

***БАКАЛАВРСКА
РОБОТА***

БР.ІІ – 01.00.00.000 ІІЗ

Група ІІ-22-4

Кантор Максим

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Кантор Максим Михайлович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка програмного забезпечення для моделювання поширення епідемій

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня М.М. Кантор
(підпис, ініціали та прізвище здобувача)

Науковий керівник Мельник Віталій Дмитрович, к.т.н., доцент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою

ІІЗ

доцент.

В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Кантор Максим Михайлович

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) "Розробка програмного забезпечення для моделювання поширення епідемій"

керівник проекту (роботи) Мельник Віталій Дмитрович, к.т.н., доцент

затверджені наказом вищого навчального закладу від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз вимог до програмного забезпечення

2. Аналіз вимог і постановка задачі

3. Проектування системи

4. Реалізація проекту

5. Тестування та впровадження

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

1 Топологія SAL завдань та вузлів (рис. 2.2, ст. 32)

2 Стек мережі SAL (рис. 2.3, ст. 38)

3 Схематичне зображення логічної структури моделі K-SEIR-Sim (рис. 3.14, ст. 71)

4 Схематичне зображення інтерфейсу користувача під час запуску програми (рис. 3.16, ст. 73)

5 Модельний аналіз ситуації з COVID-19 (рис. 3.21, ст. 80)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання _____ 2026 р. _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	17.01.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	21.02.2026	виконано
3	Побудова моделі або алгоритму власного рішення	01.03.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	21.04.2026	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	02.05.2026	виконано

Студент _____ Кантор М.М.
(підпис)

Керівник роботи _____ Мельник В.Д.
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 87 сторінок, 26 рисунків, 1 таблиця, список використаних джерел із 40 найменувань.

Метою роботи є дослідження методів моделювання поширення епідемій та розробка програмного забезпечення для симуляції епідемічних процесів.

Об'єкт дослідження: процеси поширення епідемій та методи їх комп'ютерного моделювання.

Предмет дослідження: технології, методи та інструменти розробки програмного забезпечення для моделювання епідемій, зокрема математичні моделі, агентно-орієнтовані підходи та засоби аналізу біомедичних даних.

Результати дослідження: проведено аналіз сучасних методів моделювання епідемій, розглянуто підходи до аналізу статистичних даних та реалізовано програмне забезпечення для симуляції епідемічних процесів..

У першому розділі розглянуто теоретичні основи математичного та комп'ютерного моделювання епідемій, методи імітації поширення інфекційних захворювань та сучасні підходи до прогнозування епідемічних процесів.

У другому розділі досліджено методи та інструменти розробки програмного забезпечення для моделювання епідемій, зокрема аналіз біомедичних даних, алгоритми моделювання та агентно-орієнтована структура.

У третьому розділі розглянуто реалізацію симуляційної системи, аналіз статистичних даних щодо COVID-19, а також оцінку ефективності програмного забезпечення та результатів моделювання.

Висновок: використання сучасних методів моделювання епідемій та програмних засобів симуляції дозволяє підвищити точність прогнозування поширення інфекційних захворювань.

КЛЮЧОВІ СЛОВА: МОДЕЛЮВАННЯ ЕПІДЕМІЙ, COVID-19, СИМУЛЯЦІЯ, АГЕНТНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ, БІОМЕДИЧНІ ДАНІ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОГНОЗУВАННЯ, МАТЕМАТИЧНІ МОДЕЛІ, ЕПІДЕМІОЛОГІЯ.

ANNOTATION

The bachelor's thesis contains 87 pages, 26 figures, 1 tables, and a reference list containing 40 sources.

The aim of the work is to study epidemic spread modeling methods and to develop software for the analysis, forecasting, and simulation of epidemic processes.

Research object: epidemic spread processes and methods of their computer modeling.

Research subject: technologies, methods, and tools for developing software for epidemic modeling, including mathematical models, agent-based approaches, and biomedical data analysis tools.

Research results: modern epidemic modeling methods were analyzed, mathematical and agent-based models of infectious disease spread were studied, and software for epidemic process simulation was implemented.

The first section describes the theoretical foundations of mathematical and computer epidemic modeling, methods of infectious disease spread simulation, and modern approaches to forecasting epidemic processes.

The second section analyzes methods and tools for developing epidemic modeling software, including biomedical data analysis, modeling algorithms, and agent-based simulation structures.

The third section covers the implementation of the simulation system, analysis of COVID-19 statistical data, and evaluation of software effectiveness and modeling results.

Conclusion: the use of modern epidemic modeling methods and simulation software improves the accuracy of forecasting infectious disease spread, enhances epidemic process analysis, and supports decision-making in the healthcare sector.

KEY WORDS: EPIDEMIC MODELING, COVID-19, SIMULATION, AGENT-BASED MODELING, BIOMEDICAL DATA, SOFTWARE DEVELOPMENT, FORECASTING, MATHEMATICAL MODELS, EPIDEMIOLOGY.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЮВАННЯ ПОШИРЕННЯ ЕПІДЕМІЙ	9
1.1 Основи математичного та комп'ютерного моделювання епідемій	9
1.2 Методи моделювання та імітації поширення епідемій	13
1.3 Сучасні підходи до прогнозування та моделювання епідемічних процесів	17
1.4 Висновок по розділу	21
РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЕПІДЕМІЙ	24
2.1 Аналіз та обробка біомедичних даних для моделювання епідемій	24
2.2 Методи та алгоритми моделювання епідемічних процесів	39
2.3 Агентно-орієнтовані структури моделювання поширення епідемій	44
2.4 Висновок по розділу	53
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	54
3.1 Проектування симуляційної системи моделювання епідемій	54
3.2 Аналіз статистичних даних щодо поширення COVID-19	60
3.3 Аналіз результатів моделювання та оцінка ефективності програмного забезпечення	70
3.4 Висновок по розділу	85
ВИСНОВКИ	86
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	88

					БР.ІП – 01.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка програмного забезпечення для моделювання поширення епідемій Пояснювальна записка	Літ.	Арк.	Акрушів
Розроб.		Кантор М.М.						
Перевір.		Мельник В.Д.					7	
Реценз.		Саманів Л.В.				ІФНТУНГ ІП-22-4		
Н. Контр.		Піх М.М.						
Затверд.		Бандура В. В.						

ВСТУП

У сучасному світі поширення інфекційних захворювань становить одну з найбільших загроз для суспільства, систем охорони здоров'я та глобальної економіки. Пандемії та епідемії, такі як COVID-19, SARS, H1N1 та інші інфекційні захворювання, продемонстрували необхідність оперативного аналізу епідемічних процесів, прогнозування розвитку захворювань та прийняття ефективних рішень для мінімізації наслідків їх поширення. У зв'язку з цим особливого значення набуває використання сучасних інформаційних технологій і програмного забезпечення для моделювання епідемій, що дозволяє аналізувати поведінку епідемічних процесів, оцінювати вплив різних факторів та прогнозувати можливі сценарії розвитку ситуації.

Сучасні методи математичного та комп'ютерного моделювання дозволяють створювати симуляційні моделі поширення інфекційних захворювань, враховуючи демографічні показники, мобільність населення, соціальні контакти та інші фактори, що впливають на динаміку епідемій. Особливо актуальним є застосування агентно-орієнтованих моделей та аналізу великих обсягів біомедичних даних, що дозволяє підвищити точність прогнозування та адаптувати моделі до реальних умов.

Актуальність теми зумовлена необхідністю створення ефективного програмного забезпечення для моделювання поширення епідемій, яке б забезпечувало аналіз статистичних даних, прогнозування розвитку епідемічних процесів та підтримку прийняття рішень у сфері охорони здоров'я. У контексті глобалізації та високої мобільності населення створення сучасних програмних засобів моделювання епідемій є важливим кроком для підвищення ефективності моніторингу та управління епідемічними процесами.

Метою роботи є розробка програмного забезпечення для моделювання поширення епідемій, яке забезпечує аналіз епідемічних процесів, прогнозування розвитку інфекційних захворювань та оцінювання ефективності заходів протидії поширенню епідемій.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Завданнями дослідження є аналіз предметної області та існуючих методів моделювання епідемій, дослідження математичних і агентно-орієнтованих моделей поширення інфекційних захворювань, реалізація основних модулів програмного забезпечення, тестування системи та оцінювання ефективності моделювання на основі статистичних даних щодо COVID-19.

Об’єктом дослідження є процеси поширення епідемій та їх комп’ютерне моделювання з використанням сучасних інформаційних технологій.

Предметом дослідження є методи, алгоритми та програмні засоби моделювання епідемічних процесів, зокрема математичні моделі, агентно-орієнтовані підходи, засоби аналізу біомедичних даних і технології реалізації симуляційних систем.

Методи дослідження включають аналіз наукових джерел для вивчення сучасних підходів до моделювання епідемій, порівняльний аналіз математичних моделей, моделювання структури програмної системи, експериментальний метод для реалізації та тестування програмного забезпечення, а також статистичний аналіз результатів симуляції епідемічних процесів.

Розроблене програмне забезпечення передбачатиме можливість моделювання поширення інфекційних захворювань, аналізу статистичних даних, побудови прогнозів розвитку епідемій та оцінювання ефективності заходів протидії їх поширенню. Очікується, що впровадження системи дозволить підвищити ефективність аналізу епідемічних процесів та підтримати прийняття рішень у сфері охорони здоров’я.

Бакалаврська робота містить 87 сторінок, 26 рисунків, 1 таблиця, 3 розділи, список використаних джерел із 40 найменувань.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЮВАННЯ ПОШИРЕННЯ ЕПІДЕМІЙ

1.1 Основи математичного та комп'ютерного моделювання епідемій

Епідемія – це спалах хвороби, яка швидко та широко поширюється. Епідемії відіграли вирішальну роль в історії людства. Різні епідемії, такі як бубонна чума, пташиний грип, грип H5N1, свинячий грип H1N1 та SARS, століттями гальмували суспільство. Розуміння способів поширення хвороб має низку переваг для різних суспільств, таких як покращення систем охорони здоров'я, збільшення тривалості життя та зменшення впливу біологічної війни. Таким чином, для державних органів та установ охорони здоров'я дуже важливо розуміти, як поширюються хвороби, і, отже, як обмежити це поширення, вибираючи найкращі стратегії пом'якшення наслідків та соціальні стратегії. Прикладами стратегій пом'якшення наслідків є вакцинація, противірусне лікування та профілактика вдома. Стратегії соціального дистанціювання включають закриття шкіл, карантин та ізоляцію [1, 2].

Епідемії моделювали математично для розуміння їхньої динаміки. Ці моделі можуть варіюватися від кількох простих рівнянь до складних систем, які необхідно моделювати на суперкомп'ютерах [3]. Аналіз епідемії зазвичай вимагає моделювання біологічних систем, таких як віруси, бактерії, паразити та господарі. Математичні моделі епідемій можна класифікувати на дві категорії: господар та поширення. Моделі класу *господарів* зосереджені на впливі хвороби на окрему людину, тоді як моделі класу *поширення* прогнозують, як хвороби поширюються серед групи людей [2].

Наскільки нам відомо, останнім часом не проводилося досліджень щодо моделювання та симуляції поширення епідемій. Досліджено практику агентно-орієнтованого моделювання (ABM) з січня 1998 року по липень 2008 року. Узагальнюючи останні інструменти ABM. Обидві роботи досліджували ABM загалом, не застосовуючи його до поширення епідемій зокрема. Крім того, було розглянуто кілька досліджень чотирьох захворювань з різними методологіями:

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

кір (моделі плям), ящур (моделі дистанційної передачі), пандемічний грип (багатогрупові моделі) та віспа (мережеві моделі). Запропонованої інтуїтивне введення в процес передачі захворювання та те, як його математичне представлення можна використовувати для аналізу динаміки спостережуваних епідемій. Таким чином, у цій роботі ми узагальнюємо та аналізуємо останні зусилля щодо моделювання та симуляції поширення епідемій, щоб надати огляд різних моделей цієї парадигми, що з'явилися останнім часом.

Передові технології моделювання, які включають найсучасніші знання вірусології, імунології, епідеміології, вакцин, противірусних препаратів та громадського здоров'я, нещодавно вийшли на перший план у визначенні ефективних стратегій пом'якшення наслідків захворювань і все частіше використовуються експертами у галузі громадського здоров'я для вивчення як епідеміології, так і патогенезу. Моделювання еволюціонувало для роботи з даними, які є більш гетерогенними, менш грубими (на основі рівня спільноти або окремої особи) та більш складними (спільні просторові, часові та поведінкові взаємодії). Ця еволюція характеризується парадигмою агентно-орієнтованої моделі (ABM), що представляє собою розподілені за ґратками колекції автономних суб'єктів прийняття рішень (агентів), взаємодія яких розкриває динаміку та емерджентні властивості досліджуваного спалаху інфекційного захворювання. Гнучкість ABM дозволяє ефективно представляти додаткові взаємодії між індивідами, що характеризуються локалізованими властивостями, та популяціями на глобальному рівні.

Однак, гнучкість приходить зі складністю; отже, програмна реалізація ABM вимагає суворіших вимог до розробки програмного забезпечення, ніж звичайні (і простіші) моделі поширення та контролю інфекційних захворювань, особливо щодо відтворюваності результатів, виявлення помилок та управління системою. Відтворюваність результатів є проблемою, оскільки емерджентні властивості не піддаються аналітичному контролю, що ще більше посилюється тонкими та важко виявлюваними помилками в логіці алгоритмів та розробці програмного забезпечення. Системне управління програмним забезпеченням, що моделює популяції/окремих істот та біологічні/фізичні взаємодії, є серйозною

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

проблемою, оскільки реалізація включатиме розподілені (паралелізовані), нелінійні, складні та численні процеси, що працюють узгоджено. Враховуючи ці проблеми, очевидно, що будь-яка реалізація АВМ повинна відповідати трьом цілям: надійність, ефективність та адаптивність. Надійність передбачає стійкість, відтворюваність та достовірність отриманих результатів із заданими початковими умовами. Ефективність є важливою для своєчасного проведення численних експериментів (моделювання). Адаптивність також є необхідною вимогою для коригування системи АВМ у міру внесення змін до фундаментальних знань. Попередній досвід розробки програмного забезпечення та новітня література пропонують кілька рекомендацій, яких слід дотримуватися під час розробки АВМ. До них належать:

i. *Спіральна методологія.* Програмні системи АВМ вимагають швидкого розвитку з постійними змінами вимог користувачів та поступовим удосконаленням серії тестованих прототипів. Це вимагає спіральної методології розробки програмного забезпечення, починаючи з початкового прототипу та закінчуючи зрілим випуском програмного забезпечення АВМ, через поступову та ітеративну послідовність уточнених фаз вимог, проектування, впровадження та валідації.

ii. *Потоки діяльності.* Протягом усього життєвого циклу розробки програмного забезпечення знадобляться три паралельні та взаємодоповнюючі потоки діяльності (концептуальний, алгоритмічний та інтеграційний). Високорівневі аналітичні концепції АВМ стимулюють створення функціонально релевантних алгоритмів, які впроваджуються та тестуються, а в разі перевірки інтегруються в існуючу кодову базу. Зазвичай це вважається низхідним підходом, але в спіральній методології також актуальні міркування «знизу вгору». Наприклад, вибір з набору конкуруючих концептуальних представлень для моделі АВМ може бути зроблений на основі аналізу базових алгоритмів або продуктивності відповідних реалізацій.

iii. *Контроль версій.* За спіральної методології розробки має бути запроваджена стандартна для галузі стратегія контролю версій для ретельного аудиту змін, внесених до програмного забезпечення (включаючи зміни, що

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

стосуються обґрунтувань, архітекторів та дат).

iv. *Перевірка коду.* Оскільки код інтегрується в систему АВМ, критичні перевірки програмного забезпечення слід проводити регулярно, щоб переконатися, що реалізація програмного забезпечення правильно відображає функціональність та мету загальної АВМ.

v. *Валідація .* Необхідно розробити стратегію для регулярного та частого тестування програмної системи на наявність логічних помилок та помилок проектування. Наприклад, поведінку імітаційної моделі можна перевірити, порівнявши її вихідні дані з відомими аналітичними результатами для великомасштабних мереж. Валідація програмного забезпечення має бути актуальною та поширеною в усіх рекомендаціях (i)-(iv).

vi. *Стандартизовані засоби розробки програмного забезпечення .* Середовища математичного програмування, такі як Matlab, Mathematica та Maple, є чудовими інструментами розробки для швидкого створення прототипів АВМ. Однак проблеми з продуктивністю виникають, оскільки прототипи зростають у розмірі та складності, щоб стати програмними системами. Структура розробки повинна забезпечувати зручний посередник між цими інструментами створення прототипів та зрілими ефективними системами АВМ.

vii. *Системний детермінізм.* У паралельному або розподіленому середовищі відтворюваність результатів важко досягти в системах, що містять стохастичні компоненти. Тим не менш, системний детермінізм є вимогою, навіть якщо він виконується на іншому комп'ютерному кластері.

viii. *Профілювання системи.* Важливо спостерігати та оцінювати продуктивність частин системи під час її роботи. Наприклад, які компоненти виконуються часто; який їхній час виконання; чи розподілено навантаження на обробку між вузлами в комп'ютерному кластері?

Щоб дотримуватися цих рекомендацій та досягти вищезазначених цілей, ми розробили платформу розробки програмного забезпечення для агентно-орієнтованих методів лікування інфекційних захворювань. У наступному розділі цього розділу описується Scorpa, загальна бібліотека розробки додатків, розроблена нашою дослідницькою групою як високорозширюваний інтерфейс

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

прикладного програмування з повністю вбудованим транспортним рівнем, який повністю масштабований від окремих машин до розподілених кластерів по всій території сайту. Ця бібліотека була використана для реалізації платформи агентно-орієнтованого моделювання, детальна інформація про яку наведена в наступному розділі. Ми завершуємо розділом, що описує майбутню дослідницьку діяльність.

1.2 Методи моделювання та імітації поширення епідемій

Два основні типи моделей епідемій у масштабі популяції називаються стохастичними та детермінованими. Детерміновані моделі описують та пояснюють, що відбувається в середньому в масштабі популяції (макроскопічний підхід), тоді як стохастичні моделі дозволяють спостерігати за кожною особою в популяції на випадковій основі (мікроскопічний підхід). Останні використовуються, коли випадкові коливання або неоднорідності є важливими в популяціях. Існуючі детерміновані моделі вимагають менше даних і їх відносно легко налаштувати. Однак стохастичні моделі є складнішими та потребують багатьох симуляцій для отримання корисних і точних прогнозів [9]. Це дослідження зосереджено на стохастичному моделюванні. У цьому підрозділі ми підсумовуємо ряд методів моделювання поширення епідемій.

А. Моделі з відсіками

Найтиповішою математичною моделлю поширення хвороби є компартментна, або станова, модель, де час поділяється на періоди, і кожна особа класифікується до певного стану в кожному періоді. Найпростішою моделлю станів є модель «Сприйнятливі-Інфіковані-Одужані» (SIR). Популяція поділяється на три класи: Сприйнятливі (S), Інфіковані (I) та Одужані (R), як показано на рисунку 1.1. Людина, яка інфікується, переходить з класу S до класу I зі швидкістю, яка залежить від інфекційності вірусу та поширеності інфекції. Інфікована особа одужує та переходить до класу R, коли вона стає стійкою до будь-якої майбутньої інфекції [11].

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

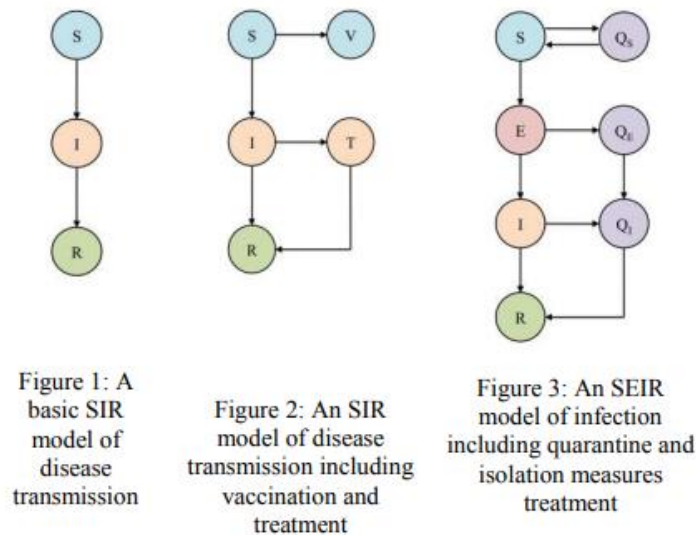


Рисунок 1.1 - Модель SEIR інфекції, включаючи карантин та ізоляційні заходи лікування

Розширення моделі SIR, яке включає два біомедичні втручання, показано на рисунку 1.2. Сприйнятлива особа (S) може бути вакцинована (V), а інфікована особа (I) може бути лікована противірусними препаратами (T), а потім переведена до стану одужання (S). Рисунок 1.3 показує модель «Сприйнятлива-Під впливом-Інфікована-Одужала» (SEIR) з розширенням, яке включає два поведінкові втручання: карантин (QS) та ізоляцію (QI).

Моделювання на основі рівнянь

Моделювання на основі рівнянь (EBM) є дуже поширеним методом моделювання поширення інфекційних захворювань. Модель EBM складається з набору рівнянь, які виражають взаємозв'язки між спостережуваними величинами. Оцінка цих рівнянь призводить до зміни цих спостережуваних величин з часом [12]. Таким чином, характеристика моделі SIR представлена наступними диференціальними рівняннями [13]:

$$\frac{dS}{dt} = -pSI, \frac{dI}{dt} = pSI - \gamma I, \frac{dR}{dt} = \gamma I, S + I + R = N \quad (1.1)$$

де $S = S(t)$ – кількість сприйнятливих осіб на момент часу t , $I = I(t)$ –

кількість інфікованих осіб на момент часу t , $R = R(t)$ – кількість осіб, що одужали на момент часу t , N – загальна чисельність усієї популяції, p – глобальний рівень передачі, а u – рівень одужання (або смертності).

Моделювання поширення хвороби за допомогою контакту. Мережі.

Контактні мережі зазвичай використовуються для моделювання епідемій. Контактна мережа моделює шанс або ймовірність того, що одна людина може заразити іншу людину. Для n людей будується матриця ймовірностей P розміром $n \times n$, де p_{ij} – ймовірність того, що інфікована людина i поширить хворобу на неінфіковану людину j [2]. Соціальні мережі представляють взаємозв'язки між особами популяції. Інші аспекти моделі, такі як щоденна діяльність кожної людини, інтегровані в соціальні мережі, щоб забезпечити конкретні випадки, коли особи контактують з інфікованою людиною i , отже, інфікуються хворобою [1].

У роботі була представлена ідея соціальних мереж малого світу, що характеризуються висококластерними з'єднаннями та короткими шляхами між вузлами. За цією роботою було опубліковано [15] опис безмасштабних мереж, що дотримуються степеневих законів розподілу зв'язності. У роботі було запропоновано використовувати рівняння середньої швидкості поля для представлення однорідних складних мереж. Вони використовували спрощене рівняння для представлення поведінки інфікованого вузла з часом у певних сценаріях захворювання. Для балансування відмінностей у щільності між вузлами вони представили вдосконалену версію та застосували її до моделі поширення епідемії відповідно до безмасштабної мережі ВА [17].

Агентно-орієнтоване моделювання

Підходи агент-орієнтованого моделювання (ABM) нещодавно були застосовані для вивчення процесів передачі епідемій там, де неагентні підходи моделювання не можуть бути застосовані [17]. Модель ABM складається з набору агентів, які інкапсулюють поведінку різних індивідів системи, а виконання полягає в емуляції цієї поведінки. ABM більше підходить для областей, що характеризуються високим ступенем локалізації та розподілу та регулюються дискретними рішеннями. Ці моделі більше підходять для областей,

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

де природною одиницею розкладання є індивід, а не спостережуване або рівняння. Внутрішня поведінка агента не обов'язково має бути видимою для решти системи. ЕВМ найприродніше застосовується до систем, які можна моделювати централізовано, і в яких динаміка контролюється фізичними законами, а не обробкою інформації [12]. У той час як ЕВМ припускає однорідність та ідеальне змішування в межах компартментів, АВМ може відображати гетерогенність між індивідами та контактними мережами. АВМ послаблює припущення щодо агрегації, але вимагає обчислювальних та когнітивних витрат, які можуть обмежувати аналіз чутливості та обсяг моделі [18]. Однією з причин популярності ЕВМ є наявність кількох інтуїтивно зрозумілих інструментів перетягування для побудови та аналізу моделей системної динаміки.

Прикладами використання методів, заснованих на агентах, у поширенні епідемії є роботи, в яких розроблені агентні моделі поширення ВІЛ. У роботі агентну модель використовували для вивчення того, як динамічна взаємодія між діяльністю людини впливає на пандемію пташиного грипу в Індонезії.

Базовий репродуктивний номер

Щоб визначити, чи має відбутися епідемія, чи хвороба зменшиться, розраховується базове репродуктивне число R_0 . R_0 – це порогова величина, яку можна визначити як очікувану кількість нових інфекцій, спричинених однією інфекцією в популяції, де всі особини сприйнятливі. Ця інфікована особина здійснює контакти з ΔY за одиницю часу, спричиняючи нові інфекції із середнім інфекційним періодом 1 рік. Отже, базове репродуктивне число становить

$$R_0 = \Delta Y / Y \quad (1.2)$$

Значення R_0 вимірює потенціал передачі хвороби, оскільки визначає швидкість поширення та співвідношення інфікованої популяції. Якщо $R_0 < 1$, інфікована особа може не передати інфекцію протягом інфекційного періоду, і тому інфекція зникає. Якщо $R_0 > 1$, у популяції виникне епідемія. Якщо $R_0 = 1$, хвороба стає ендемічною, тобто хвороба залишається в популяції з постійною

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

швидкістю, оскільки одна інфікована особа передає хворобу одній сприйнятливій [9].

1.3 Сучасні підходи до прогнозування та моделювання епідемічних процесів

Дослідження в галузі моделювання та симуляції поширення епідемій варіюються в широкому діапазоні залежно від типу модельованого захворювання, методики моделювання, моделі компартменту, інструменту моделювання та модельованого географічного розташування. У цьому розділі обговорюються та оцінюються найновіші підходи.

А. Агентно-орієнтоване моделювання та симуляція грипу

Розроблено агентну модель епідемій для типового китайського міста з використанням динамічної мережі контактів для моделювання поширення та контролю епідемій. Міські розташування поділено на чотири класи: домогосподарство, школа, робоче місце та місце розваг. Усі розташування далі поділено на підлокації, які називаються змішувальними групами. Кожна людина в місті представлена як агент у віртуальній спільноті та має контакт з іншими людьми, які проживають в одному домогосподарстві та змішувальній групі. Контакти класифіковано як звичайні контакти, що відбуваються в змішувальній групі, та випадкові контакти, що відбуваються між змішувальними групами.

Було розроблено пакет моделювання QAST, який використовує технологію паралельних обчислень для моделювання поширення інфекційних захворювань серед мільйонів людей та дослідження ефективності певної стратегії контролю. Модель була оцінена шляхом моделювання поширення грипу в громаді міста Сіань з населенням 30 000 осіб, які проживають у 10 252 домогосподарствах. Вони дійшли висновку, що політика контролю впливає на поведінку людей і, в свою чергу, може призвести до пом'якшення перебігу захворювання. Автори повідомили, що ця модель корисна для вивчення впливу контрзаходів проти інфекційних захворювань, а індивідуальні моделі є точнішими, ніж математичні моделі, для опису деталей великомасштабної

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

популяції.

Цей підхід передбачає, що людина переважно переміщується лише між своїм будинком та іншим призначеним їй місцем проживання. Наприклад, дитина шкільного віку переважно залишається вдома або в школі, до якої вона належить. Крім того, передбачається, що кожна людина перебуває в одному з дванадцяти різних станів, включаючи як стани, пов'язані із захворюванням, так і стани, пов'язані з втручанням, з припущенням експоненціально розподіленої тривалості кожного стану. Також передбачається, що за умови масової політики попередньої вакцинації 80% населення отримало вакцину.

в. Моделювання стратегій стримування пандемії 2009 року

Свинячий грип H1N1

У роботі, використовується стохастична симуляційна модель пандемічного грипу (свинячий грип H1N1 2009 року) для дослідження реалістичних стратегій втручання, які можна застосувати у відповідь на зростання спалахів. Симуляційна модель представляла типове північноамериканське місто середнього розміру та прогнозувала середні показники захворюваності та економічні витрати за різних стратегій втручання, включаючи реактивну вакцинацію з низьким охопленням, обмежене використання противірусних препаратів, руйнівну політику соціального дистанціювання та короткострокове закриття окремих шкіл. Модель базувалася на компартментній моделі SIR. Було зроблено висновок, що певні комбінації цих стратегій можуть бути значно ефективнішими, ніж вакцинація сама по собі, як з епідеміологічної, так і з економічної точки зору. Було виявлено, що всі втручання можуть знизити рівень захворюваності. Однак усі ці стратегії мають пов'язані з ними витрати на впровадження, які вплинуть на їхнє впровадження.

Спрощуючі припущення в цьому підході такі: незалежність серед інфікованих членів домогосподарства, латентний період збігається з інкубаційним періодом, противірусні курси доступні для 10% населення, а закриття школи відбувається через появу симптомів у п'яти дітей у будь-який день.

с. Адаптивне моделювання прямого циклу ланцюгів Маркова в Монте-

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

Карло для епідемічного моделювання вірусу папіломи людини

У роботі представлено байєсівську статистичну модель, засновану на методі Монте-Карло з адаптивним ланцюгом Маркова прямої проекції (MCMC), для калібрування високовимірної нелінійної системи звичайних диференціальних рівнянь (oDE), яка представляє епідемічну модель для вірусу папіломи людини 6 та 11 типів (HPV-6, HPV-11). Модель базується на компартменталізації SIGPN, яка включає п'ять станів: сприйнятливий, інфікований, генітальні бородавки, серопозитивний та серонегативний. Розроблена байєсівська модель оцінює параметри епідемії HPV-6 та HPV-11, такі як ймовірність передачі, інкубаційний період HPV, тривалість інфекції, лікування генітальних бородавок та імунітет, а також ймовірність сероконверсії за статтю, віковою групою та групою сексуальної активності. Ця робота розширила розробку матриці змішування статей. Також запропонували розширення класу адаптивних алгоритмів MCMC, щоб включити стратегію моделювання прямої проекції для траєкторій станів ODE. Вони виконали апостеріорний прогностичний висновок щодо впливу вакцинації на змодельовану популяцію. Їхню роботу оцінювали за допомогою моделювання як на нещодавніх реальних, так і на синтетичних даних в Австралії.

Вищеописана модель базується на таких припущеннях: віковий діапазон населення становить 15-59 років з припущенням відсутності сексуальної активності у людей віком до 15 років та старше 59 років, змодельоване населення Австралії вважається постійним у часі, кількість чоловіків у загальній популяції та в кожній віковій групі, що займає певну сексуальну активність, дорівнює кількості жінок, і перехід між статями заборонено, змодельоване населення вважається гетеросексуальним, і всі люди належать до однієї з груп сексуальної активності, люди звертаються за лікуванням негайно після того, як дізнаються про наявність генітальних бородавок, а імунітет є довічним для всіх осіб у популяції.

D. Підхід DTMC до поширення захворювань через контакт

Представили структуру для контактного поширення захворювань у складних мережах на основі ймовірнісних дискретних ланцюгів Маркова

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

(DTMC). Ця структура узагальнює існуючий підхід гетерогенного середнього поля (HMF) та застосовує його до зважених та незважених складних мереж. Автори розглядали дискретний двостанний, сприйнятливий (S) та інфікований (I), контактний процес. Кожен вузол мережі представляє особу або місце (наприклад, місто чи аеропорт), а кожне ребро є з'єднанням, вздовж якого поширюється інфекція. На кожному кроці часу інфікований вузол робить X спроб передати хворобу своїм сусідам з ймовірністю β за одиницю часу. Це утворює ланцюг Маркова (M_c), де ймовірність зараження вузла залежить виключно від останнього кроку часу. Для перевірки свого підходу вони виконали M_c -моделювання на різних безмасштабних (SF) мережах. У статті повідомляється, що числове розв'язання аналітичних рівнянь долає обчислювальні витрати M_c -моделювання. Крім того, представлений формалізм використовується для вивчення поведінки критичного порогу епідемії для різних значень ймовірності контакту з часткою сусідів за крок часу.

Одним з обмежень цього підходу є те, що він базується на простій моделі компартментів з двома станами. Друге обмеження полягає в тому, що хоча всі сусіди в реактивному процесі контактують, мережа контактів була розроблена відповідно до припущення про один контакт за одиницю часу, $X = 1$. У реальному житті кількість стохастичних контактів за одиницю часу є змінною задачі, і коли $X = 1$, модель не є стандартним процесом контактів, оскільки повторні інфекції не враховуються. Крім того, ця модель розглядає лише інфекції, що виникають через прямі контакти між вузлами.

Е. Агент-орієнтоване моделювання пташиного грипу у В'єтнамі

На основі щоденно зареєстрованої кількості мертвої птиці внаслідок пташиного грипу на півночі В'єтнаму в листопаді 2005 року, [13] використали математичну модель для оцінки базового репродуктивного числа R_0 хвороби. Значне значення розрахункового R_0 пояснює вибуховий спалах у популяції птиці. Розробили компартмент SIR, використовуючи комбінацію ЕВМ та АВМ, щоб повторно зібрати зареєстровані дані про спалах пташиного грипу та оцінити ефективність існуючих заходів контролю. Їхня модель припускає повністю однорідну та добре змішану популяцію птиці, де взаємодія між інфікованими та

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

сприйнятливими особинами є випадковим процесом. У роботі зроблено висновок, що процес інфекції пташиного грипу у птиці суттєво не залежить від зовнішніх факторів. Результати дослідження показали, що для контролю над хворобою слід негайно вжити комплексної стратегії вибракування, контролю біобезпеки та масштабної кампанії вакцинації.

Переваги цього підходу включають оцінку стратегій контролю та вакцинації, а також поєднання ЕВМ та АВМ. Одним з обмежень цього підходу є припущення, що популяція птиці є повністю однорідною та добре змішаною.

Е. Багатоагентне моделювання просторової та тимчасова передача серед споживачів ін'єкційних наркотиків

Запропоновано просторову та часову модель передачі ВІЛ/СНІДу серед споживачів ін'єкційних наркотиків (СІН), засновану на багатовалентній системі та геоінформаційних системах (ГІС) у міському середовищі. Вони моделювали просторовий та часовий процес передачі ВІЛ/СНІДу серед споживачів ін'єкційних наркотиків протягом десяти років на платформі Repast (REcursive Porous Agent Simulation Toolkit) Symphony1.2. Особи були розділені на 5 типів: здорова людина, ВІЛ-інфікована людина, людина зі СНІДом, особа, яка вживає ін'єкційні наркотики (СІН), та ВІЛ-інфікована людина, яка вживає СІН. Особи, представлені агентами, пов'язані з місцями, де вони взаємодіють один з одним. Кожен агент має свої особливості, такі як вік, стать, освіта, кар'єра та географічне розташування. Відносини між агентами були описані за допомогою випадкової соціальної мережі. Для кожного кроку часу визначені специфічні правила поведінки агентів. Модель була реалізована для поширення ВІЛ/СНІДу серед СІН у Куньміні, столиці провінції Юньнань, на південному заході Китаю, як тематичне дослідження. Моделювання було проведено для покращення розуміння динаміки передачі ВІЛ та забезпечення основи для запобігання та контролю поширення цієї хвороби. Результати моделювання показали, що кількість осіб, які вперше інфікувалися ВІЛ серед споживачів ін'єкційних наркотиків, частка спільного використання голочок серед них та індивідуальний соціальний вплив мають значний вплив на поширення ВІЛ/СНІДу.

У цьому підході є кілька спрощувальних припущень: для опису відносин

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

між людьми використовувалася випадкова мережа, розмір популяції становить 1000 осіб, кожен агент має двадцять друзів і щодня контактує з п'ятьма-вісьмома з них, усі здорові люди діють позитивно, заохочуючи своїх друзів кинути вживати наркотики, а всі наркомани діють негативно, надихаючи інших почати або продовжувати вживати наркотики. Основним обмеженням є те, що передача через секс між наркоманами не враховувалася.

г. Порівняння великомасштабних обчислювальних підходів з моделювання епідемій

Порівняно агентну модель зі структурованою метапопуляційною стохастичною моделлю для поширення базової пандемічної події в Італії як прикладі великої та неоднорідної європейської країни. Агентна модель базується на явному представленні італійського населення за допомогою дуже детальних даних про соціально-демографічну структуру. Метапопуляційне моделювання використовує моделювання глобальної епідемії та мобільності (GLEaM) та інтеграцію даних про потоки подорожей авіакомпаній із моделями короткострокової мобільності людей у глобальному масштабі. Модель також враховує дані вікової структури Італії. У цій роботі запропоновано розробку гібридних моделей, які поєднують агентні та метапопуляційні підходи.

Порівняння агентних та метапопуляційних стохастичних моделей є однією з переваг цієї роботи. Хоча цей підхід припускає залежність від відстані випадкових контактів у загальній популяції, він також припускає однорідне змішування в домогосподарствах, школах та на робочих місцях. У той час як агентна модель є високоструктурованою та враховує домогосподарства, школи та робочі місця, модель GLEaM враховує лише просторову та вікову структури. Крім того, ядро відстані для випадкових контактів у популяції, що розглядається в агентній моделі, може бути не в змозі відтворити деякі складні властивості, що спостерігаються в потоках повітряних перевезень з неоднорідностями Північ-Південь. Одним з обмежень цього підходу є те, що він не реалізував стратегії втручання.

н. Моделювання епідемії холери 2010-2011 років на Гаїті

Побудовано модель компартментальної передачі епідемії вібріональної

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

холери на Гаїті у 2010 та 2011 роках та дослідили потенційний вплив стратегій боротьби з хворобою. Вони розробили модель SIR з додаванням водного компартменту. Водний компартмент міг бути забруднений інфікованими або інфекційними особами та, своєю чергою, міг заразити сприйнятливих осіб. Моделі представляли населення кожного з десяти адміністративних регіонів Гаїті. Ці популяції були об'єднані для формування метапопуляційної моделі, в якій хвороба могла поширюватися як у межах певного регіону, так і між регіонами, що відображає переміщення людей між різними регіонами. Було виявлено, що поширення холери між двома регіонами пропорційне чисельності населення обох регіонів та обернено пропорційне квадрату відстані між регіональними центроїдами. Аналіз змін динаміки хвороби з часом свідчить про те, що втручання громадського здоров'я суттєво вплинули на цю епідемію. Очікувалося, що обмежене постачання вакцини, забезпечене наприкінці епідемії, матиме помірний ефект.

Одне з основних припущень полягає в тому, що холера може передаватися або через тісні контакти, або через забруднену воду. Однак передача через воду та вживання харчових продуктів, забруднених інфекційною водою, є важливими шляхами передачі холери, які не розглядалися в цій роботі.

1.4 Висновок по розділу

У першому розділі розглянуто теоретичні основи моделювання поширення епідемій. Проаналізовано математичні та комп'ютерні підходи до опису епідемічних процесів, методи їх імітації та сучасні концепції прогнозування динаміки захворюваності. Визначено ключові принципи побудови моделей поширення інфекційних хвороб. Отримані результати створюють теоретичну базу для подальшої розробки програмного забезпечення моделювання епідемій.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЕПІДЕМІЙ

2.1 Аналіз та обробка біомедичних даних для моделювання епідемій

У широкій галузі застосувань аналізу біомедичних даних попередні прототипи програмних рішень зазвичай розробляються з використанням інтерпретованої мови або середовища програмування (наприклад, Matlab®). Коли продуктивність стає проблемою, деякі компоненти прототипу згодом переносяться на компільовану мову (наприклад, C) та інтегруються в решту інтерпретованих компонентів. На жаль, цей процес може призвести до логічних помилок та помилок проектування, а функціональність результуючої гібридної системи часто буває важко розширити або адаптувати. Крім того, також стає важко скористатися такими функціями, як управління пам'яттю, об'єктна орієнтація та генерики, які є важливими вимогами для створення великомасштабних, надійних застосувань. Щоб вирішити ці проблеми, ми розробили Scorira, фреймворк з відкритим кодом на C++, що підходить для застосувань аналізу біомедичних даних, таких як методи аналізу біомедичних даних (ABM) для інфекційних захворювань. Scorira надає високопродуктивні функції комплексної розробки застосувань у формі розширюваної бібліотеки C++. Ця бібліотека надає загальні утиліти програмування, числові матриці та алгоритми, засоби паралелізації та елементи графічного інтерфейсу користувача.

Нашою мотивацією для розробки Scorira було задоволення потреб трьох категорій користувачів у спільноті біомедичних досліджень: архітекторів програмного забезпечення; вчених/математиків; та аналітиків даних. Під час розробки та впровадження нового програмного забезпечення архітекторам зазвичай доводиться впроваджувати застарілі системи, часто написані інтерпретованими мовами. У поєднанні з тим фактом, що вимоги кінцевих користувачів у дослідницькому середовищі часто змінюються (іноді радикально), а біомедичні дані стають дедалі складнішими та об'ємнішими, платформа для розробки програмного забезпечення повинна бути

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

універсальною, розширюваною та використовувати розподілені, універсальні та об'єктно-орієнтовані парадигми програмування. Для вчених або математиків інструменти аналізу даних повинні бути інтуїтивно зрозумілими з адаптивними інтерфейсами, які працюють ефективно та результативно. Нарешті, аналітик даних має вимоги, що відрізняються від вимог інших категорій користувачів. Маючи середній рівень компетенції програмування, їм потрібне відносно інтуїтивно зрозуміле середовище розробки, яке може приховати деякі деталі програмування низького рівня, водночас дозволяючи їм легко налаштовувати та проводити числові експерименти, що включають налаштування параметрів та високорівневі цикли/конструкції рішень. В результаті цієї мотивації, акцент у *Scopira* був зроблений на високій продуктивності, розробці з відкритим вихідним кодом та можливості легкої інтеграції інших бібліотек C/C++, що використовуються в галузі аналізу біомедичних даних, шляхом надання спільного об'єктно-орієнтованого інтерфейсу прикладного програмування (API) для додатків. Ця бібліотека надає широкий спектр послуг, які поділяються на наступні чотири категорії компонентів.

Інструменти Scopira пропонують розширені утиліти програмування та ідіоми, корисні для всіх типів програм. Ця категорія містить систему керування пам'яттю з підрахунком посилань, гнучку/перенаправляючу систему вводу/виводу потоку, яка підтримує файли, відображення файлів у пам'яті, мережевий зв'язок, серіалізацію та збереження об'єктів, універсально унікальні ідентифікатори (UUID) та парсинг та обробку XML.

Числові *функції* базуються на основній концепції *n*-вимірного масиву (див. нижче). Узагальнене програмування на C++ використовується для створення користувацьких, високопродуктивних масивів будь-якого типу та розмірності даних. Загальні математичні функції базуються на масиві . Розробнику також доступний великий набір функцій аналізу біомедичних даних та розпізнавання образів. Кілька API для *паралельної обробки* надаються разом з об'єктно-орієнтованим фреймворком *Scopira Agents Library* (SAL), який дозволяє алгоритмам масштабуватися з доступними ресурсами процесора та кластера. *Scopira* забезпечує легку інтеграцію з власними потоками операційної системи, а

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

також бібліотеками інтерфейсу передачі повідомлень (MPI) та паралельної віртуальної машини (PVM). Крім того, цю бібліотеку можна вбудовувати в настільні програми, що дозволяє їм автоматично використовувати обчислювальні кластери при їх виявленні. На відміну від інших інтерфейсів паралельного програмування, таких як MPI та PVM, можливості Scoripa забезпечують об'єктно-орієнтовану стратегію з підтримкою поширених шаблонів та підходів паралельного програмування.

Нарешті, надається *бібліотека графічного інтерфейсу користувача* (GUI) на базі GTK+. Ця бібліотека містить колекцію корисних віджетів, включаючи масштабований редактор числових матриць, плоттери, засоби перегляду зображень та програмне забезпечення, а також платформу плагінів та 3D-полотно на базі OpenGL®. Scoripa також надає класи інтеграції з популярною бібліотекою графічного інтерфейсу Qt.

Інтрузивний підрахунок посилань (запис кількості посилань об'єкта всередині самого об'єкта) забезпечує основу для управління пам'яттю в застосунках на базі Scoripa. На відміну від багатьох систем підрахунку посилань (таких як VTK та GTK+), система Scoripa використовує виразно симетричну концепцію. Посилання додаються лише через виклики *add_ref* та *suber_ref* — зокрема, сам об'єкт створюється з нульовою кількістю посилань. Це значно спрощує реалізацію інтелектуальних вказівників та легко дозволяє використовувати виділену пам'ять у стеку (шляхом передачі кількості посилань), на відміну від VTK та GTK+, де об'єкти створюються з лічильником посилань та зміненним лічильником посилань відповідно. Scoripa реалізує шаблонний клас *count_ptr*, який емулює *стандартну* семантику вказівників, забезпечуючи при цьому неявний підрахунок посилань на будь-якому цільовому об'єкті. Завдяки інтелектуальним вказівникам управління посиланнями стає значно простішим та безпечнішим, що є значним покращенням порівняно з ручним управлінням пам'яттю в C.

Scoripa забезпечує гнучку, поліморфну та багатоваріантну систему вводу/виводу. Об'єкти потоку можуть бути динамічно пов'язані для формування потоків вводу/виводу. Scoripa включає об'єкти *завершення потоку*, які

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

завершують або ініціюють потік даних для стандартних файлів, мережевих сокетів та буферів пам'яті. Об'єкти *потоку перетворення* виконують перетворення даних з однієї форми в іншу (наприклад, з двійкової в шістнадцяткову), консолідацію буферів та кодування ASCII. Об'єкти *потоку серіалізації* надають інтерфейс для об'єктів для кодування своїх даних у постійний потік. Через цей інтерфейс великі складні об'єкти можуть швидко та легко кодувати себе на диск або через мережу. Після реконструкції система серіалізації повторно створює екземпляри об'єктів з інформації про тип, що зберігається в потоці. Спільні об'єкти - об'єкти, які мають кілька посилань - серіалізуються лише один раз та належним чином пов'язуються з кількома посиланнями.

Платформно-незалежна система конфігурації надається через центральний клас парсингу, який приймає вхідні дані з різних джерел (наприклад, файлів конфігурації та параметрів командного рядка) та представляє їх програмісту в одному узгодженому інтерфейсі. Програміст також може зберігати налаштування та інші опції через цей інтерфейс, а також створювати графічні інтерфейси, щоб допомогти кінцевому користувачеві маніпулювати ними. Використовуючи комбінацію системи реєстрації типів серіалізації та власних функцій RTTI C++, Scoripa здатна динамічно (під час виконання) забезпечувати реєстрацію та перевірку типів об'єктів та їх зв'язків в ієрархії класів. На основі цього можна створити систему плагінів застосунків, дозволяючи зовнішнім бібліотекам динамічного компонування реєструвати власні типи як сумісні з застосунком, забезпечуючи платформу для розширень застосунків сторонніх розробників.

Мови C та C++ забезпечують найбазовішу підтримку одновимірних масивів, які тісно пов'язані з вказівниками в C. Хоча вони придатні для числових обчислень, вони не забезпечують додаткової функціональності, якої вимагають вчені та математики, такої як просте керування пам'яттю, математичні операції або фундаментальні функції, такі як зберігання власних розмірів. Багатовимірні масиви ще рідше використовуються в C, оскільки вони вимагають специфікацій розмірів під час компіляції, що різко обмежує їхню гнучкість. Мова C++, замість

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

того, щоб розробляти новий тип числового масиву, надає всі необхідні мовні функції для розробки такого масиву в бібліотеці. Узагальнене програмування (через шаблони C++, які дозволяють використовувати код для будь-яких типів даних під час компіляції), перевантаження операторів (наприклад, можливість перевизначення операторів додавання "+" або присвоєння "=") та інлайнінг (для продуктивності) забезпечують усі інструменти, необхідні для створення високопродуктивного, зручного класу масивів. Замість того, щоб змушувати розробника додавати ще одну залежну бібліотеку для класу масивів, *Scopira* надає n-вимірні масиви через свій клас *narray*. Цей клас використовує простий підхід, реалізуючи n-вимірні масиви, як це зробив би будь-який програміст на C, але надаючи типобезпечний, шаблонний інтерфейс для зменшення помилок програмування та складності коду. Внутрішні механізми легко зрозуміти, і клас добре працює зі стандартними ітераторами бібліотеки C++, а також з масивами C, мінімізуючи блокування та максимізуючи можливості інтеграції коду. Використовуючи базове програмування шаблонів C++, ми можемо побачити основні ідеї реалізації в наступному фрагменті коду:

Лістинг 2.1 - Базове програмування шаблонів C++

```
template <class T, int DIM> class narray {
    T* dm_ary;           // actual array elements
    nindex<DIM> dm_size; // the size of each of the dimensions

    T get(nindex<DIM> c) const {
        assert(c<dm_size);
        return dm_ary[dm_size.offset(c)];
    }
}
```

З цього фрагмента коду ми бачимо, що *narray* - це шаблонний клас з двома параметрами часу компіляції: *T*, типом даних елемента (*int*, *float* тощо), та *DIM*, кількістю вимірів. Фактичні елементи зберігаються в динамічно розподіленому масиві *C*, *dm_ary*. Довжини вимірів зберігаються в типі *nindex*, для зберігання зміщень масиву використовується узагальнений клас *at*. Надається узагальнений аксесор, який використовує метод *nindex-offset* для перетворення специфічного

					БР.ІІ – 01.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

для виміру індексу та розміру масиву на зміщення в масиві *C*. Це узагальнення працює для будь-якого розміру виміру.

Ще однією особливістю, показаною тут, є використання макросу *assert* у *C* для перевірки дійсності наданого індексу. Ця перевірка меж підтверджує, що індекс справді дійсний, інакше це призведе до збою та завершення програми, одночасно попереджаючи користувача. Ця перевірка значно допомагає програмісту на етапах розробки та тестування програми, а під час високопродуктивної/оптимізованої збірки програми ці макроси прозоро видаляються, що усуває будь-які втрати продуктивності з кінцевого розгорнутого коду. Також передбачені більш зручні засоби доступу (наприклад, ті, що безпосередньо приймають значення *x* або *xu*). Нарешті, можливості перевантаження операторів у *C++* використовуються для перевизначення операторів дужок `[]` та дужок `()`, щоб надати масивам більш лаконічного вигляду, порівняно з явними викликами методів *get* та *set*.

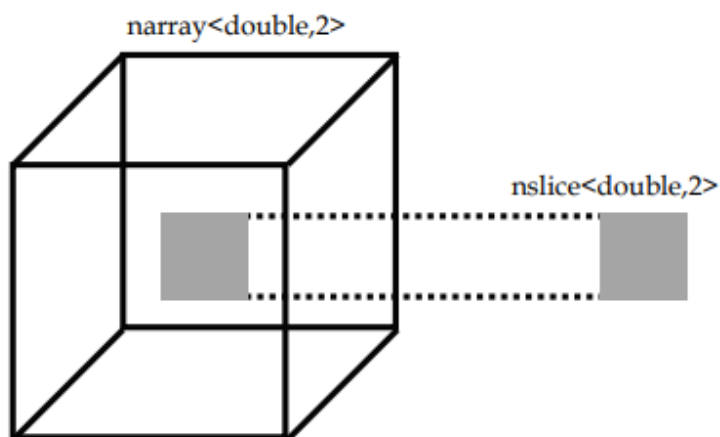


Рисунок 2.1 - Посилання *nslice* на елемент даних типу *narray*.

Шаблонний клас *nslice* - це віртуальний *n*-вимірний масив, який є посиланням на масив (*narray*). Клас містить лише інформацію про специфікацію розмірності та може бути скопійований та прийнятий як параметри функції. Доступ до елементів безпосередньо перетворюється на доступ до елементів у масиві *host*. *Nslice* завжди повинен бути того ж числового типу, що й його масив *host*, але може мати будь-яку розмірність, меншу або рівну розмірності *host*. Це забезпечує значну гнучкість; наприклад, можна мати одновимірний векторний

зріз з матриці, куба або п'ятивимірною масиву. Матричні зрізи з томів є досить поширеними (див. рис. 2.1). Ці підзрізи також можуть охоплювати будь-які виміри/осі, що неможливо з простими масивами вказівників (наприклад, матричні зрізи з масиву куба не обов'язково повинні дотримуватися природного порядку розміщення пам'яті структури масиву).

Клас *parray* надає можливості для альтернативних систем розподілу пам'яті. Однією з таких систем є система відображення *DirectIO*. Використовуючи можливості відображення пам'яті операційної системи (зазвичай через функцію *mmap* у системах POSIX), файл на диску може бути відображений у пам'ять. Коли здійснюється доступ до цього простору пам'яті, сторінки файлів прозора завантажуються в пам'ять. Запис в область пам'яті призведе до запису у файл. Це дозволяє завантажувати файли порціями та за запитом. Операційна система подбає про завантаження/вивантаження порцій за потреби. Файли, розмір яких перевищує розмір пам'яті системи, також можуть бути завантажені (операційна система зберігатиме в пам'яті лише робочу частину масиву). Програміст повинен знати про це та дбати про те, щоб робочий набір залишався в межах розміру пам'яті машини. Якщо робочий набір перевищує доступний розмір пам'яті, продуктивність значно постраждає, оскільки операційна система переносить частини сторінок на диск і з нього (це надмірне жонгливання відображенням диска та пам'яті називається "перевантаженням сторінок").

Зі зростанням кількості процесорів як на робочих столах користувачів, так і в кластерних серверних кімнатах, обчислювально ресурсомісткі програми та алгоритми повинні розроблятися паралельно, якщо вони мають бути актуальними в майбутньому, яке залежить від багатоядерних та кластерних обчислень як засобу масштабування продуктивності обробки. Щоб скористатися перевагами різних процесорів в одній системі або спільному адресному просторі (SAS), розробникам потрібно використовувати лише потоковий API операційної системи або служби спільної пам'яті. Однак для програм, які також хотіли б використовувати ресурси кластера для досягнення більшої масштабованості, використовується явна передача повідомлень. Хоча застосування моделі SAS до

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		30

кластерних обчислень є можливим, для досягнення найкращих результатів обчислювальної продуктивності та масштабованості перевага надається моделі передачі повідомлень. Scoripa включає підтримку двох добре налагоджених інтерфейсів передачі повідомлень, MPI та PVM, а також спеціальний, вбудований, об'єктно-орієнтований інтерфейс передачі повідомлень, розроблений для зручності використання та розгортання.

SAL - це розширення фреймворку паралельного виконання з кількома помітними цілями, особливо корисними для програм на базі Scoripa. API, який є повністю об'єктно-орієнтованим, включає функціональність для: використання системи потоків для обміну повідомленнями; переміщення завдань; інтеграції програм з графічним інтерфейсом; підтримки багатоплатформного зв'язку; та системи реєстрації для створення екземплярів завдань. SAL представляє високопродуктивні обчислення ширшій аудиторії кінцевих користувачів, дозволяючи розробникам програмного забезпечення вбудовувати стандартні можливості кластерів у настільні програми, що дозволяє цим програмам об'єднувати як власні, так і кластерні ресурси. Це суперечить цілям MPI (забезпечення спеціалізованого та швидкого стандарту API зв'язку для комп'ютерних кластерів) та PVM (забезпечення архітектури віртуальної машини серед різноманітних потужних комп'ютерних платформ).

За своєю суттю, SAL запозичив різноманітні концепції як з MPI, так і з PVM. SAL, як і PVM, намагається побудувати єдину та масштабовану систему управління "завданнями" з акцентом на динамічне управління ресурсами та сумісність. Користувачі розробляють взаємопов'язані об'єкти завдань. Завдання можна розглядати як окремі процеси або екземпляри обробки, за винятком того, що вони реалізовані як мовні об'єкти, а не як процеси операційної системи. Вузол SAL керує одним або кількома завданнями, а групи вузлів взаємодіють один з одним, утворюючи обчислювальні мережі (див. рис. 2.2). Завдання поєднані з потужним API передачі повідомлень, натхненним MPI. На відміну від PVM, SAL також зосереджується на простоті використання: підкреслюючи автоматичне виявлення конфігурації та зменшуючи потребу в інфраструктурних процесах. Коли ресурси кластерних або мережевих обчислень недоступні, SAL

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

використовує потоки операційної системи, щоб забезпечити багатопрограмне програмування в межах одного процесу операційної системи та тим самим вбудовуючи повну реалізацію передачі повідомлень у додаток (що значно зменшує складність розгортання). Додатки завжди мають доступну реалізацію SAL, незалежно від наявності або доступу до ресурсів кластера. Розробники завжди можуть використовувати інтерфейс передачі повідомлень, і їхні програми працюватимуть без змін конфігурації як на одному комп'ютері, так і на паралельному розгортанні обчислювального кластера.

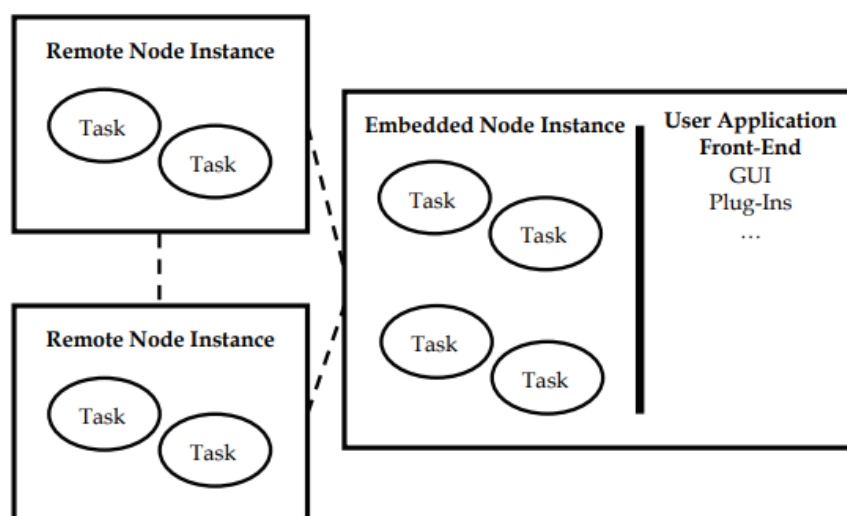


Рисунок 2.2 - Топологія SAL завдань та вузлів.

Механіка та реалізація вузлів SAL та їхньої системи балансування навантаження вбудовані в бібліотеку SAL, а отже, і в програми Scoripa. Користувачам не потрібно встановлювати додаткове програмне забезпечення, а також явно налаштовувати чи налаштовувати паралельне середовище. Це має першорядне значення для забезпечення доступності кластерних та розподілених обчислень для нетехнічних користувачів, оскільки це робить їх прозорою функцією в їхніх графічних програмах.

SAL надає об'єктно-орієнтований, пакетно-орієнтований та маршрутизований (подібно до PVM, але на відміну від MPI) API для передачі повідомлень. Цей API надає все необхідне для створення багатопотокових, кластероорієнтованих алгоритмів, які можна вбудовувати в програми. Завдання

– це основні об'єкти, які розробники створюють для системи SAL. Завдання являє собою окреме завдання або екземпляр у системі вузлів SAL, що аналогічно процесу в операційній системі. Однак вони майже ніколи не є окремими процесами, а радше згруповані в один або кілька процесів вузла SAL, які вбудовані в головну програму. Це відрізняється від більшості існуючих паралельних API, які виділяють один процес операційної системи на кожне завдання, що, хоча концептуально простіше для програміста, несе більше накладних витрат на зв'язок та запуск, а також робить керування завданнями більш складним та залежним від операційної системи. Самі завдання є об'єктами мовного рівня, але зазвичай їм призначаються власні потоки операційної системи для досягнення превентивного паралельного виконання.

Об'єкт контексту – це шлюз завдання до системи передачі повідомлень SAL. В одному процесі може бути багато завдань, тому кожне з них матиме різні контекстні інтерфейси – що неможливо з API з єдиною моделлю «одне завдання на процес» (як використовується в PVM або MPI). Цей клас надає кілька можливостей, зокрема: створення та моніторинг завдань; надсилання, перевірку та отримання повідомлень; реєстрацію служб; та управління групами. Це основний інтерфейс, який розробник повинен використовувати для створення паралельних програм за допомогою SAL.

Розробники часто запускають групу екземплярів одного й того ж завдання, а потім систематично розділяють проблемний простір для паралельної обробки. Для підтримки цієї популярної парадигми розробки система ідентифікації SAL підтримує концепцію *груп*. Група - це просто набір з N екземплярів завдань, де кожен екземпляр має $groupid=\{0,...,N\}$. Концепція групи аналогічна комунікаторам MPI (але без підтримки складної топології) та іменованим групам PVM. Така послідовна нумерація екземплярів завдань дозволяє розробнику легко зіставляти одиниці проблемної роботи із завданнями. Подібно до того, як засоби групування PVM доповнюють концепцію ідентифікатора завдання, групи SAL базуються на системі UUID, оскільки кожне завдання функціонально зберігає свій базовий UUID для ідентифікації.

Система обміну повідомленнями в SAL побудована як на загальному

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

рівні вводу/виводу Scopira, так і на системі ідентифікації UUID. SAL використовує пакетну (подібну до PVM) систему обміну повідомленнями, де система надсилає та маршрутизує лише повні повідомлення, а не окремі примітиви даних (як це може та зазвичай робить MPI) та об'єкти в них. Тільки після того, як завдання надсилання завершиться та зафіксує повідомлення, воно обробляється підсистемами маршрутизації та доставки. Вузол використовує потоки операційної системи для транспортування даних, звільняючи потік користувача для продовження роботи.

Надсилання (фіксація) даних під час знищення об'єкта *send_msg* (тобто через його деструктор) було результатом навмисного рішення, вжитого розробниками програмного забезпечення. У C++ об'єкти стеку знищуються, коли вони виходять з області видимості. Тому користувач повинен помістити об'єкт *send_msg* у власний набір фігурних дужок області видимості, що становитиме свого роду "блок відправлення". Усі передачі даних для повідомлення виконуватимуться в межах цього блоку відправлення, і програміст може бути впевнений, що повідомлення буде надіслано в кінці блоку області видимості, не пам'ятаючи про виконання операції фіксації вручну. Аналогічно, приймач використовує об'єкт *recv_msg* для отримання, декодування та розбору пакета повідомлення, все це в межах "блоку отримання", ув'язаного фігурними дужками. Наведений нижче код містить приклад об'єкта завдання, який через свій контекстний інтерфейс (інтерфейс до мережі повідомлень) надсилає різноманітні об'єкти даних за допомогою об'єктно-орієнтованого API обміну повідомленнями.

Лістинг 2.2 - Приклад об'єкта завдання

```

// declare my task object and its run method
class mytask : public agent_task_i
{ public: virtual void int run(task_context &ctx); };
int mytask::run(task_context &ctx) {
    // send some data to the master task
    narray<double,2> a_matrix;
    {
        // this scope (or send) block encapsulates the sent message
        send_msg msg(ctx, 0);    // prepare the message to task #0

        msg.write_int(10); // send one integer
        msg.save(a_matrix);    // send a matrix – type safe
        msg.save(user_object); // send a user object – via serialization

        // at this point, msg's destructor will be called (automatically)
        // triggering the sending of the message
    }
}

```

SAL наразі має два типи механізмів планування: «локальний» механізм, який використовує потоки операційної системи на одному хост-комп'ютері, та «мережевий» механізм, який може використовувати мережу робочих станцій. «Локальний» механізм - це базова багатопотокова реалізація API SAL. Він використовує потоки операційної системи для реалізації багатопроцесорності в процесі хост-додатку. Оскільки цей механізм міститься в одному процесі, він є найшвидшим і найпростішим у використанні для розробки та налагодження додатків. Програміст може повністю спроектувати та протестувати свій паралельний алгоритм та його логіку обміну повідомленнями, перш ніж перейти до багатовузлового розгортання. Локальний механізм завжди доступний і не потребує налаштування від користувача. Розробникам не потрібно писати спеціальні версії своїх алгоритмів без передачі повідомлень, щоб просто задовольнити користувачів, які можуть не витратити час на розгортання кластера. Оскільки локальний механізм забезпечує стільки ж робочих потоків, скільки й активних завдань, він покладається на здатність операційної системи керувати потоками в процесорах. Це досить добре працює, коли кількість завдань, що створюються в системі, залежить від кількості фізичних процесорів, як це заохочується API.

Мережевий механізм реалізує SAL API на наборі машин, з'єднаних мережею на основі IP; зазвичай Ethernet. Кластер може бути виділеним

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

комп'ютерним кластером та/або набором робочих станцій користувачів. Сам механізм забезпечує міжвузлову маршрутизацію та управління, залишаючи локальні рішення щодо планування в межах кожного вузла менеджера, похідному від локального механізму. Мережевий стек SAL має два рівні (див. рис. 2.3). Нижній транспортний рівень містить самі вузли SAL (об'єкти, які керують усіма завданнями та адмініструванням в одному процесі) та їхні з'єднання на основі TCP/IP. Завдання можуть надсилати повідомлення один одному, використовуючи свої UUID, не знаючи про IP-рівень або топологію з'єднання самих вузлів. Для простоти та ефективності мережа SAL (наприклад, PVM) має головний вузол, розташований на одному процесі. Цей головний вузол відповідає за розподіл, відстеження та міграцію всіх завдань у системі. Мережевий механізм використовує комбінацію прямої адресації, подібної до URL, та автоматичного виявлення на основі широкомовлення UDP/IP для побудови мережі вузлів. Найпростіша та найпопулярніша послідовність - запустити програму в режимі автоматичного виявлення. Коли мережевий механізм запускається, він шукає в локальній мережі будь-які інші вузли-однорівні та, якщо їх знайдено, приєднується до їхньої мережі. Якщо вузли не знайдено, він запускає мережу, що складається з нього як єдиного члена та бере на себе роль головного вузла. Користувачі також можуть безпосередньо вводити URL-адресу головного вузла, явно підключаючи їх до певної мережі. Окрім своїх критично важливих функцій маршрутизації, головний вузол також відповідає за відстеження та управління всіма завданнями в мережі. Завдяки централізації цієї інформації, рішення щодо навантаження та розподілу ресурсів можна приймати миттєво та рішуче. Головний вузол обробляє всі запити на створення екземплярів завдань. Коли завдання всередині вузла запитує створення додаткових завдань, запит маршрутизується хост-вузлом до головного вузла. Потім вузли створюють фактичні завдання та звітують головному вузлу, який, у свою чергу, звітує початковому вузлу та завданню.

Ця підсистема надає базовий графічний API, побудований на GTK+, і складається з класів віджетів та вікон, які стають основою для всіх віджетів графічного інтерфейсу в Scopira. Також передбачені більш спеціалізовані та

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		36

складні віджети, особливо корисні для числових обчислень та візуалізації. Це включає віджети, корисні для відображення матриць, 2D-зображень, стовпчастих та лінійних діаграм. Розробники можуть використовувати основні компоненти графічного інтерфейсу для створення складніших програм для перегляду певної області застосування.

Підсистема графічного інтерфейсу користувача *Scopira* надає корисні інструменти інтерфейсу користувача (віджети) для створення графічних наукових програм. Ці віджети доповнюють загальні віджети, що надаються бібліотекою віджетів GTK+, додатковими віджетами для візуалізації та перевірки даних числових масивів.

Віджет, подібний до матриці/таблиці, може переглядати та редагувати масиви даних (часто, але не обмежуючись, матриці) будь-якого розміру. Цей розширюваний віджет може безпосередньо працювати з *масивами Scopira*. Віджет підтримує розширені функції, такі як масове редагування, за допомогою простої у використанні макромови на основі стеку. Ця макромова підтримує різноманітні операції налаштування, копіювання та фільтрації, вибираючи дані в масиві. Загальний віджет побудови графіків дозволяє відображати значення *масивів Scopira*. Плотер підтримує різноманітні стилі та критерії побудови графіків, а інтерфейс користувача дозволяє масштабувати, панорамувати та інші налаштування графіка даних користувачем. Переглядач зображень дозволяє повністю масштабувати, панорамувати та масштабувати *масиви*, що корисно для відображення даних зображення. Переглядач підтримує довільне відображення кольорів, включає відображення легенди та підтримує плитковий вигляд для одночасного відображення колекції багатьох зображень. Включено спрощений інтерфейс полотна для малювання, який дозволяє розробникам програмного забезпечення швидко та легко створювати власні віджети. Нарешті, *Scopira* надає *лабораторні* можливості для швидкого прототипування та впровадження алгоритмів, які потребують звичайного графічного виводу. Користувачі реалізують свої алгоритми як завжди, а фоновий потік обробляє оновлення графічної підсистеми та циклу подій.

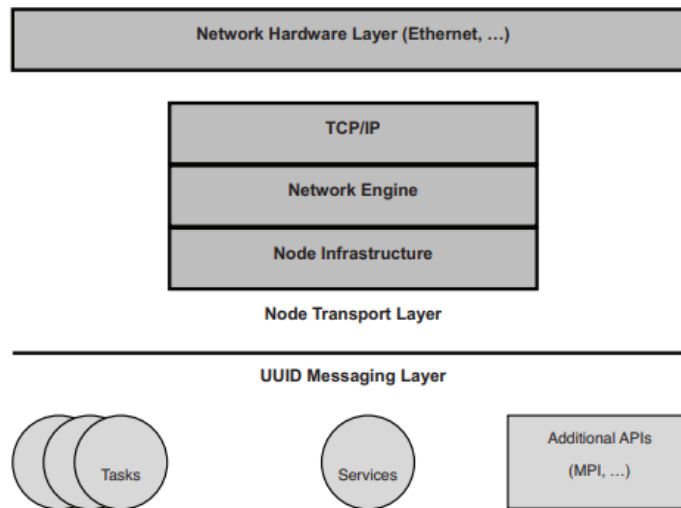


Рисунок 2.3 - Стек мережі SAL.

Scoripa надає архітектуру для логічного розділення моделей (типів даних) та представлень (графічних віджетів, які представляють або працюють з цими даними) у застосунку. Цей зв'язок між представленням та моделлю потім реєструється під час виконання. Під час виконання Scoripa поєднує сумісні моделі та представлення для представлення користувачеві. Надається колекція допоміжних класів для легкої реєстрації типових типів об'єктів, таких як моделі даних та представлення. Цей механізм реєстрації працює успішно незалежно від того, як було завантажено код: як частина застосунку, як пов'язана бібліотека коду чи як зовнішній плагін.

Сторонні розробники можуть легко розширити застосунок Scoripa, який широко використовує моделі та представлення. Стороннім розробникам потрібно лише зареєструвати нові представлення на існуючих моделях даних у застосунку, а потім завантажити свій плагін разом із застосунком, щоб одразу додати нові функції до застосунку. Бібліотека обробки та реєстрації зображень C++ з відкритим кодом ІТК була успішно інтегрована в застосунки Scoripa під час виконання за допомогою підсистеми реєстрації.

Модель визначається як об'єкт, що містить дані та може контролюватися нульовою або більшою кількістю *представлень*. Представлення – це об'єкт, який може прив'язуватися до моделі та прослуховувати її. Зазвичай представлення мають графічний характер, але в Scoripa можливі також неграфічні

представлення. *Проект* – це спеціалізована модель, яка може містити колекцію моделей та організовувати їх ієрархічно. Повністю графічні програми Scorira зазвичай орієнтовані на проекти, що дозволяє користувачеві легко працювати з багатьма моделями даних колективно. Розробникам надається базова структура програм на основі проектів, що дозволяє швидко створювати програми з графічним інтерфейсом за допомогою *моделей та представлень*.

Додаткова підсистема забезпечує базовий клас віджетів з підтримкою OpenGL, який використовує бібліотеку GTKGLExt. Бібліотека GTKGLExt дозволяє програмам на базі GTK+ використовувати OpenGL для 2D та 3D візуалізації. Розробники Scorira можуть використовувати цю систему для створення 3D-поглядів та віджетів візуалізації, що дозволяє краще досліджувати та обробляти дані. Також можлива інтеграція з більш повними пакетами візуалізації, такими як VTK.

2.2 Методи та алгоритми моделювання епідемічних процесів

По-перше, модель K-SEIR поділяє популяцію на такі шість категорій. (1) Сприйнятливі (S): ті, хто схильний до інфекції. Спалах заразив людей різного віку; таким чином, усі люди загалом сприйнятливі. Тому в цьому програмному забезпеченні S представляє всю популяцію в певній області. S має середньодобовий рівень інфекції k . (2) Під впливом (E): Безсимптомний носій вірусу. Ці особи перебувають в інкубаційному періоді після зараження вірусом. Інкубаційний період збільшує складність прогнозування та контролю епідемії. Ми знаємо, що рівень інфекції серед підданої інфекції зазвичай такий самий, як і серед інфікованої популяції; таким чином, модель пропонує таке ж значення k для E, як і для S. Ймовірність того, що носій перетвориться на інфікованого випадок за день, дорівнює r , що відповідає нормальному розподілу із середнім днем інкубації. (3) Інфекційні (I): ті, хто проявив клінічні симптоми після зараження вірусом. Наразі не повідомляється про різницю між рівнем інфекції у легких та важких випадках; отже, ця модель не розрізняє рівень інфекції у легких та важких пацієнтів. Як опромінений (E), так і інфікований (I) можуть заразити

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

S. (4) Видалені (R): ті, хто був видалений з інфікованої популяції. Середньодобовий коефіцієнт видалення інфікованих пацієнтів дорівнює u , який поділяється на дві частини: коефіцієнт виліковування a та коефіцієнт смертності p . Таким чином, $a + p = u$. Кількість видалення в класичній моделі SEIR розкладається на число самовиліковування H (вилікуваний) та число смертей D (смерть). (5) Вилікуваний (H): ті, хто одужав після інфекції. Хоча були повідомлення про повторне зараження [15,16], немає однозначного висновку, чи можуть пацієнти з COVID-19 бути повторно інфікованими після одужання [17]. Для узагальнення моделі пацієнти, які одужали від хвороби, мали середню - добову ймовірність повторного зараження p . Поки відбувається повторне зараження, якщо коефіцієнт повторного зараження отримано з епідеміологічного дослідження, коефіцієнт повторного зараження можна встановити на це значення під час використання програмного забезпечення; Якщо цілитель виробляє антитіла і більше не інфікований, значення p дорівнює 0. (6) Смерть (D): ті, хто помер після зараження. З цією категорією населення необхідно належним чином поводитися, щоб уникнути їх зарахування до людей, які можуть заразити інших.

Далі ми встановлюємо початкові умови для моделі K-SEIR. Чисельність популяції в момент часу t дорівнює s, e, i, r, h та d . У момент часу $t = 0$, відповідні параметри позначаються як s_0, e_0, i_0 та r_0 . Тоді $s_0 + e_0 + i_0 + r_0 = N$, де N – це загальна чисельність населення в певному регіоні. Параметр K було додано для врахування втручання людини, оскільки нереально очікувати, що спалах епідемії повністю відповідатиме природним законам передачі; отже, втручання людини неминуче. По-перше, уряд і громадськість часто зменшують міжособистісне спілкування або посилюють соціальне дистанціювання через виконавчі накази чи спонтанні дії, або ж вони можуть носити маски, щоб запобігти передачі вірусу. Ці заходи можуть знизити щоденний рівень природного зараження (k), зменшуючи рівень міжособистісних контактів, на який впливає захід фізичної ізоляції (k_1). По-друге, уряд може покращити рівень видалення, пришвидшивши медичне лікування. Примітно, що рівень видалення лише прискорить завершення епідемії, а не збільшення рівня

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

одужання. Однак рівень видалення не допоможе покращити рівень виживання населення. Тому ми розділили медичне лікування на два параметри: місткість лікарень (k_2) та можливість медичного лікування (k_3). Параметри втручання людини можна розрахувати за допомогою параметрів підгонки наступним чином:

Фактичний рівень зараження kk = природний рівень зараження kx ($1 - k_1$), отже, $k_1 = 1 - kk / k$ (рівняння 1, рівняння 1); фактичний рівень видалення uk = початковий рівень видалення uk_2 , отже, $k_2 = uk / u$ (рівняння 2, рівняння 2); та фактичний рівень одужання ak = початковий рівень одужання axk_3 , отже, $k_3 = ak / a$.

Теоретичну модель K-SEIR було перетворено на просте програмне забезпечення, реалізоване мовою PYTHON, з використанням програмної інженерії. Зокрема, були виконані завдання проектування графічного інтерфейсу користувача, логіки керування, логіки операцій, точного керування, керування швидкістю, візуалізації даних, імпорту та експорту даних, підбору параметрів, відображення ключових даних та інші специфічні завдання. Завдяки постійній модифікації та налагодженню ми розробили стабільну версію програмного забезпечення: K-SEIR Simulator V2.5. Програмне забезпечення та інструкції доступні онлайн для безкоштовного завантаження лише для некомерційного використання.

Аналіз інфекційних захворювань на основі даних є ключовим застосуванням запропонованого програмного забезпечення. Спочатку було виконано узагальнення аналізу на відповідність. Ми зібрали вибіркові дані з 1 березня по 30 квітня 2020 року з офіційної статистики епідемії в Сполучених Штатах. Після цього було проведено відповідну операцію апроксимації з використанням зібраних даних. Параметр N вказує на загальну чисельність населення, яка в Сполучених Штатах становить приблизно 330 мільйонів. Вразливе населення дорівнює загальній чисельності населення мінус несприйнятливий населення. Несприйнятливий населення відповідає загальному чисельності населення, помноженому на коефіцієнт поширення, а коефіцієнт поширення зазвичай становить 5% [18]. Виходячи з теоретичного моделювання,

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		41

Е дорівнює 3 на 1 випадок на ранній стадії епідемії (див. Додаткові дані S1). Отже, коли діагностоване число I 1 березня 2020 року становило 69, експоноване число було встановлено як 207, тобто $69 \times 3 = 207$. Параметром кількості інфекцій дорівнює I. Параметр для кількості цілителів H встановлено як 1, а параметр для кількості померлих D також встановлено як 1. Після імпорту зібраних даних зразків з 1 березня по 30 квітня 2020 року середній інкубаційний період було встановлено вручну, який зазвичай становить від 4 до 7 днів згідно з наявними даними для цієї епідемії [19] . Тут інкубаційний період встановлено на 5 днів, а функція сканування параметрів використовується для автоматичного підбору оптимальних значень параметрів k, r, y, a та p. Коефіцієнт невдалої антитіл p встановлено на 0. Далі розвиток епідемії визначається відповідно до цих параметрів.

На наступному кроці було проведено узагальнення помилок аналізу підгонки. Через відсутність можливостей раннього виявлення та даних виявлення для пацієнтів з легким перебігом захворювання у Сполучених Штатах, дані, опубліковані на ранніх стадіях, можуть бути значно спотворені. Тому для аналізу помилок були перехоплені статистичні дані з 1 березня по 30 квітня 2020 року.

Для узагальнення параметрів аналізу підгонки, спочатку параметри необхідно отримати, використовуючи офіційні дані Сполучених Штатів. Оскільки фактичні дані епідемії пов'язані із заходами втручання людини, параметри підгонки повинні враховувати зміни k_1 , k_2 та k_3 . Як правило, k_1 лінійно зростає, що вказує на поступове збільшення соціальної ізоляції [5]; k_2 залишається майже постійним, що вказує на те, що коефіцієнт видалення не змінився; k_3 значно зростає, що вказує на поступове збільшення здатності до загоєння. Фактичний рівень інфекції залежить від початкового рівня інфекції та інтенсивності ізоляції k_1 . Початковий рівень інфекції був розрахований як 0,502, який був отриманий шляхом ділення базового репродуктивного числа R_0 на кількість днів інфекції. Ми вважали $R_0 = 3,77$ [20,21], а кількість днів інфекції = інкубаційний період + час, необхідний для діагностики та лікування, де інкубаційний період становив 5 днів, а час, необхідний для діагностики та

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

лікування, – 2,5 дні. Отже, початковий рівень інфекції становив 0,502. Інтенсивність ізоляції k_1 можна розрахувати, використовуючи рівень зараження. Далі, надійність цих параметрів слід підтвердити, використовуючи офіційні дані Сполучених Штатів. Після отримання оптимальних параметрів шляхом підгонки історичних даних проводиться 10-денний прогностичний висновок. Шляхом узагальнення помилок, отриманих за допомогою аналізу підгонки 10-денних даних, перевіряється як раціональність параметрів, так і надійність моделі. Під час підгонки параметрів вагову коефіцієнт параметра можна встановити на основі вимоги щодо похибки даних підгонки. Під час виконання аналізу підгонки ми можемо пожертвувати поточною узгодженістю заради довгострокової узгодженості, що особливо корисно, коли фактичні дані явно не відповідають епідемічному правилу через відсутність оновлених даних протягом логарифмічного періоду. Більше того, оскільки значення відкритих даних часто менші за значення фактичних даних, вагову коефіцієнт позитивних і негативних параметрів можна скоригувати, щоб зробити дані підгонки близькими до опублікованих. Параметри підгонки епідемічної ситуації включають параметри втручання людини.

Для отримання базових даних про інфекційні захворювання потрібен аналіз на основі моделей. 23 січня 2020 року кількість інфікованих становила 830, кількість тих, хто одужав, – 34, а кількість смертей – 25 (дані надані Національною комісією охорони здоров'я Китаю). Латентний день було встановлено на 5, день виведення – на 10, а R_0 – на 3,77. Щоб максимально відобразити наслідки локдауну, самоізоляції, носіння масок та перебування в кабінних лікарнях, ми встановили $k_1 = 0,80$ та $k_2 = 1,4$. Оскільки специфічних ліків немає, ми встановили $k_3 = 1,6$. Загальну чисельність населення було встановлено на рівні 100 мільйонів.

2.3 Агентно-орієнтовані структури моделювання поширення епідемій

Моделювання за допомогою методу АВМ полягає в емуляції реальних систем, в яких колекції автономних об'єктів прийняття рішень, розподілених у

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

сітці або решітці, взаємодіють одна з одною, розкриваючи динаміку досліджуваної системи. Цей вид моделювання виник з необхідності аналізу та моделювання систем, які стають все складнішими з точки зору їхньої взаємозалежності, водночас дані стають більш організованими в бази даних на більш тонких рівнях деталізації. Цей тип моделювання та моделювання виник у галузі багатоагентних систем, які виникли в результаті досліджень розподіленого штучного інтелекту. Ідея розподіленого штучного інтелекту полягає у вирішенні проблем (або пов'язаного набору підзадач) шляхом їх розподілу між низкою відносно простих програм або агентів, кожен з яких має свій власний тип знань або досвіду. Крім того, дослідження АВМ спираються на результати кількох суміжних галузей, включаючи інформатику, науку про управління, соціальні науки, системну динаміку, теорію складності, складні мережі та традиційне моделювання та симуляцію.

Першим кроком у визначенні АВМ є встановлення значення терміна «агент». Хоча не існує одного загальноприйнятого визначення, гарною характеристикою може бути:

«Агент - це інкапсульована комп'ютерна система, розташована в певному середовищі та здатна до гнучких, автономних дій у цьому середовищі для досягнення цілей свого проектування». Найфундаментальнішою рисою агента є здатність приймати самостійні рішення та взаємодіяти з іншими агентами та їхнім середовищем. З практичної точки зору моделювання, агент - це окрема особистість, яку можна ідентифікувати та яка має певне місце розташування; цілеспрямована та автономна; живе в певному середовищі; має набір внутрішніх характеристик (властивостей), можливості прийняття рішень, систему правил, що керують його поведінкою, та здатність навчатися та адаптуватися на основі досвіду (що вимагає певного виду внутрішньої пам'яті); може взаємодіяти з іншими агентами та реагувати на зміни в середовищі.

Залежно від об'єкта дослідження, можуть існувати різні типи агентів, наприклад, дерева, комахи, люди, віруси, бактерії або клітини. Можливі навіть підтипи, такі як клітини тканин, клітини імунної системи, стовбурові клітини або нервові клітини. Враховуючи широкий спектр можливих агентів та ситуацій,

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

система може складатися з кількох типів агентів одночасно. Наприклад, припускаючи інфекційне захворювання в популяції сприйнятливих осіб, мінливість виду може бути включена, враховуючи, що кожен агент може мати різні рівні сприйнятливості до конкретного захворювання, спричинені відмінностями в їх імунній системі, способом життя, попередніми станами здоров'я, специфічними факторами ризику для здоров'я тощо.

Унікальність кожного агента встановлюється його змінними стану або атрибутами, такими як ідентифікаційний номер (бажано унікальний та незмінний, щоб агента можна було відстежити під час усіх відповідних симуляцій), вік, стать, місцезнаходження, розмір, вага, енергетичні резерви, політичні погляди, стадія захворювання тощо. Ці атрибути відрізняють об'єкт від інших такого ж типу, відстежують, як агент змінюється з часом, та керують його поведінкою. Крім того, ці змінні стану можуть бути збережені, що дозволяє перезапустити симуляцію після її переривання. Саме з гетерогенності агентів та відповідного різноманіття поведінки виникають колективні аспекти інтересу, що стосуються реальної системи, що моделюється та моделюється.

З унікальності та автономної природи агентів стає цілком зрозуміло, що застосування агентно-орієнтованого підходу вимагає кількох агентів для представлення децентралізованої перспективи проблеми. Більше того, агенти повинні взаємодіяти один з одним та з навколишнім середовищем. Для цього агенти повинні бути просторово розташовані в певному типі сітки або решітки, в якій вони можуть контактувати з іншими агентами на основі визначених правил та припущень.

Усвідомлення агентами ґратки обмежене та локалізоване. Агенти завжди розміщені в певній позиції ґратки та мають обмежену видимість свого оточення, що означає, що лише кілька інших агентів знаходяться в області впливу агента. Це відрізняється від типового двовимірного клітинного автомата, в якому кожна позиція ґратки складається з комірки; кожна комірка може приймати різні стани, які можуть оновлюватися відповідно до певних правил, заснованих лише на її безпосередніх сусідах. В клітинному автоматі з АВМ (агент з АВМ) у кожній позиції ґратки одночасно може бути більше одного агента. Агенти не є

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

статичними, що гарантує їх вільне переміщення, тому їхній радіус взаємодії в довгостроковій перспективі не обмежується їхніми найближчими сусідами-агентами.

На відміну від багатьох аналітичних моделей, які загалом легше передавати та аналізувати, оскільки вони описуються за допомогою точних математичних формул, описи АВМ часто є неповними і тому менш доступними для читача. У спробі вирішити цю проблему доречності, група моделюючих фахівців у галузі екології запропонувала стандартний протокол для опису АВМ. Протокол «Огляд, концепція дизайну та деталі», або просто ODD, пропонує, щоб АВМ (та її різні правила взаємодії) описувалися стандартизованим способом, що робить описи моделей легшими для розуміння та завершення. Досягнення цієї мети робить АВМ відтворюваними та набагато надійнішим науковим інструментом для досліджень.

Перш ніж моделювати проблему за допомогою агентно-орієнтованого підходу, необхідно врахувати кілька моментів. По-перше, слід врахувати природу проблем, які можуть бути корисними для цього методу, і як симуляції можуть надати корисну інформацію про досліджувані сценарії. Агентно-орієнтований підхід (АВМ) може забезпечити реалістичний та гнучкий опис системи, а також може враховувати емерджентні явища. «Емерджентне явище» можна концептуалізувати як великомасштабну групову поведінку системи, яка, здається, не має чіткого пояснення з точки зору складових частин системи. Ці явища можуть виникати через нелінійний характер поведінки агентів, який можна описати розривами в поведінці агента, які важко врахувати за допомогою диференціальних рівнянь. Крім того, агент демонструє пам'ять (немарковський процес), залежність від шляху, навчання та адаптацію. Таким чином, моделі на основі агентів (АВМ) є адекватними для опису явищ реального світу, де поведінка агентів є складною. Після висновку, що АВМ підходить для конкретного випадку, діаграма на рисунку 2.4 пропонує деякі рекомендації щодо розробки АВМ. Моделювання симуляцій на основі агентів вимагає створення представлення досліджуваного «підсвіту». Щоб це представлення було реалістичним і точним, повинен бути набір доступних даних для підтримки

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

різних міркувань, прийнятих для кількох аспектів і частин, включених до моделі на основі агентів. У процесі проектування АВМ для поширення інфекційного захворювання в міському центрі ми проілюструємо, які дані можна використовувати на кожному кроці.

Враховуючи поширення інфекційного захворювання в популяції, можна отримати дані щодо віково-статевої структури популяції та використовувати їх для створення популяції *in silico*. Однак відсутність детальної інформації щодо фактичної події (наприклад, оцінки епідеміологічних параметрів для нового інфекційного захворювання) часто створює значні труднощі для моделювання симуляцій на основі агентів. З іншого боку, експериментування з моделлю за різних припущень (наприклад, різних стратегій втручання) може виявити нові явища, які призводять до кращого розуміння системи та її поведінки. Саме цей циклічний зв'язок залежності робить методи впливу на агенти потужними та одночасно складними.

Хоча ця структура моделювання була створена як універсальна для основних типів АВМ, можливо, що деякі аспекти дуже специфічних випадків не були включені. Тим не менш, завдяки цій структурі читач повинен мати відправну точку для проектування АВМ та виконання симуляцій шляхом додавання специфічних характеристик системи. Як і в будь-якому проекті з розробки програмного забезпечення, головним моментом, який потрібно визначити, є мета застосування. У нашому прикладі метою моделі є розуміння ключових характеристик популяції з її моделями соціальної взаємодії, які впливають на поширення інфекційного захворювання та призводять до епідемічних сценаріїв. На цьому етапі необхідно вибрати відповідну модель захворювання для конкретного захворювання. Якщо досліджуване захворювання не є абсолютно новим та невідомим, можна вибрати, чи достатньо простої моделі «сприйнятливий-інфекційний-одужаний» (SIR), чи потрібна більш уточнена модель, що включає інші проміжні стани. Важливо зазначити, що припущення, засновані на параметрах глобальної спільноти, впливають на деякі локальні індивідуальні параметри. У цьому прикладі вибір моделі SIR означає, що з точки зору захворювання, агенти можуть перебувати лише в

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

одному з трьох станів (сприйнятливий, інфекційний або одужав). З точки зору локальних індивідуальних параметрів, можна враховувати такі аспекти, як вагітність, лікування аспірином тощо.

Цей опис дає загальне уявлення про те, що буде моделюватися. На цьому етапі важливо знати точні питання, на які має відповісти модель. Таким чином, буде зрозуміліше, які дані, отримані в результаті моделювання, слід зберігати для аналізу. Після визначення мети ми матимемо чітке уявлення про те, якими будуть агенти в нашому моделюванні. Загалом, моделювання включатиме різні види агентів. У типовому поширенні деякого вірусу в популяції, як-от у випадку пандемії грипу H1N1 2009 року, задіяний лише один тип агента, а саме люди. Ці агенти або уражені хворобою, або ні. Будучи інфікованими, вони можуть передавати хворобу сприйнятливому агенту, якщо вони потрапляють у близький контакт (визначення близького контакту - це всі взаємодії, які можуть призвести до інфекції). У цьому випадку немає потреби мати більше одного агента, оскільки інфекційне захворювання можна розглядати як властивість агента. І навпаки, якщо інтерес полягає в дослідженні закономірності передачі хвороби від свиней до людей, то можуть бути задіяні два типи агентів (свині та люди). У більш конкретних сценаріях інтерес може представляти сама вірусна популяція. У цьому випадку можуть існувати підтипи агентів, тобто штами вірусу, які є підтипами ширшого класу певного вірусу.

Незалежно від типу агентів, вони матимуть ознаки (змінні стану), які визначають їх як індивідів. Для всіх агентів моделювальник повинен перерахувати всі змінні стану, включаючи не лише ті, що визначають їхні характеристики (такі як вік, вага та температура), але й ті, що описують поведінкові риси агента, такі як швидкості, стратегії, ймовірності, сприйнятливість тощо. Крім того, важливо визначити, як агенти взаємодіють із середовищем протягом заданих періодів часу. Демографічні дані використовуються для створення агентів, щоб результуюча популяція *in silico* відповідала тим самим розподілам, що й реальна популяція. У цьому сенсі змінні стану визначаються як примітивні характеристики індивідів, які пізніше можна описати з точки зору демографічної інформації, такої як вікові та гендерні

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

профілі. Крім того, необхідно визначити кілька інших, не демографічно специфічних змінних. Деякі з цих змінних стосуються захворювання, такі як тривалість симптомів, сприйнятливість тощо. Інші види змінних стосуються поведінкових рис. На основі цих змінних та відповідної інформації, отриманої з середовища та інших агентів, агент може змінювати свій стан та приймати рішення щодо своїх наступних кроків. Коротше кажучи, змінні стану є найважливішою частиною моделі, враховуючи, що вони визначають агентів і надають їм відповідну інформацію для автономної дії.

Після визначення агентів наступним кроком є визначення їхнього розміщення, таким чином визначаючи їхнє середовище. Якщо зв'язок між агентом і середовищем не є важливим для цілей моделювання, середовищем має бути лише місце, яке сприяє взаємодії між агентами. У деяких випадках цей зв'язок може бути важливим, як у випадку, коли демографічний розподіл може відігравати важливу роль у поширенні інфекційного захворювання. Після визначення середовища необхідно визначити його масштаби, щоб встановити його розмірність та розмір кожного виміру. Крім того, представлення середовища не обмежується n -вимірним простором, а визначається як складна мережа або якась інша структура даних і може містити інші властивості, що впливають на рішення та поведінку агентів. В епідемічних сценаріях характеристики міського центру, такі як відстань поїздки на роботу, демографічна щільність та межі міста, можуть впливати на поширення інфекційного захворювання, а також на поведінку агента. Таким чином, дані щодо моделей мобільності можуть бути використані для визначення властивостей решітки, які, у свою чергу, можуть використовуватися агентами у своїх рішеннях та діях.

Планування агентів та середовища не вважається завершеним, доки не визначено процеси, пов'язані з ними, що призводить нас до наступного блоку на блок-схемі на рисунку 2.4. Набір процесів є ядром моделювання. На цьому етапі слід зазначити, що всі дії, які виконує кожен агент, стосуються оновлення свого стану, взаємодії з іншими агентами та взаємодії з середовищем. Аналогічно, слід перерахувати всі механізми оновлення середовища.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

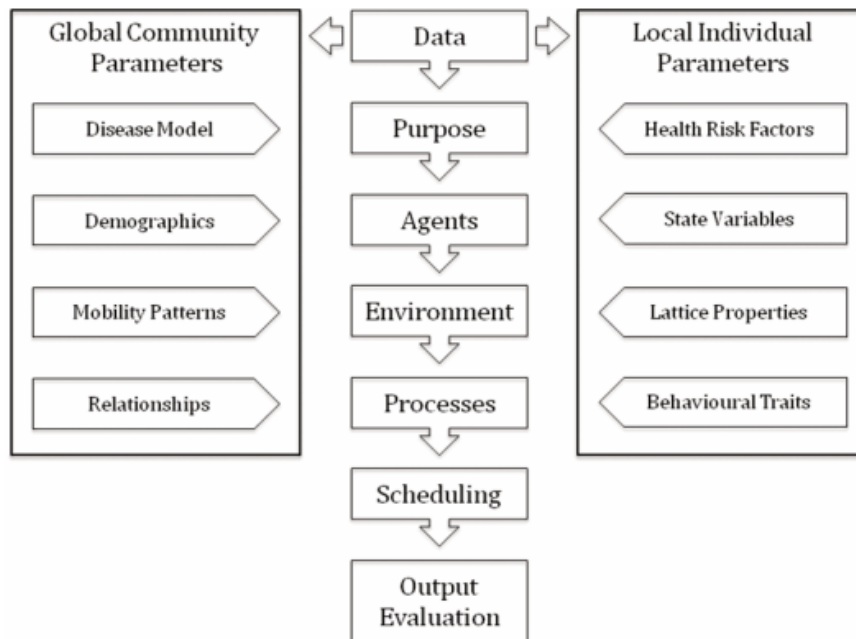


Рисунок 2.4 - Цикл розробки моделі інфекційного захворювання на основі агента.

Щодо оновлення стану агента, можна розглянути кілька кроків, таких як визначення того, чи досягнуто мети агента, або визначення найкращого плану дій для досягнення мети. Наприклад, розглядаючи випадок інфекційного захворювання, необхідно перевірити, чи інфікований агент, і, якщо так, вирішити, чи настав час звернутися за медичною допомогою. Здійснення взаємодії агент-агент (або агент-середовище) вимагає зв'язку між агентами (або з середовищем), щоб залучені суб'єкти могли отримати доступ до інформації один одного та вибрати свої наступні дії. У випадку взаємодії сприйнятливого агента з інфекційним агентом необхідно визначити, чи заразиться сприйнятлива особа.

Після визначення всіх пов'язаних процесів виникає природне питання про час, у який ці дії відбуватимуться. Це питання представлено блоком планування на рисунку 2.4. Тут блок-схема процесів буде корисною для визначення порядку їх виконання. Кілька процесів можуть відбуватися одночасно, тоді як інші можуть залежати від інших процесів. Знаючи загальну картину подій, можна визначити, чи змінні стану оновлюються синхронно чи асинхронно. Крім того, тип часової схеми слід визначати як дискретний, безперервний або обидва.

У контексті паралельних обчислень зв'язок між процесами визначатиме легкість паралелізації коду (наприклад, тісно пов'язані або просторово близькі зв'язки важче паралелізувати порівняно зі слабо пов'язаними або просторово віддаленими зв'язками). АВМ можуть починатися з невеликих і простих симуляцій. Однак, з часом АВМ може стати великим і обчислювально складним, що вимагатиме паралелізації виконання коду. Таким чином, структурування моделі перед кодуванням і вибір правильних інструментів для завдання, безумовно, заощадить час зрештою.

У цьому підрозділі ми описуємо програмну платформу АВМ для моделювання поширення заразного захворювання в людських популяціях. Агенти в цій моделі є спрощеним представленням реальних мешканців міського центру, в який було введено інфекційний патоген. У цій системі три основні компоненти моделювання: агенти, решітка (або сітка), в якій розміщені ці агенти, та правила взаємодії між агентами та агентами з середовищем. Враховуючи, що моделювання АВМ спирається на взаємодію між агентами, які мають подібні характеристики, але різну поведінку, природно вибрати будь-яку мову об'єктно-орієнтованого програмування (ООП) для запуску таких моделювань. Використовуючи ООП, кожен агент може бути створений як представлення загального класу агентів, в якому кожен програмний об'єкт має однаковий набір властивостей або змінних стану. Припускаючи різний набір значень для властивостей кожного агента, встановлюється унікальність кожного агента та враховується мінливість виду.

У контексті АВМ одиницею симуляцій є агент (часто званий індивідом), а в контексті ООП ця одиниця є об'єктом визначеного класу. Однак важливо зазначити, що агенти та програмні об'єкти є різними компонентами АВМ. Програмний об'єкт є відповідним способом представлення агентів у тому сенсі, що об'єкти інкапсулюють як стан агента (як члени даних), так і поведінку агента (як методи), і спілкуються через передачу повідомлень. Але агенти, з іншого боку, є «раціональними системами прийняття рішень», які здатні реагувати та поводитися відповідно до ситуації.

Залежно від конкретної моделюваної проблеми, властивості агентів, що

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

нас цікавлять, можуть відрізнятись. Для епідемічного поширення в популяції сприйнятливих агентів деякі з відповідних характеристик цих агентів включають вік, стать, сприйнятливість до захворювання, стадію захворювання, серед кількох інших властивостей, що стосуються природи захворювання у інфекційних агентів. У такому сценарії дослідження динаміки та еволюції інфекційного захворювання вимагає розробки реалістичної моделі популяції зі значно великою кількістю агентів (як зазначено у відповідних демографічних даних), властивості яких відображають віково-статевий розподіл у цьому конкретному місці. Отже, продуктивність системи моделювання стає важливим питанням при виборі належних інструментів для побудови моделювання.

Як і в кожному проекті з розробки програмного забезпечення, вибір відповідної мови програмування та бібліотек для побудови АВМ-додатку має вирішальне значення для створення надійного, ефективного та адаптивного програмного коду. В АВМ використання ООП є відповідним спрощеним підходом до логіки симуляцій та процесів кодування. Крім того, вибір правильних даних моделі та бібліотек впливатиме на продуктивність та використання ресурсів, що, зрештою, відображається на часі виконання симуляції. Беручи до уваги ці вимоги, ми обрали C++ як найбільш підходящу мову програмування для АВМ, описаної тут. З точки зору продуктивності та універсальності, мова C++ порівнянна з мовою C з додаванням деяких спеціалізацій ООП. Крім того, використання функціональних можливостей фреймворку Scopira та структур даних, таких як інтелектуальні вказівники, *масиви* та його розподілені обчислювальні інструменти, спрощує розробку програмного забезпечення АВМ-додатків, особливо щодо паралелізації коду.

Ще однією важливою характеристикою методів випадкових чисел (АВМ) є стохастичність, присутня в кожній частині моделювання. Мінливість серед агентів може виникати внаслідок вибірки, наприклад, розподіл за віком та статтю, розподіл сприйнятливості або розподіл періоду інфекційності. Стан агента може бути змінений за допомогою взаємодії з іншими агентами, де результат цих взаємодій залежить від значення параметрів, вибірових з деякого заданого розподілу. Як результат, генератори випадкових чисел повинні бути

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

ретельно вибрані. Для генерації випадкових чисел гарним вибором числової бібліотеки є *GNU Scientific Library*, яка широко застосовується в наукових та інженерних застосуваннях і має додаткову перевагу в тому, що вона є відкритим кодом, потокобезпечною, добре протестованою та портативною.

2.4 Висновок по розділу

У другому розділі розглянуто методи та інструменти розробки програмного забезпечення для моделювання епідемій. Досліджено підходи до аналізу біомедичних даних, алгоритми моделювання епідемічних процесів та агентно-орієнтовані структури симуляції. Визначено особливості побудови моделей, що дозволяють враховувати складну динаміку поширення інфекцій. У результаті сформовано методичну основу для створення симуляційних систем епідеміологічного моделювання.

					БР.ІІ – 01.00.00.000 ІЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Прорєктування симуляційної системи моделювання епідемій

Агенти моделюються як об'єкти класу під назвою *Agents (Агенти)*, в якому змінні-члени представляють характеристики агентів, такі як вік, стать, схильність до захворювання та стан захворювання. Методи цього класу є простими маніпуляторами, які змінюють значення цих змінних-членів. Таким чином, кожен агент представляє людину з однаковими основними характеристиками, які можуть приймати різні значення. Використовуючи розподіл за віком та статтю для певної популяції, вводиться мінливість віку та статі. У сценарії епідемічного захворювання агент може приймати різні стани щодо свого епідеміологічного статусу, які можуть бути одним із наступних: сприйнятливий, підданий впливу (але ще не заразний), заразний та одужав (імунітет до повторного зараження). Оскільки кількість агентів у симуляції може коливатися від кількох тисяч до мільйонів, важливо економити пам'ять, коли це можливо. Враховуючи, що ці стани інфекційності можуть приймати лише два значення (*true* або *false*), що вказують на те, чи знаходиться агент у певному стані чи ні, найкращим способом зберігання цієї інформації є використання одного беззнакового цілого числа як набору бітів (або ж використання класу *bitset* зі стандартної бібліотеки C++), в якому кожен біт представляє один із можливих станів. Крім того, стан агента можна оцінити за допомогою побітових операцій, що тестують лише конкретні біти, що цікавлять, заощаджуючи таким чином кілька операцій процесора на агента за цикл.

Подібно до реального сценарію, агенти, загалом, живуть у місті (або громаді), де вони повинні бути розташовані десь і можуть вільно переміщатися в різні місця. Тому місто моделюється як інший клас під назвою *Lattice (Решітка)*, в якому двовимірний n-масив використовується для представлення карти міста та є контейнером для зберігання агентів. Найпростіший спосіб побудови АВМ-моделювання - це розглянути лише одного агента на сайт, який

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

взаємодіє зі своїми сусідніми сусідами, як у клітинному автоматі, або навіть з віддаленими сусідами, але здійснює ці взаємодії, не рухаючись зі своєї позиції. Інший підхід, який використовується в представленій тут моделі, полягає в тому, щоб розглянути кілька вільно пересуваних агентів в одній позиції. Ця модель намагається імітувати поведінку людей у місті, де люди можуть взаємодіяти з іншими в одному місці, як це було б у будинку, офісі чи торговому центрі. Враховуючи, що кілька агентів можуть одночасно перебувати в одній ху позиції міської сітки, і що ці агенти можуть переміщатися по карті, кожна точка в решітці містить стандартний вектор інтелектуальних вказівників *Scopira* на агентів, тобто решітка є двовимірним n-масивом стандартних векторів вказівників на агентів, як показано на лівій панелі рисунка 3.1. Згідно з цією схемою, переміщення агента з одного місця в інше здійснюється шляхом призначення вказівника агенту, що переміщується, вектору в місці призначення, а потім видалення вказівника на цього агента з вектора в початковій позиції, як показано на правій панелі рисунка 3.1.

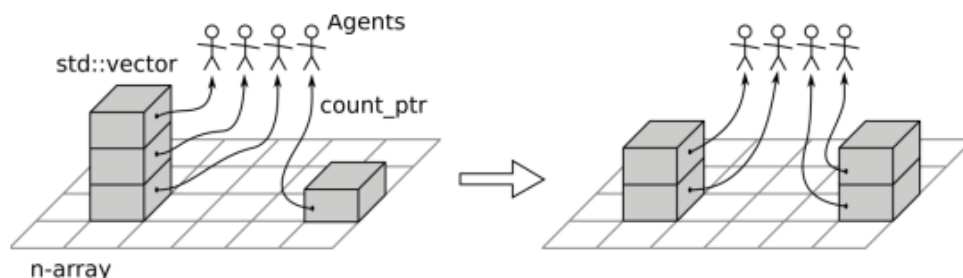


Рисунок 3.1 - Зображення рухів агента в решітці.

Використовуючи цю стратегію інтелектуальних вказівників, не буде додаткових накладних витрат на переміщення або копіювання даних агента в пам'ять. Ще однією перевагою інтелектуальних вказівників є те, що немає потреби вручну звільняти пам'ять після завершення програми, оскільки вона оновлюється, як тільки інтелектуальний вказівник виходить за межі області видимості. Крім того, можна перевірити, чи програмний об'єкт активний, що значно зменшує кількість помилок сегментації під час фази розробки програмного забезпечення через спроби доступу до сторінок пам'яті, які вже

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

були звільнені. Крім того, використовуючи область видимості як спосіб контролю життєвого циклу програмних об'єктів, отриманий застосунок є краще структурованим, надійним та розширюваним.

Методи класу *Lattice* використовуються для керування доступом до локацій на карті та до агентів, розташованих у кожній позиції. Використовуючи, наприклад, карту міста, на якій задані його межі та демографічний розподіл, представлення ґратки визначатиме всі сайти (тобто позиції x, y) відповідно до меж міста та кількість агентів, дозволених у кожному сайті, відповідно до демографічної щільності. У цьому підході агенти розміщуються в решітці, але вони можуть не бути безпосередньо пов'язані. Зв'язки агент-агент та агент-решітка зберігаються в іншому класі під назвою *Simulation*. У цьому класі використовується інтелектуальний вказівник на решітку, а правила щодо руху агентів у решітці та взаємодії між агентами розглядаються як методи цього класу. Крім того, цей клас відповідає за оновлення лічильників, які відстежують кількість агентів у кожному стані в симуляції, та за запис звітів у файли на жорсткому диску комп'ютерної системи. Для цих двох завдань генеруються класи *Counters* та *Reports*.

Оскільки симуляція була побудована для використання більш ніж одного процесора за допомогою потокового програмування, класи *Counters* та *Reports* необхідно створювати потокобезпечними, тим самим підтримуючи паралельність. Під час роботи з кількома потоками можливо, що різні потоки намагатимуться одночасно отримати доступ до однієї й тієї ж області пам'яті; наприклад, щоб збільшити значення однієї й тієї ж змінної одночасно; це може призвести до непередбачуваних результатів. Один зі способів уникнути невідповідностей даних - використовувати блокування взаємного виключення або м'ютекси. Перш ніж потік працювати зі змінною, він повинен отримати блокування на цю змінну, щоб запобігти одночасній роботі інших потоків з тією ж змінною. Це забезпечує серіалізацію доступу до певної області даних. Після того, як потік виконав потрібні операції, йому потрібно розблокувати цю змінну, щоб інші потоки могли отримати до неї доступ. Визначення того, які ресурси слід блокувати та розблоковувати, і в якому порядку, є фундаментальними

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

міркуваннями при паралельному програмуванні. Крім того, необхідно прийняти стратегію взаємодії з цими спільними ресурсами, щоб мінімізувати кількість одночасних звернень до однієї спільної області. В іншому випадку це уповільнить симуляцію, враховуючи, що коли потік очікує отримання блокування, він зупиняє інші свої операції. Хоча цю ситуацію можна вирішити різними способами, це ускладнює проблему та може призвести до інших непотрібних проблем синхронізації.

Counters та Reports виконують дії над областями спільного доступу потоків. Змінні-члени класу Counters відстежують кілька аспектів симуляції, такі як поточний крок часу, на якому виконується симуляція, загальна кількість вразливих агентів та кількість інфекційних агентів. Scopira надає кілька методів паралельного програмування в SAL, що спрощує цей процес. У випадку захисту певних областей даних, таких як змінні, він визначає клас *shared_area*, з яким пов'язаний м'ютекс, до якого пізніше можуть отримати доступ об'єкти класу *lock_ptr*. Допоміжний до класу Counters, він визначає структуру даних під назвою *SimulationCounters*, яка містить усі змінні лічильника, до яких може отримати доступ більше одного потоку одночасно. У класі Counters створюється екземпляр класу *shared_area*, пов'язаний зі структурою даних *SimulationCounters*. Методи класу Counters, під час маніпулювання цими змінними, викликають їх через об'єкт класу *lock_ptr*, пов'язаний з об'єктом спільної області. Коли створюється вказівник блокування, він автоматично блокує спільну область, з якою він пов'язаний, а коли він виходить за межі області видимості, він автоматично знищується, тим самим розблоковуючи пов'язану спільну область. Аналогічно, коли викликається метод, він створює вказівник блокування та отримує блокування, потім виконуються потрібні операції, і в кінці виконання методу вказівник блокування автоматично знищується, тим самим розблоковуючи спільну область. За допомогою цього методу програмісту не потрібно вручну блокувати та розблоковувати змінні, а натомість потрібно визначити вказівник блокування всередині області видимості, призначеної для маніпулювання спільною областю.

Дотримуючись тієї ж ідеї, клас Reports використовується для запису

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

інформації про симуляцію у файли. Якщо два потоки намагаються одночасно отримати доступ до одного файлу, повідомлення може бути спотворене, тому необхідно використовувати м'ютекси. Для цього Scorpiа надає клас *fileflow* , який, у свою чергу, надає методи для маніпулювання файлами, а також включає м'ютекси. З кожним файлом пов'язаний екземпляр об'єкта класу *Reports* . Він, по суті, відповідає за відкриття та закриття файлів і перевантажує оператор вставки потоку "<<". Таким чином, щоразу, коли викликається оператор вставки для об'єкта *Reports* , він викликає метод *fileflow::write_lock()* , передає повідомлення об'єкту *fileflow* , а потім розблоковує його за допомогою методу *fileflow::unlock()* .

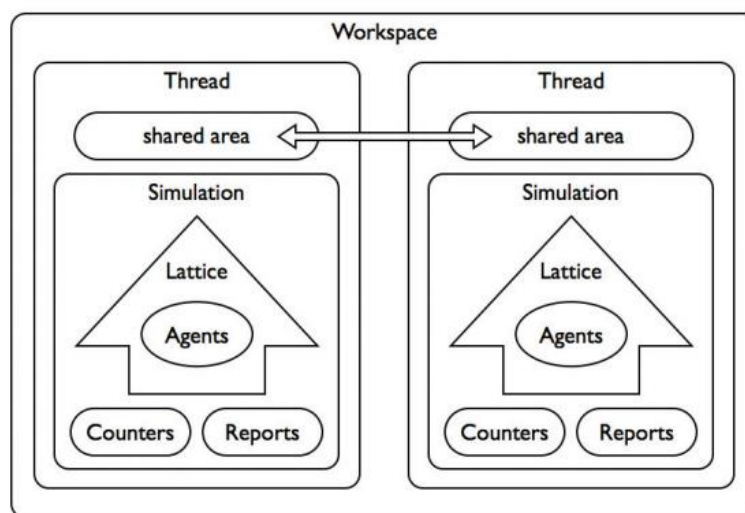


Рисунок 3.3 - Середовище моделювання, що показує взаємодію двох потоків.

На рисунку 3.6 показано схему середовища моделювання з його кількома компонентами. У цьому представленні клас *Simulation* об'єднує класи *Lattice*, *Reports* та *Counters* . Ці три класи включені до класу *Simulation* через інтелектуальні вказівники на відповідні об'єкти кожного класу. Оскільки клас *Lattice* містить інтелектуальні вказівники на всіх агентів у симуляції, до яких можна отримати доступ через об'єкт *Lattice* , правила, що стосуються взаємодії агент-агент та агент-ґратка, є методами класу *Simulation* .

Для запропонованої тут інфраструктури симуляції на основі агентів ми представляємо механізми паралелізації симуляцій. Стратегія паралелізму,

прийнята в цьому прикладі, полягала в тому, щоб розділити решітку на фрагменти та призначити їх різним потокам обробки. Щоб збалансувати обсяг роботи, що виконується кожним потоком, масив, *що* представляє решітку, був розділений на приблизно рівну кількість активних сайтів, які відповідають визначеним місцям на карті міста. Агентам дозволено переміщатися між сайтами на карті, але розташування сайту призначення може бути поза межами контролю фактичного потоку обробки. Як результат, необхідно, щоб усі потоки були оснащені спільною областю, до якої інші потоки можуть надсилати агентів, що переміщуються (див. рис. 3.2). Клас *Threads* успадковує базову функціональність класу *scopira::thread*, який забезпечує механізми для створення паралельного середовища та керування всіма аспектами потоків незалежно від операційної системи.

Лістинг 3.1 - Метод *Threads::run()*

```
Threads::run() {
    CheckSharedArea()

    foreach site:
        simulation.InteractAgentsToAgents( site )
        simulation.InteractAgentsWithLattice( site )
        simulation.MoveAgents( site )

    CheckSharedArea()
}
```

Як результат, у класі *Threads* потрібно реалізувати лише кілька методів, що стосуються конкретної досліджуваної проблеми, та спільну область для обміну агентами між потоками. Найважливішим методом з точки зору моделювання є метод *Threads::run()*, який є *перевантаженням методу thread::run()* *Scopira*, де відбувається власне моделювання. Псевдокод цього методу наведено вище.

У цьому випадку кожен потік ітерує по всіх сайтах під своїм контролем та виконує завдання, необхідні для симуляції. У простому випадку він виконуватиме взаємодії агент-агент та агент-іратка, а також переміщуватиме агенти, якщо необхідно. Останній крок переміщення агентів є найважливішим

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		59

завданням з точки зору розподіленого програмування, оскільки він включає інші потоки та проблеми синхронізації.

Як згадувалося раніше, агент може переміститися в місце поза межами контролю для поточного призначеного йому потоку. З цієї причини керуючий потік повинен перевірити, чи агент переміщується за межі своїх доменів, і направити його до спільної області відповідального потоку. Отже, потрібні додаткові кроки в методі *run()* , включаючи перевірку власної спільної області, яка виконується методом *CheckSharedArea()* , та присвоєння агента, що переміщується, його новому місцю розташування, що виконується методом *MoveAgents()* . На цьому етапі виникають проблеми синхронізації, що вимагають суворого контролю над часовим кроком агентів. Хоча потоки використовують один і той самий адресний простір, вони не знають про інші потоки та повинні керуватися з центрального місця, а саме з батьківського процесу. Клас *Workspace* відповідає за створення, керування, знищення та відстеження потоків та їхніх доменів, а також за сприяння їхньому зв'язку. Для виконання цих дій кожен потік повинен мати посилання на екземпляр *Workspace* , який можна використовувати як посередник між усіма активними потоками.

3.2 Аналіз статистичних даних щодо поширення COVID-19

Неповне повідомлення про інфекції є поширеною проблемою під час збору даних про інфекційні захворювання, особливо на початку спалаху. Коли медичні діагностичні інструменти стають точнішими та надійнішими, а також підвищується прозорість профілактичних заходів на користь добровільного домашнього карантину, зазвичай відбуваються певні коригування даних. Це показано на рисунку. 3.4 що 12 лютого сукупна та щоденно додана кількість випадків інфікування в Хубеї мала чіткі стрибки зі значно великими розмірами. Такі значні стрибки не можуть відбутися протягом одного дня, вони радше являють собою накопичення випадків, про які не повідомлялося раніше, до 12 лютого. Щоб виправити цю проблему заниження показників, ми розробили

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

процедуру калібрування з детальною інформацією в Додатку С. Нижче ми коротко опишемо наш підхід до калібрування випадків інфікування.

Ми припускаємо експоненціальну криву зростання кумулятивної кількості інфікованих випадків до 12 лютого виду $y(t) = a \exp(b \cdot t)$, де параметри A , a , b мають бути оцінені. За граничних умов $y(t = 12 \text{ січня}) = 0$ та $y(t = 12 \text{ лютого}) = a \exp(31A) \cdot b$, ми хотіли б мінімізувати похибку екстраполяції на один крок вперед 13 лютого. Обмежений оптимальний розв'язок можна отримати за допомогою множників Лагранжа; оцінки становлять $A = 0,06605$, $a = 7142,80$, $b = -7142,80$. Отримані калібрувальні криві для кумулятивної та щоденно доданої кількості інфікованих випадків показані у вигляді суцільних червоних кривих на рисунку. 3.4. Наприклад, станом на 31 січня зареєстрована сукупна кількість інфікованих випадків становила 7153, але калібрована кількість інфікованих випадків становить 17911, зі збільшенням на 10758 випадків. Як буде показано далі, цей крок контролю якості даних (QC) допомагає покращити роботу MCMC.

Ми застосували запропоновані нами моделі, алгоритми та пакет R eSIR для аналізу даних щодо COVID-19, зібраних із загальнодоступного вебсайту. Найдавніші публічні записи про провінційні дані доступні з 20 січня 2020 року. Згідно з існуючим пакетом R на GitHub, загальна кількість підтверджених інфекцій та смертей датується 13 січня 2020 року. Припущено, що до 17 січня всі зареєстровані випадки та смерті були з провінції. Імпутували відсутні випадки за 18-19 січня за допомогою лінійної інтерполяції. Таким чином, дані, використані в нашому аналізі, починаються з 13 січня. Дані, використані в аналізі для інших провінцій, починаються з 23 січня, що є найдавнішим часом з ненульовими значеннями у видаленому компартменті. Зауважте, що існують деякі незначні розбіжності між різними джерелами даних, а також проблема заниження звітності вирішуються в Додатку С за допомогою процедури калібрування. Цей підрозділ має на меті забезпечити демонстрацію нашого програмного забезпечення для аналізу поточних публічних даних щодо COVID-19, за допомогою якого користувачі можуть зрозуміти запропоновані методи. Ми зосереджуємося на ілюстрації способів експорту та інтерпретації результатів

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

МСМС. Пакет R може бути застосований для аналізу даних про інфекції з інших країн.

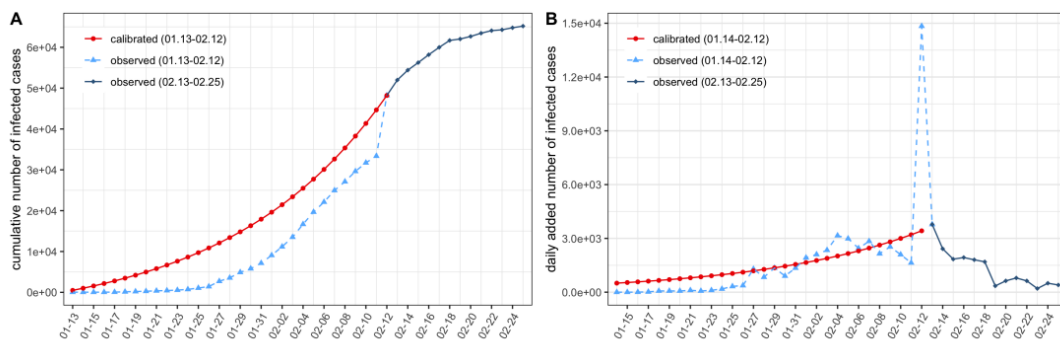


Рисунок 3.4 - Калібрування даних щодо заниженої кількості зареєстрованих випадків інфікування.

(А) Сукупна кількість зареєстрованих випадків інфікування. (В) Щоденна кількість нових зареєстрованих випадків інфікування. Калібрування даних проводилося з 13 січня по 12 лютого, що відображено червоними кривими.

Спочатку ми показуємо аналіз даних про COVID-19. Зауважте, що опція `dic=T` дозволяє розрахувати критерій інформації про відхилення (DIC) для вибору моделі, тоді як опції `save_files=T` та `save_mcmc` дозволяють зберігати вихідні таблиці, графіки, зведену статистику та навіть повні малюнки МСМС, які можна зберегти через шлях `file_add` або іншим чином через поточний робочий каталог. Основні результати, що повертаються пакетом, включають прогнозовані кумулятивні пропорції, прогнозовані точки повороту, два об'єкти `ggplot2` [36] зведених графіків, пов'язаних як з інфекцією, так і з видаленими компартментами, зведену вихідну таблицю, що містить усі апостеріорні середні значення, медіану та достовірні інтервали параметрів моделі, а також DIC, якщо увімкнено. Також надаються трасування та інші корисні діагностичні графіки, які автоматично зберігаються, якщо увімкнено `save_files=T`. У пакеті функція `tvteSIR()` працює над епідеміологічною моделлю зі змінною в часі швидкістю передачі, та `qh.eSIR()` для іншої епідеміологічної моделі з карантинним компартментом. Зверніть увагу, що у функції `tvteSIR()` з вибором `exponential=F`, у МСМС виконується ступінчаста функція, коли вказані як `change_time`, так і

π_0 . Щоб підігнати модель до функції модифікатора безперервної швидкості передачі, користувач може встановити $\text{exponential}=T$ та вказати значення λ_0 . За замовчуванням запускається базова епідеміологічна модель без карантину або $n(t) = 1$. death_in_R зазвичай встановлюється як середнє співвідношення часток смертності та видалених тварин у кожній точці часу спостереження, яке використовується для оцінки кривої смертності на графіку прогнозу видаленого відсіку. Нижче наведено R-скрипти, що використовуються в аналізі.

У наведених вище сценаріях кодування на R для врахування часу виконання задано лише дуже короткі ланцюги MCMC. На практиці рекомендується встановлювати $M=5e5$ та $\text{nburnin}=2e5$ для досягнення бажаних результатів вигорання та стабільних результатів MCMC. Ми спробували ступінчасту функцію, наведену на панелі C на рисунку де $\pi_0=c(1,0.9,0.5,0.1)$, експоненціальна функція, задана панеллю B зі швидкістю $\lambda_0=0.05$, обидві з яких порівнювалися з базовою моделлю з $n(t) = 1$. Результати трьох модифікаторних функцій, отриманих з функції `tvteSIR`. На цих прогностичних графіках інфікованих та видалених компартментів (панель A та панель C) чорні точки ліворуч від синьої вертикальної лінії позначають спостережувані пропорції інфікованих та видалених компартментів на останню дату доступних спостережень або раніше. Тобто, синя вертикальна лінія позначає час t_0 .

Зелена та фіолетова вертикальні лінії позначають першу та другу поворотні точки відповідно. Площа лососевого кольору позначає 95% достовірний інтервал прогнозованих пропорцій Y_i та Y_i^R після часу t_0 відповідно, тоді як площа блакитного кольору представляє або 95% достовірні інтервали поширеності ($0^{^1}$) або ймовірності видалення ($0^{^R}$) до часу t_0 .

Лістинг 3.2 - R-скрипти, що використовуються в аналізі

```
### Example 1: Step function pi(t)
### Y and R are observed proportions of infected and removed compartments
change_time <- c("01/23/2020", "02/04/2020", "02/08/2020")
pi0<- c(1.0,0.9,0.5,0.1)
res.step <- tvteSIR(Y,R,begin_str="01/13/2020",death_in_R = 0.4,
  T_fin=200,pi0=pi0,change_time=change_time,dic=T,
  casename="Hubei_step",save_files = T,
```

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

        save_mcmc=F,M=5e2,nburnin = 2e2)
res.step$plot_infection
res.step$plot_removed
res.step$dic_val

### Example 2: continuous exponential function pi(t)
res.exp <- tvteSIR(Y,R,begin_str="01/13/2020",death_in_R = 0.4,
  T_fin=200,exponential=TRUE,dic=F,lambda0=0.05,
  casename="Hubei_exp",save_files = F,save_mcmc=F,
  M=5e2,nburnin = 2e2)
res.exp$plot_infection
#res.exp$plot_removed

### Example 3: the basic state-space SIR model, pi(t)=1
res.nopi <- tvteSIR(Y,R,begin_str="01/13/2020",death_in_R = 0.4,
  T_fin=200,casename="Hubei_nopi",save_files = F,
  M=5e2,nburnin = 2e2)
res.nopi$plot_infection
#res.nopi$plot_removed

```

Сіра та червона криві – це криві апостеріорного середнього та медіанного значень. Чорна крива на графіку (Панель С) – це частка смертей, оцінена за заздалегідь визначеним співвідношенням death_in_R . Середня панель В відображає важливі динамічні особливості інфекції за допомогою спагеті-діаграми, на якій 20 випадково вибраних МСМС малюють похідну першого порядку апостеріорної поширеності інфекції, а саме $0'$. Чорна крива – це апостеріорне середнє похідної, а вертикальні лінії позначають час поворотних точок, що відповідають відповідно тим, що показано на панелі А та панелі С. Крім того, 95% достовірні інтервали цих поворотних точок також виділені напівпрозорими прямокутниками на панелі В. Як показано на цих рисунках, модифікатор швидкості передачі $n(t)$ відіграв важливу роль у скороченні ключових поворотних точок епідемії, і його вплив як на оцінку, так і на прогнозування динаміки інфекції COVID-19 був чітко продемонстрований.

Далі ми проаналізували дані решти населення, починаючи з 23 січня. Ми встановили $\text{begin_str} = "01/23/2020"$ у tvteSIR . Щоб врахувати затримку часу початку, ми включили дві точки зміни для ступінчастої функції $n(t)$ на [4 лютого, 8 лютого] з $n_0 = (0,8, 0,1)$. Експоненціальна функція залишилася незмінною.

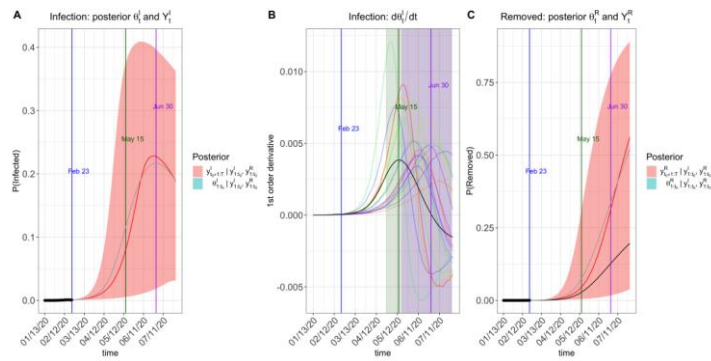


Рисунок 3.5 - Графіки прогнозів без втручання.

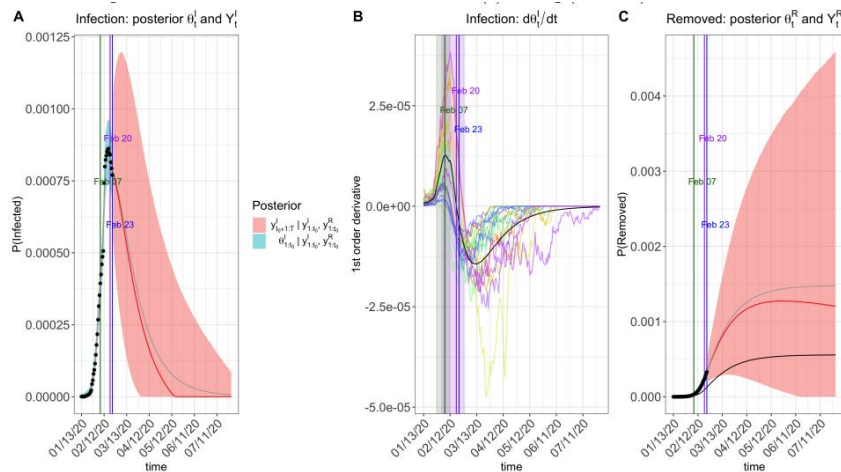


Рисунок 3.6 - Прогнозні графіки з експоненціальним модифікатором швидкості передачі $n(t) = \exp(-0,05t)$.

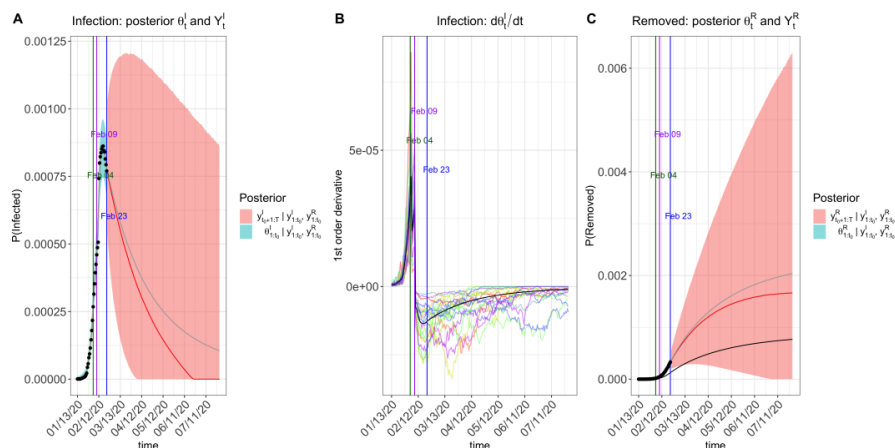


Рисунок 3.7 - Діаграми прогнозування) зі східчастою функцією модифікатора швидкості передачі, що визначається $n_0 = (1, 0,9, 0,5, 0,1)$ у точках зміни

Лістинг 3.3 - Епідеміологічна модель з доданим карантинним компартментом

Змн.	Арк.	№ докум.	Підпис	Дата

БР.ІІ – 01.00.00.000 ПЗ

Арк.

65

```

### Example 4: Dirac delta function of the quarantine process
change_time <- c("01/23/2020","02/04/2020","02/08/2020")
phi <- c(0.1,0.4,0.4)
res.q <- qh.eSIR (Y,R,begin_str="01/13/2020",death_in_R = 0.4,
                  phi0=phi0,change_time=change_time,
                  casename="Hubei_q",save_files = T,save_mcmc = F,
                  M=5e2,nburnin = 2e2)
res.q$plot_infection

```

Іншу епідеміологічну модель з доданим карантинним компартментом як поглинаючим станом було підібрано за допомогою нашої R-функції qh.eSIR у пакеті eSIR . Аргументи, що використовуються в qh.eSIR() , майже ідентичні аргументам у tvf.eSIR() . Зауважте, що якщо функція карантинного коефіцієнта встановлена на константу 0, ця модель буде зведена до базової епідеміологічної моделі SIR. COVID-19 за межами поки що протікає значно менш тяжко. Можливі причини такого низького рівня інфікування та смертності включають (i) припинення транспортного сполучення між містами, (ii) своєчасне вжиття запобіжних заходів та профілактичних заходів, а також (iii) порівняно велику чисельність населення, що розріджує групу, що зазнала впливу. Коли панель А на рисунку 3.8 збільшено, деякі спостережувані пропорції (чорні крапки) відхиляються від апостеріорного середнього значення або медіани підбраної поширеності, хоча всі вони потрапляють у 95% достовірні інтервали, як показано на панелі

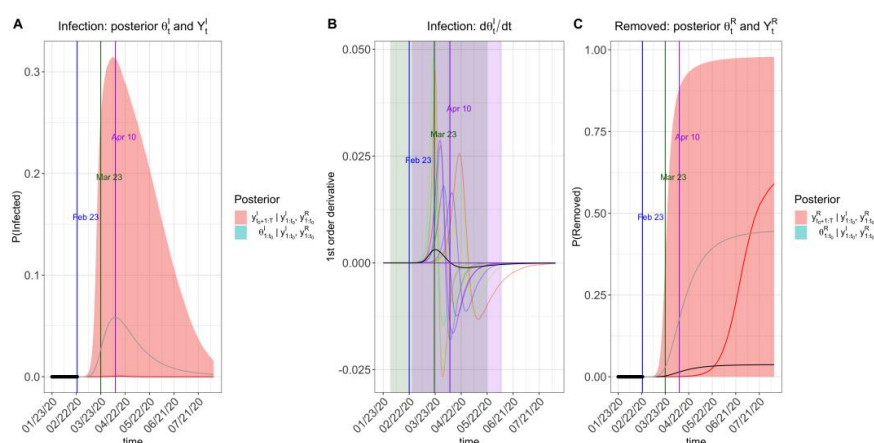


Рисунок 3.8 - Графіки прогнозів для інших міст межами з $\text{pr}tq \approx 1$

А на рисунках 3.9 та 3.10. Оскільки латентний процес відповідає диференціальним рівнянням SIR, модель SIR може бути непридатною для дуже великої та складної популяції в 1,3 мільярда людей, в якій більшість суб'єктів не

					БР.ІІ – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		66

перебувають у групі ризику. Запропоновані моделі повинні працювати набагато краще для окремих провінцій, але ми не проводили таких аналізів.

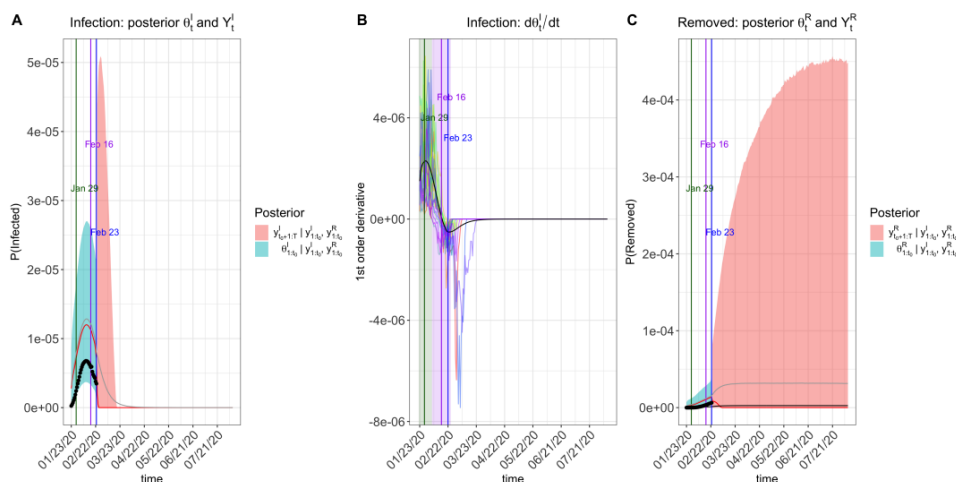


Рисунок 3.9 - Графіки прогнозів для інших з експоненційним модифікатором швидкості передачі $\gamma(t) = \exp(-0,051t)$.

Лістинг 3.4 - Функція R qh.eSIR для аналізу даних у межах

```
#res.q$plot_removed

### Example 5: basic state-space SIR model
res.noq <- qh.eSIR (Y,R,begin_str="01/13/2020",death_in_R = 0.4,
                    T_fin=200,casename="Hubei_noq",
                    M=5e2,nburnin = 2e2)
res.noq$plot_infection
```

Ми застосували функцію R qh.eSIR для аналізу даних у межах та за межами. Результати зведені на рисунках. 3.11-3.13 . Наші аналізи знову чітко показали , що суворі карантинні протоколи можуть значно зменшити поширення COVID-19 як у межах провінції Хубей , так і за її межами. Однак відомо, що надто суворий карантин може мати зворотний ефект; люди можуть втратити довіру та терпіння до своєї відданої системи, і, як наслідок, можуть спробувати зменшити дотримання або навіть уникнути карантину. Ми також представляємо апостеріорну середню ймовірність перебування в карантинному відсіку на рисунку. 17 років у межах Хубея та за його межами. Зауважте, що 23 січня не було встановлено як точку зміни для випадку за межами Хубея, що призвело

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		67

лише до двох стрибків. Очевидно, що до 8 лютого понад 90% населення Китаю перебувало в домашній ізоляції або, як таке, відповідно до дуже суворого карантинного протоколу, що діє по всій країні.

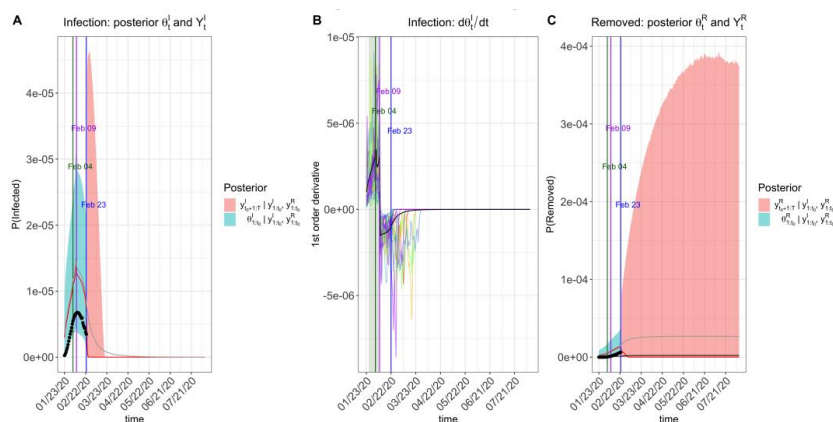


Рисунок 3.10 - Діаграми прогнозування для інших з модифікатором швидкості передачі зі східчастою функцією, що визначається як $7 \text{ TQ} = (1, 0,9, 0,5, 0,1)$ у точках зміни

Як описано виправлено занижену частку інфекцій до 12 лютого, коли в один день стався значний стрибок. Ми повторили той самий аналіз, використовуючи калібровані дані про інфекції, і відповідні результати показано на рисунках. 3.11-3.13 .

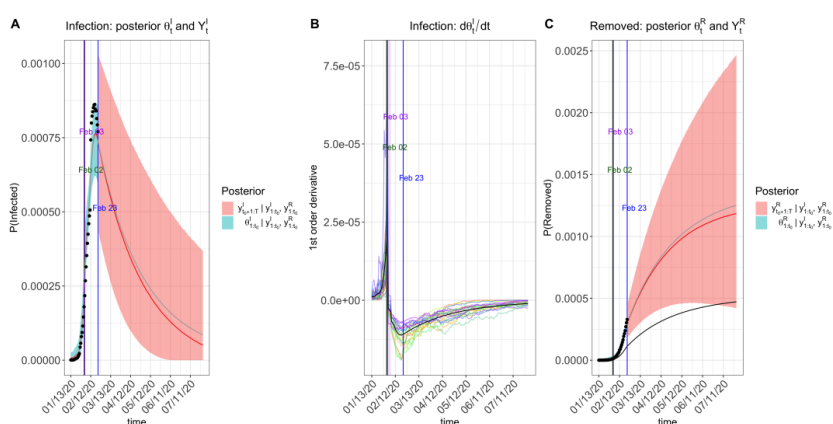


Рисунок 3.11 - Графіки прогнозів із карантинними обмеженнями, що змінюються в часі, визначеними за допомогою параметра $\phi_0 = [0,1; 0,9; 0,5]$ у точках зміни

Цікаво спостерігати, що різке зростання частки інфікування 12 лютого зникло на всіх панелях А в цих нових аналізах, а спостережувані дані (тобто чорні точки) краще узгоджуються з достовірні інтервали всіх латентних процесів. Ми також помічаємо, що після корекції оцінені числа відтворення R_0 стали меншими. Числа відтворення, оцінені за різними моделями для провінції Хубей та за її межами, з калібруванням даних та без нього, разом з їхніми 95% достовірними інтервалами, зведені в таблиці 3.1.

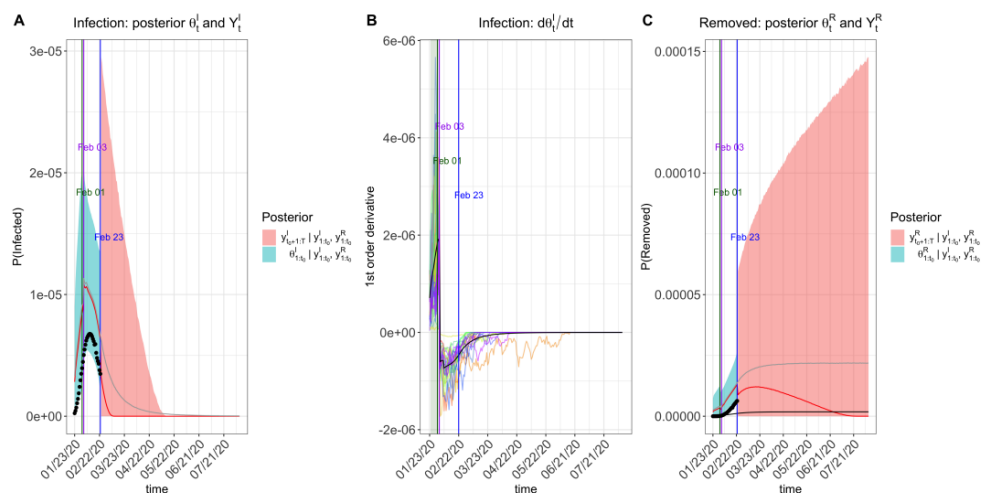


Рисунок 3.12 - Графіки прогнозування для інших з карантинними обмеженнями, що змінюються в часі, визначеними за допомогою $\phi_0 = [0,9; 0; 5]$ у точках переходу.

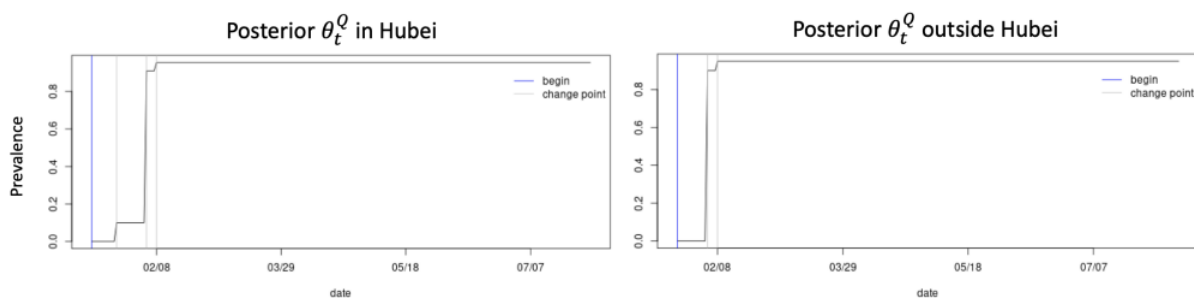


Рисунок 3.13 - Середня ймовірність перебування в карантинній групі в сістах та за межами.

Варто зазначити, що оцінки репродуктивних чисел, отримані з епідеміологічних моделей зі змінними в часі показниками передачі або карантину, видаються більшими, ніж ті, що отримані з базової моделі без

карантину. Це не дивно, оскільки наші нові моделі явно враховують втручання, так що оцінений R_0 є скоригованим числом без впливу втручань.

Таблиця 3.1

Апостеріорне середнє значення та достовірні інтервали репродуктивного числа R_0 , отримані з різних карантинних моделей та наборів даних.

Модель	В межах міста: Без калібрування даних (Mean 95% CI)	В межах міста: З калібруванням даних (Mean 95% CI)	За межами міста (Mean 95% CI)
Без карантину (No quarantine)	3.00 [1.91, 4.50]	2.96 [1.82, 4.49]	2.58 [1.52, 4.26]
Експоненціальна (Exponential)	6.30 [2.80, 10.78]	4.88 [2.35, 8.42]	3.36 [1.96, 5.26]
Ступінчаста функція (Step-function)	5.61 [2.89, 8.95]	4.31 [2.34, 6.90]	3.02 [1.74, 4.89]
Карантинний відсік (Quar. Compart.)	6.00 [2.35, 13.24]	5.04 [2.30, 9.36]	3.58 [1.85, 6.10]

Навпаки, базова модель без використання модифікатора карантину неявно інтегрує вплив втручань у показник передачі β , і, отже, оцінений R_0 зменшується завдяки внеску втручань. Наші аналізи показують, що репродуктивні показники R_0 COVID-19 без втручань у сфері охорони здоров'я становили б близько 4-5 у межах міста та близько 3-3,5 за його межами з відносно великими достовірними інтервалами. Ці результати узгоджуються з висновками [17]. Як зазначалося вище, розмір достовірного інтервалу може бути зменшений за рахунок більш доступних даних про COVID-19, що дозволяє користувачам вказувати менші відхилення в попередніх розподілах.

3.3 Аналіз результатів моделювання та оцінка ефективності програмного забезпечення

Запропонована модель K-SEIR є вдосконаленою моделлю SEIR, яка враховує вплив як природного поширення, так і втручання людини. Параметр K людського втручання додатково поділяється на три параметри: k_1 , k_2 та k_3 .

Одночасно ми розклали параметр R на два параметри: H та D. Потім ми створили модель K-SEIR (рис. 3.14).

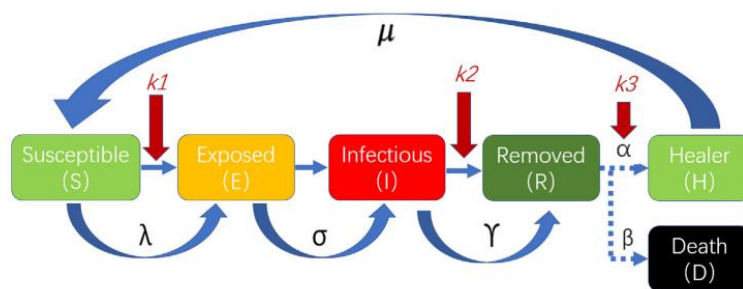


Рисунок 3.14 - Схематичне зображення логічної структури моделі K-SEIR-Sim.

Модель K-SEIR-Sim складається з таких категорій: сприйнятливі (S), інфіковані (E), заразні (I), вилікувані (R), ті, хто одужав (H), та померлі (D). Параметром заходів втручання людини є K, який включає три параметри: k_1 - міра фізичної ізоляції; k_2 - міра пропускної спроможності лікарень; k_3 - міра ефективності медичного лікування.

Для загальної чисельності населення 100 000 осіб параметр рівня зараження (k) та рівня видалення (y) було змодельовано як 0,01 та 1 відповідно. На рис. 2 показано результати запуску прикладу програми. На основі чисто числового та теоретичного моделювання ми можемо визначити численні особливості цієї моделі. Загальна кількість інфікованих випадків дуже низька або дуже висока, а проміжний стан дуже вузький. Коли загальна кількість інфікованих випадків перевищує 90% або менше 10%, інфекція швидко досягає піку, і епідемія також швидко закінчується. Коли кількість інфікованих випадків становить від 10% до 90%, інфекція повільно досягає піку, а епідемія триває довше. Щоб стримати поширення епідемії, рівень зараження слід мінімізувати, а рівень видалення – покращити. Однак для цього слід враховувати співвідношення витрат і вигод. Рівень смертності є основним фактором при оцінці співвідношення витрат і вигод. Наприклад, грип має дуже низький рівень смертності, і йому можна дозволити розвиватися незалежно від темпів зараження та видалення. Зміни k та y також можна спостерігати за втручання людини; отже, модель все ще доступна за умови втручання людини.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

Наведений вище аналіз моделювання моделі K-SEIR свідчить про те, що для стримування поширення епідемії. По-перше, захід фізичної ізоляції (k^1) можна використовувати для зниження щоденного природного рівня зараження (k). По-друге, рівень видалення можна покращити, збільшивши доступність медичного лікування. Однак збільшення рівня видалення, а не збільшення рівня одужання, прискорить ліквідацію епідемії, що не допоможе покращити рівень виживання населення. Отже, медичне лікування поділяється на два параметри: k^2 (місткість лікарень) та k^3 (можливість медичного лікування), що є дуже цінним у застосуванні.

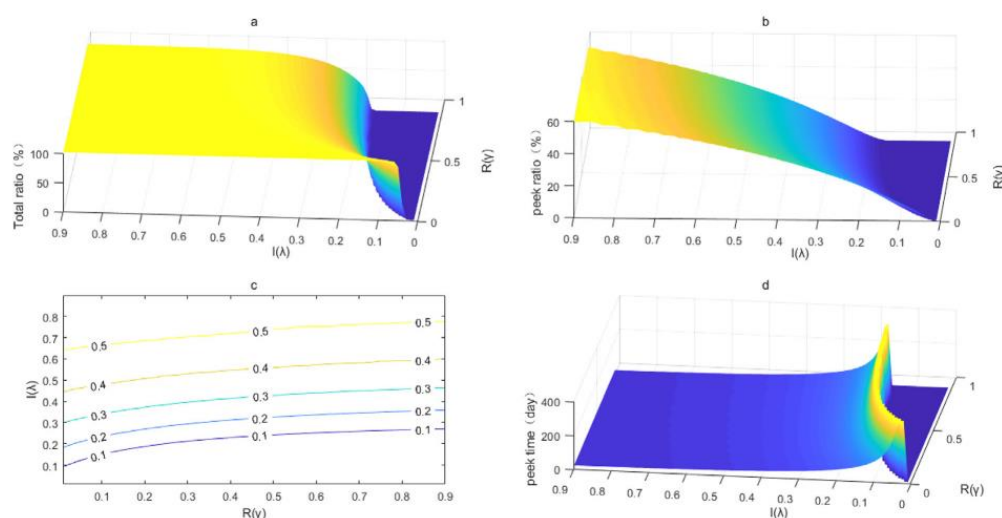


Рисунок 3.15 - Характеристики епідемії за результатами моделювання в умовах природного поширення.

а) Загальна кількість інфікованих. Коли коефіцієнт інфікування перевищує 0,3, у природних умовах зрештою інфікуються майже всі. Коли коефіцієнт інфікування менший за 0,3, вища швидкість виведення з популяції відповідає меншій загальній кількості інфікованих. Спостерігається різке зростання від невеликої кількості інфекцій у нижній частині до майже всіх інфекцій, з вузькою зоною переходу параметрів. б, Пікова кількість інфікованих осіб. Чим вищий коефіцієнт інфікування, тим нижчий коефіцієнт видалення і тим більша пікова кількість інфікованих осіб (до 60% від загальної кількості інфікованих осіб). с, Контурна залежність між часткою пікових інфекцій та

коефіцієнтом інфікування (k) і коефіцієнтом видалення (c). Щодо контурної залежності між часткою пікових інфекцій та k і c , частка пікових інфекцій менше 10 % спостерігається лише в дуже вузькому діапазоні розподілу параметрів. d , Час піку. Щодо часу піку, коли кількість інфікованих людей є високою або низькою, час піку є коротким. У регіонах, де кількість підтверджених випадків зростала стрімко, пік наставав пізніше, а спалах тривав довше, навіть понад 365 днів.

Оскільки поточні моделі базуються на мовах математичного програмування, вони підходять лише для професіоналів. Тому було створено універсальне програмне забезпечення для суспільного добробуту (K-SEIR-Sim V2.53) для моделювання розвитку інфекційних захворювань на основі нашої вдосконаленої моделі K-SEIR (рис. 3.16).



Інтерфейс користувача включає три основні функціональні модулі: налаштування параметрів, моделювання епідемічних ситуацій та сканування параметрів. Програма може працювати у трьох режимах: на основі моделі, на основі даних або у змішаному режимі, залежно від потреб користувачів.

Дослідники можуть вручну встановлювати комбінації параметрів в інтерфейсі програмного забезпечення, щоб отримати результати моделювання або прогнозування. Вони також можуть довільно коригувати параметри під час

виконання процесу, без необхідності навичок програмування. Крім того, програмне забезпечення має функцію автоматичного сканування параметрів, завдяки чому після імпорту фактичних даних епідемії (наприклад, підтверджених випадків, випадків одужання та смертей) функція сканування параметрів може бути використана для отримання поточної оптимальної комбінації параметрів, що дозволяє згодом робити прогноз. Таким чином, залежно від потреб користувачів, програмне забезпечення може функціонувати двома способами: на основі моделі та на основі даних. Воно також може виконувати аналіз у режимі реального часу, прогнозний та ретроспективний, вирішуючи практичні потреби.

Програмне забезпечення працює або на основі даних, або на основі моделі. Обидва вони складаються з п'яти операційних кроків (рис. 3.17). Операція на основі даних складається з наступних п'яти

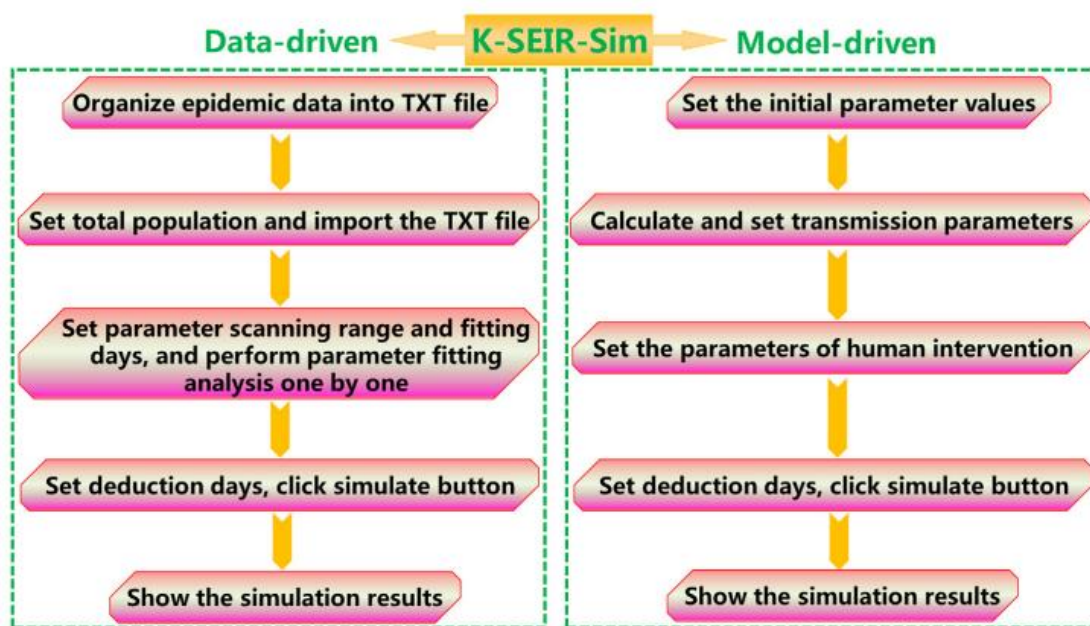


Рисунок 3.17 - Посібник користувача до програмного забезпечення K-SEIR-Sim

Як режим роботи на основі даних, так і режим роботи на основі моделей складаються з п'яти етапів, проте деякі операції відрізняються.

Кроки: створити ТХТ-файл епідемічних даних, імпортувати ТХТ-файл, иконати аналіз параметрів, виконати моделювання, відобразити результати моделювання. П'ятиетапна операція методу на основі моделі полягає в наступному: встановити початкові параметри, встановити параметри передачі встановити параметри втручання людини, виконати моделювання, відобразити результати моделювання. ТХТ-файл на основі епідемічних даних для методу на основі даних охоплює вразливе населення, латентні інфекції, підтверджені - випадки, смерті та дату початку. Для методу на основі моделі початкові параметри включають загальну чисельність населення, вразливе населення, латентні інфекції, підтверджені випадки, смерті та дату початку; параметри передачі базуються на епідеміологічному обстеженні.

Дані включають рівень зараження, захворюваність та рівень видалення; а параметри втручання людини включають інтенсивність ізоляції, місткість лікарень та переваги одужання. Результати моделювання для обох режимів роботи показують кумулятивну кількість та щоденний приріст кожної популяції, дату піку, дату ремісії, дату закінчення тощо.

Коли програмне забезпечення вперше запускається недосвідченою людиною, і коли детальна інформація недоступна на початковій стадії епідемії, як використовувати це програмне забезпечення для моделювання поширення? Є кілька застережень, на які слід звернути увагу, див. Додаткові дані S2 . Для загального використання див. приклади нижче.

Ми провели аналіз підгонки, щоб підтвердити надійність запропонованого програмного забезпечення. Під час аналізу підгонки ми повинні спочатку оцінити, чи збігаються змодельовані та фактичні дані (рис. 3.18). Ми взяли дані з Франції як приклад для проведення аналізу підгонки, і результати показують, що програмне забезпечення працює добре, а результати моделювання є надійними (див. Додаткові дані S3). Крім того, ми зібрали дані з офіційних статистичних даних епідемії у Сполучених Штатах з 1 березня по 30 квітня 2020 року для проведення аналізу підгонки. За результатами цього аналізу підгонки ми отримали оптимальні параметри природної передачі на кожному етапі та відповідні параметри втручання людини. На основі цих параметрів ми

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

провели порівняння між результатами дедукції та офіційними даними, запустивши програмне забезпечення. Порівняння показало, що дані дедукції добре узгоджуються з офіційними даними для підтверджених випадків (I), випадків одужання (H) та смертей (D).

На наступному кроці ми повинні дослідити похибки аналізу підбору (рис. 3.19). З 1 по 22 березня 2020 року було 12 днів (що становить приблизно половину днів), коли похибки зараження та смерті були в межах 10%, тоді як похибка одужання в межах 10% траплялася лише протягом 3 днів. З 23 березня по 30 квітня (коли уряд запровадив карантинні заходи) 15 днів (37,5%) відображали похибку зараження в межах 4%, понад 40% - з похибкою одужання в межах 4%, а 25 днів (більше 50%) відображали похибку смерті в межах 4% (рис. 6). Загалом, точність підбору була вищою за 96%. Відсоток днів з похибками від 4% до 10% був таким: 50% (помилка зараження), 30% (помилка одужання) та 37,5% (помилка смерті). На завершення, коли модель, запропонована в цій роботі, використовується для апроксимації існуючих даних, точність апроксимації може бути збережена на рівні понад 90%.

Ми також повинні узагальнити параметри аналізу підбору. Спочатку ми отримуємо параметри, запускаючи програмне забезпечення (рис. 3.15). На рис. 3.16 показано оптимальні параметри динаміки передачі спалаху в Сполучених Штатах у різні періоди, починаючи з 1 березня по 30 квітня 2020 року. Параметр рівня зараження зменшується, що свідчить про ефективність заходів домашнього карантину, прийнятих урядом, та поступове покращення обізнаності людей щодо профілактики та контролю. Оскільки уряд запровадив заходи профілактики та контролю 23 березня, коефіцієнт k_1 збільшився з 0 до 1 і досяг 0,948 30 квітня (згідно з рівнянням 1). Це означає, що середня людина обмежує 95% своєї соціальної інтенсивності, що показує, що заходи домашньої ізоляції, прийняті урядом та громадськістю, досягли максимального стану, і можливість подальшого збільшення k_1 є незначною. Якщо уряд відновить роботу та виробництво, пік може зрости. Інший параметр – коефіцієнт видалення – є незначним і відносно стабільним. Таким чином, кількість людей, вилучених з підтвердженої кількості, дуже мала, що свідчить про те, що велика кількість

					БР.ІІ – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		76

інфікованих пацієнтів все ще перебуває у стані діагнозу. Вплив параметра медикаментозного лікування (k^2) не є очевидним (фактичний коефіцієнт вилучення = початковий коефіцієнт вилучення $\times k^2$ (рівняння 2); коефіцієнт вилучення майже однаковий; отже, k^2 майже однаковий, оскільки як тільки місткість лікарні досягає насичення, вона більше не може приймати пацієнтів. Однак рівень смертності невисокий. Третій параметр рівня одужання зростає. Оскільки фактичний коефіцієнт одужання дорівнює початковому коефіцієнту одужання $\times k^3$, початковий коефіцієнт одужання становить приблизно 0,33, виходячи з попереднього аналізу даних. Можна зробити висновок, що станом на 30 квітня збільшення коефіцієнта одужання, k^3 , становить 2,23, що свідчить про покращення рівня медикаментозного лікування та коефіцієнта успішності одужання.

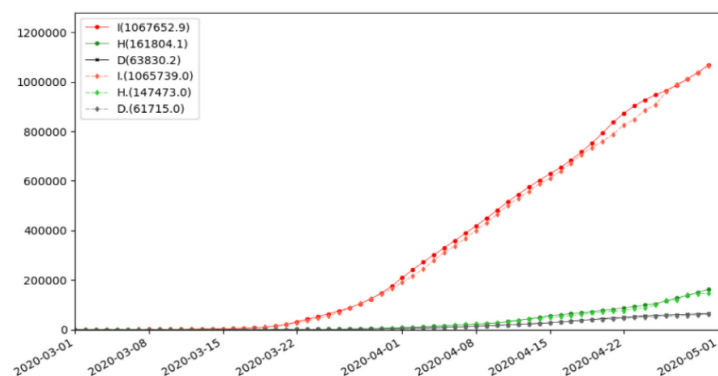


Рисунок 3.18 - Порівняння результатів розрахунків та офіційних даних за допомогою імітаційного аналізу.

Надійність параметрів необхідно перевірити, запустивши програмне забезпечення. На основі даних про оптимальні параметри, отримані шляхом підгонки історичних даних, було проведено 10-денний аналіз прогнозування та дедукції. Порівнюючи результати цього аналізу з опублікованими даними та відповідним розрахунком похибки, ми бачимо, що результати, передбачені моделлю, добре узгоджуються з опублікованими даними, а відповідна похибка здебільшого знаходиться в межах 10%.

Модельно-орієнтований аналіз інфекційних захворювань є ще одним важливим застосуванням запропонованого програмного забезпечення. Після введення розрахованих параметрів у програмне забезпечення з використанням відкритих даних ми запускаємо програмне забезпечення та отримуємо результати. Ми беремо дані з Diamond Princess як приклад для проведення моделювання, і результати показують, що програмне забезпечення працює добре, а результати моделювання є надійними. Крім того, параметри, розраховані з використанням даних з Уханя, Китай, 23 січня, в день локдауну в Ухані, вводяться в програмне забезпечення для прогнозування ситуації через 60 днів.

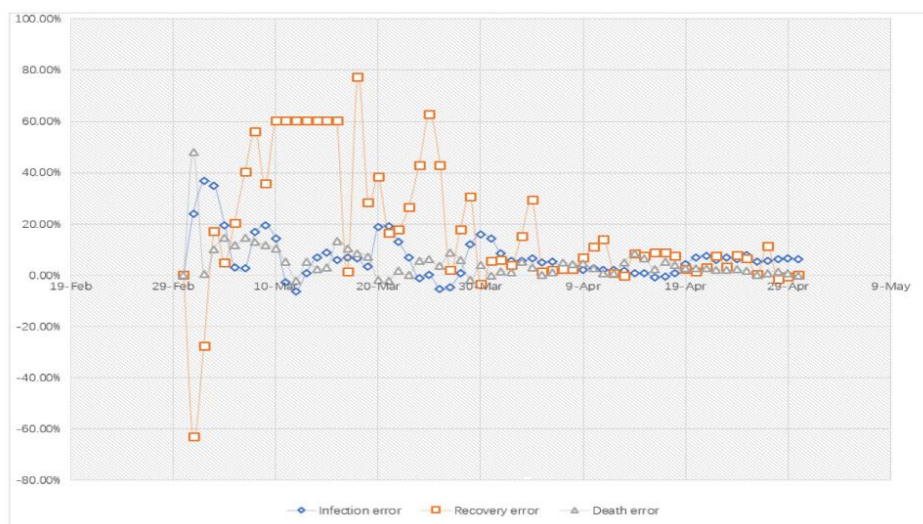


Рисунок 3.19 - Коливання похибок узгодження між модельованими та офіційними даними з 1 березня по 30 квітня.

Порівняння показало, що розрахункові дані добре узгоджуються з офіційними даними, оскільки відповідні криві в основному збігаються. На осі абсцис показано час, а на осі ординат - чисельність населення. Три верхні криві на графіку у верхньому лівому куті рисунка відображають розрахункові дані, тоді як три нижні криві - офіційні дані. І - кількість підтверджених випадків (висновок); Н - кількість вилікуваних, тобто кількість випадків одужання (висновок); D - кількість померлих (висновок). І - кількість підтверджених

випадків (офіційні дані); Н - кількість вилікуваних, тобто кількість випадків одужання (офіційні дані); D - кількість померлих (офіційні дані).

З 1 по 22 березня похибки щодо випадків зараження та смертності утримувалися в межах 10 % протягом 12 днів, тоді як похибка щодо одужань утримувалася в межах 10 % лише протягом 3 днів. З 23 березня по 30 квітня похибка щодо інфікування не перевищувала 4% протягом 15 днів, тоді як похибки щодо одужання та смертності не перевищували 4% протягом 25 днів. Усі ці показники точності моделювання можна підтримувати на рівні понад 90%. Негативна похибка вказує на те, що значення моделювання є меншим за офіційні дані.

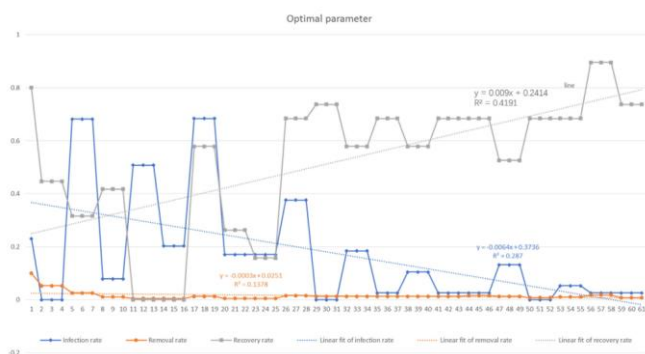


Рисунок 3.20 - Оптимальні динамічні параметри поширення вірусу в різні періоди.

Темпи зараження знижуються. 23 березня коефіцієнт k зріс з 0 до 1 і 30 квітня досяг 0,948. Коефіцієнт виведення з організму дуже низький і відносно стабільний. Ефект від медичного лікування (k_2) не є очевидним, а рівень смертності не є високим. Коефіцієнт одужання зріс з 0,33 1 березня до 0,74 30 квітня. На осі абсцис показано дні, починаючи з 1 березня (рис. 3.21). У разі вільної передачі було б понад 5 мільйонів одужань та понад 4 мільйони смертей. Однак, локдаун спричинив лише 3961,8 смертей та 46 312,1 одужань, що близько до відповідних показників 3814 та 43 214. Ці результати повністю демонструють ефективність запобіжних заходів ізоляції та контролю.

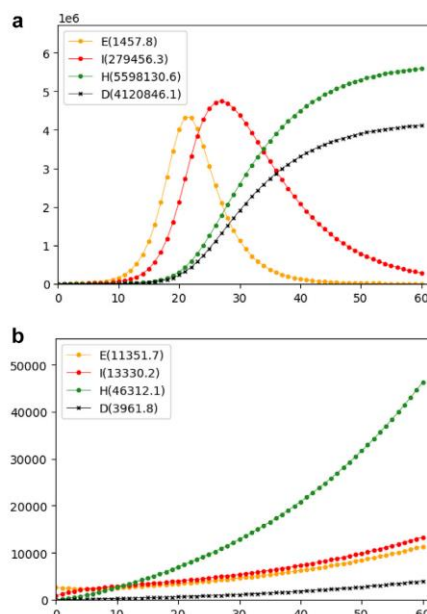


Рисунок 3.21 Модельний аналіз ситуації з COVID-19

Модельний аналіз ситуації з COVID-19 проводиться шляхом введення розрахованих параметрів у програмне забезпечення з використанням загальнодоступних даних з міста Ухань, Китай, станом на 23 січня. Програмне забезпечення змоделювало ситуацію через 60 днів. У разі вільного поширення вірусу було б понад 5 мільйонів одужавших і понад 4 мільйони смертей, тоді як карантин призвів до 3961,8 смертей і 46 312,1 одужавших. На осі абсцис показано час (у днях), а на осі ординат - чисельність населення.

Для подолання кризи в галузі охорони здоров'я та соціальної кризи, спричиненої інфекційними захворюваннями, такими як COVID-19, було запропоновано багато інструментів підтримки рішень, які зазвичай базуються на клінічних математичних моделях [9]. Ці моделі класифікуються на діагностичні та прогностичні [22]. На жаль, нещодавній огляд продемонстрував низьку якість цих моделей, і жодна з них не може бути рекомендована для клінічного використання [9]. Методологічні недоліки та невідомі характеристики нових захворювань у невідкладних ситуаціях можуть бути першопричиною цього. Розробка моделі клінічного аналізу повинна бути наукою та мистецтвом [22]. Для покращення якості моделі прогнозування багато вчених ввели такі фактори, як технології штучного інтелекту [10,23], R-пакети [24], заходи втручання [25], госпіталізація та потреби у відділенні інтенсивної терапії [26]. Також були

створені програми для допомоги в цій галузі, включаючи відкритий шаблон електронної медичної картки для покращення взаємодії в Китаї [27], додаток для смартфонів для відстеження контактів між особами у Сполучених Штатах [28] та інтерактивний веб-додаток для комплексної звітності в Чеській Республіці [29]. Тут ми розробили узагальнене та просте в експлуатації програмне забезпечення, яке важливе для розвитку клінічних застосувань. Зазначимо, що під час перегляду нашого рукопису ми знайшли онлайн- програмне забезпечення під назвою Epidemic Calculator, який також розроблений на основі моделі SEIR. Він дуже плавний з точки зору налаштування параметрів та представлення результатів. Важливо зазначити, що розроблене нами програмне забезпечення відрізняється від програмного забезпечення веб-сайту трьома основними аспектами. По-перше, наше програмне забезпечення додає три контрольні параметри: k_1 , k_2 та k_3 , що може краще відображати реальну ситуацію. По-друге, наше програмне забезпечення може проводити аналіз на основі даних. Після імпорту існуючих фактичних даних ми можемо отримати параметри передачі, що краще сприяє аналізу подальшого розвитку епідемії. По-третє, наше програмне забезпечення можна завантажити для локального використання, не залежно від мережевих обмежень. Тому ми вважаємо наше програмне забезпечення дуже цінним для епідеміологічного аналізу.

Щоб краще відповідати фактичним потребам, ми додали до програмного забезпечення штучні параметри та деякі функції. Коли моделюється лише природна передача, штучні параметри не потрібно вибирати. Коли моделюється передача під втручанням людини, штучні параметри можна скоригувати відповідно до фактичних вимог. Більше того, всі параметри можна скоригувати в будь-який час під час процесу моделювання, що може імітувати попередню ситуацію та прогнозувати майбутню тенденцію захворювання. Програмне забезпечення також може виконувати аналіз у реальному часі та підходить для COVID-19 та інших інфекційних захворювань. Таким чином, це програмне забезпечення може виконувати аналіз інфекційних захворювань у реальному часі, прогнозний та ретроспективний. Воно може працювати двома способами: на основі даних та на основі моделі. Метод, керований моделлю, означає запуск

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

програмного забезпечення для аналізу розвитку захворювання шляхом введення значень параметрів, тоді як метод, керований даними, означає запуск програмного забезпечення для отримання параметрів шляхом введення відкритих даних. Ми можемо вибрати метод на основі наших потреб, що доповнює особливу практичну цінність програми.

У методі, керованому даними, фактичні дані епідемії за певний період часу розглядалися як вхідні дані для програмного забезпечення для проведення аналізу підгонки з метою отримання поточних параметрів R_0 , рівня зараження, рівня видалення, рівня смертності, штучних параметрів тощо. Програмне забезпечення може оцінювати ефективність контролю з боку людини та ситуацію з епідемією. Хоча можуть виникнути деякі проблеми, пов'язані зі статистичним аналізом, щодо достовірності фактичних короткострокових даних, закон поширення епідемії можна отримати, використовуючи порівняно довгострокові дані. Наприклад, що стосується похибки відновлення, коли ми проводили аналіз похибок на прикладі даних Сполучених Штатів, ми виявили, що з 1 по 22 березня 2020 року існувала 3-денна похибка 10%. Значення похибки відновлення було відносно великим для решти днів, причому деякі перевищували 40%. Огляд офіційних даних показав, що в ці дні були значні прогалини в даних, які не відповідали фактичному поширенню вірусу. Розглядаючи кількість одужань як приклад, кількість тих, хто зцілювався на день, становила 7 з 2 по 4 березня, 9 з 5 по 8 березня та 12 з 9 по 16 березня. Більше того, офіційні дані не оновлювалися протягом кількох днів, що є основною причиною великої похибки одужання. Однак похибки зараження та смерті залишалися в межах відносно стабільного діапазону. Крім того, похибки одужання 12 та 18 березня аномально зросли, оскільки кількість виписаних пацієнтів, які одужали, не оновлювалася з 2 по 4 березня та з 17 по 18 березня; отже, спотворення даних призвело до коливань похибки. З 30 березня дані про похибку одужання стабілізувалися в межах розумного діапазону. Тому в Сполучених Штатах аналіз запропонованої моделі на основі даних має певне опорне значення для моделювання та прогнозування епідемічної ситуації. Його також можна використовувати в інших країнах або регіонах.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		82

Аналіз на основі даних стосується спроби зіставити фактичні дані. Однак на фактичні дані впливають багато статистичних аспектів; протягом деяких днів дані відсутні. Тому ми також розробили функцію аналізу на основі моделей.

Аналіз на основі моделей дуже простий у використанні та вимагає лише введення параметрів у програмне забезпечення. Примітно, що коли ми моделювали ситуацію через 60 днів після карантину в Ухані, кількість тих, хто одужав, була близькою до фактично зареєстрованої кількості, але кількість інфекцій була приблизно в 3 рази вищою за фактично зареєстровану кількість. Це пояснюється тим, що безсимптомні інфекції на той час не враховувалися. Згідно з великою кількістю аналізів літератури, безсимптомні інфекції становили приблизно 10%-60% від загальної кількості інфекцій. Таким чином, кількість наших змодельованих інфекцій також є відносно точною. Крім того, незважаючи на те, що кількість смертей становила 2524 за офіційними даними на 23 лютого, уряд додав 1290 до кількості смертей, коли пандемія в Ухані закінчилася. Таким чином, кількість смертей була близькою до фактично зареєстрованої кількості. Звичайно, якщо припустити, що існує абсолютно вільна передача і ми абсолютно нічого не можемо зробити для боротьби з цією хворобою, кількість інфекцій та смертей вражає. Однак така можливість не існує в сучасному суспільстві. Незважаючи на це, наш симуляційний аналіз показав корисність карантину в Ухані та впроваджених заходів контролю.

Мета полягає в широкому використанні цього програмного забезпечення. Використовуючи наше програмне забезпечення, ми провели аналіз однієї хвилі спалаху, моделюючи розвиток епідемії в різних регіонах різного розміру, починаючи від країни, міста і закінчуючи невеликим замкнутим простором. Ми отримали результати моделювання, які добре узгоджуються з фактичною ситуацією. Окрім моделювання однієї хвилі епідемії, згаданої вище, ми також використовували це програмне забезпечення для моделювання розвитку кількох хвиль епідемії, спричинених карантинном та його скасуванням у Франції. Це програмне забезпечення за замовчуванням моделює, використовуючи день як одиницю вимірювання. Застосування Франції показує,

що воно може моделювати, використовуючи тиждень як одиницю вимірювання за тривалої тривалості епідемії. Все, що нам потрібно зробити, це змінити початкові значення параметрів, розраховані за днями, на значення параметрів, розраховані за тижнем (що дорівнюють значенням параметрів, розрахованим за днями, помноженим на 7), та виконати ту саму операцію моделювання. Ми сподіваємося, що це корисний інструмент для широкого використання.

Теоретично, це програмне забезпечення може симулювати інші епідемії, окрім COVID-19. Для епідемії, яка не потребує втручання людини, ми просто не встановлюємо жодних параметрів K . Для епідемій без інкубаційного періоду достатньо встановити коефіцієнт захворюваності r на 0. Для епідемій, коли пацієнти можуть виробляти антитіла, але антитіла зрештою не дають результатів, нам потрібно лише встановити коефіцієнт невдачі антитіл m , який отримується за допомогою статистики. Одним словом, наше програмне забезпечення має велику практичну цінність.

Також варто зазначити, що модель SEIR, фактично будь-яка модель, має припущення про однорідність, тобто, що люди мають однакову ймовірність заразитися хворобою, і що люди дотримуються однорідної схеми контактів. Якщо однорідність у регіоні недостатня, це може вплинути на природні параметри поширення. У різних однорідних середовищах параметри повинні бути різними. Слід зазначити, що незалежно від того, наскільки малий регіон, ідеальної гомогенізації не існує, а гетерогенність існує об'єктивно. Але, незалежно від того, наскільки вона неоднорідна, це зрештою призведе до статистично гомогенізованих параметрів моделі в межах певного регіону. Наприклад, епідеміологічне поширення відбулося за дуже неоднорідними географічними та часовими рамками. Хоча регіони дуже різні, вони не більше відрізняються, тому ми все ще можемо розглядати як систему. Хоча фактична ситуація не є однорідною, після великої кількості статистичних даних вона вважається однорідною. Важливо, що результати моделювання близькі до реальної ситуації, тому ми вважаємо, що програмне забезпечення все ще має практичну цінність. Незважаючи на це, ми не можемо ігнорувати його при

					БР.ІП – 01.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		84

інтерпретації та аналізі результатів моделювання, уникаючи надмірного ідеалізму.

3.4 Висновок по розділу

У третьому розділі реалізовано програмне забезпечення для моделювання поширення епідемій та проведено оцінку його ефективності. Виконано проєктування симуляційної системи, проаналізовано статистичні дані щодо COVID-19 та перевірено адекватність моделі. Проведений аналіз результатів підтвердив здатність системи відтворювати основні закономірності поширення епідемічних процесів. Отримані результати свідчать про практичну придатність розробленого програмного забезпечення для дослідницьких та аналітичних задач.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання дипломної роботи було досліджено та реалізовано програмне забезпечення для моделювання поширення епідемій, спрямоване на аналіз епідемічних процесів, прогнозування розвитку інфекційних захворювань та підтримку прийняття рішень у сфері охорони здоров'я. У роботі проаналізовано сучасні підходи до математичного та комп'ютерного моделювання епідемій, досліджено методи симуляції поширення інфекційних захворювань, а також реалізовано програмну систему, що дозволяє проводити аналіз статистичних даних і прогнозувати можливі сценарії розвитку епідемічних процесів. Процес розробки охопив аналіз вимог, моделювання структури системи, реалізацію функціональних модулів, тестування та оцінювання ефективності програмного забезпечення, що забезпечило створення цілісного та функціонального рішення.

На початковому етапі роботи було проведено аналіз предметної області та існуючих систем моделювання епідемій. Особливу увагу приділено математичним моделям поширення інфекційних захворювань, зокрема SIR, SEIR та агентно-орієнтованим моделям. Було визначено основні функціональні та нефункціональні вимоги до програмного забезпечення, серед яких точність прогнозування, масштабованість, швидкість обробки даних і зручність використання. Моделювання бізнес-логіки дозволило структурувати процеси аналізу статистичних даних, симуляції поширення епідемій та формування прогнозів для різних сценаріїв розвитку захворювань.

У процесі реалізації програмного забезпечення було розроблено симуляційну систему, яка забезпечує обробку статистичних даних, моделювання поширення інфекційних захворювань та візуалізацію результатів моделювання. Для реалізації системи використано сучасні технології програмування та інструменти аналізу даних, що забезпечили гнучкість архітектури, можливість масштабування та ефективну взаємодію між компонентами системи. Особливу увагу приділено розробці модулів аналізу біомедичних даних та побудові моделей прогнозування на основі статистичних даних щодо COVID-19.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

Важливим етапом роботи стало тестування програмного забезпечення, яке включало модульне, інтеграційне та функціональне тестування. Проведене тестування дозволило перевірити коректність роботи алгоритмів моделювання, точність обробки даних і стабільність роботи системи при великих обсягах інформації. Аналіз результатів моделювання показав, що використання сучасних методів симуляції дозволяє ефективно прогнозувати динаміку поширення інфекційних захворювань та оцінювати вплив різних факторів на розвиток епідемій. Отримані результати підтвердили доцільність використання агентно-орієнтованих підходів і статистичних методів аналізу для підвищення точності прогнозування.

Загалом розроблене програмне забезпечення відповідає поставленим цілям та забезпечує можливість моделювання епідемічних процесів, аналізу статистичних даних і формування прогнозів щодо поширення інфекційних захворювань. Перевагами запропонованого підходу є гнучкість архітектури системи, можливість адаптації моделей до різних типів епідемій, підтримка масштабування та ефективна обробка великих обсягів даних. Разом із тим існують певні обмеження, пов'язані з точністю вхідних статистичних даних, складністю врахування всіх соціальних та демографічних факторів, а також потребою у додатковій оптимізації продуктивності системи при складних симуляціях.

У майбутньому програмне забезпечення може бути вдосконалене шляхом інтеграції методів машинного навчання та штучного інтелекту для автоматичного прогнозування епідемічних процесів, використання великих даних для підвищення точності моделей, а також розширення функціональних можливостей системи для підтримки прийняття рішень у сфері охорони здоров'я. Це дозволить підвищити ефективність моніторингу епідемій, оперативність реагування на надзвичайні ситуації та якість прогнозування розвитку інфекційних захворювань.

					БР.ІП – 01.00.00.000 ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Anderson, R. M., & May, R. M. (1992). Infectious Diseases of Humans: Dynamics and Control. oxfordacademic.com. <https://academic.oup.com/book/27619>
2. AnyLogic Documentation. (2025). anylogic.help. <https://anylogic.help>
3. Bootstrap Documentation. (2025). getbootstrap.com. <https://getbootstrap.com/docs/>
4. Brooks-Pollock, E., Danon, L., & Keeling, M. (2021). A fast stochastic model of COVID-19 transmission dynamics. PLOS Computational Biology, 17(2), e1008609. <https://doi.org/10.1371/journal.pcbi.1008609>
5. Centers for Disease Control and Prevention (CDC). (2025). COVID Data Tracker. cdc.gov. <https://covid.cdc.gov/covid-data-tracker/>
6. Chen, J., & Wang, Y. (2024). Machine learning approaches for epidemic prediction and analysis. IEEE Access, 12, 45672–45689. <https://doi.org/10.1109/ACCESS.2024.3378451>
7. Clean Code Principles in .NET. (2023). devblogs.microsoft.com. <https://devblogs.microsoft.com/dotnet/clean-code-principles/>
8. Cypress End-to-End Testing. (2025). docs.cypress.io. <https://docs.cypress.io/guides/overview/why-cypress>
9. Eubank, S., Guclu, H., Kumar, V., et al. (2004). Modelling disease outbreaks in realistic urban social networks. Nature, 429(6988), 180–184. <https://doi.org/10.1038/nature02541>
10. European Centre for Disease Prevention and Control (ECDC). (2025). Surveillance Atlas of Infectious Diseases. ecdc.europa.eu. <https://www.ecdc.europa.eu/en/surveillance-atlas-infectious-diseases>
11. Fowler, M. (2002). Patterns of Enterprise Application Architecture. martinowler.com. <https://martinfowler.com/eaCatalog/>
12. Freeman, E., & Robson, E. (2021). Head First Design Patterns. oreilly.com. <https://www.oreilly.com/library/view/head-first-design/9781492077992/>

					БР.ІІІ – 01.00.00.000 ІІЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. pearson.com. <https://www.pearson.com/store/p/design-patterns-elements-of-reusable-object-oriented/P100000252058>
14. Google Cloud Healthcare API Documentation. (2025). cloud.google.com. <https://cloud.google.com/healthcare-api/docs>
15. Hassan, M., & Khan, A. (2023). Scalable architectures for epidemiological simulation systems. Journal of Computational Science, 68, 101987. <https://doi.org/10.1016/j.jocs.2023.101987>
16. Hethcote, H. W. (2000). The Mathematics of Infectious Diseases. SIAM Review, 42(4), 599–653. <https://doi.org/10.1137/S0036144500371907>
17. IBM. (2024). Artificial Intelligence for Healthcare Analytics. ibm.com. <https://www.ibm.com/solutions/healthcare-analytics>
18. Johns Hopkins Coronavirus Resource Center. (2024). coronavirus.jhu.edu. <https://coronavirus.jhu.edu/>
19. Keeling, M. J., & Rohani, P. (2011). Modeling Infectious Diseases in Humans and Animals. princeton.edu. <https://press.princeton.edu/books/paperback/9780691116174/modeling-infectious-diseases-in-humans-and-animals>
20. Kumar, S., & Sharma, R. (2023). Agent-based simulation models for epidemic spread analysis. Journal of Biomedical Informatics, 141, 104336. <https://doi.org/10.1016/j.jbi.2023.104336>
21. Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. pearson.com. <https://www.pearson.com/store/p/clean-architecture-a-craftmans-guide-to-software-structure-and-design/P100001465201>
22. MATLAB Documentation – Epidemiological Modeling. (2025). mathworks.com. <https://www.mathworks.com/discovery/epidemiological-model.html>
23. Microsoft Azure Documentation. (2025). learn.microsoft.com. <https://learn.microsoft.com/en-us/azure/>
24. MongoDB Documentation. (2025). mongodb.com. <https://www.mongodb.com/docs/>

					БР.ІІІ – 01.00.00.000 ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		89

25. .NET Documentation. (2025). learn.microsoft.com.
<https://learn.microsoft.com/en-us/dotnet/>
26. NetLogo Agent-Based Modeling Platform. (2025). ccl.northwestern.edu.
<https://ccl.northwestern.edu/netlogo/>
27. Osherove, R. (2019). The Art of Unit Testing. manning.com.
<https://www.manning.com/books/the-art-of-unit-testing-third-edition>
28. OWASP Top Ten Security Risks. (2024). owasp.org.
<https://owasp.org/www-project-top-ten/>
29. Postman API Testing Guide. (2025). learning.postman.com.
<https://learning.postman.com/docs/>
30. Python Documentation – Scientific Computing. (2025). docs.python.org.
<https://docs.python.org/3/library/>
31. Rahmandad, H., & Sterman, J. (2008). Heterogeneity and Network Structure in the Dynamics of Diffusion. Management Science, 54(5), 998–1014.
<https://doi.org/10.1287/mnsc.1070.0787>
32. React Documentation. (2025). react.dev. <https://react.dev/>
33. Selenium Documentation. (2025). selenium.dev.
<https://www.selenium.dev/documentation/>
34. Singh, P., & Verma, R. (2024). Real-time epidemic monitoring systems using cloud technologies. ACM Transactions on Computing for Healthcare, 5(1), 1–19. <https://doi.org/10.1145/3651123>
35. TensorFlow Documentation. (2025). tensorflow.org.
<https://www.tensorflow.org/>
36. The Lancet Digital Health. (2024). AI and epidemic forecasting technologies. thelancet.com. <https://www.thelancet.com/journals/landig/home>
37. Wang, L., & Zhang, Y. (2025). Big data analytics for infectious disease prediction systems. ACM Transactions on Knowledge Discovery from Data, 19(2), 1–24. <https://doi.org/10.1145/3648821>
38. World Health Organization (WHO). (2025). Epidemic and Pandemic Preparedness. who.int. <https://www.who.int/emergencies/diseases>

39. Yang, Z., Zeng, Z., Wang, K., et al. (2020). Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China. Journal of Thoracic Disease, 12(3), 165–174. <https://doi.org/10.21037/jtd.2020.02.64>

40. Zhang, T., & Liu, H. (2024). Software architectures for large-scale epidemic simulation platforms. IEEE Transactions on Software Engineering, 50(4), 1123–1138. <https://doi.org/10.1109/TSE.2024.3367812>

					БР.ІІ – 01.00.00.000 ІЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: "Розробка програмного забезпечення для моделювання поширення епідемій"

Обсяг пояснювальної записки: 87 аркушів

Дата закінчення дипломної роботи 10 червня 2026р.

Підпис студента _____