МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНФОРМАЦІЙНА МОДЕЛЬ ПРОГНОЗУВАННЯ ПОШИРЕННЯ ЕПІДЕМІОЛОГІЧНИХ ЗАХВОРЮВАНЬ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910105

Виконав студент 4-го курсу, групи 401
_____ М. В. Блідар
«28» червня 2023 р.

Керівник: д-р. фіз.-мат. наук, проф.
_____Е. А. Лисенков
« 28» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Блідару Миколі Володимировичу.

1. Тема кваліфікаційної роботи «Інформаційна модель прогнозування поширення епідеміологічних захворювань».

Керівник роботи Лисенков Едуард Анатолійович д-р. фіз.-мат. наук, проф.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: експертні оцінки моделей стохастичного зростання; пріоритетність критеріїв.

Очікуваний результат: система моделювання зростання бактеріальної колонії.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз сучасного стану задачі та вибору технології;

– огляд існуючих методів моделювання колонії бактерій;

– експертне оцінювання експертні оцінки моделей стохастичного зростання;

– порівняльний аналіз результатів застосування обраних методів для розв'язання поставленої задачі.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Захист від іонізуючих випромінювань».

7. Консультанти розділів роботи.

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці		

Керівник роботи __д-р. фіз.-мат. наук, проф.Лисенков Е. А. __
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Блідар М. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Блідара Миколи Володимировича

Тема: «Інформаційна модель прогнозування поширення епідеміологічних захворювань»

Дана бакалаврська кваліфікаційна робота присвячена розробці інформаційної моделі для прогнозування поширення епідеміологічних захворювань. Основна мета полягає в розробці точної та ефективної системи, яка здатна передбачити та прогнозувати поширення захворювань з використанням інформаційних технологій.

Об’єктом роботи є процес прогнозування поширення епідеміологічних захворювань.

Предметом роботи є моделі стохастичного зростання.

Метою роботи було моделювання та прогнозування процесів поширення епідеміологічних захворювань шляхом оптимізації існуючих інформаційних моделей.

У роботі проводиться аналіз та дослідження сучасних методів та підходів до прогнозування епідеміологічних захворювань з використанням інформаційних технологій. Розглядаються різні моделі та методи, включаючи статистичні, математичні, які допомагають в розробці ефективних прогнозних моделей.

У роботі буде розроблена інформаційна модель, яка заснована на методах розмноження кластерів бактеріальної колонії.

Для розробки інформаційної моделі були використані програмні платформи та інструменти, що забезпечують обробку та аналіз даних, такі як мови програмування C#.

Висновки роботи надають узагальнену інформацію про результати дослідження та ефективність розробленої інформаційної моделі. Вони включають порівняння з існуючими методами та аналіз точності та надійності прогнозів моделі.

Бакалаврська кваліфікаційна робота містить 70 сторінок, 23 рисунків, 2 таблиці, 20 використаних джерел та 1 додаток

Ключові слова: інформаційна модель, прогнозування, епідеміологічні захворювання, інформаційні технології.

ABSTRACT

Bachelor's qualification work of student Blidar Mykola Volodymyrovych from group 401 at Petro Mohyla Chernivtsi National University.

Topic: "Information model for predicting the spread of epidemiological diseases."

This bachelor's thesis focuses on developing an information model for predicting the spread of epidemiological diseases. The main goal is to create an accurate and efficient system capable of predicting and forecasting the spread of diseases using information technologies.

The research object is the process of predicting the spread of epidemiological diseases.

The subject of the research is stochastic growth models.

The aim of the thesis was to model and forecast the spread of epidemiological diseases by optimizing existing information models.

The thesis analyzes and investigates modern methods and approaches to forecasting epidemiological diseases using information technologies. Various models and methods, including statistical and mathematical ones, are examined to develop effective predictive models.

An information model based on bacterial colony cluster propagation methods will be developed in the thesis.

The development of the information model utilized software platforms and tools for data processing and analysis, such as the C# programming language.

The conclusions of the thesis provide a comprehensive overview of the research results and the effectiveness of the developed information model. They include comparisons with existing methods and an analysis of the accuracy and reliability of the model's predictions.

The bachelor's thesis consists of 70 pages, 23 figures, 2 tables, 20 references, and 1 appendix.

Keywords: information model, forecasting, epidemiological diseases, information technologies.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Інформаційна модель прогнозування поширення епідеміологічних захворювань

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	29.10.2022	29.10.2022	Виконано
2	Отримання завдання на виконання БКР	18.11.2022	18.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	5.12.2022	5.12.2022	Виконано
4	Отримання завдання на переддипломну практику	01.05.2023	01.05.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: аналіз сучасного стану задачі, огляд методів, розробка ПЗ	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	Виконано

Розробив студент __Блідар М. В._____ _____

(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ д-р фіз.-мат. наук, проф. Лисенков Е. А. _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_9_» _____12_____ 2022 р

ЗМІСТ

ВСТУП

Епідеміологічні захворювання і моделювання їх поширення є важливими аспектами в галузі громадського здоров'я, які допомагають нам краще розуміти, передбачати і контролювати поширення захворювань серед населення. Епідеміологія є науковою дисципліною, що вивчає поширення та вплив захворювань на популяцію, а також фактори, які спричиняють їх появу. Моделювання поширення захворювань займається розробкою математичних та статистичних моделей, які допомагають прогнозувати майбутній розвиток епідемій, оцінювати ефективність заходів по контролю захворюваннями та визначати стратегії втручання.

Епідеміологічні захворювання можуть поширюватися швидко та ефективно серед населення, особливо в умовах глобалізації та масового пересування людей. Такі захворювання можуть бути викликані вірусами, бактеріями, грибками або паразитами і можуть мати різні шляхи передачі, такі як контакт з інфікованою людиною, розповсюдження через воду або повітря, передача через комах або інших векторів.

Моделювання поширення захворювань залежить від доступності якісних епідеміологічних даних, таких як інформація про випадки захворювання, кількість інфікованих осіб, розподіл хворих за віком, статтю та іншими факторами. Ці дані дозволяють побудувати математичні моделі, які описують динаміку поширення захворювання в популяції. Використання математичних моделей дозволяє епідеміологам прогнозувати можливий розвиток епідемії, оцінювати ефективність різних контрольних стратегій та розробляти оптимальні втручання для зменшення поширення захворювання.

Моделювання поширення епідемій також враховує такі фактори, як інкубаційний період захворювання, період заразності, швидкість передачі імунізації, а також можливість мутацій вірусів. Для більш точного моделювання

можуть використовуватись різні методи, включаючи статистичні моделі, машинне навчання та агентно-орієнтовані моделі, які дозволяють враховувати індивідуальні характеристики та поведінку окремих осіб у популяції.

Моделі поширення епідемій мають велике значення для прийняття рішень щодо громадського здоров'я. Вони допомагають прогнозувати можливий обсяг захворюваності, розробляти стратегії вакцинації та ізоляції, оцінювати ефективність різних інтервенційних заходів і визначати оптимальний час для їх введення. Наприклад, моделі можуть допомогти визначити, які групи населення мають бути вакциновані в першу чергу, а також які заходи по соціальному дистанціюванню та обмеженню масових зборів можуть бути найбільш ефективними для зниження поширення захворювання.

Моделювання також може бути використане для вивчення ефективності публічної інформації та освітніх кампаній щодо профілактики та контролю захворюваннями. Аналізуючи різні сценарії та варіанти втручання, можна оцінити їх потенційні наслідки та вплив на здоров'я громади.

Важливим елементом моделювання поширення епідемій є збір якісних та точних даних. Для побудови достовірних моделей необхідно зібрати і аналізувати інформацію про випадки захворювання, контакти між особами, показники мобільності населення та інші фактори, які можуть впливати на поширення захворювання. Також важливо пам'ятати про неоднорідність популяції і враховувати різні соціальні, економічні та демографічні фактори, які можуть впливати на поширення захворювання в різних регіонах.

Усі ці аспекти моделювання епідеміологічних захворювань допомагають нам краще розуміти динаміку епідемій, прогнозувати їх поширення та приймати обґрунтовані рішення з метою забезпечення громадського здоров'я та добробуту населення.

1. ТИПИ МОДЕЛЕЙ СТОХАСТИЧНОГО ЗРОСТАННЯ

Моделі стохастичного росту є важливим інструментом для дослідження імітації різних процесів у природі. Вони базуються на випадкових процесах та розподілах, що використовуються для опису поведінки системи. Основною перевагою стохастичних моделей росту є їхні здатності відтворювати нелінійні залежності між різними параметрами системи та моделювати невизначеність у процесах росту і розвитку.

Історія використання стохастичних моделей у науці сягає 19 століття, коли математики та статистики почали розробляти теорії випадкових процесів. У 20 столітті ці ідеї були детально розвинуті в рамках теорії ймовірностей та стохастичних процесів.

У сучасних дослідженнях стохастичні моделі є важливим інструментом для дослідження складних природних систем, які мають стохастичний характер. Наприклад, вони застосовуються для дослідження динаміки лісових екосистем, зміни водного режиму річок та інші.

Використання стохастичних моделей у природних науках почало набувати популярності у 1960-х роках, коли науковці почали розробляти стохастичні моделі популяцій тварин. Одним з найбільш відомих прикладів використання стохастичних моделей в біології є модель Гаузінгера-Холла-Руджера для моделювання чисельності популяцій риб у водоймах.

Першу модель такого типу запропонував Алан Тьюрінг у 1952 році, в якій генерувалися стохастичні плями, подібні до чорно-білих плям на шкурі тварин [1]. Пізніше, Іденом була розроблена спрощена модель зростання бактеріальних колоній, яка була наочно продемонстрована у вигляді прямокутних отворів на перфокарті Голлеріта [2]. На початку свого існування ця модель не привертала до

себе багато уваги, оскільки розглядалася як надто спрощена, а також комп'ютери того часу не мали достатньої потужності для проведення інформативних досліджень. Зараз же, завдяки постійному розвитку технологій, стохастичні моделі росту є важливим інструментом для різноманітних наукових досліджень і прогнозування розвитку різних галузей, включаючи економіку, медицину, техніку та інші. Хоча стохастичні моделі росту можуть бути корисним інструментом для дослідників, їхня точність може бути обмеженою тим, що вони базуються на випадкових процесах. Також важливо враховувати, що стохастичні моделі можуть бути дуже складними та вимагати значної обчислювальної потужності для їхнього застосування.

1.1 Базисна модель Ідена

Базисна модель Ідена є однією з основних моделей, що використовуються для вивчення розвитку структурних змін в різних фізичних системах. Ця модель була запропонована фізиком Джоном Іденом в 1949 році та базується на моделюванні зростання кластерів за допомогою послідовного додавання нових частинок до уже існуючих кластерів.

У найпростішому варіанті моделі Ідена, яку представлено на рис. 1.1, ріст бактеріальної колонії імітується за послідовним заповненням вузлів квадратної решітки по периметру кластера.

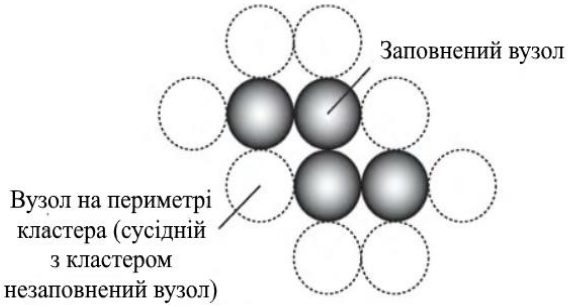


Рисунок 1.1 – Варіант моделі Ідена

Характеристики росту кластерів у базисній моделі Ідена залежать від ймовірності приєднання частинок до кластеру. Наприклад, якщо ймовірність приєднання кожної частинки до кластеру однакова, то кількість частинок у кластері зростає експоненційно з часом. Але якщо ймовірність приєднання залежить від розміру кластеру, то кластери можуть досягти максимального розміру і зупинитися на цьому рівні.

У початковий момент часу $(t = 1)$ кластер складається всього з однієї частинки ("зерна" зростання). Надалі зростання кластера відбувається за рахунок послідовного приєднання нових частинок у вузлах по його периметру. Для квадратної решітки в початковий момент часу кластер складається з частинки і має периметр, що складається з 4 вузлів. У наступний момент заповнюється якийсь із цих вузлів, далі визначається новий периметр тощо. У процесі моделювання складають список вузлів, що належать до периметра, які потенційно можуть бути заповнені. Показано кластер, що складається з 4 частинок із периметром, що включає 8 вузлів. Зазначимо, що в більш складних варіантах моделі випадковим чином можна вибирати не тільки вузли, але також зв'язки між вузлами, розташованими на периметрі [3].

1.2 Модель з придушенням шуму

Модель Ідена з придушенням шуму – це модифікована версія базисної моделі Ідена, яка дозволяє моделювати ріст кластерів у наявності додаткового шуму, такого як мікроскопічні нерівності на поверхні підкладки. Ця модель використовує дві ймовірності приєднання, які залежать від числа частинок у кластері та параметра " придушенням шуму", що відображає взаємодію з шумом. Чим більше значення цього параметра, тим менша ймовірність приєднання до кластеру, тим сильніше шум придушується.

Модель Ідена з придушенням шуму використовує дві ймовірності приєднання: ймовірність приєднання частинок до вільних місць на підкладці та

ймовірність приєднання до кластеру. Ймовірність приєднання до кластеру залежить від числа частинок у кластері та від значення параметра " придушенням шуму", який відображає ефект взаємодії з шумом. Чим більший цей параметр, тим менша ймовірність приєднання до кластеру, і тим сильніше придушується шум.

Одним з популярних методів придушення шуму є метод адаптивного фільтра, наприклад, алгоритм Левінсона-Вінера або алгоритм LMS (алгоритм найменших середньоквадратичних помилок). Ці алгоритми можуть бути використані для виявлення і придушення шуму вхідних сигналів перед їх подачею на вхід моделі Ідена.

Застосування моделі з придушенням шуму для моделі Ідена може поліпшити якість прогнозування моделі, зменшити вплив шуму і допомогти досягти кращої точності в різних завданнях, де присутні шуми вхідних даних.

Граткові моделі допускають надзвичайно просту програмну реалізацію і дозволяють проводити великомасштабні вичищення для великих розмірів кластерів. Разом з тим істотним їх недоліком є потенційна можливість впливу симетрії ґрат на структуру утвореного кластера [4,5]. Цей вплив легко продемонструвати за допомогою так званої моделі Ідена з придушенням шуму. У цій моделі вузол на периметрі кластера займається не відразу, а тільки через певну кількість візитів m у певний вузол. Для цього включається рахунок числа візитів у кожен вузол на периметрі, але список потенційних вузлів на периметрі поповнюється лише після успішної спроби заповнення. Приклади кластерів, отриманих при різних значеннях m , наведені на рис. 1.2. При збільшенні m кластер набуває форми квадрата, що відбиває вплив симетрії підкладки на структуру кластера.

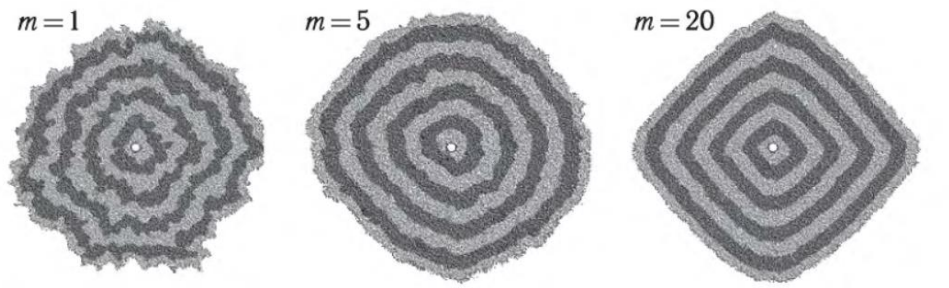


Рисунок 1.2 – Структура кластерів Ідена для ґраткової моделі з подавленням шуму(ріст на квадратній решітці)

1.3 Неґраткові моделі Ідена

Моделі Ідена можуть бути поділені на дві категорії: ґраткові та неґраткові. Неграткові моделі Ідена використовуються для моделювання росту кластерів на неперіодичних поверхнях, де не можна застосовувати ґраткову модель Ідена.

У неграткових моделях Ідена замість рівняння ґраткової моделі використовуються рівняння для інтенсивності потоку частинок на поверхню. Ці рівняння враховують різні фактори, такі як дифузія частинок, реакції між частинками на поверхні та інші фізичні процеси.

Нерешіточні моделі Ідена є складнішими в програмної реалізації і вимагають значно більших обчислювальних ресурсів. Один з варіантів такої моделі продемонстровано на рис. 1.3 [6]. У даній моделі випадково вибирається пробна клітина, і ідентифікуються її сусіди з відривом, не перевищую ще двох діаметрів. Потім, навколо обраної клітини з певним кроком $(2\pi/k)$ скануються кути з метою вибору вдалого напрямку зростання. Кожна сусідня клітина виключає певний інтервал напрямків зростання поблизу обраної клітини. За наявності такого напрямки нова клітина приєднаються до обраної клітини на відстані, що дорівнює її діаметру. За відсутності вдалого спрямування обрана клітина оголошується "мертвою" і далі не розглядається.

Процедура надалі повторюється.

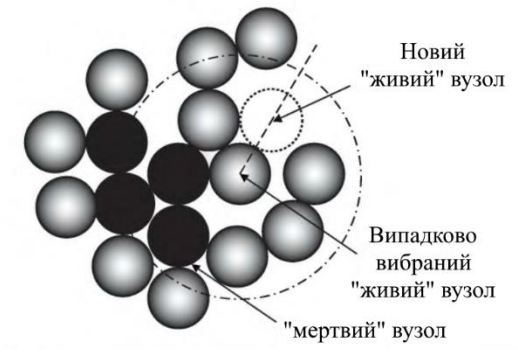


Рисунок 1.3 – Неґраткова модель Ідена

Вибір параметра k є досить критичним, оскільки при малих значеннях k деякі з клітин можуть оголошуватися «мертвими» передчасно, що може призводити до зменшення щільності кластера. Розрахунки показують, що для даної моделі при $k \approx 24$, досягається асимптотичний режим із граничним значенням щільності заповнення $\rho = 0{,}8276$. Ця щільність перевищує щільність випадкової упаковки для дисків $\rho = 0{,}772$, але нижче за максимально досяжну $\rho = 0{,}9069$ [7, 20].

1.4 Структура кластерів Ідена

Кластери Ідена – це структури, які виникають у процесі зростання бактеріальних колоній. Вони мають компактну внутрішню структуру, що проявляється у залежності між кількістю частинок у кластері та його радіусом гірації. При цьому поверхня кластерів Ідена має фрактальні властивості та сильно порізана, а кількість вузлів на периметрі зростає як з евклідовою розмірністю. Для вивчення зростання бактеріальних колоній та інших природних стохастичних процесів застосовуються різні варіанти моделі Ідена. Наприклад, модель клітинних автоматів, яка базується на покроковому рості клітин на сітці.

У методі Ідена кожен об'єкт даних представлений точкою в n-вимірному просторі, де n – кількість ознак. Спочатку визначається мінімальна густота

кластеру, тобто кількість об'єктів, які мають бути в заданій відстані один від одного, щоб бути включеними до кластеру. Потім визначається густота кожної точки, тобто кількість сусідніх точок, які знаходяться в заданій відстані.

Крім того, структура кластерів Ідена може бути використана для виявлення аномальних об'єктів, які не належать жодному кластеру. Вони можуть бути виключені з аналізу або досліджені окремо, що дозволяє виявити потенційні помилки у даних або незвичайні відхилення.

Для використання структури кластерів Ідена необхідно встановити декілька параметрів, таких як мінімальна густота кластеру, радіус заданої відстані та крок збільшення густоти. Невірне встановлення параметрів може призвести до недооцінки або переоцінки кількості та розміру кластерів.

В цілому, структура кластерів Ідена є корисним інструментом для кластеризації та аналізу даних з різною структурою та формою. Вона знайшла застосування в областях, таких як обробка природних мов, машинне навчання, генетика, біоінформатика, фінанси та інші.

Одним із застосувань моделі Ідена є симулювання процесів лікування та регенерації біотканин, вивчення зростання популяції ракових клітин та інших природних стохастичних процесів, таких як агрегація магнітних та заряджених частинок, еволюція басейнів річок, зростання міст. Для того, щоб зрозуміти ці процеси, дослідники використовують різні методи аналізу кластерів Ідена та їх властивостей. Наприклад, вони можуть вивчати динаміку зміни розміру та форми кластерів, або аналізувати залежність кількості частинок у кластері від часу. Ці дослідження допомагають краще зрозуміти природу та поведінку кластерів Ідена та застосовувати їх для розуміння природних явищ [8, 15].

Як ілюстрацію розглянемо деякі найпростіші варіанти моделі Ідена.

1.5 Перколяційна модель

Перколяційна модель – це математична модель, що використовується для опису процесу проникнення рідини, газу чи інших речовин через в'язкість і тверді матеріали, такі як пористі породи, тканини, композитні матеріали та інше. Модель базується на теорії перколяції, яка вивчає залежність властивостей системи від кількості та розміру включень (так званих перколяційних кластерів) в матриці [9].

Модель Ідена використовується для моделювання росту клітинних структур, зокрема бактеріальних колоній. Дослідження показали, що кластери Ідена мають компактну внутрішню структуру, що залежить від кількості частинок та радіусу гірації, що визначається як відстань до центра мас кластера. Одночасно, поверхня кластерів Ідена є фрактальною та сильно порізаною. Кількість вузлів на периметрі зростає як функція евклідової розмірності кластера.

Існує багато різних варіантів моделі Ідена, які включають облік міграції клітин, їх обертання, ріст та поділ, а також інші аспекти клітинної динаміки. Ці моделі використовуються для симуляції процесів лікування та регенерації біотканин, вивчення зростання популяції ракових клітин, а також інших природних стохастичних процесів.

У перколяційній моделі має місце фазовий перехід – зміна властивостей системи з плавним збільшенням включень до певної критичної кількості, після якої система перестає бути ізольованою і починає проводити речовину. Цей критичний поріг залежить від розміру включень, форми та орієнтації їх відносно матриці.

Перколяційна модель може бути використана для опису процесів, пов'язаних з рухом рідини, газу чи тепла через матеріали, що мають різну пористість та структуру. Також вона знаходить застосування в інших областях, таких як фізика м'яких матеріалів, екологія, геологія та інші.

Одним з найпростіших варіантів моделі Ідена є модель із блокуванням, в якій певна частина центрів зростання на периметрі кластера може блокуватися з певною

ймовірністю. Це призводить до обмеження зростання, порушення компактності кластерів та можливого повного припинення зростання при високих значеннях. Збіг величини з перколяційною концентрацією p (для квадратної грати p = 0,5927) дозволяє використовувати модель із блокуванням для генерації перколяційних кластерів.

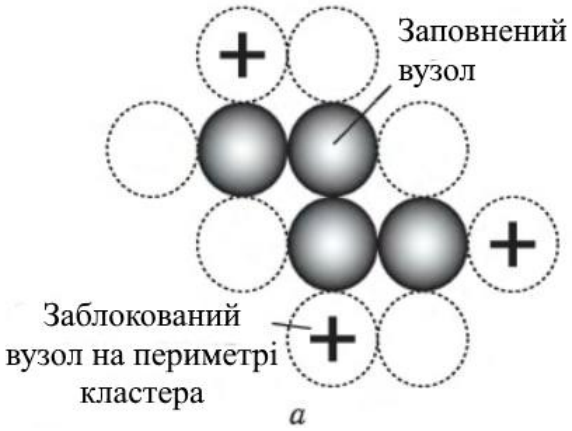


Рисунок 1.4 – Модель з блокуванням

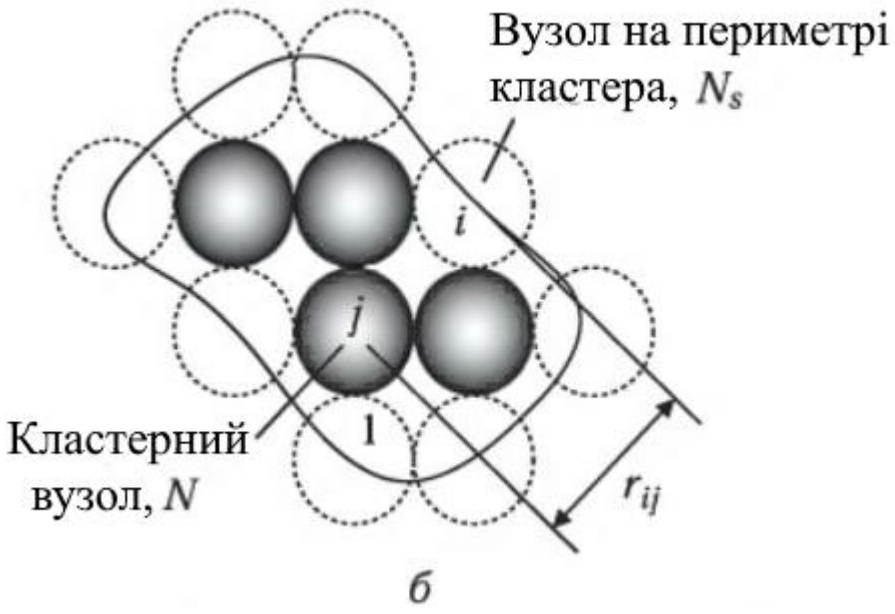


Рисунок 1.5 – Модель екраного росту

1.6 Модель екранованого зростання

Модель екранованого зростання – це математична модель, яка описує зростання популяції організмів в умовах обмеженої доступності ресурсів.

У цій моделі припускається, що зростання популяції залежить від швидкості народження нових особин і швидкості смертності старих особин, а також від кількості доступних ресурсів, необхідних для життя і розмноження популяції.

Модель передбачає, що на початку зростання популяції швидкість зростання є пропорційною кількості доступних ресурсів, але з часом, коли популяція збільшується, кількість доступних ресурсів стає обмеженою, і швидкість зростання починає спадати [10,14].

За моделлю екранованого зростання розмір колонії є важливим фактором, що впливає на швидкість зростання. Коли колонія досягає певного розміру, екрануючий ефект стає значним, що зменшує доступ бактерій до харчових ресурсів і забезпечує зниження швидкості зростання.

Модель екранованого зростання може бути застосована для дослідження динаміки зростання колонії бактерій у різних умовах середовища. Вона дозволяє оцінити вплив різних факторів на зростання бактерій, таких як наявність харчових ресурсів, конкуренція між бактеріями, наявність антибіотиків тощо.

Наприклад, застосування моделі екранованого зростання може допомогти визначити ефективність антибіотиків проти бактерій, оцінити вплив різних факторів на конкуренцію між різними штамами бактерій або встановити оптимальні умови для зростання бактерій у лабораторних умова

У моделі екранованого зростання передбачається, що частинки та кластери мають «заряд» і враховується відштовхування між кластером і частинкою, що приєднується. Ймовірність зростання на периметрі визначається виразом

$$f_i = \prod_{j=1}^{N} \exp\left(-\frac{a}{r_{ij}^{\psi}}\right) / \sum_{i=1}^{N_s} \times \prod_{j=1}^{N} \exp\left(-\frac{a}{r_{ij}^{\psi}}\right), \quad (1.1)$$

де N, N_s – кількість частинок у кластері та кількість вузлів на периметрі відповідно;
r_{ij} – відстань між зайнятим вузлом(i) та вузлом на периметрі (j);
α, ψ – параметри, що характеризують потенціал взаємодії.

Для даної моделі може спостерігатися зростання некомпактних кластерів, при $\psi < d$ (де d – евклідова розмірність простору) кластери виявляють фрактальні властивості з фрактальною розмірністю $d_i \approx \psi$.

1.7 Модель випадкового послідовного зростання

Модель випадкового послідовного зростання – це математична модель, що використовується для моделювання різних природних процесів, таких як адсорбція молекул на поверхні, формування покриття колоїдних частинок, або заростання кристалів. У цій моделі, розглядається послідовна додавання об'єктів на випадкові місця на певній поверхні, яка вже містить інші об'єкти. Ця модель дозволяє отримати статистичні характеристики системи, такі як площа зайнятої поверхні і кількість незайнятих місць на поверхні.

Основна ідея цієї моделі полягає в тому, що клітини в колонії ростуть випадковим чином і можуть додаватись в будь-яку точку на поверхні колонії. Коли клітина додається до колонії, вона розширюється і утворює нову поверхню, яка може стати місцем прикріплення наступної клітини. Оскільки місце додавання наступної клітини визначається випадково, то з часом утворюється випадкова мережа поверхонь, які перетинаються між собою.

Модель випадкового послідовного зростання може бути використана для вивчення різних властивостей зростання колоній бактерій, таких як швидкість зростання, форма колонії, щільність клітин тощо. Вона також може бути

використана для порівняння зростання колоній різних типів бактерій або для вивчення впливу різних умов на зростання колоній.

У цій моделі, пробний вузол на периметрі заповнюється з певною ймовірністю, а ступінь заповнення грат обмежується умовою максимального згущення. При черговій спробі обчислюється локальна щільність заповнення

$$\rho = N/N_{max}, \tag{1.2}$$

де N, N_{max} – відповідно кількість заповнених і максимальна кількість вузлів усередині кола з центром на пробному вузлі та радіусом R.

Кластери, отримані за допомогою цієї моделі, можуть виявляти фрактальні властивості. Приклади структури кластерів, одержаних за допомогою цієї моделіОтримані за допомогою цієї моделі кластери можуть мати фрактальні властивості, що означає, що їх структура повторюється на різних масштабах. Науковці застосовують модель випадкового послідовного зростання для дослідження різних властивостей кластерів та розв'язання різних практичних завдань. На рисунку нижче наведені приклади структури кластерів, які були отримані за допомогою моделі випадкового послідовного зростання.

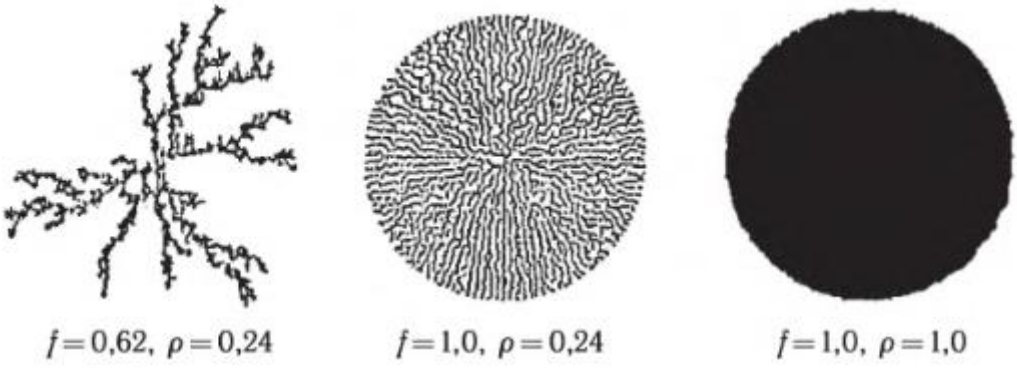


Рисунок 1.6 – Приклади структур кластерів зроблених за допомогою цього методу

На першому зображенні показано кластер з гладкою фрактальною структурою, який складається з великої кількості дрібних гілок. На другому зображенні показано кластер з більш грубою структурою, який складається з декількох великих гілок та декількох менших. На третьому зображенні показано кластер, який має підобов'язкову структуру та складається з великої кількості менших кластерів. Кожен з цих кластерів може мати унікальні властивості та застосування в різних галузях науки та техніки.

1.8 Модель «літаючого метелика»

Модель «літаючого метелика» – це стохастична модель, що використовується для дослідження властивостей кластерів. У цій моделі частинки приєднуються послідовно на периметрі, з обмеженням на відстань між ними. В даній моделі "літаючого метелика" встановлюється обмеження на відстань r між послідовно приєднуваними частинками на периметрі. Ймовірність приєднання до певного вузла на периметрі менша за 1 і дорівнює $f(r) \sim 1/r^{\psi}$, яка характеризує середню довжину стрибків. Якщо $\psi = 0$, то модель відповідає звичайній моделі Ідена, а якщо $\psi > d$, то стрибки відбуваються переважно між найближчими вузлами. У даній моделі компактність внутрішньої структури кластерів зберігається, але структура периметра істотно залежить від ψ [11]. Ця модель використовується для дослідження різних фізичних процесів, таких як зростання кристалів, агрегація частинок, дифузія речовини та інші. Вона дозволяє вивчити властивості кластерів, такі як їх розмірність, форма та компактність. Одна з головних переваг цієї моделі полягає в тому, що вона може бути застосована до дослідження різноманітних систем, які мають різні фізичні властивості.

У контексті колонії бактерій, модель "літаючого метелика" означає, що навіть незначні випадкові зміни або подразники, такі як зміна середовища, можуть спричинити радикальні зміни в поведінці та структурі колонії бактерій. Це може призводити до емержентності нових властивостей або функцій, які не були

присутніми в колонії раніше. Таким чином, навіть малі зміни можуть мати значний вплив на поведінку та еволюцію бактеріальних колоній.

Ця модель була запропонована в наукових дослідженнях для пояснення динаміки розвитку бактеріальних колоній та їх адаптації до змінних умов. Вона підкреслює важливість врахування незначних змін та випадковості в еволюції та динаміці бактерій, а також в інших комплексних системах.

Для даної моделі компактність внутрішньої структури кластерів зберігається, але структура периметра істотно залежить від значення у. Наприклад, при у < 1 периметр буде більш «зубчастим», а при у > 1 периметр буде більш «гладким». Одним з результатів моделі "літаючого метелика" є збереження компактності внутрішньої структури кластерів, що дозволяє використовувати її для опису фізичних процесів, таких як фільтрація, провідність, дифузія тощо. Однак структура периметра кластерів значно залежить від значення параметру у, тому відмінності в структурі периметра можуть відігравати важливу роль у певних фізичних процесах.

Крім фізичних застосувань, модель «літаючого метелика» може бути використана для дослідження біологічних процесів, таких як зростання бактеріальних колоній та морфогенез. Зокрема, можна застосовувати ідеологію клітинних автоматів для моделювання цих процесів.

1.9 Модель DLA для моделі Ідена

Модель DLA (Diffusion Limited Aggregation) – це математична модель росту, що описує процес утворення складних структур в результаті дифузії частинок. Ця модель була винайдена в 1981 році Віталіем Курном та Євгеном Сімонсом як модель формування галактик [12]. Ця модель базується на процесі дифузії та збору частинок у випадковому порядку.

Модель Ідена, з іншого боку, є моделлю зменшення розміру системи, де об'єкти зникають з випадковою інтенсивністю. Модель Ідена використовується для

моделювання різноманітних процесів, таких як флуктуації цін на фінансовому ринку, розширення руйнівних лавин та інші.

Застосування моделі DLA для моделювання зростання системи в моделі Ідена дає можливість досліджувати різні властивості зростання, такі як швидкість зростання, форма системи та її вплив на навколишнє середовище. Це дозволяє вченим отримувати нові знання про різні фізичні процеси та розвивати нові методи дослідження складних систем [13].

Крім того, модель DLA знайшла застосування в біології, де її використовують для дослідження структури та росту рослин, а також моделювання нейронних мереж. Використання цієї моделі дозволяє отримати нові інсайти щодо розвитку та росту біологічних систем.

У загальному, модель DLA є потужним інструментом для дослідження складних систем, який дозволяє отримувати нові знання та розробляти нові методи моделювання фізичних та біологічних процесів.

1.10 Метод кластерного аналізу пропалювання

Метод кластерного аналізу пропалювання є одним з методів аналізу даних, який використовується для виявлення схожості між об'єктами і групування їх в кластери на основі їх подібності. Цей метод використовується в багатьох галузях, включаючи маркетинг, соціологію, біологію та інші.

Принцип кластерного аналізу пропалювання полягає у наступному. Спочатку об'єкти аналізуються на основі деяких характеристик або ознак. Наприклад, можуть бути виміряні різні параметри, такі як вік, стать, заробітна плата, кількість дітей, релігійна належність тощо. Далі, за допомогою математичних методів, які базуються на відстанях між об'єктами, знаходяться групи об'єктів, які мають найбільшу подібність між собою. Ці групи називаються кластерами.

Метод кластерного аналізу пропалювання відрізняється від інших методів кластерного аналізу тим, що він використовує статистичні методи пропалювання

для визначення кількості кластерів, які є оптимальними для даного набору даних. Пропалювання полягає в запуску алгоритму кластеризації з різними кількостями кластерів та порівнянням отриманих результатів. Оптимальна кількість кластерів визначається на основі статистичних показників, таких як коефіцієнт силуету, який визначає якість кластеризації.

Метод кластерного аналізу пропалювання є досить потужним інструментом для аналізу даних та виявлення складних взаємозв'язків між ними. В основі цього методу лежить ідея про те, що об'єкти можуть бути розділені на групи на основі схожості між ними. Основними етапами методу є визначення міри схожості між об'єктами та їх подальше розбиття на кластери[13,18].

Метод кластерного аналізу пропалювання зазвичай використовується у таких областях, як соціологія, економіка, маркетинг та біологія. Наприклад, в соціології метод може бути використаний для аналізу споживацьких поведінок людей та виявлення груп споживачів зі схожими інтересами або поглядами. В економіці метод може бути використаний для аналізу фінансових показників компаній та виявлення груп підприємств зі схожими стратегіями розвитку. В біології метод може бути використаний для класифікації організмів на основі схожості їх генетичного матеріалу [14-17].

Висновки до розділу 1

У даному дослідженні були розглянуті типи моделей стохастичного зростання, зокрема типи моделей Ідена. Аналізуючи літературні джерела та проводячи дослідження в цій галузі, було виявлено, що моделі Ідена є важливим інструментом для моделювання та прогнозування стохастичних процесів зростання.

Моделі Ідена використовуються для опису змінних, що зростають з часом, і враховують стохастичний характер цих змін. Вони засновані на концепції

фрактальної геометрії та використовують стохастичні диференціальні рівняння для моделювання зростання.

Дослідження показало, що моделі Ідена мають широкий спектр застосувань у різних галузях, таких як економіка, фінанси, біологія, медицина та інші. Вони можуть бути використані для прогнозування фінансових ринків, моделювання зростання популяцій, аналізу клінічних даних та багатьох інших завдань, де важлива стохастична природа процесу зростання.

Результати дослідження свідчать про важливість моделей Ідена у розумінні та прогнозуванні стохастичних процесів зростання. Вони дозволяють краще розуміти поведінку змінних, що зростають з часом, та допомагають у прийнятті рішень на основі аналізу цих процесів. Дані моделі можуть бути корисним інструментом для дослідників, аналітиків та практиків, які працюють у галузях, де важливе розуміння та прогнозування стохастичного зростання.

2. МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Планування розробки програмного забезпечення є важливим етапом у процесі розробки. Це дозволяє розробникам та командам зрозуміти, що потрібно зробити, як це робити, які ресурси та інструменти потрібні для завершення проекту, а також уникнути можливих проблем, що можуть виникнути під час розробки. Планування також допомагає забезпечити, що проект буде завершено вчасно та в рамках бюджету, а результат буде відповідати вимогам та очікуванням замовника. Для створення якісного продукту треба створити план дій та список функціоналу який буде реалізований у проєкті.

Розробка програмного забезпечення на мові програмування C# за допомогою бібліотеки Windows Forms для візуалізації процесу росту кластерів.

Для реалізації процесу моделювання росту кластерів та їх візуалізації у графічному інтерфейсі використовується мова програмування C# та бібліотека Windows Forms. За допомогою цих інструментів буде створено програмне забезпечення, яке надасть можливість спостерігати за процесом росту кластерів на екрані комп'ютера.

Реалізація методу Ідена для моделювання росту кластерів на двовимірному просторі.[20]

Метод Ідена використовується для моделювання росту кластерів на двовимірному просторі. В процесі росту кластерів на кожному кроці обирається випадкова точка, що має сусідні точки, та перевіряється доступність сусідніх точок для зростання. Якщо сусідня точка є пустою, то кластер росте, і ця точка додається до кластеру. Процес росту кластерів триває до досягнення певних умов, наприклад, поки одна з координат точки росту не досягне певного значення.

Ініціалізація поля росту кластерів та графічного інтерфейсу.

При початку моделювання росту кластерів ініціалізується поле росту кластерів, яке відображає двовимірний простір, на якому відбувається ріст кластерів. Також створюється графічний інтерфейс, який буде використовуватись для візуалізації процесу росту кластерів.

Візуалізація процесу росту кластерів на графічному інтерфейсі.

Кожна точка кластеру представлена окремим кольором на графічному інтерфейсі. Після кожного кроку моделювання росту кластеру, зображення кластерів оновлюється на графічному інтерфейсі, що дозволяє спостерігати за процесом росту кластерів у реальному часі.

Аналіз отриманих результатів.

Після завершення моделювання росту кластерів і візуалізації результатів проводиться аналіз отриманих результатів. Важливі характеристики, які аналізуються, включають кількість ітерацій, структуру росту кластерів та їх розподіл у просторі. Цей аналіз допомагає краще зрозуміти особливості росту кластерів та їх властивості.

2.1 Розробка програмного забезпечення для візуалізації процесу росту кластерів

Розробка програмного забезпечення на мові програмування C# за допомогою бібліотеки Windows Forms для візуалізації процесу росту кластерів є захоплюючим завданням, яке дозволяє створити інтерактивну інтерфейсну програму для відображення та вивчення росту кластерів на двовимірному просторі.

Мова програмування C# використовується для реалізації логіки та алгоритмів, які керують процесом росту кластерів. Бібліотека Windows Forms надає потужні інструменти для створення графічного інтерфейсу користувача, що дозволяє відображати процес росту кластерів у реальному часі.

Процес росту кластерів може бути реалізований за допомогою класу Eden, який успадковує від класу Form з бібліотеки Windows Forms. В цьому класі

визначені різні методи та змінні, необхідні для ініціалізації, керування та відображення процесу росту кластерів.

Процес росту кластерів реалізований з використанням випадкових генераційних алгоритмів та взаємодії з двовимірним простором. Починаючи з початкової точки, кожен новий кластер випадково розширюється у чотирьох напрямках: вправо, вниз, вліво та вгору. Якщо нова точка попадає на пусту область простору, вона стає частиною кластеру, в іншому випадку появляється новий кластер. Кластери розфарбовуються у різні кольори для візуального відображення.

Процес росту кластерів відображається на графічному інтерфейсі за допомогою елемента управління PictureBox, на якому відображується двовимірний простір. Кожен новий крок росту кластерів оновлює зображення на PictureBox, що дозволяє відслідковувати процес у реальному часі. Також, для зручності користувача, програма може відображати додаткову інформацію, наприклад, кількість ітерацій та кількість кластерів.

Розробка такого програмного забезпечення вимагає систематичного підходу та дотримання кращих практик програмування. Код повинен бути структурованим та добре організованим, використовуючи принципи об'єктно-орієнтованого програмування та поділу на модулі. Налагодження та тестування програми має проводитися для перевірки правильності роботи алгоритмів та візуалізації.

Розробка програмного забезпечення на мові програмування C# з використанням бібліотеки Windows Forms для візуалізації процесу росту кластерів є захоплюючим та творчим завданням, яке вимагає розуміння алгоритмічного підходу та глибоких знань програмування. Результатом роботи буде програмний продукт, який дозволить візуально спостерігати та аналізувати процес росту кластерів, що має потенціал для застосування в різних областях, таких як комп'ютерні науки, фізика, соціологія та багато інших.

2.2 Реалізація методу Ідена для моделювання росту кластерів на двовимірному просторі

Реалізація методу Ідена для моделювання росту кластерів на двовимірному просторі є захоплюючим та цікавим завданням, яке дозволяє відтворити природні процеси росту та розвитку в контрольованій середовищі. Цей метод, який отримав назву на честь ботаніка Хафстедта Ідена, дозволяє створити візуальну модель росту кластерів, що знаходяться на двовимірній поверхні.

Перш за все, для реалізації методу Ідена потрібно створити двовимірний простір, який відповідає розміру 3 сторінок. Цей простір може бути представлений у вигляді сітки або матриці, де кожна комірка відповідає одному елементу простору.

Починаючи з початкової точки в центрі простору, процес росту кластерів відбувається шляхом додавання нових елементів до кластеру. Кожен новий елемент додається до кластеру за певними правилами, що залежать від обраної моделі росту. Наприклад, можуть використовуватись правила, які базуються на випадковому розподілі або на локальних взаємодіях з сусідніми елементами.

Після додавання нового елементу до кластеру, можуть виконуватись додаткові дії, такі як перевірка на наявність порожніх комірок або оновлення інформації про розмір та форму кластеру.

У процесі росту кластерів на двовимірному просторі можуть використовуватись різні стратегії та параметри, які впливають на зовнішній вигляд та структуру кластерів. Наприклад, можуть враховуватись фізичні чинники, такі як гравітація або розміщення кластерів у вигляді дерева.

Оскільки реалізація методу Ідена для моделювання росту кластерів є складним завданням, важливо мати глибокі знання мови програмування та використовувати потужні бібліотеки та інструменти для реалізації цієї моделі. Мова програмування C# та бібліотека Windows Forms є потужними засобами для

створення візуальних моделей та графічного відображення росту кластерів на двовимірному просторі.

Після успішної реалізації методу Ідена можна відтворити різні сценарії росту кластерів на двовимірному просторі, використовуючи різні параметри та моделі. Це відкриває широкі можливості для досліджень та аналізу різних аспектів росту кластерів у контрольованому середовищі.

Реалізація методу Ідена для моделювання росту кластерів на двовимірному просторі є захоплюючим процесом, що вимагає креативності, алгоритмічних знань та розуміння принципів росту та розвитку. Цей метод дозволяє відтворити природні процеси та досліджувати їх характеристики у віртуальному середовищі. Результатом реалізації методу Ідена є можливість візуалізувати та аналізувати рост кластерів на двовимірному просторі та отримати цінні висновки щодо їх структури та динаміки росту.

2.3 Ініціалізація поля росту кластерів та графічного інтерфейсу

Ініціалізація поля росту кластерів та графічного інтерфейсу є важливим етапом у розробці програмного забезпечення для моделювання росту кластерів. Цей етап вимагає установки початкових параметрів та налаштувань, які визначають вигляд та поведінку поля росту та графічного інтерфейсу.

Починаючи з поля росту кластерів, необхідно встановити його розміри та форму. Це може бути прямокутне поле або має більш складну форму, залежно від вимог і специфіки дослідження. Також важливо встановити початковий стан поля, наприклад, чи буде воно порожнім або міститиме початкові кластери або елементи.

Далі, ініціалізація графічного інтерфейсу включає встановлення компонентів та елементів, які будуть відображати поле росту та додаткові контроли. Це може включати в себе створення вікон, панелей, кнопок та інших елементів, які дозволяють користувачу взаємодіяти з програмою та впливати на процес росту кластерів.

Крім того, ініціалізація поля росту та графічного інтерфейсу може включати встановлення параметрів та налаштувань моделі росту кластерів. Наприклад, це може бути встановлення правил росту, таких як випадковий розподіл чи локальні взаємодії, а також налаштування параметрів, що впливають на швидкість та інтенсивність росту кластерів.

Після успішної ініціалізації поля росту кластерів та графічного інтерфейсу, програма готова до відображення та моделювання процесу росту кластерів. Користувач може взаємодіяти з графічним інтерфейсом, задавати параметри та спостерігати за змінами у полі росту. Графічний інтерфейс дозволяє зручно відображати та контролювати процес росту кластерів, що сприяє зрозумінню та аналізу отриманих результатів.

Після успішної ініціалізації поля росту кластерів та графічного інтерфейсу, можна перейти до наступних кроків у реалізації методу Ідена.

Для початку, можна здійснити визначення та налаштування параметрів, які визначають поведінку процесу росту кластерів. Це можуть бути такі параметри, як швидкість росту, ймовірність взаємодії між елементами, розміри поля росту тощо. Ці параметри можуть бути встановлені заздалегідь або налаштовуватись користувачем через графічний інтерфейс.

Наступним кроком є вибір початкового стану поля росту. Це може бути порожнє поле без будь-яких кластерів, або вже наявні кластери або елементи. Початковий стан впливає на подальший процес росту кластерів, тому його вибір має бути узгодженим з метою та цілями моделювання.

Після встановлення початкового стану та параметрів, можна перейти до самого процесу росту кластерів. Зазвичай це виконується за допомогою ітераційного циклу, де на кожній ітерації відбувається рост одного або декількох кластерів. У методі Ідена це реалізовано за допомогою алгоритму, що включає випадковий вибір початкового елементу та його зростання за певними правилами.

Під час процесу росту кластерів можна використовувати графічний інтерфейс для візуалізації та відображення змін у полі росту. Це дозволяє користувачу спостерігати за динамікою росту кластерів, спостерігати зміни у їх розмірах та формах, а також відслідковувати процес зростання в реальному часі.

Крім того, графічний інтерфейс може надати можливості взаємодії з процесом росту кластерів. Наприклад, користувач може впливати на параметри моделювання, вибирати початковий стан поля росту, зупиняти або продовжувати процес росту, зберігати та завантажувати результати моделювання тощо.

Завершуючи, ініціалізація поля росту кластерів та графічного інтерфейсу є важливим кроком у реалізації методу Ідена для моделювання росту кластерів на двовимірному просторі. Цей етап дозволяє задати початкові параметри та стан поля, а також візуалізувати процес росту та взаємодіяти з ним. Графічний інтерфейс робить моделювання більш доступним та зрозумілим, допомагаючи аналізувати результати та досліджувати характеристики росту кластерів.

2.4 Візуалізація процесу росту кластерів на графічному інтерфейсі

Візуалізація процесу росту кластерів на графічному інтерфейсі є надзвичайно важним аспектом при моделюванні та аналізі росту кластерів. Це дозволяє нам відображати та спостерігати за динамікою змін у кластерах, розмірах та формах їх росту.

При створенні графічного інтерфейсу для візуалізації процесу росту кластерів, ми можемо використовувати різноманітні графічні елементи та інструменти для надання зрозумілого та ефективного способу відображення даних. Основною метою є передача користувачеві інформації про стан та зміни в кластерах у вигляді зрозумілих візуальних образів.

Використання двовимірного поля для відображення кластерів є одним з ключових елементів графічного інтерфейсу. Кожен елемент поля може мати свою власну властивість або стан, що відображає наявність або відсутність кластеру. Це

дає змогу користувачу побачити місця, де формуються кластери та як вони поширюються з плином часу.

Кольорова палітра використовується для позначення різних властивостей або типів кластерів. Вона дозволяє візуально виділити різні аспекти росту кластерів, наприклад, великі кластери можуть бути позначені темними кольорами, а менші - світлими. Це допомагає користувачу легше розрізнити та сприйняти різні характеристики кластерів.

Анімація відіграє важливу роль у візуалізації росту кластерів. Вона дозволяє плавно переходити від одного стану до іншого, відображаючи етапи росту кластерів і зміни їх форми та розміру. Користувач може спостерігати за цим процесом та аналізувати зміни, що відбуваються у кластерах протягом певного періоду часу.

Графічний інтерфейс також має елементи управління, що дозволяють користувачам контролювати процес росту кластерів. Це можуть бути кнопки для початку, паузи або зупинки моделювання, регулятори для зміни параметрів росту або інструменти для маніпулювання кластерами на полі.

Загалом, візуалізація процесу росту кластерів на графічному інтерфейсі допомагає нам краще розуміти та аналізувати динаміку росту кластерів. Вона надає наглядний спосіб відображення даних та спрощує взаємодію з моделлю росту кластерів, дозволяючи нам досліджувати різні сценарії та варіації росту кластерів у зручний та інтуїтивно зрозумілий спосіб.

2.5 Технології розробки системи

При розробці системи візуалізації процесу росту кластерів на графічному інтерфейсі можна використовувати різні технології та інструменти. Одним з ключових елементів є вибір мови програмування, наприклад, мова програмування C#. Для створення графічного інтерфейсу та взаємодії з користувачем можна використовувати бібліотеку Windows Forms. Для розробки програмного забезпечення на мові C# та роботи з бібліотекою Windows Forms можна

використовувати інтегровані середовища розробки (IDE), такі як Visual Studio або JetBrains Rider. Для відображення даних та візуалізації процесу росту кластерів можна використовувати різні графічні елементи, такі як плоский канвас, графіки та колірну палітру. Для створення плавних анімацій можна використовувати техніки зміни положення та розміру графічних елементів та плавні переходи між станами кластерів.

Використання мови програмування C# в розробці системи візуалізації процесу росту кластерів на графічному інтерфейсі має свої переваги.

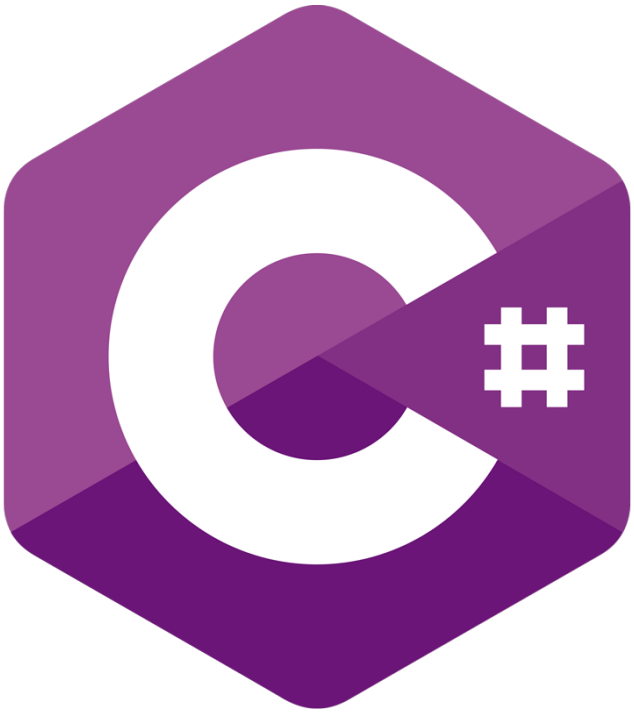


Рисунок 2.1 – Логотип мови C#

C# є об'єктно-орієнтованою мовою програмування, яка надає розробникам потужний та ефективний інструментарій для створення програмного забезпечення. Вона має зручний синтаксис та багато вбудованих функцій і бібліотек, що спрощують розробку.

C# є частиною платформи .NET, що робить його високопродуктивною та надійною мовою програмування. Інтеграція з платформою .NET дозволяє використовувати багато розширених можливостей, таких як управління пам'яттю, механізми безпеки, доступ до баз даних та багато іншого.

Крім того, C# має широку підтримку середовищ розробки, зокрема таких популярних IDE, як Visual Studio та JetBrains Rider. Ці інтегровані середовища надають розробникам зручні інструменти для написання, налагодження та тестування коду, а також допомагають будувати інтерфейс користувача.

Використання мови C# разом з бібліотекою Windows Forms дозволяє розробникам ефективно створювати графічні інтерфейси з великою кількістю готових елементів керування, таких як кнопки, тексти, таблиці, зображення і т.д. Бібліотека Windows Forms надає зручний спосіб створення і розташування цих елементів на формі, а також забезпечує можливість реагувати на події користувача.

Отже, використання мови програмування C# в розробці системи візуалізації процесу росту кластерів дозволяє розробникам отримати потужний та гнучкий інструментарій, який спрощує роботу з графічним інтерфейсом та забезпечує високу продуктивність та якість програмного забезпечення.

2.6 Windows Forms

Для створення графічного інтерфейсу та взаємодії з користувачем у проекті використовується бібліотека Windows Forms. Ця бібліотека є одним з найпоширеніших інструментів для розробки десктопних додатків на платформі .NET.

Рисунок 2.2 – Логотип бібліотеки Windows Form

Основна перевага використання Windows Forms полягає у його простоті та легкості використання. Вона надає розробникам набір готових елементів керування (кнопки, поля введення, списки, таблиці тощо), які можна легко розміщувати на формі та взаємодіяти з ними за допомогою подій та методів. Це спрощує процес створення інтерфейсу, забезпечує швидку реалізацію функціональності та прискорює розробку додатків.

Windows Forms також надає розробникам можливість налаштувати зовнішній вигляд та поведінку елементів керування. За допомогою властивостей і стилів можна змінювати кольори, розміри, шрифти, а також додавати анімацію та ефекти переходів. Це дозволяє створювати привабливі та інтуїтивно зрозумілі інтерфейси для користувачів.

Крім того, Windows Forms підтримує обробку подій, що дозволяє реагувати на дії користувача, такі як натискання кнопок, переміщення миші, введення тексту тощо. За допомогою обробників подій можна виконувати відповідні дії, наприклад, змінювати вміст елементів керування, виконувати обчислення, оновлювати дані

тощо. Це дозволяє створювати інтерактивні та реактивні додатки, які взаємодіють з користувачем у реальному часі.

Крім того, Windows Forms інтегрований з мовою програмування C#, що робить його ще більш привабливим для розробників, які вже мають досвід роботи з C# або використовують цю мову у своїх проектах. Використання Windows Forms з C# дозволяє розробникам використовувати всі можливості мови, такі як об'єктно-орієнтований підхід, багатопотоковість, винятки та інші, для створення потужних та ефективних додатків.

Загалом, використання бібліотеки Windows Forms разом з мовою програмування C# забезпечує розробникам зручну та продуктивну платформу для створення графічного інтерфейсу та взаємодії з користувачем у десктопних додатках. Вона дозволяє ефективно втілювати функціональність, створювати привабливий дизайн та забезпечувати зручну взаємодію користувача з програмою.

2.4 Visual Studio

Visual Studio є інтегроване середовище розробки (IDE), яке надає широкі можливості для створення програмного забезпечення на мові програмування C#. Воно пропонує інструменти та ресурси, які полегшують процес розробки та забезпечують ефективну роботу розробників.

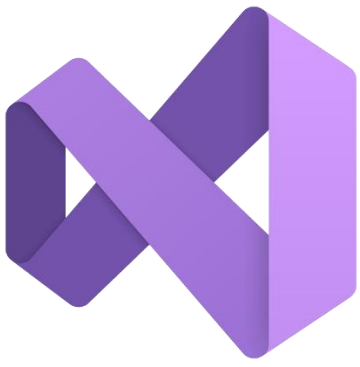


Рисунок 2.3 – Логотип середовища програмування Visual Stidio

Один з головних аргументів використання Visual Studio полягає в його інтегрованому підході до розробки. Він поєднує у собі редактор коду, компілятор, засоби налагодження та інші необхідні компоненти в одному середовищі, що спрощує розробку програмного забезпечення.

Visual Studio має потужні інструменти для побудови графічного інтерфейсу за допомогою бібліотеки Windows Forms. Це дозволяє розробникам створювати інтуїтивно зрозумілі та привабливі користувацькі інтерфейси для своїх програм.

Інтегроване середовище Visual Studio також надає засоби взаємодії з користувачем, такі як форми, кнопки, поля введення тощо. Це дозволяє розробникам створювати інтерактивні програми, які можуть реагувати на дії користувача.

Окрім того, Visual Studio надає розробникам зручність і продуктивність завдяки своєму інтерфейсу. Воно має ряд функцій та можливостей, таких як автодоповнення коду, навігація по проекту, відладка, аналіз коду тощо. Це допомагає розробникам писати код швидше, зменшує кількість помилок та полегшує розуміння коду.

Visual Studio також забезпечує зручне відстеження версій, керування проектами та спільну роботу у команді. Воно підтримує інтеграцію з системами контролю версій, такими як Git, що дозволяє командам розробників ефективно співпрацювати та керувати версіями коду.

Узагальнюючи, Visual Studio є потужним інструментом для розробки програмного забезпечення на мові програмування C#. Він надає зручне та продуктивне середовище для створення графічного інтерфейсу, взаємодії з користувачем та розробки програмного забезпечення загалом. Його функціональність та інтегрований підхід роблять його популярним вибором серед розробників.

Висновки до розділу 2

У цьому тексті було розглянуто використання Visual Studio для розробки програмного забезпечення на мові програмування C# з використанням бібліотеки Windows Forms. Детально розглянуто переваги використання Visual Studio, а саме його інтегрований підхід до розробки, можливості створення графічного інтерфейсу, взаємодії з користувачем та зручності й продуктивності, які надає це середовище.

Використання Visual Studio дозволяє розробникам ефективно створювати програмне забезпечення, зосереджуючись на функціональності та взаємодії з користувачем. Інтегрований підхід до розробки, що поєднує редактор коду, компілятор, налагоджувач та інші необхідні інструменти, спрощує роботу розробників і полегшує процес створення програмного забезпечення.

Бібліотека Windows Forms, яка доступна у складі Visual Studio, дозволяє створювати привабливий та інтуїтивно зрозумілий графічний інтерфейс для програм. Це дозволяє розробникам створювати користувацькі інтерфейси, які легко сприймаються користувачами і забезпечують зручну взаємодію з програмою.

Крім того, використання Visual Studio сприяє збільшенню продуктивності розробників завдяки різноманітним функціям, таким як автодоповнення коду, навігація по проекту, аналіз коду та інші. Ці можливості допомагають розробникам писати код швидше, зменшувати кількість помилок та підвищувати якість програмного забезпечення.

Загалом, використання Visual Studio у поєднанні з мовою програмування C# та бібліотекою Windows Forms є ефективним способом розробки програмного забезпечення з графічним інтерфейсом. Цей інструментарій надає зручність, продуктивність та можливості для створення високоякісних програм, які задовольняють потреби користувачів.

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Моделювання інформаційної системи є важливим етапом у процесі розробки та оптимізації сучасних комп'ютерних систем. Цей підхід дозволяє розробникам і фахівцям з області ІТ аналізувати, проектувати і тестувати різні аспекти системи перед її фактичною реалізацією.

Моделювання інформаційних систем використовується для створення абстрактної, спрощеної версії реальної системи з метою вивчення її поведінки, функціональності та взаємодії з користувачами та іншими системами. Під час моделювання створюються формальні описи системи, які включають в себе процеси, даних, ресурсів та взаємодії між компонентами.

Модель, що була дана на початку, базується на мові програмування C# та використовує бібліотеку Windows Forms для візуалізації процесу росту кластерів на двовимірному просторі. Ця модель дозволяє відтворити процес росту кластерів, досліджувати його властивості та вплив різних факторів на його розвиток.

Моделювання інформаційних систем має кілька переваг. По-перше, воно дозволяє виявити та усунути помилки та недоліки системи ще до її впровадження. По-друге, воно дозволяє проводити експерименти з різними варіантами системи та з'ясувати їх вплив на її продуктивність та ефективність. По-третє, воно дозволяє спростити процес розробки та зменшити витрати часу та ресурсів на реалізацію систем.

Ціль моделювання інформаційної системи полягає в тому, щоб мати чітке уявлення про те, як система буде працювати, які будуть взаємодії між її компонентами та які будуть результати роботи системи. Використання моделей дозволяє виявити можливі проблеми, помилки або недоліки ще на ранніх етапах розробки, що дозволяє зекономити час, ресурси та забезпечити успішну реалізацію системи.

У випадку даної інформаційної системи, яка моделює ріст кластерів на двовимірному просторі, основними елементами моделі є:

– простір моделювання – це двовимірний простір, на якому відбувається ріст кластерів. Він може бути представлений у вигляді сітки або матриці точок, де кожна точка має свої координати та певні властивості;

– кластери – це скупчення точок в просторі, що репрезентують об'єкти або області інтересу. Кластери можуть мати різні характеристики, такі як розмір, форма, кольори тощо;

– правила росту – це набір правил або алгоритмів, що визначають, як кластери збільшуються та розширюються в просторі. Вони можуть базуватися на таких факторах, як прив'язка до сусідніх точок, розмір кластеру, ймовірність зростання тощо.

Візуалізація: Це процес відображення моделі росту кластерів на графічному інтерфейсі. Візуалізація може включати різні елементи, такі як кольорові схеми для різних типів кластерів, анімацію зростання кластерів, можливість масштабування та перегляду простору моделі тощо.

Діаграма класів є інструментом, який графічно відображає структуру та взаємозв'язки між класами в програмному проекті. Вона допомагає нам отримати загальне уявлення про компоненти програми та їх взаємозв'язки.

Одна з важливих функцій діаграми класів – це візуалізація структури програми. Вона допомагає нам побачити всі класи, які використовуються в проекті, і їх ієрархію. Це дозволяє краще розуміти, як всі класи взаємодіють між собою.

Діаграма класів також вказує на залежності між класами, такі як успадкування, композиція або асоціації. Це допомагає нам бачити, які класи співпрацюють між собою і як зміни в одному класі можуть вплинути на інші.

Крім того, діаграма класів може служити документацією про архітектуру програми і сприяти комунікації між розробниками та іншими учасниками проекту.

Вона допомагає краще пояснити структуру програми та зберегти консистентність у спільній розробці.

Узагалі, діаграма класів є важливим інструментом для розуміння, планування та комунікації в процесі розробки програмного забезпечення. Вона надає нам візуальне представлення архітектури проекту, що сприяє його розумінню та ефективній співпраці між розробниками.

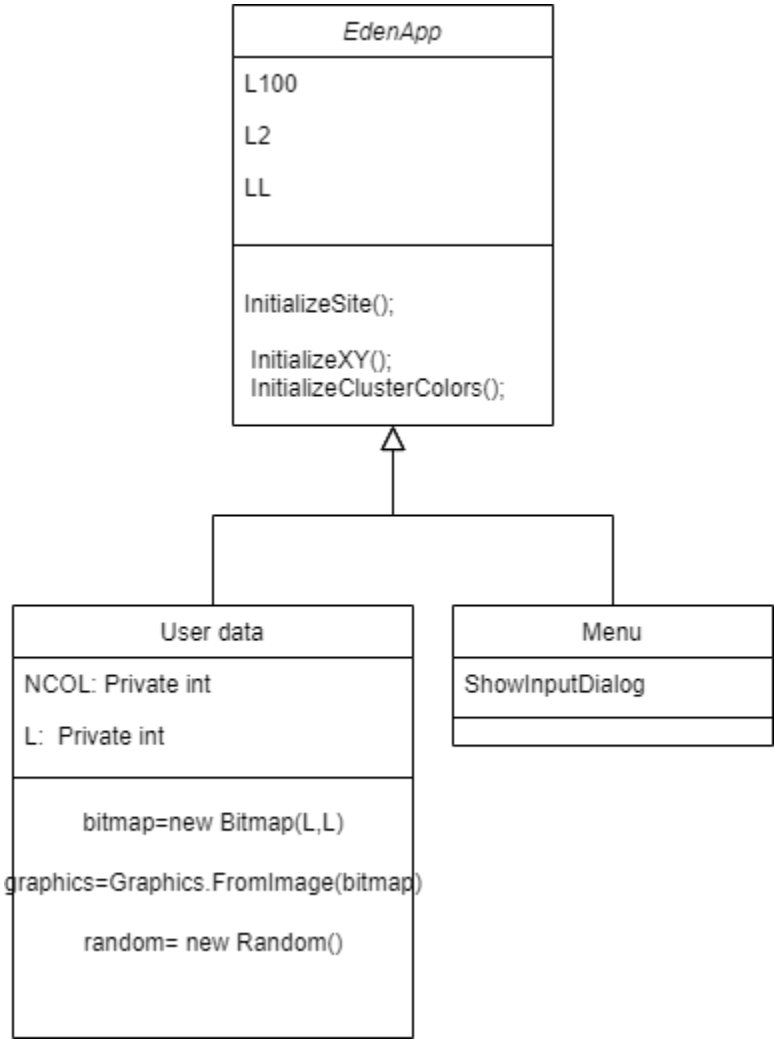


Рисунок 3.1 – Діаграма класів

Загалом, моделювання інформаційної системи для росту кластерів на двовимірному просторі допомагає розуміти процеси, що відбуваються в системі, та

дозволяє розробникам та користувачам отримати чітке уявлення про її функціональність та можливості.

3.1 Кодова реалізація

Надана кодова реалізація демонструє метод Ідена для моделювання процесу росту кластерів на двовимірному просторі. Ідея полягає у послідовному додаванні точок до кластерів за допомогою випадкового вибору напрямку руху. Починаючи з початкової точки, алгоритм генерує випадкову координату в межах заданого простору та перевіряє, чи ця точка вже належить якому-небудь кластеру. Якщо точка є новою, вона додається до поточного кластеру, а також генеруються нові точки для руху усіх кластерів.

Процес триває до досягнення заданої межі розміру простору або заданої кількості точок. Кластери ростуть та поширюються на всій площині, заповнюючи вільні місця та формуючи складні структури. Кожен кластер має свій унікальний ідентифікатор, що допомагає візуалізувати різні кластери на графічному інтерфейсі. В процесі росту кластерів використовується випадковість для визначення напрямку руху та моменту зупинки.

Головна ідея методу Ідена полягає у створенні реалістичних структур шляхом імітації природних процесів росту. Цей підхід може бути застосований у різних областях, таких як фізика, біологія, комп'ютерна графіка та інші, де необхідно моделювати рост систем або структур. Код, який наведений вище, демонструє просту реалізацію методу Ідена на двовимірному просторі з використанням мови програмування C# та бібліотеки Windows Forms для візуалізації процесу росту кластерів на графічному інтерфейсі.

Головною метою цього коду є створення інформаційної системи, яка моделює процес росту кластерів. Вхідні дані включають розміри простору, кількість точок та параметри, що визначають поведінку кластерів, такі як швидкість росту та ймовірність вибору напрямку руху.

Система складається з ряду компонентів, які взаємодіють між собою. На початку роботи системи ініціалізується простір заданого розміру та створюється початкова точка. Після цього починається ітераційний процес росту кластерів.

Кожна ітерація включає в себе додавання нової точки до кластеру. За допомогою генерації випадкових чисел визначається напрямок руху точки, яка додається до поточного кластеру. Також генеруються нові точки для руху усіх існуючих кластерів згідно з визначеними параметрами. Цей процес повторюється до досягнення заданої кількості точок або до заповнення простору.

Рисунок 3.1 – Блоксхема

Узагальнюючи, цей код демонструє реалізацію моделювання росту кластерів на двовимірному просторі з використанням мови програмування C# та бібліотеки

Windows Forms. Ця інформаційна система дозволяє візуалізувати та досліджувати процес росту кластерів, враховуючи задані параметри та розміри простору.

3.1.1 Принцип роботи коду

```
private const int L = 1024;
private const int NCOL = 10000000;
private const int L2 = L / 2;
private const int LL = L * L;
private const int L100 = L2 - 100;
private static readonly int[] dx = { 1, 0, -1, 0 };
private static readonly int[] dy = { 0, -1, 0, 1 };
private static readonly Random random = new Random();
private static readonly Bitmap bitmap = new Bitmap(L, L);
private static readonly Graphics graphics = Graphics.FromImage(bitmap);
private static readonly PictureBox pictureBox = new PictureBox();
private static readonly List<List<int>> site = new List<List<int>>();
private static readonly List<int> x = new List<int>();
private static readonly List<int> y = new List<int>();
private static readonly Dictionary<int, Color> clusterColors = new Dictionary<int, Color>();
```

Рисунок 3.4 – Змінні для ініціалізації моделі

Роль змінних у цьому коді дуже важлива, оскільки вони виконують різні функції, необхідні для коректної роботи алгоритму моделювання. Змінні дозволяють зберігати та оновлювати інформацію про доступні напрямки руху, що є важливим для визначення нових координат точки. Деякі змінні визначають поточні координати точки, яка розглядається в алгоритмі. Ці змінні змінюються під час руху по моделі. Лічильник n дозволяє відстежувати кількість кроків руху, що може бути використано для виконання певних дій після певної кількості кроків. Iite є структурою даних, що представляє модель. Вона зберігає інформацію про те, які точки вже зайняті. Це дозволяє контролювати поширення та розміщення точок у моделі. Змінна col використовується для вибору кольору, який буде використовуватись для малювання точок на графічному інтерфейсі. Нижче наведена таблиця в якій вказаний сенс кожної змінної в коді.

Таблиця 3.1 – Назви змінних та їх роль

Назва змінної	ЇЇ роль
L	константа визначає розмір сторони квадратної сітки. Це число використовується для визначення розмірів зображення та сітки для прогнозування поширення епідеміологічних захворювань. Загальна кількість точок у сітці буде L на L;
NCOL	константа визначає кількість елементів у списку site. Значення NCOL дорівнює 10000000. Список site використовується для збереження інформації про кластери в сітці. Кожен елемент у списку site представляє окремий кластер і містить список координат (X, Y) точок у цьому кластері;
L2	змінна представляє половину значення L. Значення L2 дорівнює 512. Використовується для обчислення різних позицій у сітці. Наприклад, L2 використовується для обчислення L100 (L2 - 100), яке вказує на зсув для певних операцій у сітці;
LL	змінна представляє квадрат значення L. Значення LL дорівнює 1048576. Використовується для обчислення різних позицій у сітці та розмірів зображення. Наприклад, для перетворення двовимірних координат (X, Y) у сітці у одновимірний індекс використовується формула index = Y * L + X, де L замінюється на LL;

Продовження таблиці 3.1

L100	змінна представляє значення L2 мінус 100. Значення L100 дорівнює 412. Використовується для обчислення різних позицій у сітці. Наприклад, L100 використовується для обчислення зсуву для деяких операцій у сітці;
dx	масив, що містить зсуви по осі X для пересування по сітці. Масив dx містить значення { 1, 0, -1, 0 }, масив, що містить 4 цілих числа. Використовується для збереження зсувів для обходу сусідніх точок у сітці за напрямком X. Ці значення використовуються для пересування у сусідні точки сітки;
dy	масив, що містить зсуви по осі Y для пересування по сітці. Масив dy містить значення { 0, -1, 0, 1 }. масив, що містить 4 цілих числа. Використовується для збереження зсувів для обходу сусідніх точок у сітці за напрямком Y. Ці значення використовуються для пересування у сусідні точки сітки;
random	об'єкт класу Random, який використовується для генерації випадкових чисел у коді. Використовується для генерації випадкових координат точок та випадкових кольорів для кластерів;

Закінчення таблиці 3.1

clusterColors	словник, який містить пари ключ-значення, де ключ - індекс кластера, а значення - відповідний колір для цього кластера. Використовується для призначення кожному кластеру унікального кольору для візуалізації на зображенні.

Даний фрагмент коду на мові C# містить ініціалізацію різних змінних, які використовуються в програмі. Змінна L визначає розмір сторони квадратної області, а NCOL представляє кількість стовпців даних. Інші змінні, такі як L2, LL і L100, використовуються для розрахунків та зміщень у межах області.

Також, у коді існують масиви dx та dy, які містять значення зміщення в різних напрямках, що допомагає у навігації по матриці або масиву. Об'єкт random використовується для генерації випадкових чисел. Для роботи з графікою використовуються об'єкти bitmap і graphics, які дозволяють створювати та малювати зображення. Елемент pictureBox використовується для відображення зображення в графічному інтерфейсі програми.

Крім того, у коді є списки site, x та y, які використовуються для зберігання та обробки даних. А словник clusterColors використовується для зберігання кольорів, пов'язаних з кластерами. Загальна мета цих змінних полягає в створенні та управлінні необхідними об'єктами та даними для подальшої роботи програми з моделлю Eden.

```
InitializeSite();
InitializeXY();
InitializeClusterColors();

Form form = new Eden();
form.Text = "Eden Model";
form.ClientSize = new Size(L, L);

pictureBox.Dock = DockStyle.Fill;
pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
pictureBox.Image = bitmap;

form.Controls.Add(pictureBox);
form.Show();
```

Рисунок 3.5 – Початкові методи ініціалізації

У частині коду, яку було наведено, відбувається налаштування графічного інтерфейсу для відображення моделі Ідена.

Даний фрагмент коду на мові C# відображає процес налаштування графічного інтерфейсу в програмі. Він починається зі створення форми, якій присвоюється назва "Eden".. Далі створюється елемент PictureBox, який буде відповідальний за відображення зображення моделі. Після ініціалізації елементів графічного інтерфейсу та їх налаштування, фрагмент коду встановлює форму з відповідними параметрами. Назва форми встановлюється на "Eden", а її розмір визначається за допомогою ClientSize, де L відповідає розміру 1024x1024 пікселів. Цей елемент налаштовується на розтягнення на всю доступну площу форми за допомогою властивості Dock. Зображення моделі, яке зберігається у змінній bitmap, прив'язується до елемента PictureBox. Після цього елемент PictureBox додається до списку контролів форми, щоб бути розміщеним у формі. Нарешті, за допомогою методу Show(), форма відображається на екрані, дозволяючи користувачеві бачити та взаємодіяти з програмою. Цей фрагмент коду демонструє важливі кроки для налаштування графічного інтерфейсу, зокрема розміщення

елементів та прив'язку зображень до них, що робить програму більш інтуїтивно зрозумілою та привабливою для користувачів. Нарешті, за допомогою методу Show() відображається форма на екрані.

Таким чином, цей фрагмент коду встановлює та налаштовує графічний інтерфейс програми, створює форму з заданими параметрами, додає елемент PictureBox для відображення зображення моделі та відображає форму на екраністворює форму з відповідними розмірами та додає елемент PictureBox для відображення зображення моделі. Коли форма відображається, користувач може бачити та взаємодіяти з програмою через графічний інтерфейс, що робить його більш зручним та дружнім до використання.

```
for (int i = 0; i < 4; i++)
{
    np++;
    x.Add(dx[i]);
    y.Add(dy[i]);
}

while (xp < L100 && yp < L100)
{
    double rnd = random.NextDouble();
    int rndIndex = random.Next(0, np);
    xp = x[rndIndex];
    yp = y[rndIndex];
    n++;
    site[xp + L2][yp + L2] = clusterCount;
    x[rndIndex] = x[np - 1];
    y[rndIndex] = y[np - 1];
    np--;
```

Рисунок 3.6 – Початок циклу

Завдання цього фрагмента коду - генерація випадкових координат для точок моделі та їх оновлення відповідно до певних правил.

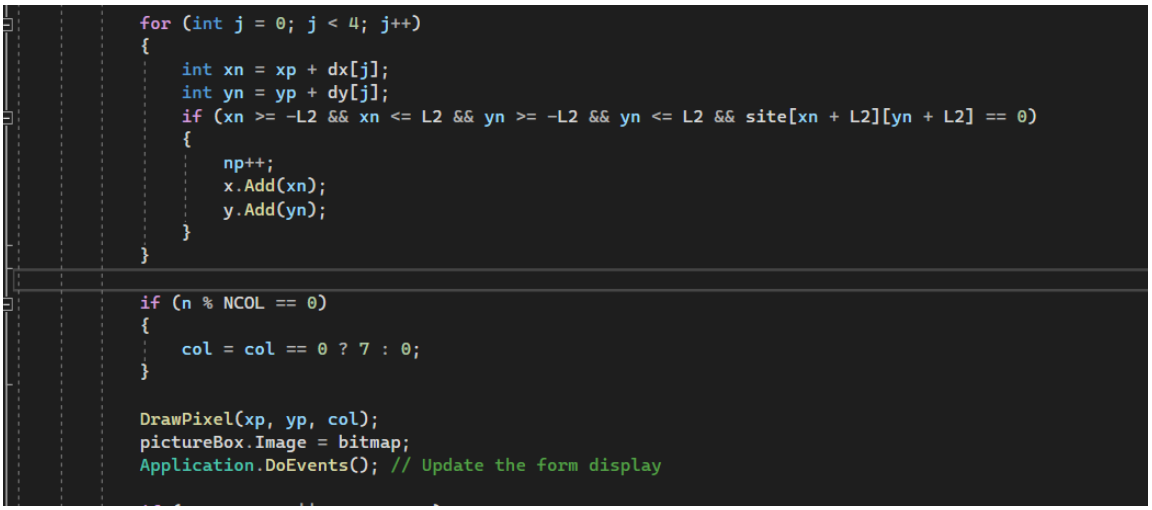
Цей фрагмент коду відповідає за генерацію випадкових координат для точок моделі та їх оновлення згідно з певними правилами. Спочатку виконується цикл for, який ініціалізує значення змінних для керування генерацією координат точок. Після цього виконується цикл while, який триває до досягнення заданих обмежень координат. У циклі генеруються випадкові числа для вибору точок та їх оновлення відповідно до визначених правил.

На початку виконується цикл for, який проходить чотири ітерації. Кожна ітерація збільшує значення лічильника np на одиницю та додає значення dx[i] та dy[i] до відповідних списків x та y. Це дозволяє ініціалізувати початкові значення для подальшої обробки.

Далі виконується цикл while, який триває, поки значення xp та yp менше L100 (значення L2 - 100). У циклі генерується випадкове дробове число rnd за допомогою методу random.NextDouble(). Також генерується випадковий індекс rndIndex в межах від 0 до np, що дозволяє вибрати випадкову точку зі списків x та y.

Значення xp та yp встановлюються на вибрану точку, лічильник n збільшується на одиницю, а значення clusterCount присвоюється елементу site[xp + L2][yp + L2]. Після цього відбувається перестановка вибраної точки з останньою точкою в списку x та y, а значення np зменшується на одиницю. Це призводить до вилучення обробленої точки з активних точок та збереження лише невикористаних точок для подальшої обробки.

Таким чином, цей фрагмент коду виконує процес генерації випадкових координат для точок моделі та оновлення відповідних змінних та структур даних для подальшої обробки та відображення моделі. Кожна точка отримує випадкові координати, і після обробки вона виключається зі списку активних точок, щоб не була використана повторно.



```
for (int j = 0; j < 4; j++)
{
    int xn = xp + dx[j];
    int yn = yp + dy[j];
    if (xn >= -L2 && xn <= L2 && yn >= -L2 && yn <= L2 && site[xn + L2][yn + L2] == 0)
    {
        np++;
        x.Add(xn);
        y.Add(yn);
    }
}

if (n % NCOL == 0)
{
    col = col == 0 ? 7 : 0;
}

DrawPixel(xp, yp, col);
pictureBox.Image = bitmap;
Application.DoEvents(); // Update the form display
```

Рисунок 3.7 – Цикл розміщення нових точок

У даному фрагменті коду відбувається перевірка можливості розміщення нових точок у моделі. За допомогою циклу for перебираються всі можливі напрямки руху (вправо, вниз, вліво, вгору) від поточної точки (xp, yp). Для кожного напрямку обчислюються нові координати (xn, yn) і перевіряється, чи вони знаходяться в межах дозволених координат та чи не зайняті вже іншими точками. Якщо ці умови виконуються, то нові координати додаються до списків x та y, а лічильник np збільшується.

Далі, умова if (n % NCOL == 0) перевіряє, чи досягнуто необхідної кількості кольорів, яка визначається за допомогою константи NCOL. Якщо умова виконується, то змінна col змінюється на протилежне значення: якщо col дорівнює 0, то стає 7, і навпаки.

Далі викликається функція DrawPixel, яка малює піксель на вказаних координатах (xp, yp) з використанням кольору col. Зображення оновлюється в елементі pictureBox.Image, а за допомогою Application.DoEvents() оновлюється відображення форми, щоб зміни стали видимими.

Цей фрагмент коду відповідає за рух та малювання нових точок на формі. Він перебирає можливі напрямки руху, обчислює нові координати та перевіряє їх допустимість. Якщо умови виконуються, нові координати додаються до відповідних списків та лічильник збільшується. Крім того, код змінює кольори та оновлює візуальне представлення моделі на формі. Таким чином, цей фрагмент забезпечує динамічний процес росту моделі та відображення його на формі.

```
            if (xp == L100 || yp == L100)
            {
                clusterCount++;
                InitializeClusterColors();
            }
        }

        Console.WriteLine("Iterations: " + n);
        Application.Run(form);
    }
```

Рисунок 3.8 – Цикл перевірки місця

У даному фрагменті коду перевіряється, чи поточні координати xp та yp дорівнюють L100. Якщо ця умова виконується, це означає, що точка дійшла до межі моделі і утворює новий кластер.

Коли утворюється новий кластер, змінна clusterCount збільшується, що використовується для відстежування кількості кластерів у моделі. Також, викликається функція InitializeClusterColors(), яка скидає кольори кластерів, щоб почати з новими кольорами для нового кластеру.

Після закінчення циклу while виводиться повідомлення про кількість ітерацій n на консоль, що є інформацією про хід виконання алгоритму моделювання. Далі виконується функція Application.Run(form), яка запускає графічний інтерфейс програми та відображає форму зі згенерованим зображенням моделі.

Цей фрагмент коду відповідає за завершення виконання алгоритму, виведення результатів та показ інтерфейсу програми зі згенерованим зображенням моделі.

На останніх рядках коду відбувається виведення на консоль кількості ітерацій (значення змінної "n"), які були виконані в процесі моделювання. Це дає інформацію про загальну кількість кроків, які були зроблені для створення моделі. Після цього, викликається метод Application.Run(form), що запускає головний цикл подій програми і відображає форму на екрані. Це забезпечує інтерактивність та взаємодію з користувачем, який може спостерігати результати моделювання через графічний інтерфейс.

Узагальнюючи, останні рядки коду виводять інформацію про кількість ітерацій та запускають головний цикл подій програми для відображення форми з результатами моделювання. Це дозволяє користувачу бачити та взаємодіяти з отриманими даними на екрані.

3.1.2 Основні методи

У цьому коді присутні основні методи, які відповідають за ініціалізацію початкових значень, обробку даних і відображення результатів. Ці методи виконують важливі функції для створення та відображення моделі на графічному інтерфейсі. Вони взаємодіють один з одним, щоб забезпечити правильний хід моделювання і коректне відображення даних. Крім того, присутні інші конструкції коду, такі як цикли та умовні оператори, які контролюють логіку виконання програми. Всі ці елементи разом допомагають реалізувати функціональність моделі та відображення результатів на екрані.

```
1 reference
static void InitializeSite()
{
    for (int i = 0; i <= 2 * L2; i++)
    {
        site.Add(new List<int>());
        for (int j = 0; j <= 2 * L2; j++)
        {
            site[i].Add(0);
        }
    }
}
```

Рисунок 3.9 – Метод ініціалізації

Метод InitializeSite() виконує створення та ініціалізацію двовимірного списку site, який представляє сітку для моделювання. У цьому методі використовується вкладений цикл для створення двовимірного списку з розміром 2 * L2 на 2 * L2. Зовнішній цикл проходить через рядки сітки, а внутрішній цикл - через стовпці. Кожному елементу сітки присвоюється початкове значення 0.

Це означає, що кожна точка на сітці починає зі значення 0, що може відповідати відсутності захворювання або кластеру в даній області. Цей метод дозволяє підготувати початковий стан моделі перед початком моделювання поширення захворювання.

Після виклику методу InitializeSite(), ми отримуємо готову сітку site, де кожен елемент представляє окрему точку на сітці. Цей метод є важливим кроком для налагодження початкових умов моделі та підготовки її до подальшого моделювання та аналізу.

```
static void InitializeClusterColors()
{
    clusterColors.Clear();
    for (int i = 1; i <= 10; i++)
    {
        int r = random.Next(0, 256);
        int g = random.Next(0, 256);
        int b = random.Next(0, 256);
        Color color = Color.FromArgb(r, g, b);
        clusterColors.Add(i, color);
    }
}
```

Рисунок 3.10 – Метод ініціалізації кольорів

Метод InitializeClusterColors() виконує ініціалізацію кольорів для кластерів у моделі. Спочатку він очищає колекцію clusterColors, щоб гарантувати, що вона буде порожньою перед налаштуванням нових кольорів.

Далі відбувається цикл, який проганяється 10 разів, від 1 до 10, для кожного потенційного кластера. На кожній ітерації циклу генерується випадковий колір для кластера. Це досягається шляхом вибору випадкових значень червоного (r), зеленого (g) та синього (b) каналів у межах від 0 до 255. Колір формується з цих випадкових значень за допомогою методу Color.FromArgb(r, g, b). Отриманий колір потім додається до колекції clusterColors разом з відповідним індексом кластера за допомогою методу Add().

Таким чином, метод InitializeClusterColors() гарантує, що для кожного кластера у моделі буде встановлений унікальний колір. Це дозволяє візуально відрізняти різні кластери під час відображення моделі та поліпшує зрозуміння та аналіз результатів моделювання. Кольорове розмежування кластерів робить візуалізацію більш зрозумілою та привабливою для користувачів.

3.1.3 Графічний інтерфейс

Графічний інтерфейс є ключовим елементом взаємодії користувача з програмним забезпеченням. Він надає можливість візуально сприймати та контролювати функціонал програми, що забезпечує зручність та ефективність використання. Графічний інтерфейс дозволяє користувачам взаємодіяти з програмою за допомогою інтуїтивно зрозумілих елементів, таких як кнопки, поля введення, списки тощо, що спрощує навігацію та виконання завдань.

Важливість графічного інтерфейсу полягає в тому, що він забезпечує зручність, ефективність та задоволення користувача. Графічний інтерфейс допомагає створити зручне та привабливе середовище, в якому користувачі можуть легко зорієнтуватись та взаємодіяти з програмою. Він дозволяє візуально представити інформацію та дозволяє користувачам здійснювати операції за допомогою простих та зрозумілих дій.

Графічний інтерфейс також сприяє поліпшенню користувацького досвіду. Завдяки зручним інтерфейсним елементам та інтуїтивному дизайну, користувачі можуть швидко оволодіти програмою, знижуючи час на навчання та підвищуючи ефективність роботи. Крім того, графічний інтерфейс може включати функціональні можливості, такі як валідація даних, повідомлення про помилки або контекстна довідка, що поліпшують користувацький досвід та забезпечують більшу задоволеність від використання програми.

Узагалі, графічний інтерфейс відіграє важливу роль у розробці програмного забезпечення, оскільки він забезпечує комфортну та ефективну взаємодію між користувачем та програмою. Він спрощує навігацію, дозволяє швидко здійснювати дії та забезпечує зручність використання програми, що є ключовими факторами для успіху програмного продукту.

```
Form inputForm = new Form();
inputForm.Width = 250;
inputForm.Height = 150;
inputForm.StartPosition = FormStartPosition.CenterScreen;
inputForm.Text = "Input";

Label labelL = new Label();
labelL.Text = "Enter value for L:";
labelL.AutoSize = true;
labelL.Left = 20;
labelL.Top = 20;

TextBox textBoxL = new TextBox();
textBoxL.Width = 150;
textBoxL.Left = 20;
textBoxL.Top = 50;

Label labelNCOL = new Label();
labelNCOL.Text = "Enter value for NCOL:";
labelNCOL.AutoSize = true;
labelNCOL.Left = 20;
labelNCOL.Top = 80;
```

Рисунок 3.11 – кодова реалізація діалогового вікна

Цей код відповідає за створення вікна для введення даних від користувача. Виклик методу ShowInputDialog() ініціює процес створення нової форми inputForm, яка буде відображатись на екрані. Налаштування форми включає встановлення її розмірів, позиції та заголовку "Input".

```
        inputForm.Controls.Add(labelL);
        inputForm.Controls.Add(textBoxL);
        inputForm.Controls.Add(labelNCOL);
        inputForm.Controls.Add(textBoxNCOL);
        inputForm.Controls.Add(buttonOk);

        if (inputForm.ShowDialog() == DialogResult.OK)
        {
            if (int.TryParse(textBoxL.Text, out int l) && int.TryParse(textBoxNCOL.Text, out int ncol))
            {
                L = l;
                NCOL = ncol;
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter valid integer values.");
                Environment.Exit(0);
            }
        }
        else
        {
            Environment.Exit(0);
        }
    }
```

Рисунок 3.12 – Парсинг даних

На формі створюються декілька елементів управління, які включають мітки Label та текстові поля TextBox. Мітки labelL і labelNCOL відповідають за відображення тексту "Enter value for L:" та "Enter value for NCOL:" відповідно. Текстові поля textBoxL і textBoxNCOL призначені для введення даних користувачем.

До форми також додається кнопка "OK" (buttonOk), яка дозволяє користувачу підтвердити введені дані. При натисканні кнопки "OK" виконується діалогове вікно ShowDialog(), що очікує на взаємодію з користувачем.

Якщо користувач натискає кнопку "OK", виконується перевірка введених даних за допомогою int.TryParse(). Це дозволяє перевірити, чи введені значення є цілими числами. Якщо перевірка пройшла успішно, значення з текстових полів textBoxL та textBoxNCOL присвоюються змінним L та NCOL відповідно. В протилежному випадку, коли введені дані є недійсними, виводиться повідомлення про невірний ввід, і програма завершує своє виконання.

Загальна мета цього коду полягає в тому, щоб дозволити користувачеві ввести значення для змінних L та NCOL, які використовуються в подальшому

виконанні програми. Використовуючи графічний інтерфейс, цей код надає зручний спосіб взаємодії з користувачем для введення необхідних параметрів.

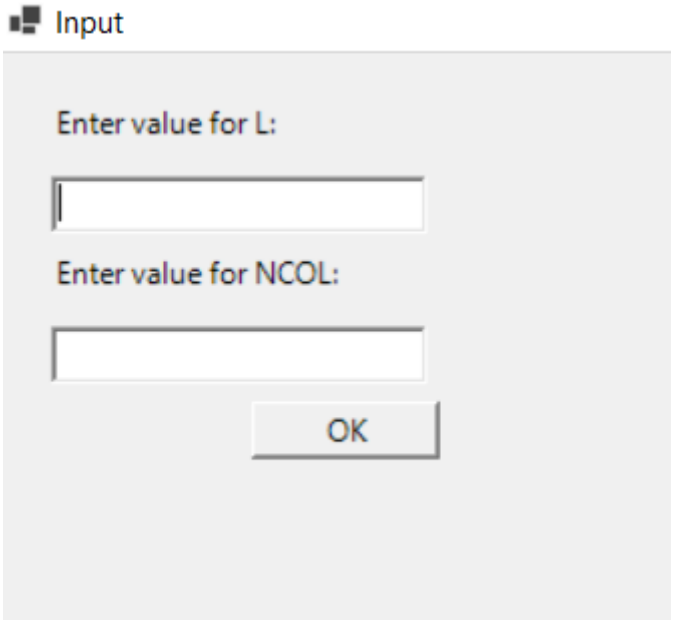


Рисунок 3.13 – Вікно взаємодії користувача

У контексті даної моделі, важливість мати графічний вивід полягає в тому, що він надає візуальне представлення результатів обчислень і сприяє кращому розумінню та візуалізації моделі.

Перш за все, графічний вивід дозволяє візуалізувати результати моделювання. В цій моделі, кожне зерно посідає певну позицію на двовимірній сітці, і графічний інтерфейс дозволяє відображати цю сітку та розташування зерен. Завдяки графічному виводу, користувач може бачити, як зерна розміщуються та рухаються на сітці, що допомагає зрозуміти поведінку системи.

Далі, графічний інтерфейс дозволяє в реальному часі відображати процес моделювання. За допомогою елемента PictureBox, зображення моделі може оновлюватись на кожній ітерації, що дозволяє відслідковувати прогрес та зміни в системі. Це особливо корисно для аналізу та наочного спостереження динаміки моделі.

Крім того, графічний інтерфейс робить модель більш доступною для користувачів без технічного бекграунду. Він дозволяє взаємодіяти з моделлю шляхом введення параметрів через текстові поля та кнопки. Це спрощує процес користування моделлю та дозволяє швидко змінювати параметри для проведення різних експериментів.

Таким чином, графічний вивід у цій моделі допомагає покращити візуалізацію результатів, спрощує взаємодію з користувачем та дозволяє краще розуміти та аналізувати динаміку системи. Все це робить його важливою складовою для наукових досліджень та роботи з даною моделлю.

Рисунок 3.14 – Результат роботи

Висновки до розділу 3

З кодовою реалізацією моделі Ідена з графічним інтерфейсом важливою є можливість візуалізації процесу моделювання та аналізу його результатів. Графічний інтерфейс дозволяє спостерігати розташування та рух зерен на сітці, що

полегшує розуміння та аналіз моделі. Крім того, взаємодія з моделлю через графічний інтерфейс дозволяє зручно вводити дані та змінювати параметри моделі. Такий підхід допомагає покращити доступність моделі для користувачів та сприяє проведенню експериментів та аналізу результатів. Кодова реалізація включає алгоритми моделювання, які дозволяють створювати та оновлювати модель, обробляти ітерації та взаємодіяти з графічним інтерфейсом. Крім того, використання віконного інтерфейсу дає можливість користувачам вводити дані та змінювати параметри моделі зручним способом. Завдяки графічному інтерфейсу користувач може спостерігати та аналізувати поведінку системи, зміни в розташуванні зерен та динаміку моделі, що допомагає встановлювати зв'язки між параметрами та поведінкою системи. Такий підхід сприяє поліпшенню розуміння моделі та виявленню нових залежностей та властивостей.

ВИСНОВКИ

У данній роботі була проведена детальна аналітична робота над моделлю Ідена. Були розглянуті різні типи моделей стохастичного зростання, зосереджуючись на моделях Ідена. Дослідження включало аналіз літературних джерел та наукових статей, що описують основні концепції та методи моделювання зростання за допомогою моделей Ідена.

У рамках роботи було проаналізовано принципи роботи моделей Ідена, їх математичну основу та основні характеристики. Були розглянуті алгоритми та методи, які використовуються для виконання обчислень та моделювання зростання за допомогою цих моделей.

Крім того, робота включала вивчення інструментів для написання коду, зокрема мови програмування C# та фреймворку Windows Forms. Було детально розглянуто можливості цих інструментів у контексті розробки програмного забезпечення з графічним інтерфейсом користувача. Пояснено основні концепції та практики програмування на мові C#, а також представлені можливості фреймворку Windows Forms для створення графічних інтерфейсів.

Наступним етапом була програмна реалізація коду, що базується на моделі Ідена. За допомогою мови програмування C# та фреймворку Windows Forms було створено програмний продукт, який надавав користувачеві можливість моделювати та прогнозувати стохастичні процеси зростання. Було розроблено графічний інтерфейс, що дозволяв вводити вхідні дані, виконувати розрахунки та відображати результати моделювання.

Висновком з роботи стало підтвердження важливості моделі Ідена у контексті стохастичного зростання, а також практична демонстрація ефективності мови програмування C# та фреймворку Windows Forms для створення програмного забезпечення на основі даної моделі. Результати дослідження та програмна реалізація коду можуть бути використані як основа для подальших досліджень та розробок в галузі стохастичного моделювання зростання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eden M., Thevenaz P. History of a stochastic growth model // Proc. SPIE. 1998. — V. 336. — P. 43-54.

2. Eden M. Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability / Ed. by F. Neyman. V. 4: Biology and Problems of Health. — Berkeley: University of California Press, 1961. — P. 223-239.

3. Iullien R., Botet R. Scaling properties of the surface of the Eden model in D=2,3,4 // J. Phys. AI8. — 1985. — P. 2279-2287.

4. Zabolitzky J. G., Stauffer D. Simulation of large Eden clusters // Phys. Rev. A. — 1986. — V. 34, No. 2. — P. 1523-1530.

5. Batchelor M.T., Henry B.1. Limits to Eden growth in 2-dimensions and 3-dimensions // Phys. Lett. A. — 1991. — P. 229-236.

6. Wang C.Y,, Liu P.L., Bassingthwaighte I. B. Off-lattice Eden-C cluster growth-model // J. Phys. A28. — 1995. — P. 2141-2147.

7. Wang C. Y., Bassingthwaighte J. B. Biological Growth on a Surface // Mathematical Biosciences. — 1997. — V. 142. — P. 91-106

8. Maini P.K., Othmer H.G. ed. Mathematical models for biological pattern formation. — New York, Berlin, Heidelberg: Springer-Verlag, 2000. — 317 p.

9. Savakis A. E. and Maggelakis S. Models of Shrinking Clusters with Applications to Wound Healing // Mathematical and Computer Modeling. — 1997. P. 1-6.

10. Drasdo D., Kree R., and McCaskill J. Monte carlo approach to tissue cell populations // Phys. Rev. E. — 1995. — V. 52. — P. 6635-6657.

11. Ausloos M., Vandewalle N. and Cloots R. Magnetic Eden Model // Europhys. Lett. — 1993. — V. 24. — P. 629-634.

12. Lebovka N.I., Ivanenko Ya.V., and Vygornitskii N. V. Deterministic Eden model of charged-particles aggregation // Europhys. Lett. — 1998. — V. 41. — P.9-24.

13. Leheny R. Simple model for river network evolution // Phys. Rev. E. — 1995. — V.52. — P. 5610-5620.

14. Benguigui L. A new aggregation model: Application to town growth // Physica A. — 1995. — V. 219. — P. 13-26.

15. Hammersley I. M., Handscomb D.C. Monte Carlo methods. — London: Methuen, 1964. — 178 p.

16. Meakin P. Formation of Fractal Clusters and Networks by Irreversible Diffusion-Limited Aggregation // Phys. Rev. Lett. — 1983. — V.51. — P. 1119-1122.

17. Rikvold P.A. Simulations of a stochastic model for cluster growth on a square lattice // Phys. Rev. A. — 1982. — V. 26. — P. 647-650.

18. Zigin W., Boquan L. Random successive grown model for pattern formation // Phys. Rev. E. — 1995. — V. 51, No. I. — P. R16-RI9.

19. Margolina A. The fractal dimension of cluster perimeters generated by a kinetic walk // J. Phys. A: Math. Gen. — 1985. — V. 18. — P. L651-1656.

20. Kree R., and McCaskill J. Monte carlo approach to tissue cell populations // Phys. Rev. E. — 1998. — V. 53. — P. 6635-6657.

ДОДАТОК А Кодова реалізація методу Ідена

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

class Eden : Form
{
    private static long L;
    private static long NCOL;
    private static long L2;
    private static long LL;
    private static long L100;
    private static long[] dx;
    private static long[] dy;
    private static Random random;
    private static Bitmap bitmap;
    private static Graphics graphics;
    private static PictureBox pictureBox;
    private static List<List<long>> site;
    private static List<long> x;
    private static List<long> y;
    private static Dictionary<long, Color> clusterColors;

    static void Main()
    {
        ShowInputDialog();

        L2 = L / 2;
        LL = L * L;
        L100 = L2 - 100;
        dx = new long[] { 1, 0, -1, 0 };
        dy = new long[] { 0, -1, 0, 1 };
        random = new Random();
        bitmap = new Bitmap((int)(L * 10), (int)(L * 10));
        graphics = Graphics.FromImage(bitmap);
        pictureBox = new PictureBox();
        site = new List<List<long>>();
        x = new List<long>();
        y = new List<long>();
        clusterColors = new Dictionary<long, Color>();

        InitializeSite();
        InitializeXY();
        InitializeClusterColors();

        Form form = new Eden();
        form.Text = "Eden Model";
        form.ClientSize = new Size((int)L, (int)L);

        pictureBox.Dock = DockStyle.Fill;
        pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
        pictureBox.ClientSize = new Size((int)(L * 10), (int)(L * 10));
        pictureBox.Image = bitmap;



        form.Controls.Add(pictureBox);
        form.Show();
```

```
        long col = 0;
        long n = 0;
        long np = 0;
        long xp = 0;
        long yp = 0;
        long clusterCount = 1;

        for (int i = 0; i < 4; i++)
        {
            np++;
            x.Add(dx[i]);
            y.Add(dy[i]);
        }

        while (xp < L100 && yp < L100)
        {
            double rnd = random.NextDouble();
            int rndIndex = random.Next(0, (int)np); // Generate a random index within the
valid range
            xp = x[rndIndex];
            yp = y[rndIndex];
            n++;
            site[(int)(xp + L2)][(int)(yp + L2)] = clusterCount;
            x[rndIndex] = x[(int)np - 1];
            y[rndIndex] = y[(int)np - 1];
            np--;

            for (int j = 0; j < 4; j++)
            {
                long xn = xp + dx[j];
                long yn = yp + dy[j];
                if (xn >= -L2 && xn <= L2 && yn >= -L2 && yn <= L2 && site[(int)(xn +
L2)][(int)(yn + L2)] == 0)
                {
                    np++;
                    x.Add(xn);
                    y.Add(yn);
                }
            }

            if (n % NCOL == 0)
            {
                col = col == 0 ? 7 : 0;
            }

            DrawPixel((int)xp, (int)yp, (int)col);
            pictureBox.Image = bitmap;
            Application.DoEvents(); // Update the form display

            if (xp == L100 || yp == L100)
            {
                clusterCount++;
                InitializeClusterColors();
            }
        }

        Console.WriteLine("Iterations: " + n);
        Application.Run(form);
    }
```

```
    static void ShowInputDialog()
    {
        Form inputForm = new Form();
        inputForm.Width = 250;
        inputForm.Height = 150;
        inputForm.StartPosition = FormStartPosition.CenterScreen;
        inputForm.Text = "Input";

        Label labelL = new Label();
        labelL.Text = "Enter value for L:";
        labelL.AutoSize = true;
        labelL.Left = 20;
        labelL.Top = 20;

        TextBox textBoxL = new TextBox();
        textBoxL.Width = 150;
        textBoxL.Left = 20;
        textBoxL.Top = 50;

        Label labelNCOL = new Label();
        labelNCOL.Text = "Enter value for NCOL:";
        labelNCOL.AutoSize = true;
        labelNCOL.Left = 20;
        labelNCOL.Top = 80;

        TextBox textBoxNCOL = new TextBox();
        textBoxNCOL.Width = 150;
        textBoxNCOL.Left = 20;
        textBoxNCOL.Top = 110;

        Button buttonOk = new Button();
        buttonOk.Text = "OK";
        buttonOk.DialogResult = DialogResult.OK;
        buttonOk.Left = 100;
        buttonOk.Top = 140;

        inputForm.Controls.Add(labelL);
        inputForm.Controls.Add(textBoxL);
        inputForm.Controls.Add(labelNCOL);
        inputForm.Controls.Add(textBoxNCOL);
        inputForm.Controls.Add(buttonOk);

        if (inputForm.ShowDialog() == DialogResult.OK)
        {
            if (long.TryParse(textBoxL.Text, out long l) &&
long.TryParse(textBoxNCOL.Text, out long ncol))
            {
                L = l;
                NCOL = ncol;
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter valid integer values.");
                Environment.Exit(0);
            }
        }
        else
        {
            Environment.Exit(0);
        }
    }
```

```
    static void InitializeSite()
    {
        for (int i = 0; i <= 2 * L2; i++)
        {
            site.Add(new List<long>());
            for (int j = 0; j <= 2 * L2; j++)
            {
                site[i].Add(0);
            }
        }
    }

    static void InitializeXY()
    {
        x.Clear();
        y.Clear();
    }

    static void InitializeClusterColors()
    {
        clusterColors.Clear();
        for (int i = 1; i <= 10; i++)
        {
            int r = random.Next(0, 256);
            int g = random.Next(0, 256);
            int b = random.Next(0, 256);
            Color color = Color.FromArgb(r, g, b);
            clusterColors.Add(i, color);
        }
    }

    static void DrawPixel(int x, int y, int col)
    {
        int scaledX = (int)(x * 10 + L2 * 10);
        int scaledY = (int)(y * 10 + L2 * 10);

        // Calculate the adjusted center coordinates for the larger circle
        int centerX = scaledX + 2;
        int centerY = scaledY + 2;

        Color color = col == 0 ? Color.Black : clusterColors[col];

        // Draw a larger circle by filling a rectangle that fits within the scaled
coordinates
        graphics.FillEllipse(new SolidBrush(color), centerX - 4, centerY - 4, 8, 8);
        pictureBox.Image = bitmap;
    }
}
```