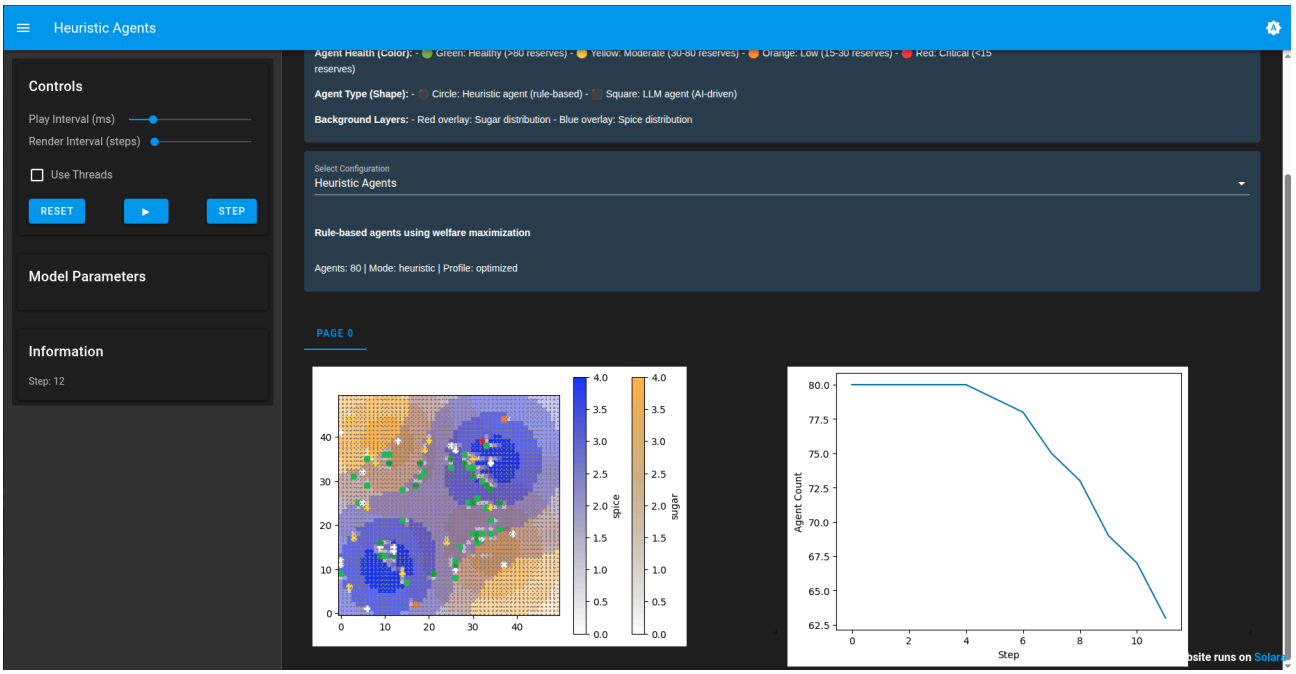

Рисунок 3.1 – Інтерфейс застосунку Sugarscape

Висновки до розділу 3

Спроєктовано та реалізовано програмну систему із трьох модулів: *agents/*, *simulations/* та *utils/*. Центральним архітектурним рішенням є патерн «Стратегія», що відокремлює тіло агента від механізму прийняття рішень через інтерфейс *DecisionStrategy* з єдиним методом $decide(context) -> action$. Визначено уніфікований контракт: контекст агента передається як словник, дія повертається як словник, що забезпечує взаємозамінність евристичних та LLM-стратегій без змін у логіці симуляції. Структуроване LLM-виведення реалізовано через instructor [17] і Pydantic-моделі, а ланцюгові міркування – через додаткове поле *reasoning* [15]. Асинхронне виконання LLM-запитів через $asyncio.gather()$ скорочує час одного кроку у 5–8 разів порівняно з послідовним виконанням. Система збору

даних поєднує Mesa DataCollector [8] для кількісних метрик і $DecisionLogger$ для якісного аналізу рішень, затримок та reasoning.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПОВЕДІНКИ LLM-АГЕНТІВ У МУЛЬТИАГЕНТНИХ СИМУЛЯЦІЯХ

4.1 Методологія та умови проведення експериментів

Для забезпечення відтворюваності та статистичної надійності результатів усі експерименти проводились за уніфікованою методологією. Кожна з п'яти симуляцій тестувалась у трьох профілях:

- heuristic: евристична стратегія з детермінованими правилами;
- optimized: LLM-стратегія без ланцюгових міркувань;
- optimized_with_reasoning: LLM-стратегія із ланцюговими міркуваннями.

Для LLM-профілів варіювались три значення параметра температури та три генератори псевдовипадкових чисел, що дало 27 незалежних прогонів на кожну симуляцію.

Загальна матриця експериментів наведена в таблиці 4.1.

Таблиця 4.1 – Матриця експериментів

№ п/п	*Профіль*	*Температура*	*Seeds*	*Прогонів*
1	heuristic	–	456, 789, 1337	3
2	optimized	0.0, 0.3, 0.7	456, 789, 1337	9
3	optimized_with_reasoning	0.0, 0.3, 0.7	456, 789, 1337	9
Усього на симуляцію				21

Кожна симуляція виконувалась протягом 50 дискретних кроків. Параметри середовища фіксувались однаковими для всіх профілів, щоб виключити вплив початкових умов. Основні параметри наведено в таблиці 4.2.

Таблиця 4.2 – Параметри симуляцій

№ п/п	*Симуляція*	*Агентів*	*Розмір сітки*	*Ключові параметри*
1	Sugarscape	20	20x20	vision=3, metabolism=3, sugar_max=4
2	Boltzmann	20	10x10	initial_wealth=1
3	Schelling	200	20x20	homophily=0.5, density=0
4	Virus SIR	20	10x10	infection_prob=0.4, recovery_time=10
5	Predator-Prey	30	10x10	sheep=20, wolves=10, wolf_reproduce=0.02

Метрики збирались на кожному кроці. Для LLM-агентів додатково логувались: прийняте рішення, поле reasoning (за наявності) та час відповіді API (latency). Усі числові результати у розділі 4.2 наведено як середні за трьома seed при температурі 0.0, якщо не вказано інше.

4.2 Результати за симуляціями

4.2.1 Модель сегрегації Шеллінга

Модель Шеллінга стала єдиною симуляцією, де LLM-агенти відтворили евристичний результат із майже ідеальною точністю. Кількісні результати наведено в таблиці 4.3.

Таблиця 4.3 – Результати моделі Шеллінга

№ п/п	*Профіль*	*% Happy*	*Segregation Index*	*Unhappy*	*Час/крок*
1	heuristic	100%	0.844	0	0.001 с
2	optimized	100%	0.840	0	1.655 с
3	optimized_with_reasoning	100%	0.840	0	3.096 с

Усі три профілі досягли 100% задоволених агентів та практично однакового індексу сегрегації (~0.84). Порівняння кінцевих метрик показано на рисунку 4.1.

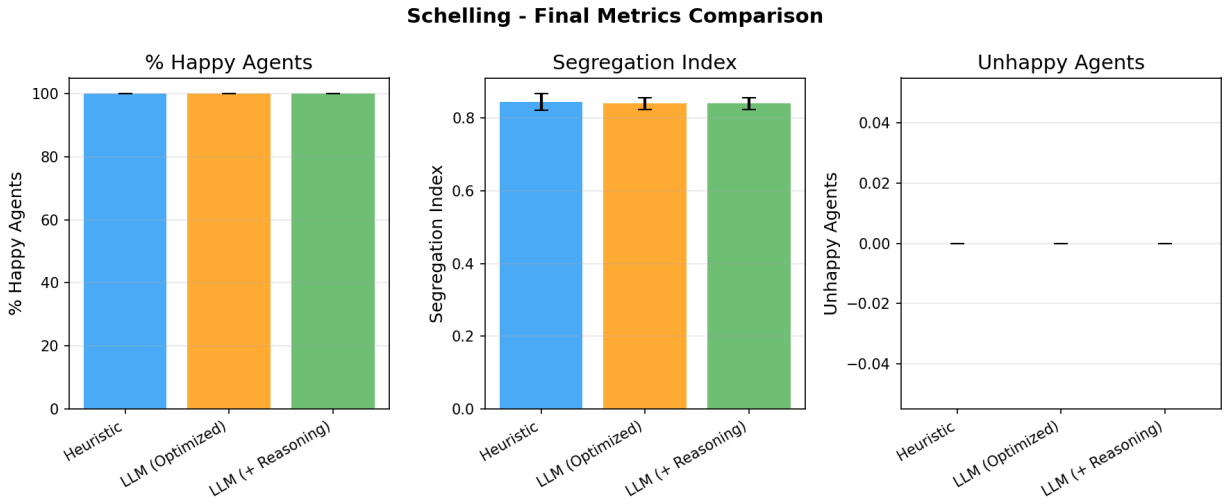

Рисунок 4.1 – Кінцеві метрики моделі Шеллінга: порівняння трьох профілів

Динаміку індексу сегрегації в часі показано на рисунку 4.2. Усі три профілі демонструють практично ідентичну траєкторію: індекс зростає від ~0.55 до ~0.84 протягом перших 7 кроків і стабілізується.

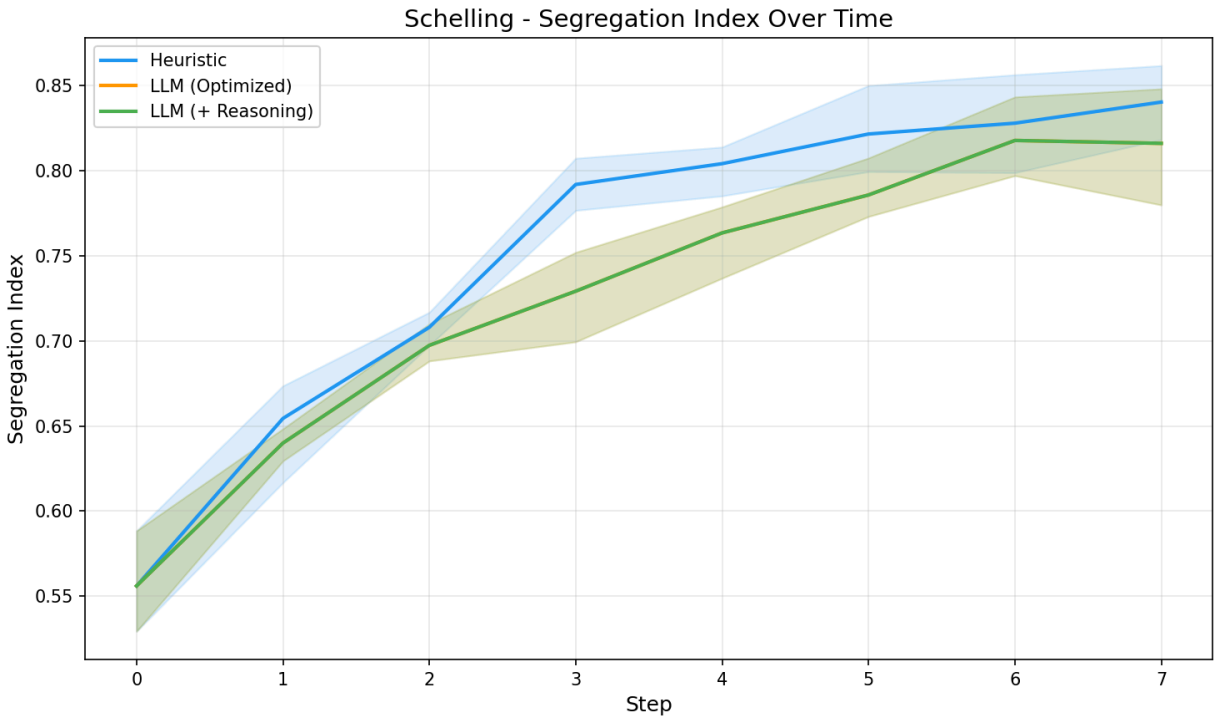

Рисунок 4.2 – Динаміка індексу сегрегації в часі

Успіх LLM пояснюється тим, що рішення агента є бінарним і спирається на чітку числову умову: якщо частка сусідів свого типу нижча за поріг 0.5 – переміщуйся. LLM легко інтерпретує цю умову: "My similar_fraction is 0.29, below my threshold of 0.50, therefore I will move." Індивідуальна раціональність агента (бажання мати схожих сусідів) збігається з механізмом колективного ефекту (сегрегацією).

4.2.2 Модель Sugarscape

У Sugarscape LLM-агенти з ланцюговими міркуваннями наблизились до евристичної виживаності, тоді як агенти без CoT зазнали значного погіршення. Результати наведено в таблиці 4.4.

Таблиця 4.4 – Результати Sugarscape по температурах (середнє за 3 seed)

№ п/п	*Профіль*	*Виживаність (t=0.0)*	*Виживаність (t=0.3)*	*Виживаність (t=0.7)*	*Gini (t=0.7)*
1	heuristic	21.67%	21.67%	21.67%	0.148
2	optimized	5.00%	8.33%	15.00%	0.100
3	optimized_with_reasoning	20.00%	23.33%	25.00%	0.200

Кінцеві метрики при температурі 0.7 показано на рисунку 4.3. При цій температурі агенти з reasoning досягають виживаності 25%, що перевищує евристичний результат (21.67%).

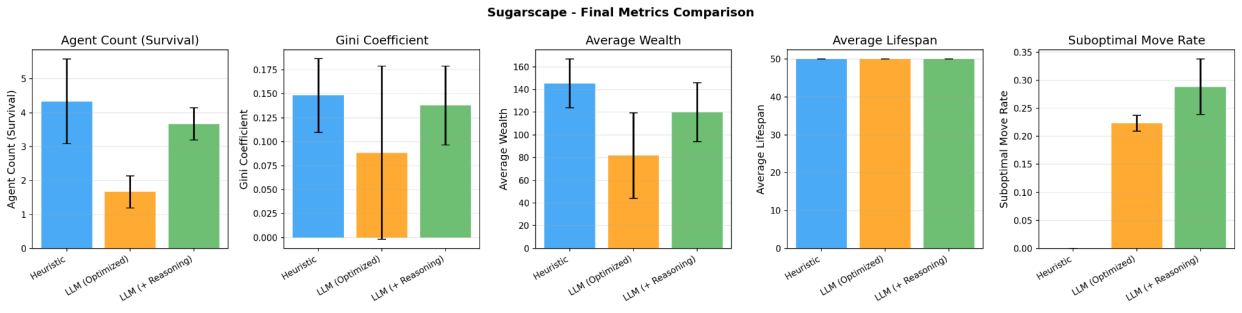

Рисунок 4.3 – Кінцеві метрики Sugarscape при температурі 0.7

Часовий ряд кількості агентів при температурі 0.7 показано на рисунку 4.4. Евристичні агенти (синя лінія) підтримують найвищу кількість протягом усієї симуляції. LLM з reasoning (зелена) відстає незначно. LLM без reasoning (помаранчева) різко падає і стабілізується біля нуля після кроку 35.

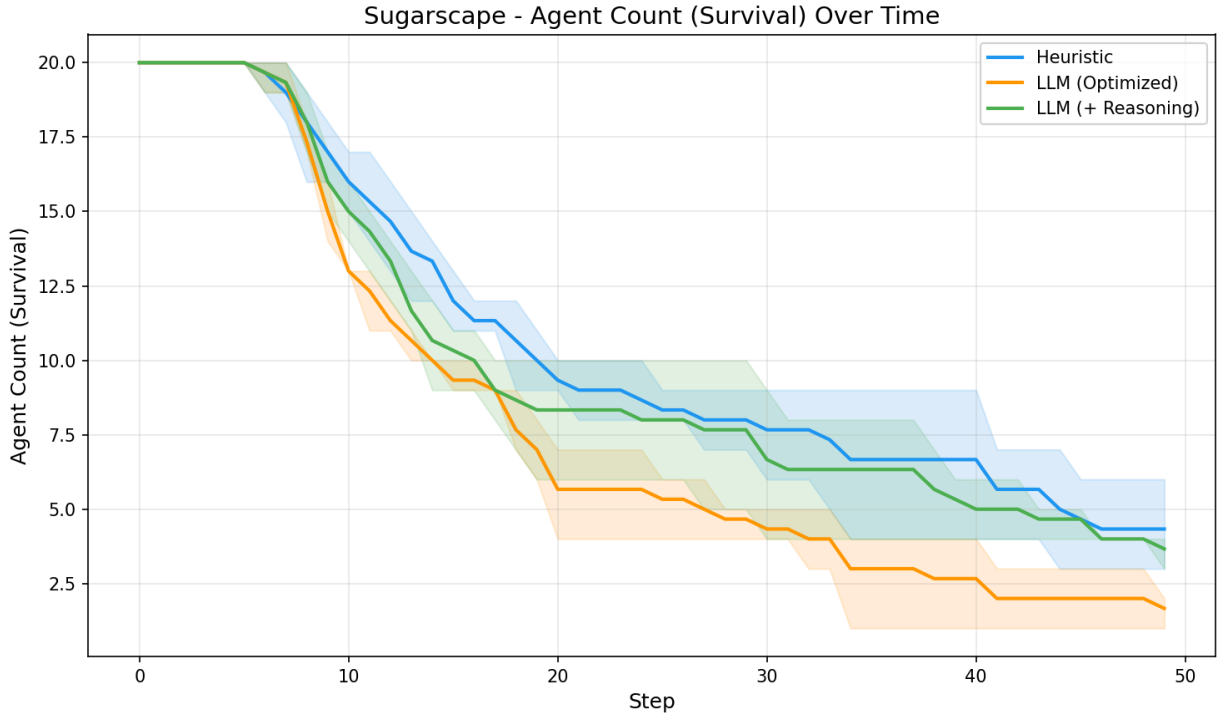

Рисунок 4.3 – Кількість агентів у часі

Ключовим є вплив CoT: при температурі 0.0 reasoning забезпечує 20.00% виживаності проти 5.00% без нього – покращення у 4 рази. Агент без reasoning часто обирає субоптимальні переміщення або залишається на місці, тоді як reasoning дозволяє сформулювати стратегію пошуку ресурсів.

4.2.3 Модель розподілу багатства Больцмана

У моделі Больцмана LLM-агенти систематично відмовляються від обміну, що руйнує розподіл Больцмана-Гіббса. Результати наведено в таблиці 4.5.

Таблиця 4.5 – Результати моделі Больцмана (середнє за 3 seed)

№ п/п	*Профіль*	*Gini (t=0.0)*	*Gini (t=0.3)*	*Gini (t=0.7)*	*Std Dev (t=0.0)*
1	heuristic	0.673	0.673	0.673	1.38
2	optimized	0.113	0.320	0.307	0.26
3	optimized_with_reasoning	0.433	0.453	0.480	0.81

Кінцеві метрики порівняно на рисунку 4.5. Евристичний Gini (~0.67) відповідає теоретичному значенню розподілу Больцмана-Гіббса [9]. LLM optimized демонструє Gini ~0.11 – майже ідеальна рівність, що свідчить про відмову від будь-якого обміну.

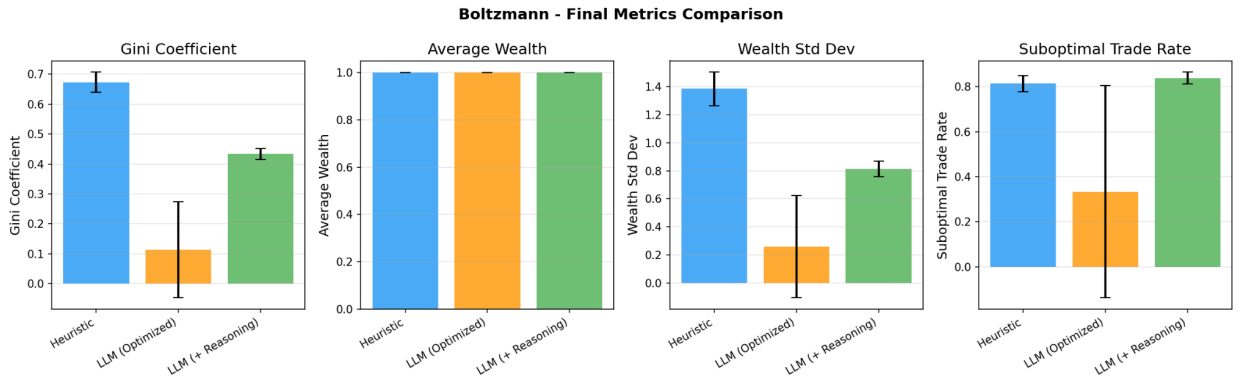

Рисунок 4.5 – Кінцеві метрики моделі Больцмана

Динаміку коефіцієнта Джині в часі показано на рисунку 4.6. Евристичний Gini (синя лінія) зростає до ~0.67 протягом 25 кроків та стабілізується – класична форма розподілу Больцмана-Гіббса. LLM optimized (помаранчева) практично не зростає. LLM з reasoning (зелена) повільно зростає до ~0.43.

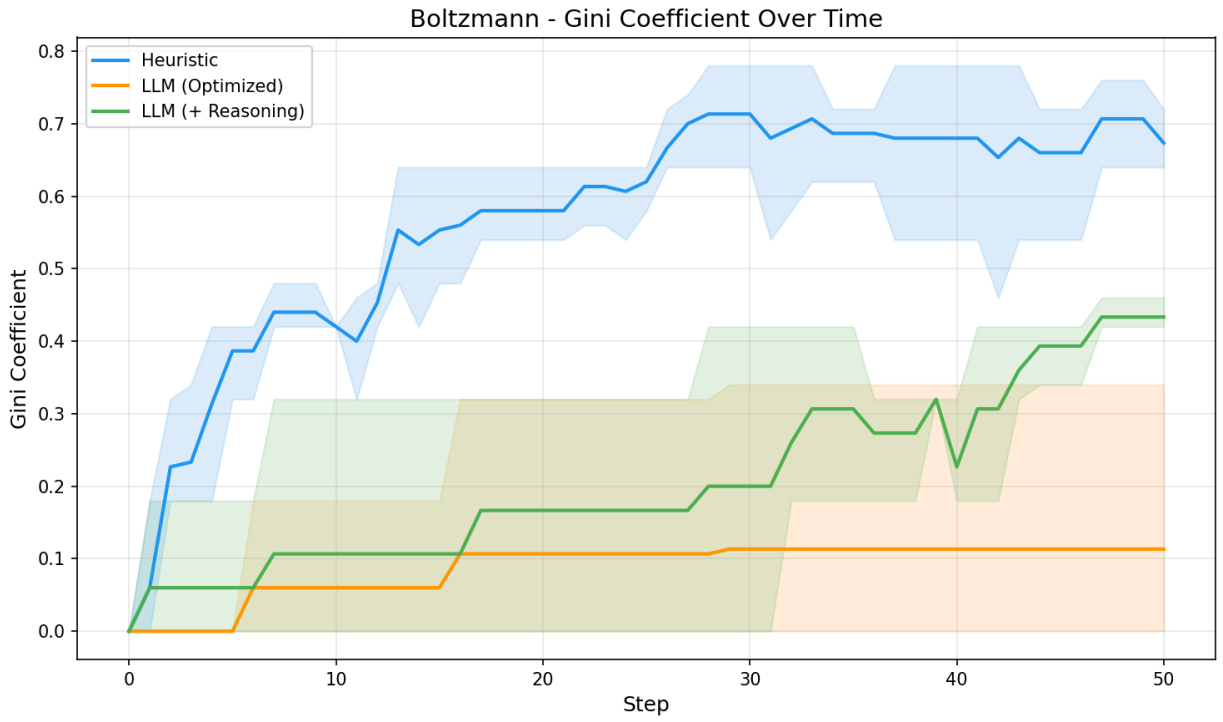

Рисунок 4.6 – Динаміка коефіцієнта Джині

Аналіз рішень виявив, що LLM optimized у 100% випадків обирає give=False. Агенти пояснюють відмову так: "I currently have 1 unit of wealth. Giving it away would leave me with 0 units, which is not beneficial." Це раціонально з індивідуальної точки зору, але руйнує колективний ефект, оскільки розподіл Больцмана-Гіббса вимагає саме безумовного обміну.

4.2.4 SIR-модель поширення вірусу

У SIR-моделі LLM-агенти практикують соціальну дистанцію, пригнічуючи епідемічну динаміку. Результати наведено в таблиці 4.6.

Таблиця 4.6 – Результати SIR-моделі (середнє за 3 seed)

№ п/п	*Профіль*	*Attack Rate*	*Susceptible (фін.)*	*Resistant (фін.)*
1	heuristic	41.7%	14.7	5.0
2	optimized	25.0%	16.7	3.3
3	optimized_with_reasoning	25.0%	16.7	3.3

Кінцеві метрики показано на рисунку 4.7. Рівень атаки LLM-агентів (25%) суттєво нижчий за евристичний (41.7%). Кількість стійких агентів у LLM значно менша – тобто менше агентів пройшли шлях інфікування і набули імунітету.

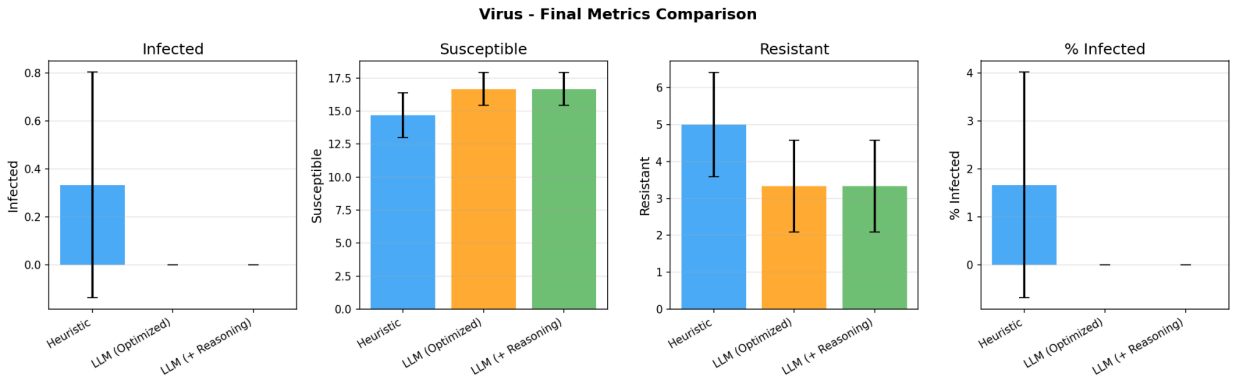

Рисунок 4.7 – Кінцеві метрики SIR-моделі

Часовий ряд кількості інфікованих порівняно на рисунку 4.8. Евристична симуляція (синя) демонструє класичну SIR-криву: пік захворюваності ~8 агентів на кроці 5, поступове зниження. LLM-профілі (помаранчева і зелена) монотонно спадають від початкового значення без жодного піку.

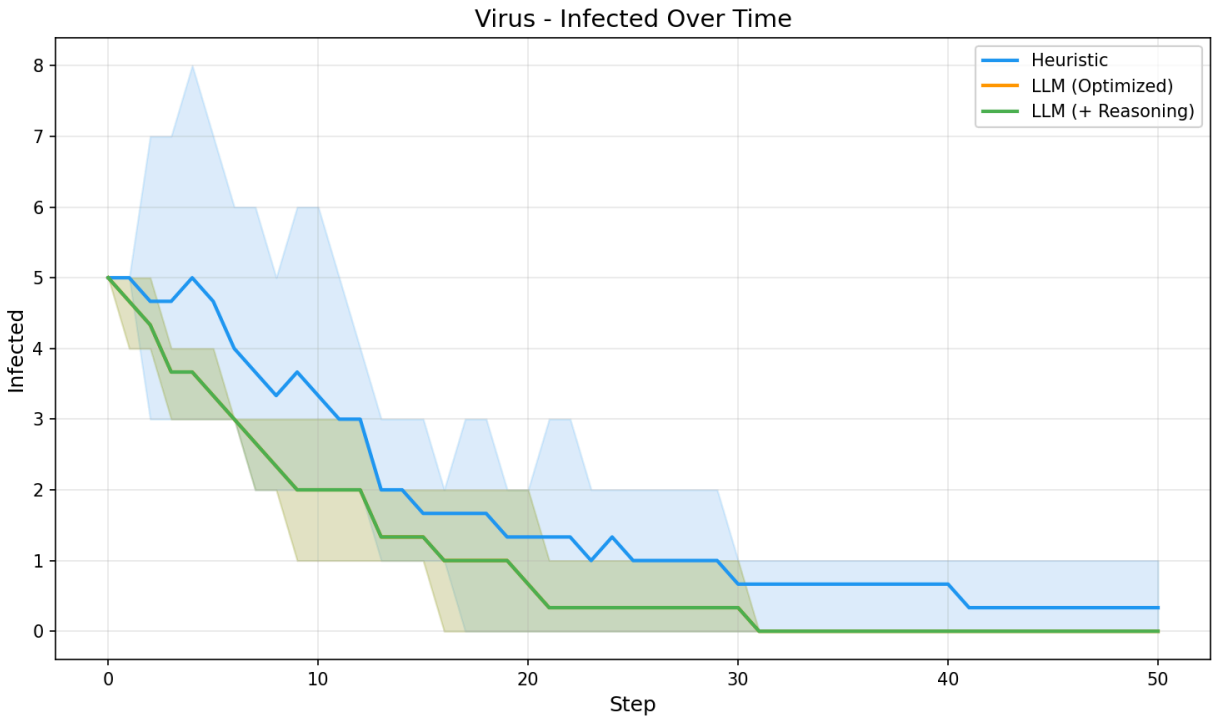

Рисунок 4.8 – Динаміка кількості інфікованих

Причина – "ефект відлюдника": LLM-агенти в 81.6–92.6% кроків залишаються на місці (heuristic: 0%). Навіть стійкі агенти уникають інфікованих сусідів – "To minimize exposure, I will move away from the infected neighbor" – хоча для стійких агентів зараження неможливе. Упередженість самозбереження є систематичною і не залежить від поточного стану агента.

4.2.5 Модель хижак-жертва

У моделі Вовк-Вівця вовки вимирають у всіх LLM-прогонах без винятку. Результати наведено в таблиці 4.7.

Таблиця 4.7 – Результати моделі Вовк-Вівця (середнє за 3 seed)

№ п/п	*Профіль*	*Вовки (фін.)*	*Вівці (фін.)*	*Усього (фін.)*	*Пік вовків*
1	heuristic	4.0	3.3	7.3	11.7
2	optimized	0.0	4.7	4.7	10.0
3	optimized_with_reasoning	0.0	4.0	4.0	10.0

Кінцеві метрики показано на рисунку 4.9. Стовпчик "Wolves" для обох LLM-профілів дорівнює нулю, тоді як евристика підтримує популяцію у ~4 вовки.

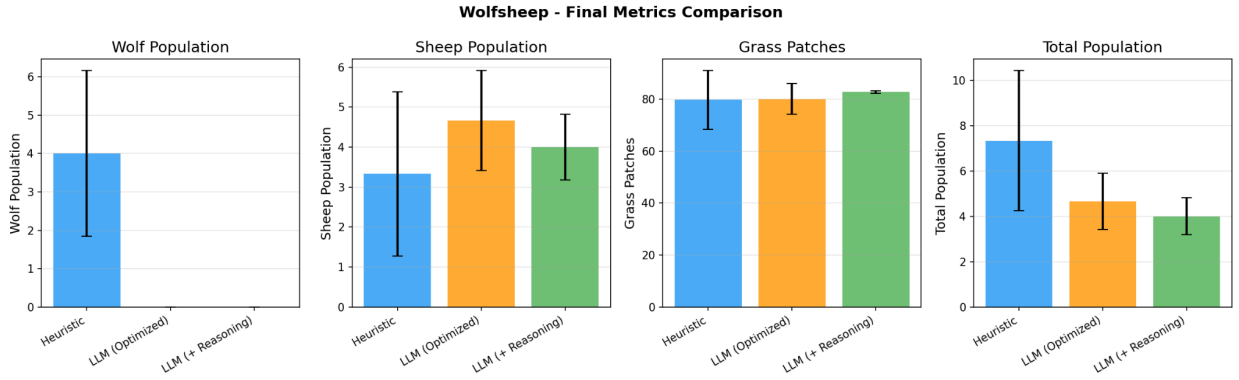

Рисунок 4.9 – Кінцеві метрики моделі Вовк-Вівця

Динаміку чисельності вовків показано на рисунку 4.10. Евристика (синя) демонструє коливання 3–12 особин упродовж 50 кроків – класичну осциляцію

Лотки-Вольтерри. Обидва LLM-профілі монотонно спадають від 10 до 0 на кроці 20–25 без жодного відновлення.

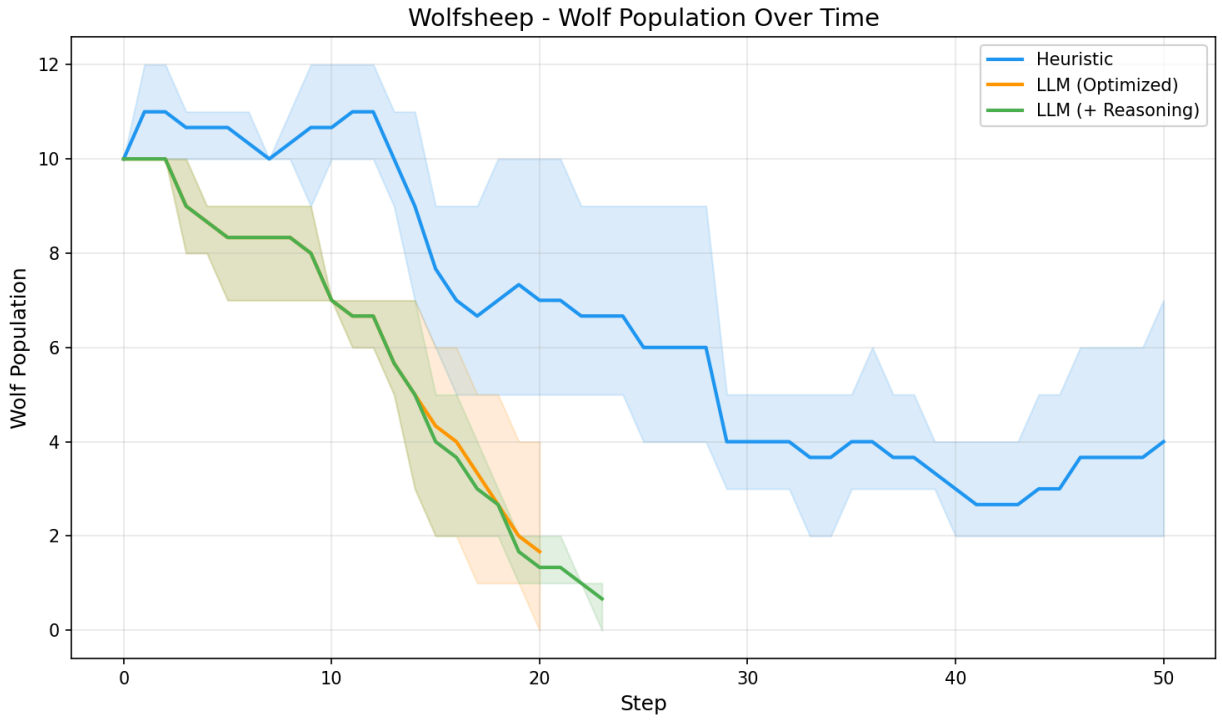

Рисунок 4.10 – Динаміка чисельності вовків у часі

Аналіз рішень розкриває причину: reproduce=False домінує у ~1100 рішень з 1500 (73%). Вовки пояснюють відмову від розмноження: "I need to conserve energy for future actions." При цьому вовки залишаються на місці у 40.8% кроків, тоді як евристика передбачає активний пошук їжі. Поєднання відмови від розмноження та пасивності призводить до необоротного вимирання: вовки поступово помирають від голоду, не відновлюючи популяцію.

4.3 Крос-симуляційний аналіз

Результати п'яти симуляцій демонструють чітку закономірність: ефективність LLM визначається тим, чи збігається індивідуальна раціональність агента з механізмом колективного ефекту. Зведене ранжування симуляцій за відповідністю LLM наведено в таблиці 4.8.

Таблиця 4.8 – Ранжування симуляцій за відповідністю LLM-агентів

№ п/п	*Ранг*	*Симуляція*	*Відповідність LLM*	*Ключова проблема*
1	1	Schelling	Ідеальна	Індивідуальна раціональність = механізм сегрегації
2	2	Sugarscape	Висока (з CoT)	CoT відновлює 4× виживаності
3	3	Boltzmann	Низька	Раціональність забороняє безумовний обмін
4	4	Virus SIR	Низька	Соціальна дистанція пригнічує SIR-динаміку
5	5	Вовк-Вівця	Відсутня	Відмова від розмноження → вимирання хижаків

Зведену таблицю упередженостей, виявлених у поведінці LLM-агентів, наведено в таблиці 4.9.

Таблиця 4.9 – Ранжування симуляцій за відповідністю LLM-агентів

№ п/п	*Упередженість*	*Прояв*	*Уражені симуляції*
1	Уникнення ризику	Залишатись на місці, уникати контакту	Virus (92.6%), Wolf (40.8%)

Кінець таблиці 4.9

№ п/п	*Упередженість*	*Прояв*	*Уражені симуляції*
2	Накопичення ресурсів	Відмова від передачі багатства	Boltzmann (100%)
3	Уникання розмноження	Не розмножуватись, щоб "зберегти енергію"	Wolf-Sheep (73%)
4	Уникання взаємодії	interact=False навіть для стійких агентів	Virus (97%)

4.3.1 Вплив ланцюгових міркувань

CoT стабільно покращує результати, але не може подолати фундаментальне неузгодження. Зведений ефект CoT наведено в таблиці 4.10.

Таблиця 4.10 – Вплив ланцюгових міркувань

№ п/п	*Симуляція*	*Ключова метрика*	*Optimized*	*Optimized + reasoning*	*Зміна*
1	Sugarscape	Виживаність	5.00%	20.00%	+4.0×
2	Boltzmann	Gini	0.113	0.433	+3.8×
3	Schelling	Segregation Index	0.840	0.840	0 (вже ідеально)

Кінець таблиці 4.10

№ п/п	*Симуляція*	*Ключова метрика*	*Optimized*	*Optimized + reasoning*	*Зміна*
4	Virus SIR	Attack Rate	25.0%	25.0%	0
5	Вовк-Вівця	Фінальна популяція	4.7	4.0	−15% (гірше)

Примітне спостереження: у моделі Вовк-Вівця reasoning погіршує результат. Агенти з CoT залишаються на місці частіше (40.8% проти 2.4% без reasoning), раціоналізуючи бездіяльність розгорнутим поясненням.

4.3.2 Вплив температури

Температура діє лише на неперервні рішення з м'якими порогами; для бінарних рішень вона неефективна. Зведені дані наведено в таблиці 4.11.

Таблиця 4.11 – Чутливість до температури (профіль optimized_with_reasoning)

№ п/п	*Симуляція*	*Ключова метрика*	*Temp 0.0*	*Temp 0.3*	*Temp 0.7*	*Чутливість*
1	Schelling	Segregation Index	0.840	0.839	0.866	Відсутня
2	Sugarscape	Виживаність	20.00%	23.33%	25.00%	Помірна
3	Boltzmann	Gini	0.433	0.453	0.480	Низька

Кінець таблиці 4.11

№ п/п	*Симуляція*	*Ключова метрика*	*Temp 0.0*	*Temp 0.3*	*Temp 0.7*	*Чутливість*
4	Virus SIR	% нерухомих агентів	81.6%	57.4%	49.6%	Висока
5	Вовк-Вівця	Фінальна кількість вовків	0.0	0.0	0.0	Відсутня

4.4 Аналіз отриманих результатів

Сукупність результатів дозволяє сформулювати парадокс раціональності й емерджентності: багато класичних ABM-ефектів виникають саме тому, що агенти поводяться ірраціонально з індивідуальної точки зору.

Розподіл Больцмана-Гіббса [9] вимагає безумовного і випадкового обміну – дії, яка ніколи не є раціональною для окремого агента. SIR-динаміка [10] потребує вільного пересування без уникання контактів – поведінки, що підвищує індивідуальний ризик. Осциляції Лотки-Вольтерри [11, 12] неможливі без ймовірнісного розмноження незалежно від поточного рівня енергії. LLM, навчена на текстах людської культури, де самозбереження є нормою, систематично відхиляє ці ірраціональні дії. Це не помилка промпт-інжинірингу, а фундаментальне обмеження, що походить із тренувального корпусу.

Навпаки, у моделях, де індивідуальна раціональність збігається з механізмом колективного ефекту, LLM працює бездоганно. Шеллінг: агент хоче мати схожих сусідів – сегрегація виникає сама. Sugarscape: агент хоче максимізувати ресурс – з reasoning він знаходить багаті клітини ефективніше за просте правило. У цих випадках LLM не лише відтворює евристику, але й потенційно перевершує її (25.00% проти 21.67% у Sugarscape при температурі 0.7).

LLM-агентів доцільно використовувати у симуляціях, де емерджентність виникає з індивідуально раціональних рішень, – і поєднувати з ланцюговими міркуваннями та помірною температурою (0.3–0.7). Для симуляцій, що потребують ірраціональної поведінки, LLM не є прийнятною заміною без явного примусу через промпт або гібридних підходів (змішані популяції евристичних та LLM-агентів).

Висновки до розділу 4

Проведено 21 прогін для кожної з п'яти симуляцій. LLM-агенти повністю відтворили евристичний результат лише у моделі Шеллінга; частково – у Sugarscape з CoT і температурою 0.7; зазнали провалу у моделях Больцмана, SIR і Вовк-Вівця через систематичну упередженість самозбереження. CoT забезпечує до 4× покращення там, де агент здатен сформулювати раціональну стратегію, але не долає фундаментального конфлікту між індивідуальною раціональністю та колективною динамікою.

ВИСНОВКИ

У роботі проведено систематичне кількісне порівняння евристичних та LLM-агентів у п'яти класичних мультиагентних симуляціях – Sugarscape, моделі Больцмана, моделі Шеллінга, SIR-моделі поширення вірусу та моделі Вовк-Вівця. Для кожної симуляції реалізовано евристичну стратегію як детерміновану базову лінію та LLM-стратегію на основі у двох варіантах – без ланцюгових міркувань і з ними. Порівняльні експерименти проводились за трьома рівнями температури та трьома незалежними генераторами псевдовипадкових чисел, що забезпечило статистичну надійність результатів.

Отримані результати дозволяють дати чітку відповідь на центральне дослідницьке питання. LLM-агенти здатні відтворювати емерджентну колективну поведінку класичних ABM-симуляцій лише тоді, коли механізм колективного ефекту збігається з індивідуальною раціональністю агента. У моделі Шеллінга, де сегрегація виникає саме через те, що кожен агент прагне до бажаного сусідства, LLM відтворила евристичний результат із майже ідеальною точністю: 100% задоволених агентів та індекс сегрегації 0.840 проти 0.844 у евристичної базової лінії. У Sugarscape, де агент раціонально максимізує ресурс, LLM-агенти з ланцюговими міркуваннями досягли виживаності 25% при температурі 0.7, що навіть перевищило евристичний результат (21.67%).

Натомість у трьох симуляціях LLM-агенти зазнали системного провалу. У моделі Больцмана коефіцієнт Джині при температурі 0.0 склав 0.113 замість еталонних 0.673: LLM-агенти у 100% випадків відмовлялись від передачі багатства, раціонально пояснюючи це недоцільністю втрати ресурсу. У SIR-моделі рівень атаки знизився до 25% проти 41.7% у евристичного агента через те, що LLM-агенти практикували соціальну дистанцію у 81.6–92.6% кроків; характеристична епідемічна крива не сформувалась жодного разу. У моделі Вовк-Вівця вовки вимерли в усіх 18 LLM-прогонах без винятку: відмова

від розмноження заради «збереження енергії» (73% рішень) унеможливила відтворення класичних осциляцій Лотки-Вольтерри.

Спільною причиною цих провалів є упередженість самозбереження – системна тенденція LLM уникати дій, що підвищують індивідуальний ризик або зменшують поточний ресурс, навіть коли такі дії є необхідними для виникнення колективних ефектів. Ця упередженість не є артефактом промпт-інжинірингу: вона закладена під час навчання на людських текстах, де самозбереження є культурною нормою. Виявлений феномен сформульовано як парадокс раціональності й емерджентності: низка класичних ABM-ефектів виникає саме тому, що агенти поводяться «ірраціонально» з індивідуальної точки зору, а LLM систематично відхиляє таку поведінку.

Ланцюгові міркування (CoT) покращують результати там, де агент здатен сформулювати раціональну стратегію, але не усувають фундаментальний конфлікт. У Sugarscape CoT забезпечує 4-кратне зростання виживаності (з 5% до 20% при температурі 0.0), у моделі Больцмана – 3.8-кратне зростання коефіцієнта Джині. Водночас у моделі Вовк-Вівця CoT погіршує результат на 15%: агенти з розгорнутим обґрунтуванням раціоналізують бездіяльність детальніше, але не ефективніше. Температура LLM ефективна лише для симуляцій із м'якими порогами рішень: у SIR-моделі підвищення температури з 0.0 до 0.7 знижує частку нерухомих агентів з 93% до 50%, тоді як для Вовк-Вівця та Шеллінга вплив температури відсутній.

На основі отриманих результатів сформульовано практичні рекомендації щодо застосування LLM у мультиагентних симуляціях. LLM-агентів доцільно використовувати у симуляціях, де емерджентна поведінка виникає з індивідуально раціональних рішень; оптимальним є поєднання профілю з ланцюговими міркуваннями та помірною температурою (0.3–0.7). Для симуляцій, що вимагають «ірраціональної» поведінки, LLM не є прийнятною прямою заміною евристиці без явного примусу через системний промпт або гібридних підходів зі змішаними популяціями агентів. Обчислювальна вартість

LLM-агентів у 1500–4500 разів вища за евристичну базу, що необхідно враховувати при плануванні масштабних симуляцій.

Перспективами подальших досліджень є: вивчення гібридних популяцій із різними частками LLM- та евристичних агентів для визначення порогів, після яких упередженість самозбереження порушує колективну динаміку; тестування інших LLM-провайдерів на тих самих сценаріях для перевірки узагальненості виявленої упередженості; дослідження архітектур із короткотерміновою пам'яттю, що дозволяють агенту враховувати динаміку попередніх кроків.

ПЕРЕЛІК ПОСИЛАНЬ

1. 1. Epstein, Joshua M., Axtell, Robert. Growing Artificial Societies: Social Science from the Bottom Up. Washington: Brookings Institution Press, 1996. 228 p.
2. 2. Schelling, Thomas C. Dynamic Models of Segregation. Journal of Mathematical Sociology. 1971. Vol. 1, No. 2. P. 143–186. DOI: https://doi.org/10.1080/0022250X.1971.9989794
3. 3. Achiam, Josh et al. GPT-4 Technical Report. arXiv:2303.08774, 2023. DOI: https://doi.org/10.48550/arXiv.2303.08774
4. 4. Park, Joon Sung et al. Generative Agents: Interactive Simulacra of Human Behavior. In Proceedings of UIST 2023. arXiv:2304.03442. DOI: https://doi.org/10.1145/3586183.3606763
5. 5. Gardner, Martin. Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game "Life". Scientific American. 1970. Vol. 223. P. 120–123. DOI: https://doi.org/10.1038/scientificamerican1070-120
6. 6. Reynolds, Craig W. Flocks, Herds, and Schools: A Distributed Behavioral Model. ACM SIGGRAPH Computer Graphics. 1987. Vol. 21, No. 4. P. 25–34. DOI: https://doi.org/10.1145/37402.37406
7. 7. Axelrod, Robert. The Evolution of Cooperation. New York: Basic Books, 1984. 241 p.
8. 8. Mesa Development Team. Mesa: Agent-based modeling in Python 3+. URL: https://github.com/projectmesa/mesa
9. 9. Dragulescu, Adrian A., Yakovenko, Victor M. Statistical mechanics of money. The European Physical Journal B. 2000. Vol. 17, No. 4. P. 723–729. DOI: https://doi.org/10.1007/s100510070114
10. 10. Kermack, William O., McKendrick, Anderson G. A Contribution to the Mathematical Theory of Epidemics. Proceedings of the Royal Society of

London A. 1927. Vol. 115, No. 772. P. 700–721. DOI: https://doi.org/10.1098/rspa.1927.0118

11. 11. Lotka, Alfred J. Elements of Physical Biology. Baltimore: Williams and Wilkins, 1925. 460 p.
12. 12. Volterra, Vito. Fluctuations in the Abundance of a Species considered Mathematically. Nature. 1926. Vol. 118. P. 558–560. DOI: https://doi.org/10.1038/118558a0
13. 13. Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Łukasz, Polosukhin, Illia. Attention Is All You Need. In Advances in Neural Information Processing Systems (NeurIPS). 2017. arXiv:1706.03762. DOI: https://doi.org/10.48550/arXiv.1706.03762
14. 14. Brown, Tom et al. Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems (NeurIPS). 2020. arXiv:2005.14165. DOI: https://doi.org/10.48550/arXiv.2005.14165
15. 15. Wei, Jason et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Advances in Neural Information Processing Systems (NeurIPS). 2022. arXiv:2201.11903. DOI: https://doi.org/10.48550/arXiv.2201.11903
16. 16. Li, Xinyi et al. A Survey on LLM-driven Multi-Agent Systems: Recent Advances and New Frontiers. arXiv:2503.03800, 2025. URL: https://arxiv.org/abs/2503.03800
17. 17. Mesa-LLM Development Team. Mesa-LLM: LLM integration extension for Mesa. URL: https://github.com/projectmesa/mesa-llm
18. 18. Bonabeau, Eric. Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences. 2002. Vol. 99, Suppl. 3. P. 7280–7287. DOI: https://doi.org/10.1073/pnas.082080899
19. 19. OpenAI. GPT-4o mini – advancing cost-efficient intelligence. OpenAI Blog. 2024. URL: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

20. 20. Стеценко І. В., Дифучин А. О. Petri-Object Simulation: Technique and Software. Information, Computing and Intelligent Systems. 2020. № 1. DOI: https://doi.org/10.20535/2708-4930.1.2020.216057

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Лістинг програми розміщено за посиланням

URL: https://github.com/makarellav/thesis