БАКАЛАВРСЬКА РОБОТА

БР. ІП - 52.00.00.000 ПЗ

Група ІП-21-3

Фролова Яна

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Фролова Яна Олександрівна

(прізвище, ім'я, по батькові)

УДК 004

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка та реалізація методу візуалізації кластеризації даних

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

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

Здобувач освітнього рівня Фролова Я.О.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Зікратий Сергій Вікторович, к.т.н., доцент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Фроловій Яні Олександрівні

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Розробка та реалізація методу візуалізації кластеризації даних”

керівник проекту (роботи) Зікратий С.В., доц., к.т.н.

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

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

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

1. Аналіз предметної області візуалізації техніки кластеризації даних
2. Розробка техніки візуалізації кластеризації даних на основі фізичних моделей
3. Представлення алгоритмів функціонування системи візуалізації кластеризації даних
4. Алгоритм візуалізації процесу кластеризації даних
5. Програмна імплементація методу візуалізації кластеризації даних

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

1. Графічне представлення процесу кластеризації даних (рис. 1.1)
2. Приклад виконання кластеризації текстової інформації (рис. 1.2)
3. Принцип роботи моделі Bag-of-Words (рис. 1.3)
4. Принцип подубови TF-IDF матриці (рис. 1.4)
5. Приклад дендограми побудованої під час ієрархічної кластеризації (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області візуалізації техніки кластеризації даних	04.05.2025	виконано
2	Розробка техніки візуалізації кластеризації даних на основі фізичних моделей	15.05.2025	виконано
3	Представлення алгоритмів функціонування системи візуалізації кластеризації даних	21.05.2025	виконано
4	Алгоритм візуалізації процесу кластеризації даних	28.05.2025	виконано
5	Програмна імплементація методу візуалізації кластеризації даних	03.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 79 сторінок, 35 рисунків, список використаних джерел із 36 найменуваннями.

Метою дипломної роботи є розробка та реалізація ефективного методу візуалізації кластеризації даних, що базується на фізичних моделях, для покращення інтуїтивного розуміння та аналізу складних мережевих структур.

Об'єктом дослідження є процеси візуалізації даних, що представляють собою мережеві структури.

Предметом дослідження є методи та алгоритми просторового розташування вершин графа, що імітують фізичні взаємодії, з метою покращення візуального виявлення кластерних структур у даних.

В першому розділі обґрунтовується актуальність кластеризації та візуалізації даних, підкреслюючи роль візуалізації у виявленні прихованих структур та аномалій у складних наборах даних

В другому розділі описано алгоритм візуалізації графіків на основі фізичних моделей та його адаптацію для ефективної кластеризації та візуалізації мережевих даних.

В третьому розділі виконано опис архітектури програмного забезпечення, вибору структур даних, реалізації компонентів GUI та представлення експериментальних результатів, що підтверджують функціональність та переваги розробленого методу.

Висновок: виконано програмну реалізацію методу візуалізації кластеризації даних, який забезпечує динамічну візуалізацію процесу кластеризації.

КЛЮЧОВІ СЛОВА: КЛАСТЕРИЗАЦІЯ ДАНИХ, ВІЗУАЛІЗАЦІЯ ІНФОРМАЦІЇ, АЛГОРИТМИ НА ОСНОВІ СИЛ, ГРАФІЧНІ МОДЕЛІ, МЕРЕЖЕВИЙ АНАЛІЗ, ХАБИ, ВИКИДИ, BIG DATA, JAVA SWING, АНІМОВАНА ВІЗУАЛІЗАЦІЯ.

ANNOTATION

The bachelor's thesis contains 79 pages, 35 figures, a list of used sources with 36 names.

The method of the thesis is the development and implementation of an effective method for visualizing data clustering, based on physical models, to improve the intuitive understanding and analysis of complex network structures.

The object of the study is the processes of visualizing data, which are network structures.

The subject of the study is methods and algorithms for spatial arrangement of graph vertices, which simulate physical interactions, in order to improve the visual detection of cluster structures in data.

The first section justifies the relevance of clustering and data visualization, emphasizing the role of visualization in detecting stored structures and anomalies in complex data sets.

The second section describes the algorithm for visualizing graphs based on physical models and its adaptation for effective clustering and visualization of network data.

The third section describes the software architecture, the choice of data structures, GUI implementation components, and presents experimental results that confirm the functionality and advantages of the developed method.

Conclusion: a software implementation of the data clustering visualization method has been implemented, which provides dynamic visualization of the clustering process.

KEYWORDS: DATA CLUSTERING, INFORMATION VISUALIZATION, FORCE-BASED ALGORITHMS, GRAPHICAL MODELS, NETWORK ANALYSIS, HUBS, OUTPUTS, BIG DATA, JAVA SWING, ANIMATED VISUALIZATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

EM – Expectation-Maximization – Експериментальна максимізація

DBSCAN – Density-Based Spatial Clustering of Applications with Noise – Просторовий кластерний аналіз на основі щільності з шумом

OPTICS – Ordering Points to Identify the Clustering Structure – Упорядкування точок для ідентифікації структури кластеризації

PAM – Partitioning Around Medoids – Кластеризація навколо медоїдів

STING – Statistical Information Grid – Статистична інформаційна сітка

JVM – Java Virtual Machine – Віртуальна машина Java

ВСТУП

У сучасному світі, що характеризується безпрецедентним зростанням обсягів даних (Big Data), здатність ефективно аналізувати та інтерпретувати цю інформацію стає ключовим фактором успіху в науці, бізнесі та суспільстві. Одним з найважливіших інструментів для вилучення знань із складних даних є кластеризація – процес групування схожих об'єктів у сукупності, або кластери. Ця методологія дозволяє виявляти приховані закономірності, структури та аномалії, що є основою для прийняття обґрунтованих рішень. Застосування кластеризації охоплює широкий спектр областей, від біоінформатики та медичної діагностики до маркетингового аналізу та виявлення шахрайства.

Однак, складність сучасних наборів даних часто ускладнює пряму інтерпретацію результатів кластеризації. Навіть найдосконаліші алгоритми можуть надавати вихідні дані, які потребують значних зусиль для візуального осмислення та виявлення реальних інсайтів. Саме тут на перший план виходить візуалізація даних. Візуалізація не просто представляє дані графічно; її мета полягає в тому, щоб підкреслити та розкрити фундаментальні явища та взаємозв'язки, використовуючи природні можливості людського зору та когнітивних процесів. Ефективна візуалізація дозволяє не тільки швидко інтерпретувати складні дані, але й виявляти неочевидні раніше структури, такі як хаби (вузли, що з'єднують різні кластери) та викиди (аномальні або ізольовані елементи), які є критично важливими для глибокого розуміння системи.

Ця дипломна робота присвячена розробці та реалізації інноваційного методу візуалізації кластеризації даних, який використовує принципи фізичних моделей, зокрема взаємодії електричних зарядів та пружин. Метою є створення візуального представлення, що не лише групує схожі елементи, але й динамічно демонструє процес формування кластерів, підвищуючи

інтуїтивне розуміння прихованих структур у складних мережевих даних. Розроблений підхід дозволяє вирішити проблему статичності традиційних візуалізацій, надаючи користувачеві анімований процес, що відображає динаміку переміщення вершин до стану рівноваги, де кластерні структури стають чітко вираженими.

Актуальність роботи

Актуальність даної дипломної роботи визначається кількома ключовими факторами, що відображають сучасні тенденції в галузі аналізу даних та візуалізації. Багато сучасних алгоритмів кластеризації, особливо для складних або багатовимірних даних, можуть діяти як "чорні скриньки", надаючи лише кінцевий результат без можливості зрозуміти логіку та динаміку процесу групування. Це обмежує довіру до результатів та ускладнює їх верифікацію. Візуалізація процесу, а не лише результату, підвищує прозорість та інтерпретованість. У багатьох реальних даних (наприклад, соціальні мережі, біологічні взаємодії, транспортні системи) елементи пов'язані між собою складними відносинами, формуючи мережі. Традиційні методи кластеризації часто примушують кожну точку належати до одного кластера, ігноруючи при цьому хаби (зв'язуючі вузли) та викиди (аномалії), які є критично важливими для розуміння динаміки та вразливостей системи. Існує гостра потреба в інструментах, які можуть візуально ідентифікувати ці особливі типи вузлів.

Таким чином, розробка методів візуалізації, що оптимізують когнітивне сприйняття результатів кластеризації, є вкрай актуальною.

Метою даної дипломної роботи є розробка та реалізація ефективного методу візуалізації кластеризації даних, що базується на фізичних моделях, для покращення інтуїтивного розуміння та аналізу складних мережевих структур, включаючи виявлення кластерів, хабів та викидів.

Завдання дослідження

Для досягнення поставленої мети було визначено наступні завдання:

1. Проаналізувати теоретичні та практичні аспекти кластеризації даних, її релевантність та роль візуалізації у цьому процесі.

2. Дослідити та порівняти сучасні алгоритми кластеризації даних.

3. Розробити модель та алгоритм візуалізації графіків даних.

4. Спроектувати загальну архітектуру програмної системи візуалізації кластеризації даних та розробити її графічний інтерфейс користувача.

5. Реалізувати розроблений метод візуалізації у вигляді програмного додатка, включаючи ефективні структури даних.

6. Провести експериментальне тестування реалізованого методу на реальних наборах даних для оцінки його ефективності.

Об'єктом дослідження є процеси візуалізації даних, що представляють собою мережеві структури.

Предметом дослідження є методи та алгоритми просторового розташування вершин графа, що імітують фізичні взаємодії, з метою покращення візуального виявлення кластерних структур у даних.

Методи дослідження

В роботі використано комплекс методів дослідження:

- Теоретичний аналіз та систематизація
- Математичне моделювання
- Алгоритмізація
- Об'єктно-орієнтоване програмування
- Експериментальне моделювання

Наукова новизна роботи полягає у розробці та програмній реалізації методу візуалізації кластеризації даних, який забезпечує динамічну візуалізацію процесу кластеризації. На відміну від статичних представлень, запропонований підхід дозволяє спостерігати за ітераційним процесом переміщення вершин, що значно підвищує інтуїтивне розуміння формування кластерів.

Практичне застосування результатів дипломної роботи охоплює широкий спектр галузей:

- Аналіз соціальних мереж - виявлення спільнот, впливових осіб (хабів) та аномальної поведінки користувачів.

- Біоінформатика - візуалізація мереж білкових взаємодій, генетичних зв'язків для ідентифікації функціональних груп.

- Маркетинг та бізнес-аналітика - сегментація клієнтів, аналіз поведінки споживачів, виявлення взаємозв'язків у базах даних.

Бакалаврська робота містить 79 сторінок, 35 рисунків, 3 розділи список використаних джерел із 36 найменуваннями.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВІЗУАЛІЗАЦІЇ ТЕХНІКИ КЛАСТЕРИЗАЦІЇ ДАНИХ

1.1. Особливості розробки та реалізації методу візуалізації кластеризації даних

Дана дипломна робота зосереджена на графічній реалізації техніки кластеризації даних, що використовує принцип фізичних законів зарядів та пружин для просторового переміщення вершин графа. Основною метою є розробка методу візуалізації, що відображає результати кластеризації, одночасно слугуючи інструментом для аналізу та розуміння прихованих структур у даних.

Розроблений підхід демонструє свою ефективність у візуалізації кластерних структур, що підтверджується якістю отриманих графічних зображень. Ці візуалізації не лише свідчать про успішність застосованого методу, але й вказують на придатність певних наборів даних для рутини кластеризації.

Основним призначенням реалізованого алгоритму є сприяння розумінню патернів групування в наборах даних через їх візуальне представлення. Такий візуальний результат є цінним інструментом для швидкого аналізу та може бути використаний як допомога у презентації виявлених тенденцій у даних, забезпечуючи інтуїтивне сприйняття складної інформації.

1.1.1. Методологічна основа та реалізація

Описаний метод візуалізації ґрунтується на моделі сил графа, де кожна вершина представляє об'єкт даних, а зв'язки між ними (або їх відсутність) відображають взаємозв'язки або схожість. Застосування фізичних законів, таких як кулонівська сила відштовхування між зарядами (для незв'язаних або

менш схожих вершин) та гукова сила притягання між пружинами (для зв'язаних або схожих вершин), дозволяє ітеративно оптимізувати просторове розташування вершин. Це призводить до формування візуально відокремлених груп — кластерів, де об'єкти всередині кластера розташовані близько, а об'єкти з різних кластерів — на значній відстані.

Реалізація цього підходу вимагає ефективних алгоритмів оптимізації та рендерингу, щоб забезпечити адекватну швидкість обчислень та високу якість графічного виведення. Важливою складовою є також інтерактивність візуалізації, що дозволяє користувачеві масштабувати, обертати та фільтрувати дані, глибше занурюючись у структуру кластерів.

1.1.2. Аналітична цінність та сфери застосування

Візуальні результати цього методу кластеризації мають значну аналітичну цінність. Вони дозволяють не тільки ідентифікувати явні кластери, але й виявляти аномалії, викиди та перехідні зони між групами, які можуть бути неочевидними при використанні лише числових метрик. Це особливо корисно в задачах розвідувального аналізу даних (EDA), де первинна мета — отримати інсайт у структуру даних без попередніх гіпотез.

Потенційні сфери застосування цього методу є широкими і охоплюють:

- Біоінформатику: для кластеризації генів, білків або пацієнтів на основі їх характеристик.

- Соціальні науки: для виявлення спільнот у соціальних мережах або групування поведінки користувачів.

- Маркетинг: для сегментації клієнтів та виявлення цільових груп.

- Кібербезпеку: для ідентифікації аномальних мережевих активностей або кластеризації загроз.

- Фінанси: для виявлення схожих за поведінкою акцій або кластеризації ризиків.

Таким чином, розроблений метод візуалізації кластеризації даних не просто відображає результати, а надає потужний інструмент для глибокого розуміння даних, сприяючи прийняттю більш обґрунтованих рішень у різних галузях.

1.2. Теоретичні та практичні аспекти кластеризації даних та її візуалізації

1.2.1. Вступ до кластеризації даних та її релевантність

Кластеризація даних є фундаментальним напрямком у галузі дослідження даних (data mining), що спрямований на вилучення корисної інформації шляхом ідентифікації та агрегації об'єктів на основі їх спільних характеристик. Ефективність успішної класифікації компонентів у наборах даних підкреслюється численними застосуваннями, що спостерігаються як у природних, так і в соціальних системах.

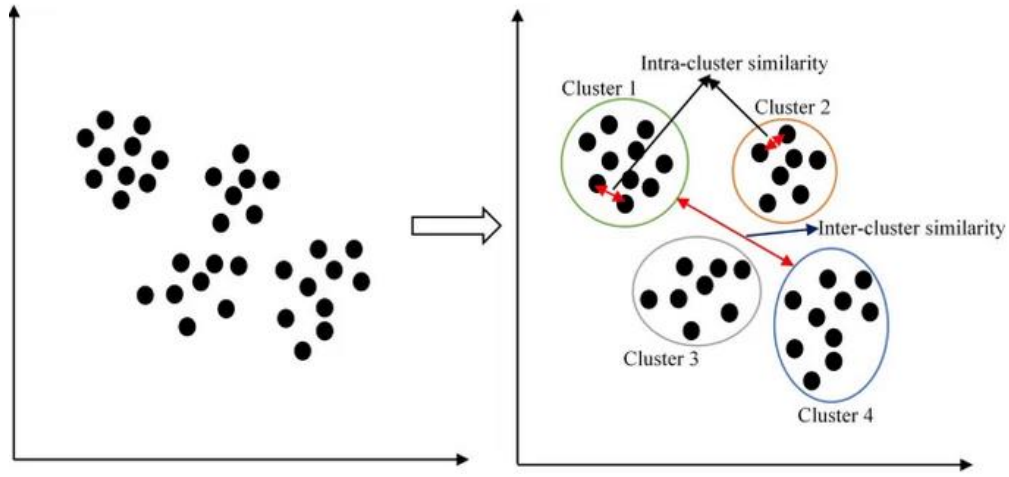

Рисунок 1.1 – Графічне представлення процесу кластеризації даних

Зокрема, в екологічних дослідженнях виникає нагальна потреба в організації кліматичних даних, даних про харчові ланцюги або міграційні процеси для ідентифікації життєвих структур, таких як біосфери або

екосистеми. Незважаючи на доступність обчислювальних інструментів для вирішення цих завдань, вибір оптимального алгоритму кластеризації, що відповідає специфічним характеристикам екологічних даних, залишається складним завданням.

Поза екологією, кластеризація є основним інструментом для організації даних у таких різноманітних галузях, як:

- Медицина: для аналізу та категоризації медичних звітів, наприклад, електрокардіограм (ЕКГ).

- Маркетингові дослідження: для сегментації споживачів та виявлення потенційних цільових груп на основі опитувань або тестових панелей.

- Інформаційний пошук: для покращення релевантності результатів пошуку шляхом інтелектуального групування веб-сторінок або документів.

З огляду на експоненційне зростання обсягів даних, потреба в ефективних техніках кластеризації стає дедалі більш критичною.

1.2.2. Роль візуалізації в кластеризації даних

Представлення результатів кластеризації даних у візуально ефективний спосіб значно підвищує їх інформативність для більшості застосувань, включаючи всі вищезгадані приклади. Основна мета візуалізації полягає не в простому відображенні сирих даних, а у підкресленні фундаментальних явищ та прихованих патернів.

З огляду на те, що приблизно п'ятдесят відсотків нейронів людського мозку задіяні в обробці зорової інформації, візуалізація не лише повинна забезпечувати точне представлення набору даних, але й сприяти його швидкій та інтуїтивній інтерпретації. Визнання можливостей та швидкості людського пізнання підкреслює необхідність того, щоб візуалізація мала бажаний ефект, відповідаючи своїй основній меті.

Важливо розрізняти графічне представлення даних та візуалізацію. Графічне представлення є, по суті, альтернативою табличному формату для

вираження результатів, де цінність полягає у структурованому відображенні вимірювань (наприклад, графік залежності температури від відстані від джерела тепла). Натомість, візуалізація прагне витягти значущість інформації, перетворюючи сирі дані на зрозумілі образи. Можливою візуалізацією тих самих даних про температуру може бути теплова карта, де кольорове кодування відображає розподіл температури, а форма зображення відповідає тестовій області. Візуалізація активно використовує графічні примітиви, координатні площини, візуальні техніки та технології відображення для надання інсайтів у представлену інформацію.

1.2.3. Дослідження візуалізації кластеризації розділених підмножин даних

Дана дипломна робота присвячена дослідженню здатності покращити візуалізацію наборів даних, що демонструють розділені підмножини (кластери). Візуальний вивід представлено у вигляді двовимірного графа, де кожна точка відповідає елементу даних, а ребра з'єднують пов'язані між собою елементи.

Для демонстрації результатів візуалізації в рамках цього дослідження були використані кілька наборів даних, зокрема:

- Дані із соціальних мереж, що відображають взаємозв'язки між користувачами.

- Дані матчів американського футболу NCAA College 2007 року, що ілюструють зв'язки між командами.

Важливо підкреслити, що ці набори даних характеризуються різними типами відносин між їх елементами. Ключовим аспектом є те, що наявність відносин між двома елементами надається рутині кластеризації апріорі, оскільки завдання алгоритму не полягає у витягуванні цих відносин із сирого набору даних. Таким чином, для згаданих наборів даних, визначення відношення між двома точками (тобто наявність ребра між двома вузлами)

базувалося на суб'єктивних критеріях, специфічних для кожного набору даних. Це підкреслює гнучкість розробленої техніки візуалізації до різних застосувань, дозволяючи акцентувати різні атрибути даних залежно від визначення відносин.

1.3. Алгоритми кластеризації текстової інформації

Кластеризація текстової інформації – це процес групування текстових документів (або фрагментів тексту) на основі їх семантичної схожості. Цей процес є ключовим у багатьох завданнях обробки природної мови (NLP) та інтелектуального аналізу даних.

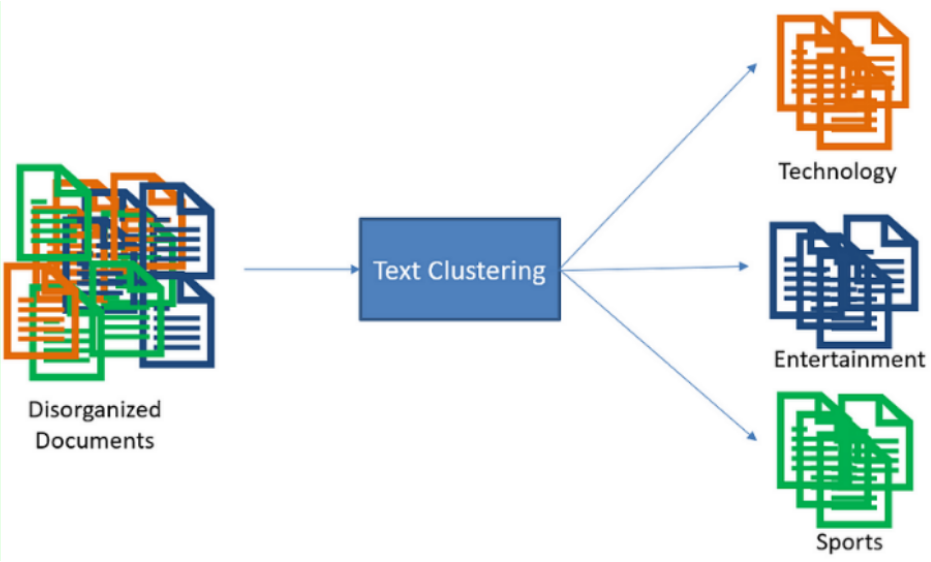

Рисунок 1.2 – Приклад виконання кластеризації текстової інформації

1.3.1. Загальні етапи кластеризації текстової інформації

Перед застосуванням будь-якого алгоритму кластеризації, текстові дані потребують попередньої обробки та перетворення в числовий формат. Типові етапи включають:

1. Збір та очищення даних: Видалення шумів, таких як HTML-теги, пунктуація, спеціальні символи, дублікати.

2. Токенізація: Розбиття тексту на окремі слова або фрази (токени).

3. Нормалізація:

- Приведення до нижнього регістру: Усі слова перетворюються на малі літери.

- Видалення стоп-слів: Видалення часто вживаних, але малоінформативних слів (наприклад, "і", "або", "є", "до").

- Стеммінг або лемматизація: Зведення слів до їхньої основи або нормальної форми (наприклад, "біг", "бігав", "біжить" до "біг" або "бігти").

- Побудова словника: Створення унікального списку всіх слів, що залишилися після нормалізації.

- Векторизація тексту: Перетворення тексту в числовий вектор. Найпоширеніші методи:

- Модель "мішка слів" (Bag-of-Words, BoW): Кожен документ представлений вектором, де кожен елемент вектора відповідає слову зі словника, а його значення – частоті цього слова в документі.

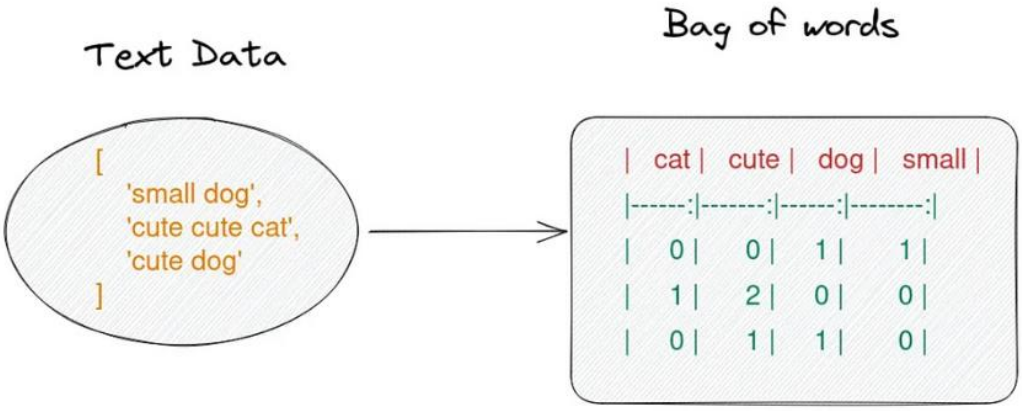

Рисунок 1.3 – Принцип роботи моделі Bag-of-Words

- TF-IDF (Term Frequency-Inverse Document Frequency): Враховує не лише частоту слова в документі (TF), а й його рідкість у всій колекції документів (IDF), що надає більшої ваги більш значущим словам.

Рисунок 1.4 – Принцип подубови TF-IDF матриці

Інверсна частота зустрічальності термінів (TFIDF) – це статистична формула для перетворення текстових документів у вектори на основі релевантності слова. Вона базується на моделі «мішка слів» для створення матриці, що містить інформацію про менш релевантні та найбільш релевантні слова в документі.

$$w_{x,y} = tf_{x,y} \times log(\frac{N}{df_x})$$

TF-IDF особливо корисна в завданнях NLP, тематичному моделюванні та завданнях машинного навчання. Вона допомагає алгоритмам використовувати важливість слів для прогнозування результатів.

- Векторні вбудовування слів (Word Embeddings): Сучасніші методи (наприклад, Word2Vec, GloVe, FastText) представляють слова у вигляді щільних векторів у багатовимірному просторі, де семантично схожі слова розташовані ближче.

- Вбудовування документів (Document Embeddings): Методи на кшталт Doc2Vec або Sentence-BERT дозволяють генерувати вектори для цілих документів або речень, зберігаючи їхній загальний зміст.

1.3.2. Популярні алгоритми кластеризації для текстових даних

Після векторизації даних можна застосовувати різні алгоритми кластеризації. Їх вибір залежить від структури даних, кількості кластерів та цілей аналізу.

1) K-means - ітеративно призначає точки до найближчих центроїдів (середніх значень кластерів) і оновлює положення центроїдів.

Переваги: Швидкий, масштабований для великих наборів даних.

Недоліки: Вимагає попереднього визначення кількості кластерів (K), чутливий до початкового розміщення центроїдів, не підходить для кластерів неправильної форми. Для текстових даних може використовувати косинусну подібність замість евклідової відстані для кращого врахування схожості за змістом.

2) Ієрархічна кластеризація (Hierarchical Clustering) - будує дерево (дендрограму) кластерів. Може бути агломеративною (об'єднує окремі точки в кластери) або дивізивною (розбиває великий кластер на менші).

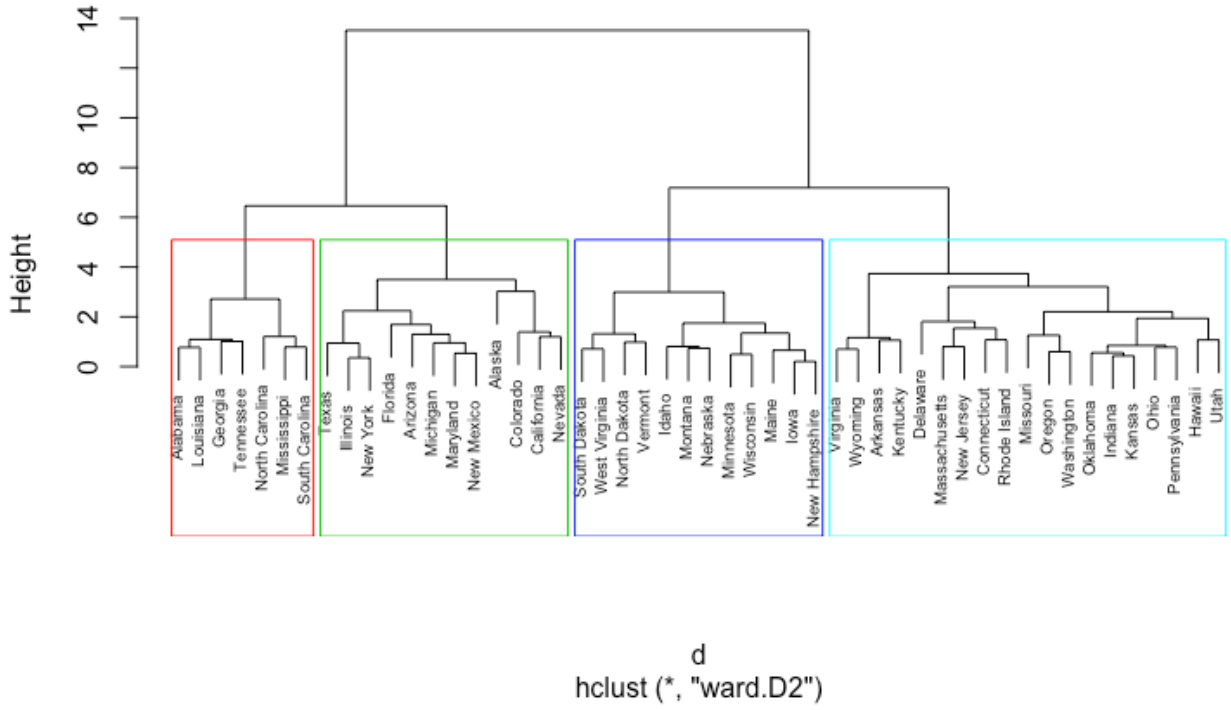

Рисунок 1.5 – Приклад дендограми побудованої під час ієрархічної кластеризації

Переваги: Не вимагає попереднього визначення кількості кластерів, дозволяє візуалізувати структуру кластерів.

Недоліки: Обчислювально дорожча для великих наборів даних, складніше інтерпретувати великі дендрограми.

3) DBSCAN (Density-Based Spatial Clustering of Applications with Noise) групує щільні області даних, позначаючи розріджені області як шум.

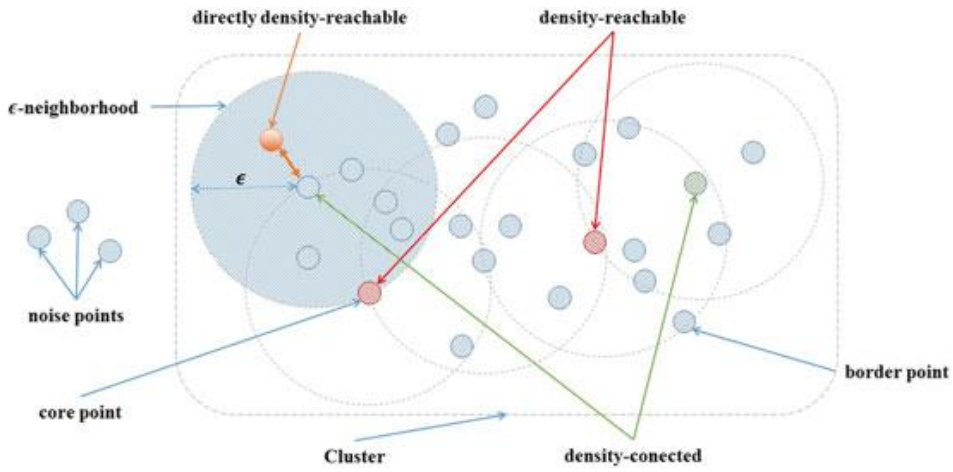

Рисунок 1.6 – Принцип роботи DBSCAN

Переваги: Не вимагає попереднього визначення кількості кластерів, може знаходити кластери довільної форми, ідентифікує викиди.

Недоліки: Складно підібрати параметри (є та min_samples), погано працює з даними різної щільності.

4) Gaussian Mixture Models (GMM) - припускає, що точки даних генеруються з суміші декількох гаусових розподілів. Визначає ймовірність належності кожної точки до кожного кластера.

Переваги: Може визначати м'які межі кластерів (одна точка може належати до декількох кластерів з різною ймовірністю), добре працює з даними, що мають складну структуру.

Недоліки: Вимагає попереднього визначення кількості компонентів (кластерів), чутливий до ініціалізації.

5) Latent Dirichlet Allocation (LDA) та Non-negative Matrix Factorization (NMF) - по суті, методи тематичного моделювання, які можуть використовуватися для кластеризації. Вони ідентифікують "теми" в колекції документів, де кожна тема є розподілом слів, а кожен документ – розподілом тем.

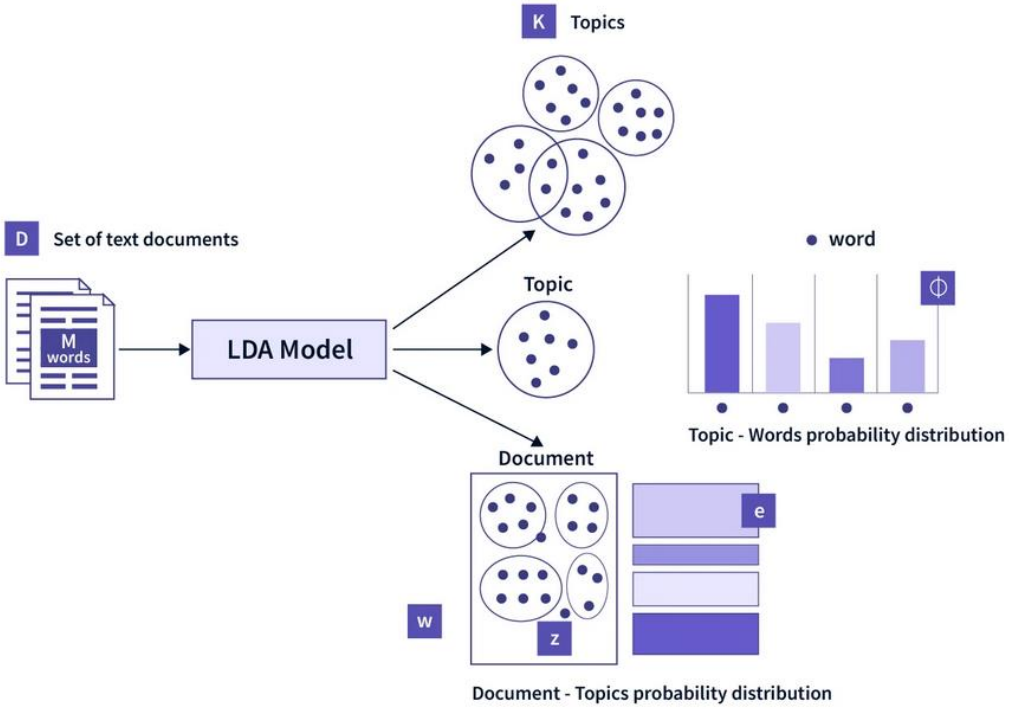

Рисунок 1.7 – Огляд роботи LDA моделі

Переваги: Виявляють приховані семантичні теми, дозволяють інтерпретувати кластери на основі слів, що їх визначають.

Недоліки: Результати можуть бути чутливі до кількості тем, що вибираються.

1.3.3. Оцінка ефективності кластеризації

Оцінка якості кластеризації є складним завданням, оскільки часто немає "правильних" міток кластерів (як у класифікації). Використовуються такі метрики:

1. Силуетний коефіцієнт (Silhouette Score).

Вимірює, наскільки об'єкт схожий на свій власний кластер порівняно з іншими кластерами. Значення від -1 до 1, де вищі значення вказують на краще визначені кластери.

2. Індекс Девіса-Болдіна (Davies-Bouldin Index).

Оцінює співвідношення між дисперсією всередині кластера та відстанню між кластерами. Нижчі значення кращі.

3. Індекс Калінського-Харабаса (Calinski-Harabasz Index).

Оцінює співвідношення між дисперсією між кластерами та дисперсією всередині кластерів. Вищі значення кращі.

4. Взаємна інформація (Mutual Information) та гомогенність/повнота (Homogeneity/Completeness).

Якщо є відомі мітки (навіть якщо вони не використовуються для кластеризації), ці метрики порівнюють отримані кластери з істинними мітками.

Вибір відповідного алгоритму та методів попередньої обробки для кластеризації текстової інформації залежить від специфіки даних, доступних обчислювальних ресурсів та кінцевих цілей аналізу. Сучасні підходи, що базуються на векторних вбудовуваннях слів та документів, відкривають нові можливості для більш точної та семантично змістовної кластеризації.

1.4. Розробка техніки візуалізації кластеризації даних на основі фізичних моделей

Представлена дипломна робота присвячена розробці та реалізації техніки візуалізації, спрямованої на поліпшення виявлення та інтерпретації групувальних тенденцій у наборах даних. Цінність візуалізації визначається її здатністю надавати глядачеві інтуїтивне розуміння неочевидних характеристик даних. Зокрема, у цьому дослідженні розглядається покращення візуального представлення наборів даних, що мають структуру

двовимірних графів, або мереж, які складаються з вершин (елементів), з'єднаних ребрами (відношеннями).

Сучасні алгоритми кластеризації зазвичай прагнуть групувати вершини мережі в дискретні підмережі (кластери), де кожна вершина належить виключно до одного кластера. Проте, набори даних, що походять з наукових вимірювань або соціального аналізу, часто містять аномальні елементи, які не підпадають під жодну чітку тенденцію. Розглянемо приклад мережі, що складається з вершин, які представляють учнів середньої школи, а ребра позначають дружні зв'язки. Застосування традиційної кластеризації може розбити цю мережу на кліки (повні підграфи). Однак, можуть існувати викиди – вершини з одним або нульовим з'єднанням, а також хаби – вершини, що з'єднані з двома або більше кліками. Примусове віднесення всіх вершин до одного підграфа призводить до того, що викиди вважаються такими, що мають асоціації, аналогічні іншим членам кліки, а численні зв'язки хабів з іншими підграфами ігноруються. У наведеному прикладі мережі середньої школи це може призвести до втрати важливої інформації, наприклад, щодо шляхів поширення чуток або вірусів усередині мережі. Цей приклад наочно демонструє обмеження існуючих рутин кластеризації щодо візуалізації всіх властивостей розділених наборів даних.

Запропонована робота визначає, що за умови наявності набору даних з визначеними відношеннями між його елементами, візуалізація даних може бути значно покращена за допомогою алгоритму, який позиціонує вузли даних на двовимірній координатній площині, імітуючи фізичну поведінку електричних точкових зарядів, з'єднаних пружинами.

Абстрагування набору даних у формі графа, що складається з вершин та ребер, є ефективним засобом для забезпечення універсальності техніки візуалізації кластеризації, що дозволяє її застосування до різноманітних типів даних. Переміщення вершин мережі має базуватися на її графічних елементах, а не на специфічних властивостях вихідного набору даних.

Цей метод використовує інформацію про вершини та ребра мережі, інтегруючи її в модифіковані фізичні рівняння, які традиційно застосовуються для моделювання поведінки електричних зарядів та пружин. В алгоритмі кожна вершина представлена як електричний заряд, а наявність ребра між двома вершинами моделюється як пружина, що з'єднує відповідні електричні заряди. Важливо зазначити, що ця техніка переміщення графічних точок не розрізняє два типи зарядів. Натомість, всі елементи набору даних представлені одним типом електричного заряду і, як наслідок, завжди відштовхуються. З точки зору графічного представлення даних, наявність двох різних типів точок є недоцільною, оскільки це розрізнення не може бути адекватно виражене в самому графі.

Іншим ключовим аспектом запропонованого підходу є те, що між двома зарядами може існувати лише одна пружина, що відповідає одному ребру між двома вершинами. У контексті набору даних це означає, що відношення між двома елементами є бінарним: воно або існує, або ні. Таким чином, мережа розглядається як система рухомих точок, з'єднаних пружинами, яка прагне до стану рівноваги, де всі сили збалансовані. Більш точне визначення рівноваги полягає у стані, коли всі сили протидіють силам еквівалентної величини. Оптимальна точка завершення для алгоритму досягається при досягненні цього стану рівноваги. Однак необхідно враховувати можливість незначних осциляцій вершин навколо точки рівноваги, що вимагає механізмів для їх компенсації.

Додаткові обмеження дизайну візуалізації були обумовлені попередньо визначеним програмним середовищем. Розробка здійснювалася в двовимірному графічному середовищі, де вершини та ребра відображаються за допомогою компонентів Swing в Java.

В другому розділі описується як функціональність методу малювання компонента Swing, так і складність поєднання малювання компонента Swing з іншими видами обробки. Через непередбачувану поведінку компонентів

Swing, рішення з використанням багато поточності суттєво вплинуло на реалізацію візуалізації.

Підхід, обраний цим дослідженням для вирішення проблеми візуалізації розділених наборів даних, може бути застосований для аналізу раніше неочевидних суспільних тенденцій. Здатність швидко ідентифікувати, наприклад, хаб, що відхиляє патерни даних, не може бути досягнута простим порівнянням степенів вершин або існуючими технологіями кластеризації.

Такий хаб може мати значне суспільне значення, наприклад, при аналізі поширення хвороби або виявленні зв'язків між групами, підозрюваними у протиправній діяльності. Гнучкість визначення ребер як будь-яких відносин без зміни функціональності алгоритму також є надзвичайно важливою. Ця техніка зменшує складність модифікації алгоритму для його адаптації під потреби конкретної наукової галузі.

1.5. Аналіз сучасних методів кластеризації даних

Існує широкий спектр алгоритмів кластеризації, кожен з яких має свої переваги, недоліки та оптимальні сценарії застосування. Нижче представлено опис найбільш відомих та широко використовуваних категорій алгоритмів кластеризації.

1.5.1. Кластеризація на основі поділу (Partitioning-Based Clustering)

Ці алгоритми поділяють набір даних на k непересічних кластерів, де k – це заздалегідь визначена кількість кластерів.

- K-means (K-середніх)

Алгоритм ітеративно розподіляє точки даних між k кластерами. На кожній ітерації він обчислює центроїди (середні значення) кластерів, а потім перепризначає кожну точку даних до найближчого центроїда.

Рисунок 1.8 – Принцип роботи алгоритму k-середніх

Переваги: Простота реалізації, висока швидкість обробки для великих наборів даних, добре підходить для сферичних кластерів.

Недоліки: Чутливий до вибору початкових центроїдів, вимагає заздалегідь задавати кількість кластерів (k), неефективний для кластерів неправильної форми або кластерів різної щільності, чутливий до викидів.

Застосування: Сегментація клієнтів, аналіз зображень, виявлення аномалій (як попередній крок).

- K-medoids (K-медоїдів)

Схожий на K-means, але замість центроїдів (середніх) використовуються медоїди – фактичні точки даних, які є найбільш центральними в кластері. Це робить його менш чутливим до викидів. PAM (Partitioning Around Medoids) є однією з найпоширеніших реалізацій.

- Переваги: Менш чутливий до викидів порівняно з K-means, добре працює з довільними метриками відстані.

- Недоліки: Повільніший для великих наборів даних, ніж K-means, все ще вимагає заздалегідь задавати k.

- Застосування: Обробка зображень, вивчення тексту.

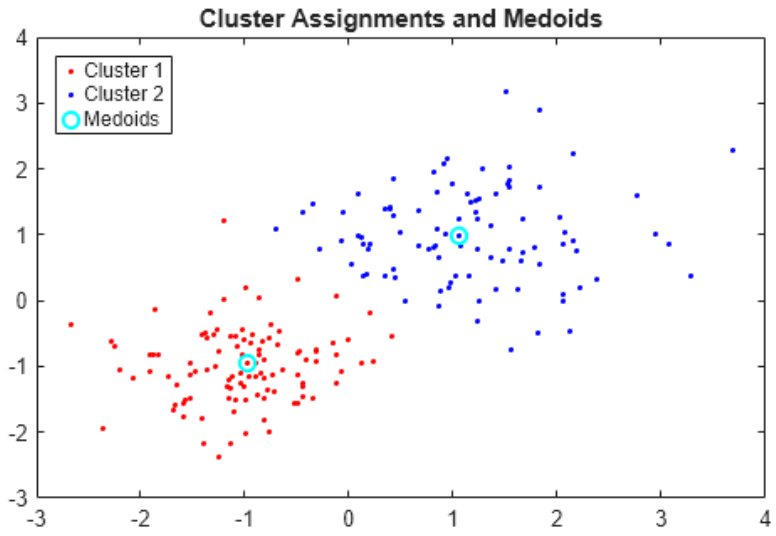

Рисунок 1.9 – Приклад K-medoids

1.5.2. Ієрархічна кластеризація (Hierarchical Clustering)

Ці алгоритми будують деревоподібну структуру (дендрограму), яка показує ієрархічні відносини між кластерами.

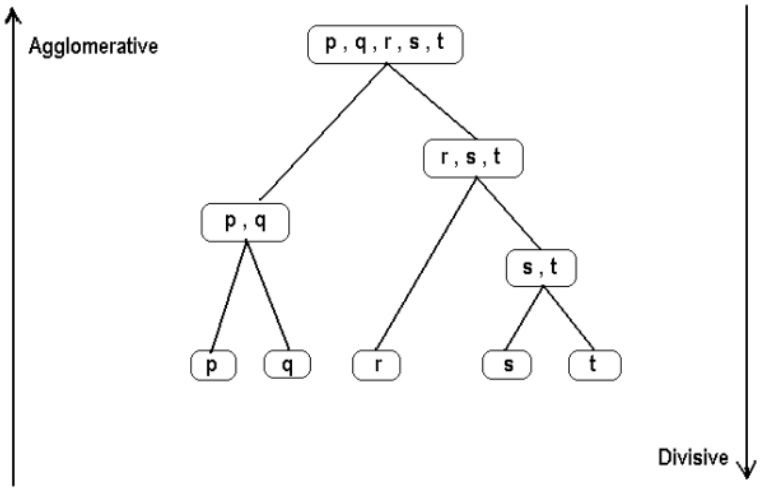

Рисунок 1.10 – Приклад здійснення ієрархічної кластеризації

- Агломеративна (Agglomerative): "Знизу вгору". Кожна точка даних спочатку розглядається як окремий кластер, а потім найближчі кластери послідовно об'єднуються, поки не буде досягнуто бажаної кількості кластерів або єдиного кластера.

- Дивізійна (Divisive): "Зверху вниз". Спочатку весь набір даних розглядається як один кластер, який потім рекурсивно поділяється на менші кластери.

Переваги: Не вимагає заздалегідь задавати кількість кластерів, візуально інтуїтивний (завдяки дендрограмам), дозволяє досліджувати кластери на різних рівнях деталізації.

Недоліки: Висока обчислювальна складність для великих наборів даних, чутливий до шуму та викидів, важко визначити оптимальну кількість кластерів без візуального аналізу дендрограми.

Застосування: Біологія (філогенетичні дерева), таксономія, медична діагностика.

1.5.3. Кластеризація на основі щільності (Density-Based Clustering)

Ці алгоритми ідентифікують кластери як області з високою щільністю точок даних, відокремлені від областей з низькою щільністю.

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - визначає кластери на основі щільності точок. Точки класифікуються як:

- Ядрові точки (Core points): Мають щонайменше `MinPts` (мінімальна кількість точок) сусідів у радіусі `eps` (епсилон).

- Граничні точки (Border points): Знаходяться в радіусі `eps` від ядрової точки, але самі не є ядровими.

- Шумові точки (Noise points): Не є ні ядровими, ні граничними точками.

Переваги: Здатний виявляти кластери довільної форми, не вимагає заздалегідь задавати кількість кластерів, може виявляти шумові точки.

Недоліки: Чутливий до параметрів `eps` та `MinPts`, погано працює з кластерами різної щільності, важко справлятися з даними високої розмірності.

Застосування: Виявлення аномалій, просторова кластеризація, аналіз географічних даних.

OPTICS (Ordering Points to Identify the Clustering Structure)

- Принцип роботи: Розширює DBSCAN, усуваючи необхідність фіксованих параметрів `eps` та `MinPts`. Він генерує "порядковий" вигляд кластерної структури, що дозволяє виявляти кластери з різною щільністю.

- Переваги: Може виявляти кластери різної щільності, не вимагає глобальних параметрів щільності.

- Недоліки: Висока обчислювальна складність, результати важко інтерпретувати без спеціалізованих візуалізацій.

- Застосування: Аналіз послідовностей, виявлення структурованих аномалій.

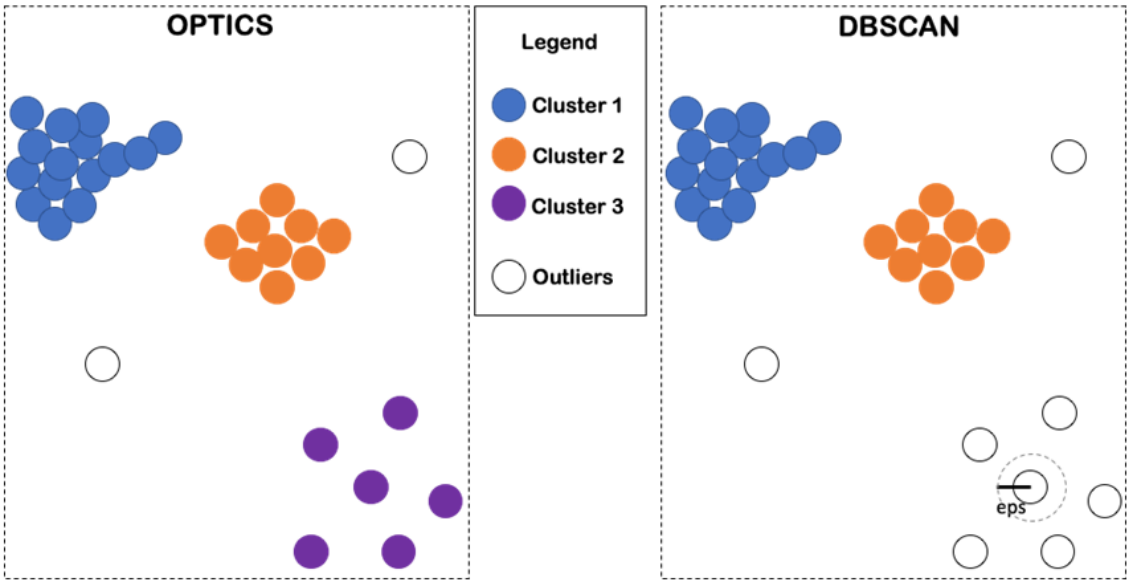

Рисунок 1.11 – Різниця між алгоритмами кластеризації DBSCAN та OPTICS

1.5.4. Кластеризація на основі моделі (Model-Based Clustering)

Ці алгоритми припускають, що дані генеруються з суміші ймовірнісних розподілів (наприклад, гаусових розподілів).

- EM (Expectation-Maximization) для гаусових сумішей

Принцип роботи: Використовує ітераційний підхід для знаходження параметрів суміші гаусових розподілів, які найкраще відповідають даним. Кожна точка даних має ймовірність приналежності до кожного кластера.

Переваги: Може виявляти кластери довільної форми, надає ймовірність приналежності до кластера, менш чутливий до викидів, ніж K-means.

Недоліки: Чутливий до початкових параметрів, схильний до локальних оптимумів, вимагає заздалегідь задавати кількість кластерів, припускає певний розподіл даних.

Застосування: Сегментація зображень, біоінформатика, аналіз фінансових даних.

1.5.5. Кластеризація на основі сітки (Grid-Based Clustering)

Ці алгоритми розділяють простір даних на сітку (набір комірок), а потім виконують кластеризацію на основі цих комірок.

- STING (Statistical Information Grid)

Розділяє простір на ієрархічні прямокутні комірки. Статистична інформація про кожну комірку зберігається, а кластеризація виконується шляхом аналізу цих статистик на різних рівнях ієрархії.

- Переваги: Висока швидкість обробки (особливо для великих наборів даних), дозволяє інкрементальне оновлення.

- Недоліки: Чутливий до розміру комірок, не дуже ефективний для кластерів неправильної форми.

- Застосування: Бази даних, що базуються на місцезнаходженні, просторовий аналіз.

1.5.6. Кластеризація на основі зв'язності/графів (Connectivity/Graph-Based Clustering)

Ці алгоритми розглядають точки даних як вузли в графі, а зв'язки між ними як ребра. Кластери визначаються як сильно зв'язані компоненти.

- Спектральна кластеризація (Spectral Clustering)

Використовує власні значення (eigenvalues) та власні вектори (eigenvectors) матриці спорідненості (similarity matrix) даних для зменшення розмірності та перетворення даних у простір, де їх легше кластеризувати за допомогою традиційних алгоритмів (наприклад, K-means).

Переваги: Здатний виявляти кластери довільної форми, добре працює з розрідженими даними.

Недоліки: Висока обчислювальна складність для великих наборів даних, вимагає заздалегідь задавати кількість кластерів, чутливий до вибору матриці спорідненості.

Застосування: Обробка зображень, біоінформатика, аналіз соціальних мереж.

1.5.7. Алгоритми на основі сил (Force-Directed Algorithms)

Моделює точки даних як частинки, що взаємодіють через сили (наприклад, відштовхування Кулона та притягання Гука). Система ітеративно наближається до стану рівноваги, де кластеризовані точки збираються разом. Зазвичай використовуються для візуалізації графів, де кластери стають візуально очевидними.

Переваги: Добре підходить для візуалізації кластерів, може виявляти хаби та викиди, гнучкий до визначення відносин.

Недоліки: Висока обчислювальна вартість для дуже великих графів, складність досягнення істинної рівноваги, візуальний результат залежить від параметрів сил.

Застосування: Візуалізація соціальних мереж, біоінформатика, аналіз складних систем.

Представлений перелік в даному підрозділі охоплює основні категорії та найбільш відомі алгоритми кластеризації, кожен з яких пропонує унікальний підхід до групування даних на основі різних припущень про

структуру даних. Вибір конкретного алгоритму залежить від характеристик даних, цілей аналізу та наявних обчислювальних ресурсів.

РОЗДІЛ 2. ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ ФУНКЦІОНУВАННЯ СИСТЕМИ ВІЗУАЛІЗАЦІЇ КЛАСТЕРИЗАЦІЇ ДАНИХ

2.1. Алгоритм візуалізації графіків даних на основі фізичних моделей

Представлений алгоритм спрямований на покращення візуального представлення графіків даних шляхом імітації фізичних властивостей електричних зарядів та пружин. Фундаментальне розуміння принципів закону Кулона та закону Гука є критично важливим для осмислення успіху цього методу візуалізації.

Реалізація візуальної складової алгоритму в середовищі Java вимагала адаптації об'єкта JLabel для специфічних завдань відображення вершин і ребер. Кастомізація цього об'єкта досягалася шляхом додавання необхідних структур даних та методів, що відповідають потребам програми. Розширення, внесені до стандартного об'єкта JLabel у даній реалізації, не є центральними для теми цієї роботи. Проте, функціональність методу repaint() об'єкта JLabel суттєво впливає на кінцевий результат візуалізації, і її деталі пояснюються в цьому розділі.

2.1.1. Закон Кулона

Атом складається з електрично заряджених частинок: протонів (що за умовністю вважаються позитивно зарядженими) та електронів (що вважаються негативно зарядженими), а також нейтронів, які не мають заряду. Хоча поняття позитивного та негативного заряду є довільним і не використовується безпосередньо в цій роботі, фундаментальна властивість електрично заряджених субатомних частинок полягає в тому, що однойменні заряди відштовхуються, а різнойменні – притягуються.

Між будь-якими двома зарядами виникає сила, що або відштовхує, або притягує, і якщо заряди мають свободу руху, вони будуть прискорюватися

один від одного або один до одного відповідно. Величина цієї сили визначає величину руху. Закон Кулона кількісно описує цю взаємодію, стверджуючи, що сила між двома електричними точковими зарядами прямо пропорційна добутку величин цих зарядів і обернено пропорційна квадрату відстані між ними. Математично це виражається рівнянням:

$$F_{A-B} = k_c \times \frac{(q_A \times q_B)}{r^2}$$

де:

F_{A-B} — сила між зарядами A і B.

k_c — стала Кулона, що дорівнює $9{\times}109$ Н · м^2/Кл^2.

q_A,q_B — величини електричного заряду A і B відповідно, в одиницях Кулона.

r — відстань між точковим зарядом A і точковим зарядом B.

Для застосування в двовимірній декартовій системі координат силу необхідно розкласти на вертикальну та горизонтальну складові. Евклідова відстань між двома точками (x_A,y_A) та (x_B,y_B) визначається за формулою, виведеною з теореми Піфагора:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

де:

d — відстань між двома точками на площині xy.

x_A,x_B — x-координати точок A і B відповідно.

y_A,y_B — y-координати точок A і B відповідно.

Нарешті, для виділення горизонтальних (Fx) та вертикальних (Fy) складових сили між двома електричними точковими зарядами використовуються тригонометричні співвідношення:

$$F_x = F_{A-B} \times \cos\Theta \quad F_y = F_{A-B} \times \sin\Theta$$

де

Θ — кут, утворений вектором сили F та його горизонтальною або вертикальною складовою.

Комбінуючи закон Кулона, застосований до декартової системи координат, та теорему Піфагора, можна отримати горизонтальні та вертикальні складові сили між двома електричними точковими зарядами.

Важливо відзначити, що ці рівняння надають лише величину сили без урахування її напрямку. У контексті цієї роботи завдання визначення напрямку спрощується завдяки припущенню, що всі елементи даних, абстраговані як електричні заряди, мають однаковий заряд. Оскільки однойменні заряди відштовхуються, то сили, що виникають між точками, завжди спрямовані одна від одної.

2.1.2. Закон Гука

Пружина, як пружний об'єкт, прагне повернутися до своєї початкової форми після деформації. Ця властивість зумовлена відновлювальною силою, яка повертає пружину до її стану спокою при розтягуванні. Закон Гука кількісно описує залежність між відновлювальною силою та величиною деформації (розтягування) пружини:

$$F = - k_s \times d$$

де:

F — відновлювальна сила.

k_s — стала пружини (коефіцієнт жорсткості).

d — зміщення внаслідок розтягування.

Зміщення від точки A до точки B, виражене через горизонтальні та вертикальні компоненти в декартовій системі координат, може бути виведено з тригонометричних властивостей:

$$\Delta x = x_A - x_B$$
$$\Delta y = y_A - y_B$$

Тоді рівняння для горизонтальних (F_x) та вертикальних (F_y) компонентів відновлювальної сили є:

$$F_x = -k_s \times (x_A - x_B)$$
$$F_y = -k_s \times (y_A - y_B)$$

2.1.3. Потоки виконання та графічний інтерфейс користувача

Клас JLabel в Java успадковує методи paint() та repaint() від інтерфейсу JComponent, який є частиною фреймворку Swing, призначеного для розробки графічних інтерфейсів користувача (GUI). JLabel є елементом GUI, що відображає текст, зображення або їх комбінацію. Типове виконання програми з GUI починається з ініціалізації, за якою слідує обробка даних, оновлення GUI, і ці останні два кроки повторюються до завершення програми. У програмі Java, що використовує JLabel як область відображення, метод repaint() викликається для оновлення вмісту. Після виклику repaint(), метод paint() JLabel виконується після завершення всіх очікуючих подій.

Розглянемо приклад використання JLabel для відображення кількості знайдених символів 'Z' у текстовому документі:

```
while ( !EOF ) {
    char = next character;
    if ( char == 'Z' ) {
        tally++;
        repaint();
    }
}
```

У цьому псевдокоді змінна tally зберігає загальну кількість 'Z', знайдених у документі, і використовується методом paint() для визначення кількості елементів для відображення. Якщо припустити, що документ

достатньо великий, щоб сканування кожного символу займало кілька секунд, очікуване виконання програми передбачало б поступове додавання елементів до порожньої області відображення. Проте, фактичне виконання демонструє порожній екран протягом майже всього часу роботи програми, відображаючи всі елементи лише безпосередньо перед завершенням, і цей останній екран може бути видимим настільки короткий проміжок часу, що спостерігач його не помітить.

Така несподівана поведінка пояснюється характеристиками методів виконання компонентів Swing. Як зазначено в документації до методу repaint(), "Компонент буде перемальований після того, як будуть відправлені всі наразі очікувані події". При виконанні псевдокоду середовище виконання Java оптимізує операції перемальовування, групуючи всі виклики repaint(). Це прискорює файловий ввід/вивід та іншу обробку, але бажана функціональність візуалізації не досягається.

Java пропонує конвенцію для використання компонентів Swing, відому як правило однопотоковості Swing, яка стверджує: "Щоб уникнути можливості взаємоблокування, ви повинні бути дуже обережними, щоб компоненти та моделі Swing створювалися, змінювалися та запитувалися лише з потоку обробки подій". Оскільки компоненти Swing не розроблені для безпечної роботи з потоками, це попередження є важливим перед спробою використання потоків, відмінних від потоку обробки подій, у програмі з компонентами Swing. Однак існують винятки з цього правила, і метод repaint() включений до списку методів, виключених з правила однопотоковості Swing.

Java включає клас Thread, який може бути інстанційований у програмі для виконання асинхронних завдань. Екземпляр класу Thread є об'єктом Thread, а завдання, що виконуватиме об'єкт Thread, деталізуються в його методі run():

```
public void run() {
    System.out.println("Hello from a thread!");
}
```

Псевдокод для наведеного вище прикладу підрахунку 'Z' може бути модифікований для використання нового потоку для виконання repaint(), тоді як потік обробки подій обробляє файловий ввід/вивід:

```
while ( !EOF ) {
    char = next character;
    if ( char == 'Z' ) {
        tally++;
        // Створити новий потік для перемальовування
        Thread repainter = new Thread() {
            public void run() {
                repaint();
            }
        };
        // Запустити потік repainter
        repainter.start();
    }
}
```

Цей новий псевдокод використовує вбудовані механізми Java для багатопотокового виконання компонента Swing, і ця концепція стає центральною для реалізації алгоритму візуалізації.

2.2. Алгоритми кластеризації на основі сил та візуалізація даних

2.1. Алгоритми на основі сил для кластеризації графів

Використання фізичних рівнянь для просторового переміщення вершин графа з метою кластеризації відоме як алгоритми на основі сил (force-directed algorithms) або пружинні алгоритми (spring-electrical algorithms). Ці алгоритми присвоюють сили елементам графа, як правило, за допомогою закону Кулона (для відштовхування) та закону Гука (для притягання).

В дослідження [12] автор детально розглядає різні силові моделі, розрізняючи при цьому мету моделі та мету алгоритму. Більшість енергетичних моделей були розроблені передусім для цілей візуалізації, а не

для безпосереднього розділення графа. Їхньою основною метою було створення макета з короткими та рівномірними довжинами ребер, де вузли розподілені рівномірно. Автор вводить два ключові визначення, що формують метод макета графа на основі енергії: енергетична модель відповідає за опис бажаного макета, тоді як алгоритм мінімізації енергії визначає фактичні обчислення для досягнення цього макета.

Автор представляє дві енергетичні моделі:

1. "Node-repulsion LinLog": Модель, де вузли відчувають сили відштовхування один від одного, аналогічно електричним зарядам у законі Кулона.

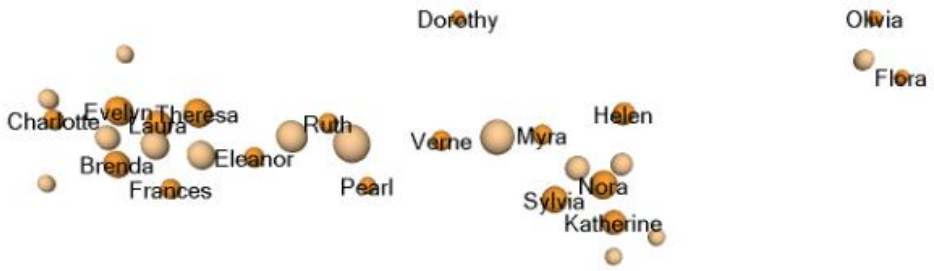

Рисунок 2.1 - Node-repulsion LinLog модель

"Edge-repulsion LinLog": Менш поширена модель, де ребра відштовхуються та притягуються одне до одного.

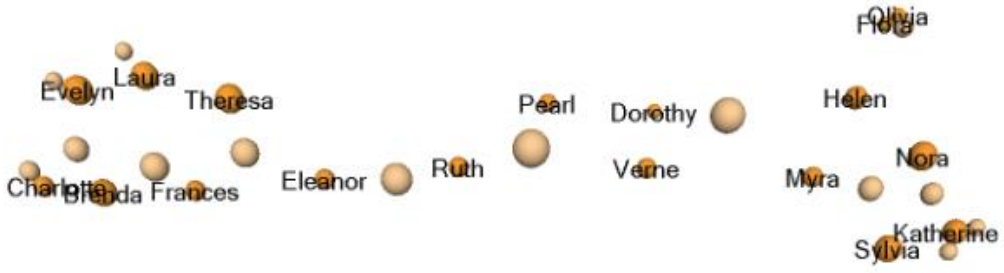

Рисунок 2.2 - Edge-repulsion LinLog модель

Обидві моделі прагнуть до внутрішньої валідності кластеризації, де щільніше з'єднані вузли групуються, а ті, що мають менше з'єднань,

розділяються. Це відрізняється від зовнішньої валідності, де групування залежить від зовнішнього, апріорі визначеного кластера. Суттєвою відмінністю обох моделей ("node-repulsion LinLog" та "edge-repulsion LinLog") від підходу, прийнятого в даній роботі, є використання ваг ребер як важливої частини моделі. У цьому дослідженні розглядаються графи з незваженими ребрами однакової ваги.

Проте, результати, отримані для двох моделей, досліджені в [12], є релевантними для цілей цієї роботи. Перша енергетична модель, "node-repulsion LinLog", створює графи, де кластери чітко відокремлені один від одного значною відстанню, а довжини ребер усередині кластера значно менші. Це свідчить про досягнення внутрішньої валідності. Друга модель, "edge-repulsion LinLog", не створює макети з настільки вираженими відмінностями між кластерами. Модель "edge-repulsion LinLog" надає симетричний макет із загалом короткими довжинами ребер і, незважаючи на інноваційну ідею зворотного застосування силових елементів у графі, не має такого сильного зв'язку з цією роботою через відмінності у візуальних характеристиках кластерів.

2.2.2. Структурна кластеризація та роль сусідства

В дослідженні [14] пропонують інноваційний алгоритм кластеризації, що відрізняється своїм визначенням проблеми. На відміну від типових технік кластеризації, які прагнуть визначити приналежність кожної вершини в графі, SCAN розрізняє різні ролі, які вершини відіграють у мережі, дозволяючи ідентифікувати кластери, хаби та викиди. Це співпадає з цілями даної дипломної роботи та є свідченням недавнього розвитку у галузі ідентифікації не тільки підграфів мережі, але й окремих типів вузлів.

Унікальність SCAN полягає не лише в кінцевому результаті, а й у підході. Тоді як більшість процедур кластеризації фокусуються на виявленні великої кількості ребер всередині кластерів та меншої кількості ребер між

ними, SCAN зосереджується на сусідстві кожної вершини як критерії для групування. SCAN групує вершини за тим, як вони розділяють спільних сусідів, створюючи розділи, що називаються структурно зв'язаними кластерами. Кожна вершина відвідується один раз для визначення структурно зв'язаних кластерів, після чого алгоритм обробляє вершини, ізольовані від будь-якого кластера, ідентифікуючи їх як хаби або викиди.

Для оцінки обчислювальної ефективності запропонованого алгоритму було згенеровано десять графів із кількістю вершин у діапазоні від 1 000 до 1 000 000 та кількістю ребер від 2 182 до 2 000 190. Ми адаптували метод побудови графів, наступним чином: спочатку генерувалися кластери таким чином, що кожна вершина з'єднувалася з вершинами всередині того ж кластера з ймовірністю Pi, і з'єднувалася з вершинами за межами свого кластера з ймовірністю Po<Pi. Потім додавалася певна кількість хабів та викидів. Приклад згенерованого графа представлено на рисунку 2.3.

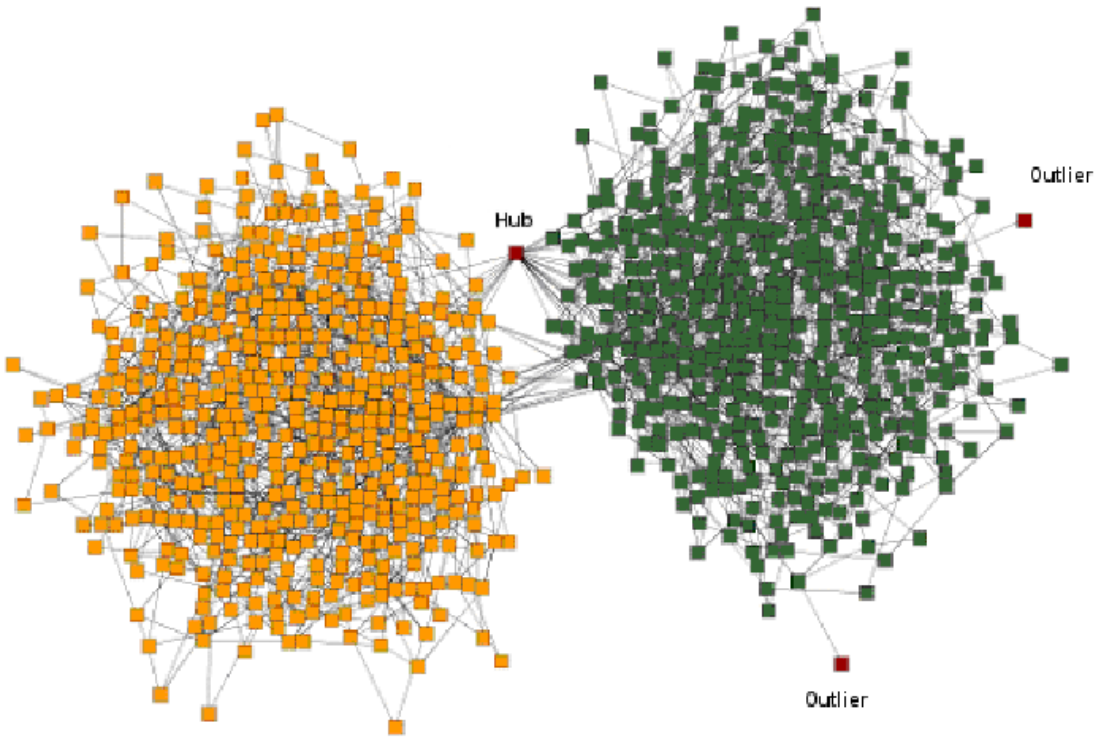

Рисунок 2.3 - Синтетичний граф з 1 000 вершинами

Цей акцент на значущості сусідства корелюється з дослідженням [18]. Ця робота вирішує проблему кластеризації графа без попереднього знання кількості підграфів – поширений підхід до розділення графів. Вона пропонує процедуру, яка спочатку аналізує сусідство кожної вершини, а потім обирає одну вершину з цього сусідства як оптимального представника кластера. Таким чином, для кожного шаблону в наборі даних генерується "нейбограма" (neighborgram), а потім підхід найближчої відстані до представника кластера завершує групування вершин, які чітко не належать до одного кластера.

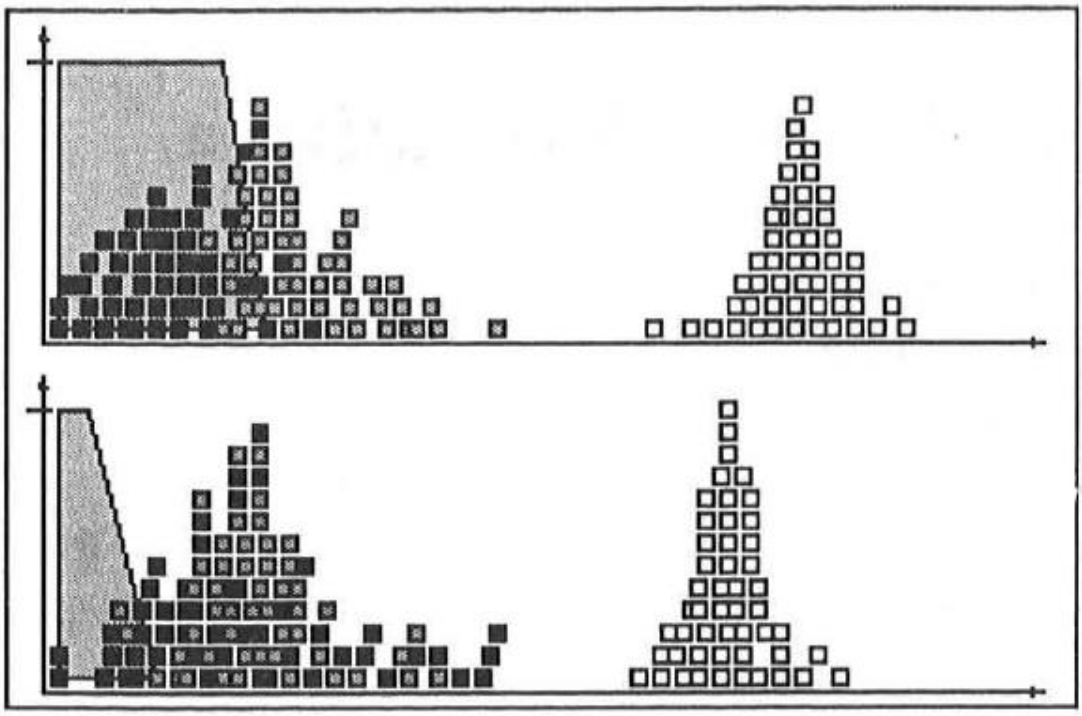

Рисунок 2.4 - Дві нейбограми, побудовані для даних Iris

Це дослідження релевантне для даної роботи, оскільки вона визначає, що кластеризація графа фактично полягає в розділенні набору даних за тенденцією або шаблоном, який не є явно деталізованим у самому графі. Це відповідає меті візуалізації – виразити приховане явище даних, а одновимірному графічному представленні що є початком дослідження візуалізації інформації.

2.2.3. Принципи візуалізації інформації

В роботі [20] автор досліджує важливість візуалізації даних, порівнюючи техніки абстрактного живопису з візуалізацією інформації. Він вводить концепцію естетичних обчислень – застосування художніх цілей до обчислень. Автор вважає, що групування даних є "одним з найважливіших процесів у візуальному аналізі даних та візуалізації інформації". З цією метою він окреслює п'ять факторів, що визначають групування у візуальному контексті:

1. Близькість (Proximity): Елементи сприймаються як згруповані, якщо вони розташовані близько один до одного.

2. Подібність (Similarity): Елементи групуються разом, якщо вони подібні за певною мірою.

3. Замикання (Closure): Елементи групуються разом, якщо вони мають тенденцію доповнювати якусь сутність або форму.

4. Безперервність (Continuity): Сприйняття того, що елементи, розташовані вздовж лінії або кривої, належать до однієї групи.

5. Простота (Simplicity): Тенденція групувати точки разом у прості геометричні фігури, такі як кола, трикутники або квадрати.

Як приклад ієрархічної візуалізації інформації, що базується на треемапах, на рисунку 2.5 зображено мапу вебсайту всього Техаського університету в Далласі на його п'яти верхніх рівнях (приблизно 8 115 сторінок). Користувачі можуть переміщатися цією мапою, зберігаючи контекст та фокусуючись на певних ділянках.

Ця візуалізація надає інформативний абстрактний огляд сайту та взаємозв'язків між його сторінками. Проміжні сторінки організовані у вигляді вкладених прямокутників. Таким чином, веб-адміністратор може легко ідентифікувати структурну поведінку сайту. Наприклад, ця візуалізація одразу показує, що сторінка "search" (пошук) у нижньому лівому куті містить найбільшу кількість підсторінок.

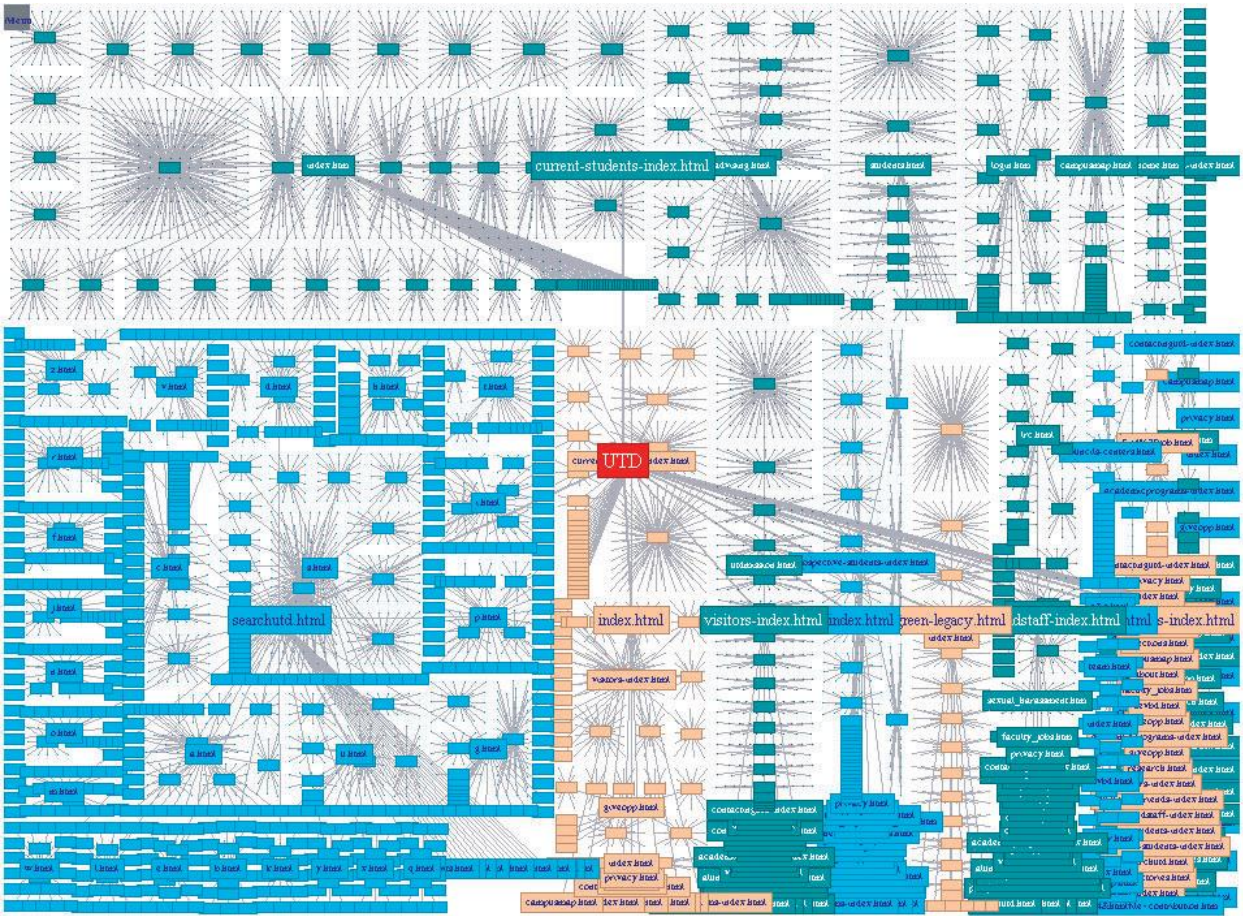

Рисунок 2.5 - Приклад візуалізації карти веб-сайту Техаського університету

Ці "Закони організації" є важливими в галузі абстрактного мистецтва і можуть бути екстрапольовані на візуальний аналіз даних, так що симетрія та мінімальне перетинання в графі сприяють створенню корисної та візуально привабливої презентації. Ці визначення допомогли в розумінні того, як кожна частина отриманого графа сприяє розумінню даних глядачем. Загалом, розглянуті дослідження сприяли формуванню цілей цієї роботи та її узгодженню з поточними розробками у галузі аналізу даних.

2.3. Алгоритм візуалізації процесу кластеризації даних

На рисунку 2.6 подано алгоритм візуалізації процесу кластеризації даних як ітераційний цикл, що веде до отримання нових знань. Процес

починається з "початкових даних", які представляють сирі, неструктуровані або слабоструктуровані дані.

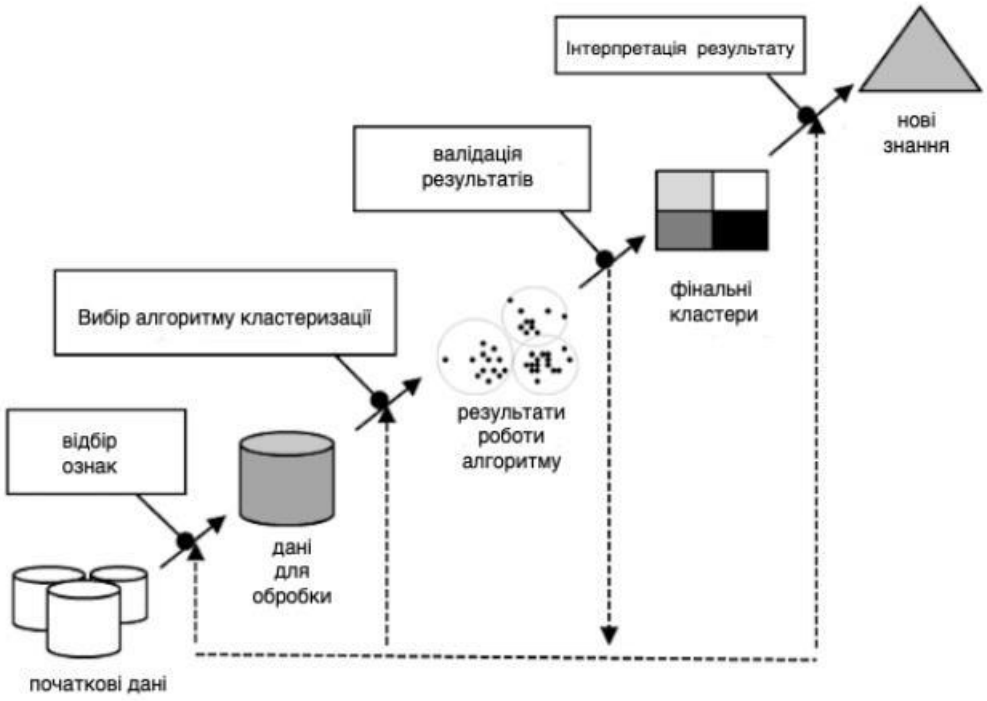

Рисунок 2.6 - Алгоритм візуалізації процесу кластеризації даних як ітераційний цикл

Наступний етап – "відбір ознак". Це критично важливий крок, де з початкових даних вибираються або конструюються релевантні атрибути (ознаки), які будуть використані для кластеризації. Це дозволяє зосередитися на найбільш інформативних аспектах даних, відфільтровуючи шум та надлишкову інформацію.

Після відбору ознак підготовлені "дані для обробки" передаються до наступного етапу – "Вибір алгоритму кластеризації". На цьому етапі відбувається вибір відповідного методу кластеризації, який найкраще підходить для типу даних та поставленої задачі. Вибір алгоритму суттєво впливає на кінцеві результати групування.

Застосування обраного алгоритму до підготовлених даних призводить до "результатів роботи алгоритму". На схемі це зображено у вигляді трьох

кругових областей з розсіяними точками, що символізують виявлені кластери з їхніми елементами.

Наступний етап – "валідація результатів". Цей крок є обов'язковим для оцінки якості та достовірності отриманих кластерів. Валідація може включати різні метрики та методи для перевірки когерентності кластерів, їх роздільності та відповідності предметній області.

Після валідації формуються "фінальні кластери". Це очищені та підтверджені групи даних, які відображають знайдені закономірності. На схемі це представлено чотирма квадратами різних відтінків, що символізують окремі, чітко визначені кластери.

Останній, але не менш важливий етап – "Інтерпретація результату". На цьому етапі людський аналітик або експерт предметної області осмислює отримані фінальні кластери, надаючи їм значення та контекст. Метою інтерпретації є перетворення абстрактних кластерів на "нові знання", які можуть бути використані для прийняття рішень, побудови гіпотез або подальших досліджень.

Зі схеми також видно петлю зворотного зв'язку. Від "Інтерпретації результату" пунктирна лінія веде назад до "відбору ознак" та "вибору алгоритму кластеризації". Це вказує на ітеративний характер процесу кластеризації: отримані знання можуть спонукати до перегляду вихідних ознак або вибору іншого алгоритму для подальшого покращення результатів та глибшого розуміння даних. Також існує зворотний зв'язок від "валідації результатів" до "вибору алгоритму кластеризації", що свідчить про можливість налаштування або зміни алгоритму, якщо початкові результати валідації є незадовільними.

Загалом, схема наочно демонструє, що візуалізація процесу кластеризації даних – це не просто лінійна послідовність кроків, а динамічний, циклічний процес, що поєднує автоматизовані обчислення з людською експертизою для генерації значущих інсайтів.

РОЗДІЛ 3. ПРОГРАМНА ІМПЛЕМЕНТАЦІЯ МЕТОДУ ВІЗУАЛІЗАЦІЇ КЛАСТЕРИЗАЦІЇ ДАНИХ

3.1. Загальний дизайн та архітектура візуалізації

Представлена візуалізація реалізована як інтегрована частина на базі Java, що містить додаткові інструменти для маніпулювання та аналізу відображуваних даних. Знімок екрана програми Java наведено на рисунку 3.1, а функціональні можливості кожного розділу графічного інтерфейсу користувача (GUI) детально описані в наступному підрозділі.

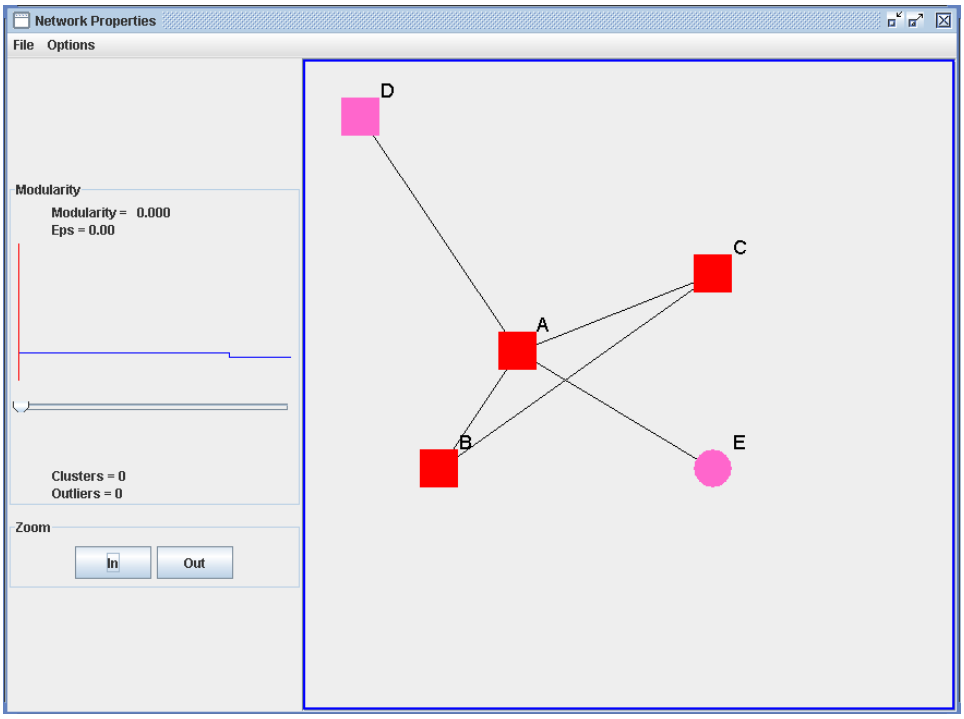

Рисунок 3.1 – Скріншот екрану системи

Єдиним дизайнерським рішенням, що випливало безпосередньо з обраного підходу, а не з обмежень середовища реалізації, була необхідність обмеження кількості ітерацій переміщення вершин. У запропонованому алгоритмі оптимізації кожен елемент даних абстрагується як електричний заряд, розташований у двовимірному просторі, а відношення між елементами

даних моделюються як пружини, що з'єднують ці електричні заряди. Поведінка електричних зарядів у просторі, детермінована фізичними силами, детально розглянута в другому розділі. З точки зору графічного представлення наборів даних у GUI програми, вершини відповідають електричним зарядам одного типу, а ребра – пружинам, які використовуються алгоритмом оптимізації. Згідно з цією абстракцією, вершини будуть відштовхуватися одна від одної, і лише наявність пружини між двома точковими зарядами утримуватиме їх поблизу.

Зрештою, система зарядів і пружин досягне точки рівноваги, де всі сили компенсують сили еквівалентної величини. Більш точне визначення рівноваги в даному контексті – це стан, де всі сили протидіють силам майже еквівалентної величини, що забезпечує стабільність системи. Оптимальна кінцева точка для алгоритму досягається при досягненні цього стану рівноваги. Проте існує ймовірність незначних осциляцій вершин навколо точки рівноваги, що вимагає впровадження механізмів для їх обліку та, при необхідності, зупинки ітерацій.

Додаткові обмеження дизайну для візуалізації були зумовлені архітектурою існуючої програми. Середовище відображення являє собою двовимірний макет вершин, з'єднаних ребрами, причому як вершини, так і ребра малюються в GUI за допомогою класів, що є частиною компонентів Java Swing. Ініціація алгоритму здійснюється вибором опції "Optimize" через випадаюче меню "Options", розташоване безпосередньо під рядком заголовка програми у верхній частині вікна, як показано на рисунку 3.2.

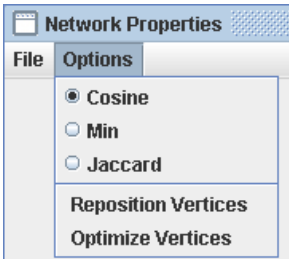

Рисунок 3.2 – Меню опцій

В другому розділі було детально описано як функціональність методу малювання компонента Swing, так і складності, пов'язані з поєднанням малювання компонента Swing з будь-яким іншим типом обробки. Внаслідок цієї непередбачуваної поведінки компонентів Swing, рішення щодо багатопотоковості, суттєво вплинуло на реалізацію візуалізації.

3.2. Компоненти графічного інтерфейсу користувача

Графічний інтерфейс користувача (GUI) складається з трьох основних компонентів: панелі меню, бічної панелі та великої області відображення. При запуску програми область відображення містить повідомлення "(No Network Loaded)", що вказує на відсутність завантаженої мережі. Ця область призначена для візуалізації графічного представлення завантаженої мережі.

Меню "File", доступне на панелі меню, включає опції "Open" та "Save", які дозволяють користувачеві завантажувати або зберігати файли відповідно. При виборі будь-якої з цих опцій відкривається стандартне вікно навігатора файлів, що дозволяє користувачеві обирати файли форматів GraphML або DAT, як показано на рисунку 3.3.

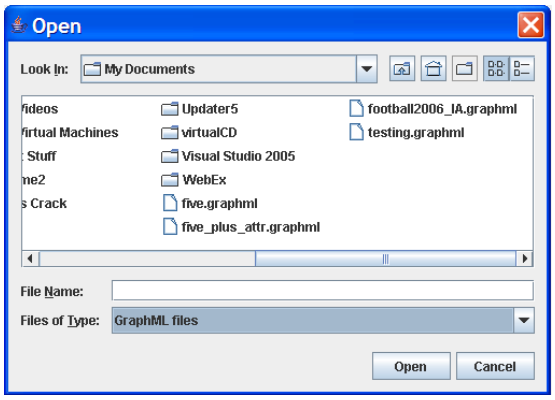

Рисунок 3.3 – Файловий навігатор системи

Друге меню на панелі меню називається "Options" і надає користувачеві вибір з трьох альтернативних методів: "Cosine", "Min" та

"Jaccard". За замовчуванням активовано "Cosine", а вибір між цими опціями здійснюється за допомогою перемикачів, що позначають активний метод, як показано на рисунку 3.2. Меню "Options" також включає дві додаткові опції: "Reposition Vertices" та "Optimize Vertices". Обидві опції мають спільну мету візуального групування будь-яких підгруп, присутніх у відображеній мережі, але реалізовані різними способами. "Reposition Vertices" характеризується довшим часом виконання і не забезпечує постійного оновлення графічного представлення мережі протягом усього процесу обробки. Натомість, "Optimize Vertices" виконує алгоритм візуалізації, детально описаний у цій роботі, забезпечуючи більш динамічне відображення.

Бічна панель GUI містить два основні компоненти: "Modularity" та "Zoom", розташований під першим. Елемент "Modularity" використовується для відображення невеликого графіка залежності модульності від подібності; приклад такого графіка показано на рисунку 3.4.

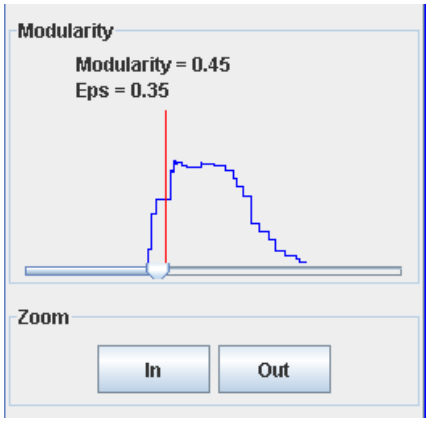

Рисунок 3.4 - Компонент модульності в системі

У контексті цієї програми, подібність є мірою сили зв'язку між двома вершинами, визначеної конкретним відношенням, присвоєним кожному ребру в мережі. Модульність – це кількісна оцінка модульності мережі, коли ігноруються всі ребра, подібність яких менше заданого значення. Таким чином, повзунок, розташований під невеликим графіком у компоненті "Modularity" GUI, дозволяє користувачеві встановлювати певне значення

подібності, і графік мережі перемальовується, щоб виключити ребра, подібність яких менша за задане значення повзунком. Компонент "Zoom" включає дві кнопки – "In" та "Out" – що дозволяють користувачеві змінювати масштаб відображення мережі.

3.3. Оптимізація розташування вершин

Вибір опції "Options → Optimize Vertices" в програмному додатку ініціює комплексний процес, що включає не лише безпосередні обчислення сил, які діють на кожну вершину, та їх подальше переміщення, але й низку необхідних попередніх та пост-обробкових кроків.

Перед виконанням обчислень сил обов'язковою є перевірка унікальності просторового розташування всіх вершин. У випадку, якщо дві вершини мають ідентичні координати, обчислення евклідової відстані між ними призведе до нульового значення. Оскільки сила, що діє на вершину з боку іншої, обернено пропорційна квадрату відстані між ними, нульова відстань спричинить ділення на нуль, що призводить до невизначеної поведінки програмного забезпечення. Для запобігання такій ситуації, перед початком ітерацій обчислення сил, необхідно забезпечити унікальне початкове розташування всіх вершин шляхом їхнього незначного переміщення, якщо виявляється колізія.

Після ініціального позиціонування вершин ключовим аспектом алгоритму є визначення критерію зупинки ітерацій. Як було зазначено в підрозділі 3.1, у теоретичному сценарії всі сили в системі з часом досягають рівноваги, внаслідок чого кожна з них стає незначною через компенсацію з боку інших, майже еквівалентних за величиною, протилежних сил у мережі. Оскільки сили безпосередньо зумовлюють рух вершин, моніторинг найбільшого переміщення, здійсненого будь-якою вершиною протягом останнього набору ітерацій, дозволяє встановити поріг. Таким чином, для

продовження ітеративного процесу обчислення та переміщення вершин, найбільший зафіксований рух повинен перевищувати встановлений поріг.

Нарешті, перед кожним новим перемальовуванням графа необхідно змістити всю графічну структуру таким чином, щоб максимізувати її видиму частину в межах області перегляду. Це досягається шляхом закріплення вершини з найменшими координатами x та y у верхньому лівому куті дисплея та подальшого коригування координат усіх інших вершин відносно цього якоря. Після цього граф може бути оновлений. Процес обчислення сил, переміщення вершин, їхнє коригування для оптимального відображення та перемальовування повторюється ітераційно, доки найбільший рух вершини не стане меншим за визначений поріг, що свідчить про досягнення стану квазі-рівноваги системи.

3.4. Представлення структури даних

Для представлення вершин (елементів) та ребер (відносин) мережі, що є основою алгоритму візуалізації, були використані структури даних, інтегровані в існуючий програмний комплекс Java. Ці структури, хоча і були попередньо визначені, виявилися оптимальними для цілей даного дослідження. Зокрема, для зберігання вершин та ребер використано дві структури TreeMap:

```
Private TreeMap<String, Vertex> vertexMap;
Private TreeMap<Pair, Edge> edgeMap;
```

Клас TreeMap у Java є реалізацією інтерфейсу SortedMap, де елементи ідентифікуються за унікальним ключем і зберігаються у відсортованому порядку за цими ключами. TreeMap забезпечує логарифмічну складність за часом, O(logn), для операцій додавання, отримання та видалення елементів.

Клас Vertex є спеціалізованим класом, розробленим для цієї реалізації. Він зберігає такі атрибути, як унікальний рядковий ідентифікатор вершини, набір ідентифікаторів вершин, що формують її безпосереднє сусідство, а також координати x та y вершини в двовимірному просторі.

Клас Edge також є оригінальним класом, що зберігає, зокрема, рядкові ідентифікатори двох вершин, з'єднаних даним ребром.

Клас Pair, що використовується як ключ для елементів у edgeMap, не є частиною стандартної бібліотеки колекцій Java. Він містить два рядки, які в цій реалізації слугують унікальними ідентифікаторами для конкретного ребра.

Оскільки клас Vertex зберігає поточне просторове розташування (координати x та y) кожного члена vertexMap, можлива ітерація по vertexMap для виконання обчислень, використовуючи поточні позиції вершин як операнди. Рух кожної вершини визначатиметься сумою всіх сил, що на неї діють. Таким чином, для обчислення загальної сили, що діє на вершину, необхідно обчислити та підсумувати сили взаємодії цієї вершини з кожною іншою вершиною в мережі.

Ефективність обчислень може бути підвищена за рахунок одноразової ітерації по всіх вершинах для обчислення всіх сил, замість послідовної ітерації по кожній вершині та обчислення сил, що діють на неї. Це стає можливим завдяки використанню принципу, що сила, яку вершина A відчуває від вершини B ($F_{A \leftarrow B}$), є рівною за величиною і протилежною за напрямком силі, яку вершина A чинить на вершину B ($F_{A \rightarrow B}$). Тобто, $F_{A \leftarrow B} = -F_{B \leftarrow A}$. Цей підхід дозволяє скоротити кількість необхідних обчислень вдвічі.

Для зберігання всіх обчислених сил використовується структура даних – двовимірний масив. Масив розміром N×N, де N – кількість вершин (number_VERTICES), може зберігати $F_{A \leftarrow B}$ у позиції array[A][B], при цьому array[B][A] міститиме $(-1) \times F_{A \leftarrow B}$. Таким чином, алгоритм використовує

окремий двовимірний масив для горизонтальних складових сил та інший двовимірний масив для вертикальних складових сил. Після обчислення всіх сил, горизонтальне зміщення вершини A може бути визначено шляхом підсумовування елементів відповідного рядка горизонтального масиву сил:

$$x_{A_{new}} = x_{A_{old}} + \sum_{i=0}^{\text{number\_VERTICES}-1} \text{HorizontalForceArray}[A][i]$$

де x_{Anew} та x_{Aold} - нове та старе значення x-координати вершини A. Аналогічно для y-координати.

З наявними структурами даних, останнім важливим аспектом реалізації цього алгоритму візуалізації є використання методів обробки для генерації покращеного графічного представлення мережі.

3.5. Обчислення сил у мережі

Застосування фізичних принципів закону Кулона та закону Гука до мережі полягає у використанні визначених структур даних та реалізації відповідних рівнянь. Фізичні властивості електричних зарядів та пружин імітуються в мережі вершин та ребер таким чином: усі вершини відштовхуються одна від одної, наче вони були електричними зарядами одного типу, і лише наявність ребра між двома вершинами забезпечує їх взаємне притягання. Отже, система складається з двох протилежних сил: сили відштовхування між усіма вершинами та відновлювальної сили, що виникає за наявності ребра.

Закон Кулона кількісно описує силу між двома точковими зарядами, як деталізовано в другому розділі. При його застосуванні до двовимірної системи координат отримані рівняння мають вигляд:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

$$F_x = \frac{k_C \times (q_A \times q_B)}{d^2} \times \frac{(x_A - x_B)}{d}$$

$$F_y = \frac{k_C \times (q_A \times q_B)}{d^2} \times \frac{(y_A - y_B)}{d}$$

де:

d — Евклідова відстань між точками A і B.

k_C — стала Кулона.

x_A,x_B — x-координати точок A і B відповідно.

y_A,y_B — y-координати точок A і B відповідно.

У цій реалізації закону Кулона передбачається, що всі вершини будуть відштовхуватися. Тому величини зарядів q_A та q_B можуть бути встановлені як константи, наприклад, одиничні, оскільки напрямок сил визначатиметься відносним положенням вершин. Це спрощує рівняння до:

$$F_x = \frac{k_C \times (x_A - x_B)}{d^3} \quad F_y = \frac{k_C \times (y_A - y_B)}{d^3}$$

Ці рівняння представляють горизонтальні та вертикальні компоненти сили відштовхування. Для отримання протилежної відновлювальної сили застосовується закон Гука. Він надає взаємозв'язок між горизонтальними та вертикальними компонентами відновлювальної сили та зміщенням розтягнутої пружини, як деталізовано в другому розділі.

У реалізації кожна вершина відчуває силу відштовхування від усіх інших вершин та відновлювальну силу від усіх ребер, кінцем яких вона є. Отже, загальну силу, що діє на вершину A, можна виразити її горизонтальними (FXA) та вертикальними (FYA) компонентами наступним чином:

$$F_{X_A} = \sum_{i \neq A} \left(Q_{X_{A,i}} + S_{X_{A,i}} \times \delta_{(A,i)} \right)$$

де Q_{XA} -компонента сили Кулона),

$S_{XA,i}$ - компонента сили Гука,

$\delta_{(A,i)} = 0$, якщо немає ребра між A і i, та 1, якщо ребро існує.

Сталу Кулона було округлено до значення 100 000, оскільки модель не є точним фізичним представленням електричних зарядів, а скоріше алгоритмом, натхненним взаємозв'язком сили та відстані. Таким чином, сила, що діє на вершину A з боку вершини B, може бути повністю обчислена на основі x та y координат вершин.

Після визначення методології обчислення сил, інтеграція всіх компонентів дизайну здійснюється шляхом послідовного виконання етапів. Перший крок – ініціалізація вершин у різні випадкові позиції, як описано в підрозділі 3.3. Далі, для виконання обчислень, створюється новий потік для перемальовування дисплея, деталізований у другому розділі. Запускається ітераційний цикл, який спочатку переміщує вершини, потім зсуває весь граф у видиму область дисплея. Дисплей оновлюється за допомогою нового потоку, і розмір найбільшого переміщення порівнюється з встановленим порогом перед початком нової ітерації. Псевдокод нижче підсумовує алгоритм:

```
Optimize(G=<V,E>) {
    // Випадкове розташування вершин
    for each vertex v in V {
        v.setXcoordinate = Math.random();
        v.setYcoordinate = Math.random();
    }
    // Створення нового потоку для перемальовування
    Thread repainter = new Thread() {
        public void run() {
            networkDisplay.repaint();
        }
    }
    // Ітераційний процес оптимізації
    do {
        // Обчислення сил
        for each vertex v1 in V {
            for each vertex v2 in V {
                calculate repelling force;
                if (v1 and v2 are neighbors) {
                    subtract restoring force;
                }
                store force in 2D array;
            }
        }
```

```
        // Переміщення вершин
        for each vertex v in V {
            sum total force on v from 2D array;
            v.setXcoordinate = ratio * totalXforce;
            v.setYcoordinate = ratio * totalYforce;
            max = size of largest move; // Оновлення найбільшого переміщення
        }
        // Зсув до видимої області (визначення найменших x та y)
        find lowest x and y;
        // Перемальовка
        repainter.start();
    } while (max > limit); // Умова зупинки ітерацій
}
```

3.6. Демонстрація переміщення вершин та візуалізація кластерів

Для наочної демонстрації механізму переміщення вершин, реалізованого запропонованим алгоритмом, на рисунку 3.5 представлено невеликий прикладний граф, що складається з п'яти вершин.

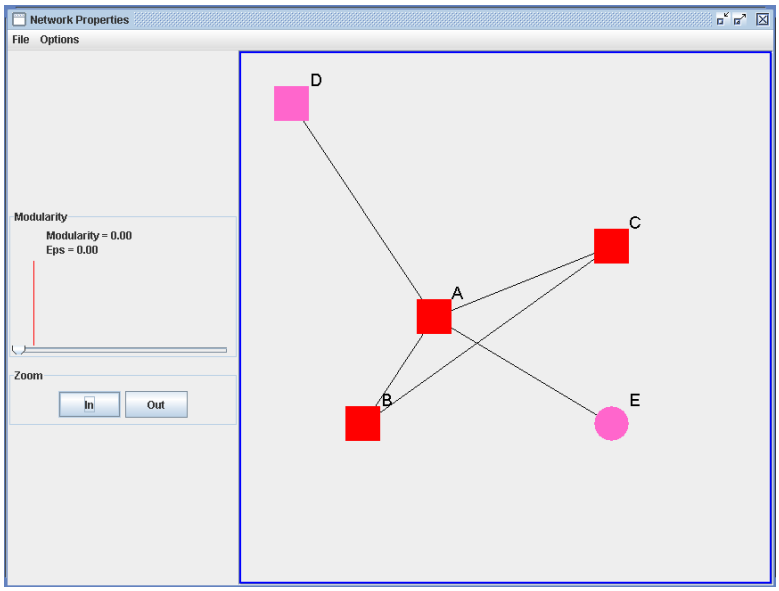

Рисунок 3.5 - Приклад графа з п'ятьма вершинами

У випадку трьох точок, з'єднаних ребрами однакової ваги (що в даній моделі відповідає силам притягання), оптимальний баланс сил призводить до утворення рівностороннього трикутника. Як показано на рисунку 3.6, після виконання процедури оптимізації, три вершини – A, B і C – формують збалансований рівносторонній трикутник.

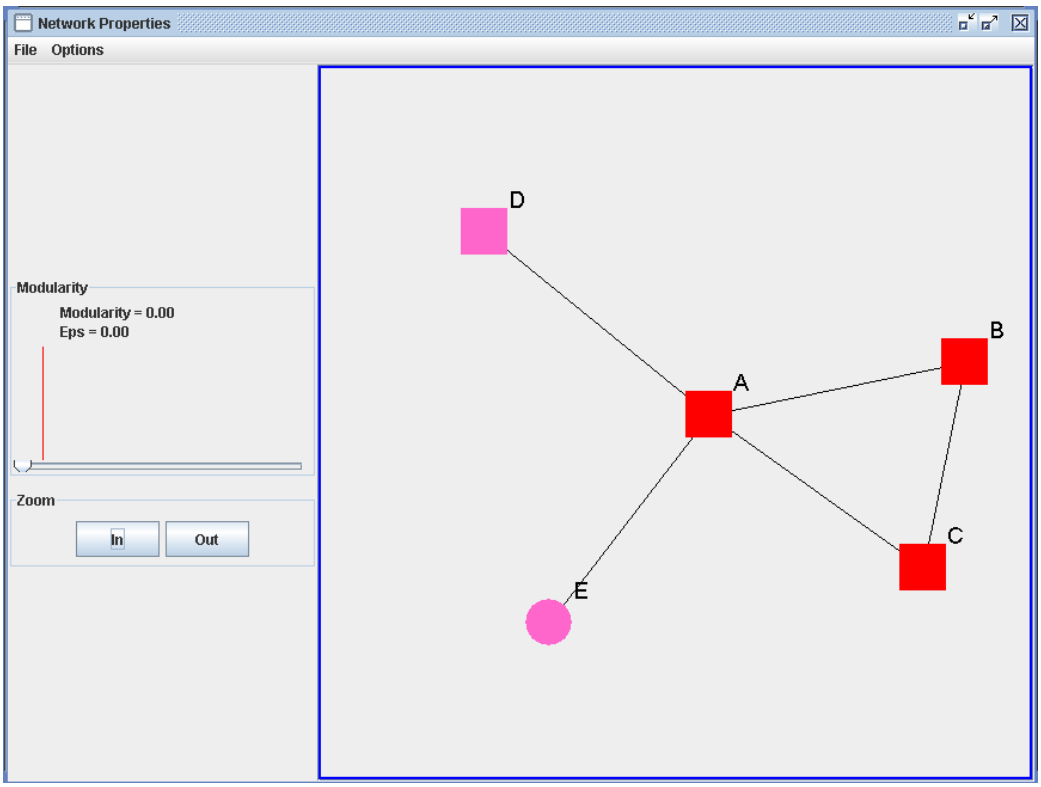

Рисунок 3.6 - Граф з п'ятьма вершинами після оптимізації

Крім того, дві вершини, D і E, які можуть бути класифіковані як викиди графа (оскільки кожна з них з'єднана лише з однією іншою вершиною), чітко візуально відокремлені від основного кластера. Ефективність такого відокремлення аномальних елементів стане ще більш очевидною при застосуванні алгоритму до більших наборів даних. Цей приклад підтверджує здатність алгоритму не лише групувати взаємопов'язані елементи, але й виділяти периферійні або аномальні вузли.

3.7. Експериментальні результати тестування та аналіз роботи системи

Після верифікації принципу переміщення вершин, реалізованого алгоритмом, було проведено застосування алгоритму до реальних наборів даних для оцінки його ефективності.

3.7.1. Аналіз мережі онлайн-блогерів

Перший тестовий набір даних був побудований на основі взаємозв'язків між онлайн-блогерами. У цій мережі кожна вершина представляє окремого блогера, а ребро встановлюється між двома блогерами, якщо принаймні один з них посилався на іншого у своїх блогах протягом останнього місяця. Вершини були забарвлені відповідно до веб-сайту, який надавав послуги хостингу їхніх блогів. Початковим припущенням було, що блогери, розміщені на одному веб-сайті, частіше посилатимуться один на одного, отже, очікувалося, що алгоритм кластеризації згрупує вершини за кольором.

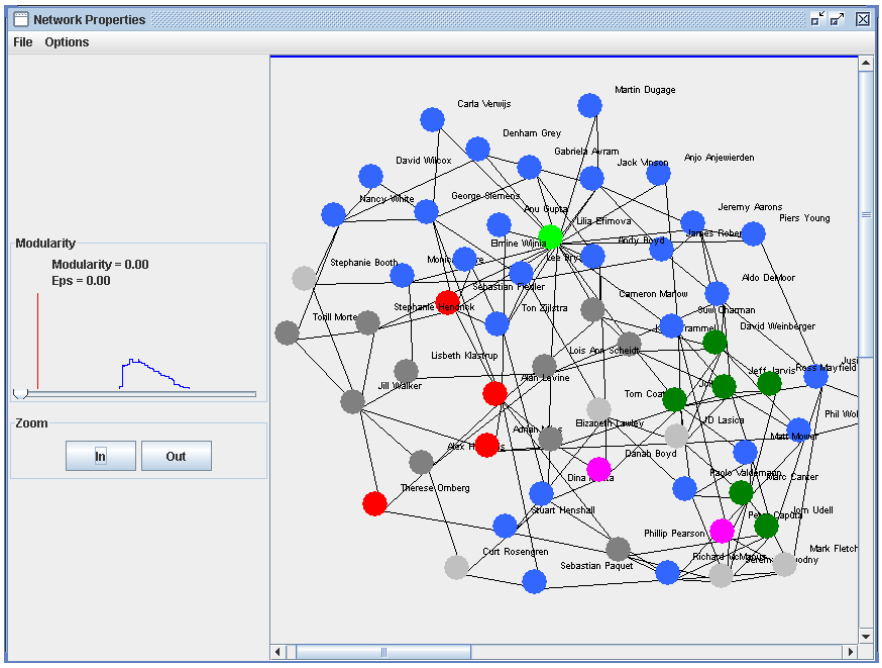

Рисунок 3.7 - Граф онлайн-блогерів після кластеризації

На практиці, отриманий граф, представлений на рисунку 3.7, не продемонстрував очікуваної чіткої кластеризації за кольором. Такий результат, на перший погляд, може здатися невтішним, проте детальний аналіз природи самих даних виявив джерело цього неочікуваного висновку. Вихідне припущення про кореляцію між хостингом блогів та взаємними посиланнями виявилося невідповідним реальним відносинам у наборі даних. Таким чином, результати на рисунку 3.7 свідчать не про повну

неспроможність алгоритму, а скоріше про хибне вихідне розуміння властивостей даних. Відсутність очікуваних тенденцій у наборі даних, як це візуалізовано на рисунку 3.7, дозволила глядачеві отримати нове розуміння даних, яке не було очевидним до застосування інструменту візуалізації. Це підкреслює здатність візуалізаційних інструментів виявляти приховані або неочевидні характеристики даних.

3.7.2. Візуалізація соціальної мережі

Альтернативний набір даних, використаний для демонстрації можливостей інструменту візуалізації у сфері соціальних мереж, представлений на рисунку 3.8. Цей граф побудований на основі дружніх зв'язків особи на ім'я Тіна, де індивіди зображені як вершини, а ребро з'єднує двох друзів. На відміну від попереднього випадку з блогерами, відносини в цьому графі чітко визначені, а не припущені.

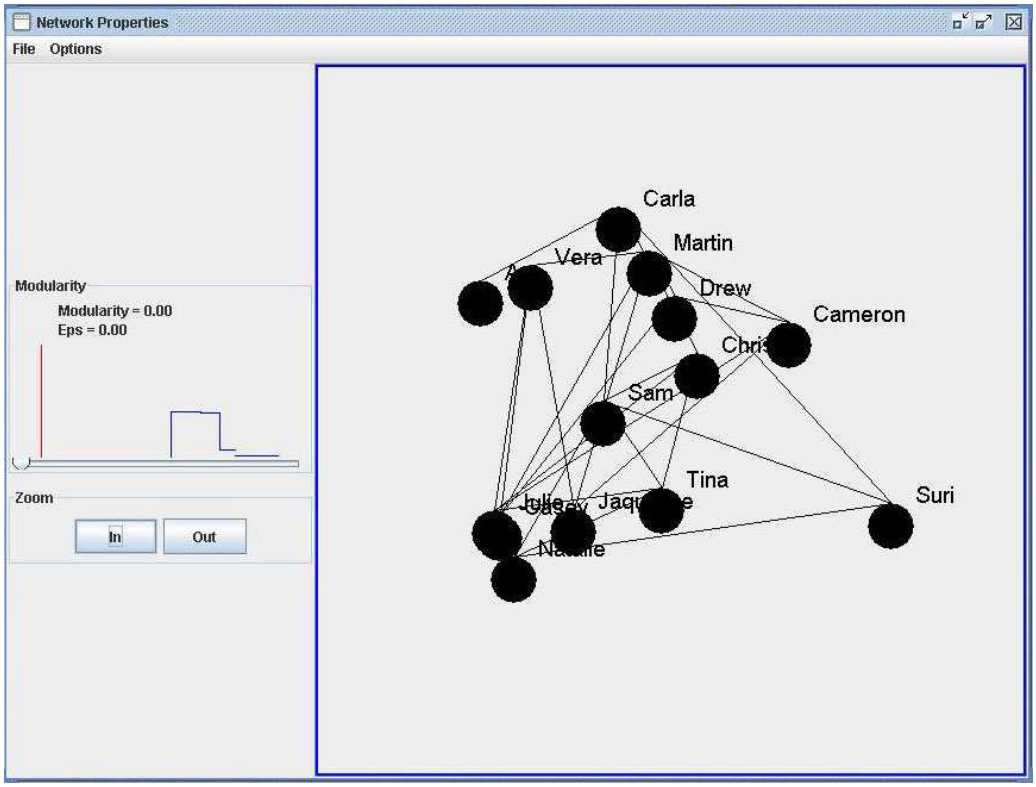

Рисунок 3.8 - Граф друзів користувача соціальної мережі

Результати застосування алгоритму оптимізації до цієї мережі, зображені на рисунку 3.9, демонструють виразне розділення на кластери. Крім чітко визначених розділів, також швидко розпізнаються вершини, що не належать до жодного явного кластера.

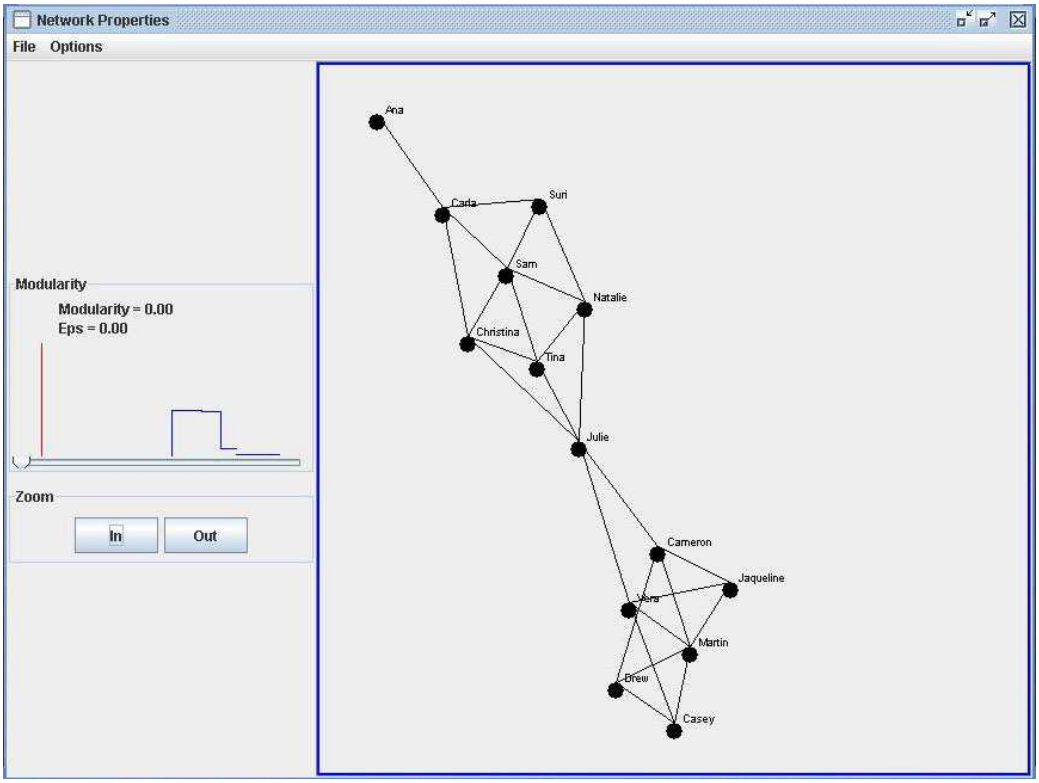

Рисунок 3.9 – Кластеризований граф друзів користувача соціальної мережі

На рисунку 3.9 легко ідентифікувати викид (outlier), представлений вузлом "Ана", та хаб (hub), позначений як "Джулі", який має рівну кількість зв'язків з обома основними підграфами. Ця мережа є чудовим прикладом здатності запропонованого алгоритму оптимізації підсилювати тенденції, які інакше були б непомітними, як це чітко видно порівняно з рисунком 3.8. Фактично, виявлену тенденцію неможливо легко ідентифікувати шляхом поелементного аналізу набору даних; хоча граф друзів Тіни містить лише чотирнадцять вершин, графи, побудовані з таких джерел соціальних мереж, як Facebook, можуть легко сягати шістдесяти дев'яти мільйонів вершин.

3.7.3. Аналіз мережі футбольних матчів

Останній набір даних, представлений для демонстрації здатності візуалізації просторово розділяти підграфи, був згенерований з розкладу матчів NCAA Football Bowl Subdivision (раніше Division 1-A) за 2006 рік. Кожна команда представлена вершиною в графі, а ребро з'єднує дві вершини, якщо між відповідними командами заплановано матч. Набір даних включає 180 команд, серед яких є команди нижчих дивізіонів, що мали заплановані матчі проти команд Division 1-A, та 787 матчів. Вершини забарвлені відповідно до конференції, до якої належить команда; команди, що належать до конференцій поза Division 1-A, забарвлені світло-сірим кольором. На початковому графі (рисунок 3.10) дані перемішані, і навіть при використанні колірного кодування вузлів, виділити окрему конференцію досить складно.

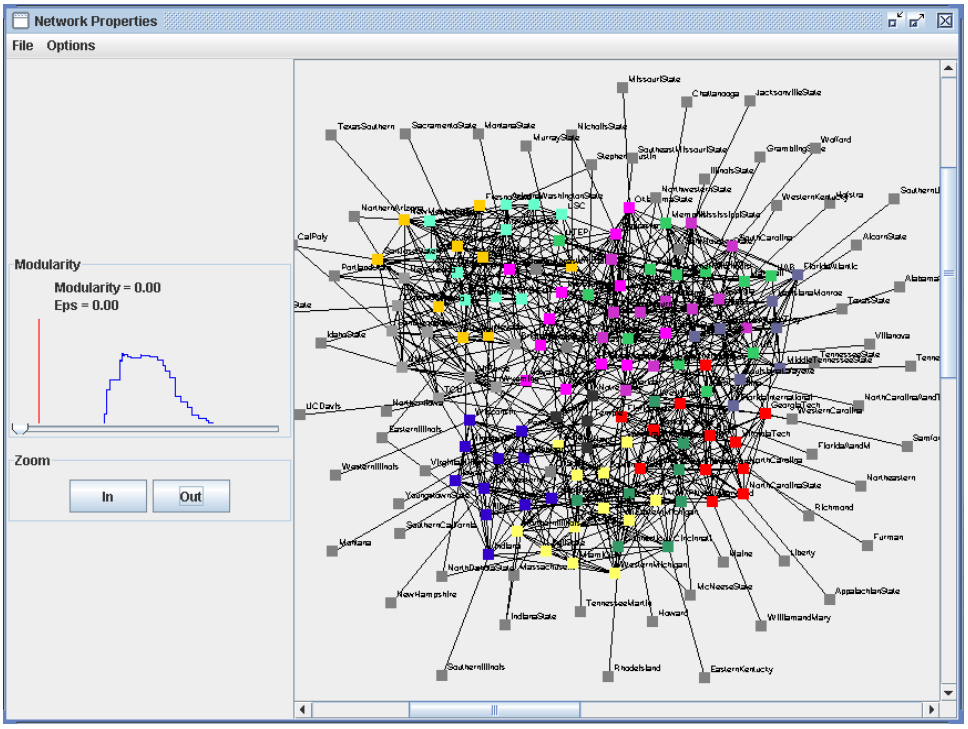

Рисунок 3.10 – Графова візуалізація розкладу футбольних матчів

Великий розмір цього набору даних робить очевидною перевагу багатопотокової реалізації алгоритму. У міру виконання обчислень, дисплей постійно оновлюється з останнім станом графа. Глядач фактично спостерігає

анімацію процесу обчислення, що дозволяє візуально відстежувати траєкторії руху вершин. Рисунок 3.11 є знімком екрана моменту цієї анімації, демонструючи, що анімація значно сприяє візуалізації, надаючи аудиторії зростаюче розуміння структури підграфів у міру того, як вони стають все більш чіткими.

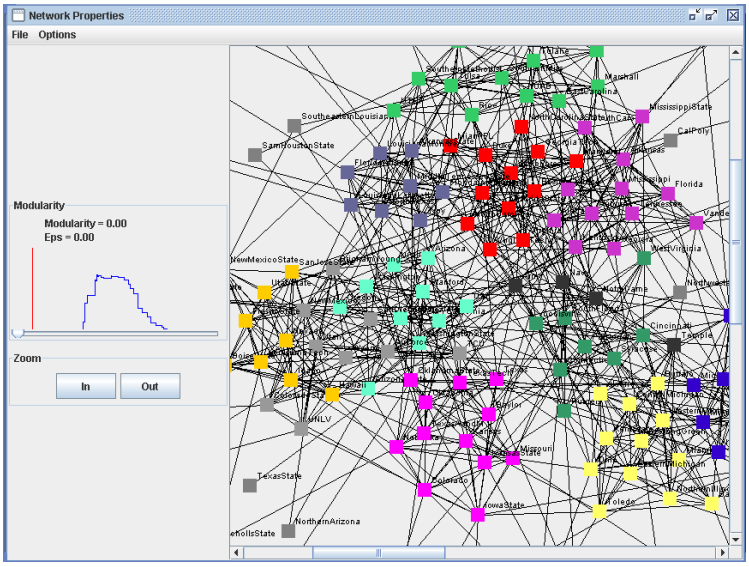

Рисунок 3.11 - Момент в анімації графа

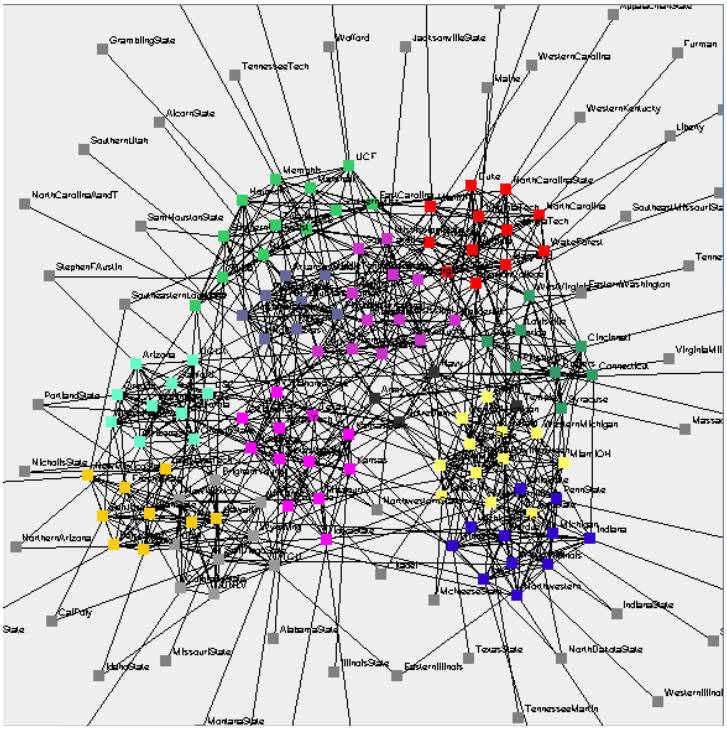

Рисунок 3.12 - Кластеризована версія графа

Кінцевий результат візуалізації графа представлений на рисунку 3.12. Найбільш вражаючими елементами є викиди та хаби, оскільки вони швидко візуально помічаються як аномалії або точки особливого інтересу в даних. Цей приклад підкреслює здатність алгоритму ефективно розрізняти та візуалізувати різні типи вузлів у складних мережах.

3.8. Опис процесу удосконалення візуалізації кластеризованих даних за допомогою алгоритмів на основі сил

3.8.1. Актуальність та застосування кластеризації даних

Кластеризація даних є фундаментальною галуззю аналізу даних, що дозволяє витягувати корисну інформацію з наборів даних шляхом ідентифікації та використання спільних характеристик між окремими елементами. Значущість успішної класифікації компонентів у наборах даних підкреслюється численними застосуваннями кластеризації як у природничих, так і в соціальних науках.

Наприклад, екологічні дослідження вимагають упорядкування кліматичних, біотичних (наприклад, харчових) або міграційних даних для ідентифікації структур життєвих систем, таких як біосфери чи екосистеми. Хоча дослідники в екології використовують обчислювальні інструменти для вирішення подібних задач, вибір алгоритму, який відповідає характеристикам екологічних даних, залишається складним завданням. Інші галузі, що використовують кластеризацію як ключовий інструмент для організації даних, включають медицину (для аналізу електрокардіограм), маркетингові дослідження (для ідентифікації потенційних сегментів споживачів на основі опитувань або тестових панелей), а також провайдерів пошукових систем в Інтернеті (для покращення результатів пошуку шляхом інтелектуального групування). Отже, існує не тільки нагальна потреба в

методах кластеризації, але й величезний обсяг даних, які потребують обробки цими методами.

3.8.2. Роль візуалізації в аналізі даних

Більшість застосувань кластеризації інформації, включаючи всі вищезазначені приклади, значно виграють від ефективного візуального представлення. Основна мета візуалізації полягає не в простому відображенні даних, а в підкресленні та розкритті прихованих закономірностей та явищ. За оцінками, до п'ятдесяти відсотків нейронів мозку людини задіяні в обробці значного обсягу інформації, отриманої через зір. Таким чином, ефективна візуалізація повинна не лише надавати точне представлення набору даних, але й забезпечувати його інтерпретацію з максимальною швидкістю. Визнання потужності та швидкості людської інтерпретації означає, що візуалізація повинна завжди мати бажаний ефект для досягнення своєї фундаментальної мети.

Певним чином, розробка методу візуалізації паралельна розробці рекламного продукту. У випадку єдиного статичного зображення, наприклад, первинний візуальний вплив є найважливішим. Хоча це може здатися суперечливим, оскільки існують графіки, що вимагають ретельного вивчення легенд, позначок осей та одиниць вимірювання, які використовуються фахівцями в різних галузях, такі графіки не кваліфікуються як візуалізація у строгому сенсі. Різниця полягає в тому, що цінність візуалізації полягає у вилученні сенсу інформації, і це не те саме, що графічне представлення, яке служить лише альтернативою таблиці або подібній структурі даних для відображення результатів. Як приклад, графічним представленням даних може бути набір вимірювань температури, нанесених як точки відносно осі, що позначає відстань від джерела тепла. Можливою візуалізацією цих самих даних може бути карта розподілу тепла, яка використовує колірне кодування для різних температур, а сама тестова область надає форму зображення.

Візуалізація використовує графічні примітиви, такі як координатні площини, а також візуальні техніки та технології відображення, щоб надати інсайти щодо представленої інформації.

Дослідження, представлене в цій роботі, зосереджується на здатності покращувати візуалізацію наборів даних, які демонструють внутрішні розділені підмножини (кластери). Відображення представлене як двовимірний граф точок, що представляють кожен елемент даних, і ребер, що з'єднують дві точки, між якими існує визначене відношення. Для демонстрації результатів візуалізації в рамках цієї роботи використовуються набори даних з кількох джерел соціальних мереж та результати футбольних матчів. Ці набори даних мають різні типи відношень, що з'єднують їх елементи. Важливо зазначити, що наявність відношення між двома елементами надається алгоритму кластеризації як вхідні дані, оскільки метою алгоритму не є вилучення відношень із заданого набору даних. Таким чином, для згаданих наборів даних, наявність відношення між двома точками (тобто ребро між двома вузлами) визначалася суб'єктивним визначенням, специфічним для кожного набору даних. Це є сильною стороною запропонованого методу візуалізації, оскільки він гнучкий до різних застосувань, і різні атрибути даних можуть бути підкреслені залежно від визначення відношення.

3.8.3. Призначення запропонованого методу візуалізації

Метод візуалізації, детально описаний у цій роботі, призначений для покращення візуального виявлення групувальних тенденцій у наборі даних. Цінність візуалізації полягає в інсайті, який вона надає глядачеві щодо характеристик даних, що інакше не були б очевидними. Дана робота спрямована на покращення представлення наборів даних, що відображаються у вигляді двовимірного графа, який також може бути названий мережею, що складається з елементів, відомих як вершини, з'єднані ребрами.

Сучасні методи кластеризації зазвичай намагаються групувати вершини мережі в набір підмереж таким чином, що кожна вершина належить лише до одного кластера. Проте набори даних, отримані в результаті наукових вимірювань або соціального аналізу, часто містять аномалії – елементи, які не відповідають жодній чіткій тенденції. Наприклад, у мережі, де вершини представляють учнів середньої школи, а ребро з'єднує двох учнів, якщо вони є друзями, алгоритм кластеризації може виділити "кліки" (тісно пов'язані групи). Однак можуть існувати вершини з одним або нульовим зв'язком, що називаються викидами (outliers), а також вершини, що з'єднані з двома або більше кліками, відомі як хаби (hubs). Примушуючи всі вершини належати до одного підграфа, викиди помилково вважаються такими, що мають стільки ж асоціацій з клікою, скільки і всі інші її члени, а множинні зв'язки хаба з іншими підграфами ігноруються. У наведеному прикладі мережі середньої школи, здатність відстежувати поширення чуток серед учнів або навіть чогось настільки важливого, як шлях поширення вірусу, буде втрачена на кластеризованій мережі. Цей приклад ілюструє наслідки неможливості візуалізувати всі властивості розділених наборів даних за допомогою існуючих алгоритмів кластеризації.

Метою цього алгоритму є переміщення вершин таким чином, щоб кластери, викиди та хаби в мережі були візуально більш очевидними.

За умови наявності набору даних з визначеними відношеннями між його елементами, візуалізація даних значно покращується за допомогою алгоритму, який розміщує вузли даних на двовимірній координатній площині, імітуючи фізичну поведінку електричних точкових зарядів, з'єднаних пружинами.

Абстрагування набору даних у граф вершин і ребер, як це має місце в середовищі, в якому має функціонувати ця візуалізація кластеризації, є ефективним засобом для забезпечення універсальності алгоритму, що дозволяє застосовувати його до різних типів даних. Переміщення вершин

мережі повинно ґрунтуватися на її графічних елементах, а не на специфічних властивостях самого набору даних. Цей метод використовує інформацію про вершини та ребра мережі у модифікованих фізичних рівняннях, які зазвичай використовуються для моделювання поведінки електричних зарядів та пружин. Кожна вершина представлена як електричний заряд в алгоритмі, а наявність ребра між двома вершинами зображується як пружина, що з'єднує два електричні заряди.

Ця техніка для переміщення графічних точок з метою досягнення бажаної візуалізації вихідного набору даних не розрізняє два типи зарядів. Натомість, усі елементи набору даних представлені як один тип електричного заряду і завжди відштовхуються, як пояснено в другому розділі. З точки зору графічного представлення даних, немає сенсу мати два різні типи точок, оскільки це розрізнення не може бути виражено в самому графі. Ще одним ключовим аспектом підходу є те, що між двома зарядами може існувати лише одна пружина, що означає одне ребро між двома вершинами. Стосовно набору даних, це означає, що відношення між двома елементами або існує, або ні. Таким чином, мережа стає набором рухомих точок, з'єднаних пружинами, і зрештою всі сили досягнуть стану рівноваги, де кожна сила врівноважується силою еквівалентної величини. Більш точне визначення рівноваги насправді є станом, де всі сили врівноважуються силами майже еквівалентної величини. Оптимальна точка завершення алгоритму досягається, коли досягається стан рівноваги, але існує можливість випадку, коли вершини повторно рухаються туди-сюди незначною мірою, і це має бути враховано.

Підхід, застосований у цьому дослідженні до проблеми візуалізації наборів даних, що мають внутрішні розділення, може бути ефективно використаний для аналізу соціальних тенденцій, які раніше не були легко помітні. Здатність швидко ідентифікувати хаб, наприклад, що впливає на шаблони даних, не може бути досягнута за допомогою простого порівняння

ступенів вершин або технологій кластеризації, розглянутих у першому розділі. Хаб може мати критичне значення для суспільства у випадку аналізу поширення захворювань або зв'язків між групами, підозрюваними у делінквентній діяльності. Гнучкість визначення ребер як будь-якого відношення без зміни функціональності алгоритму також є значною перевагою. Завдання модифікації алгоритму для відповідності конкретному науковому використанню полегшується цією технікою.

3.9. Напрямки подальших досліджень та вдосконалень

Представлена робота забезпечує базову архітектуру для візуалізації алгоритму кластеризації. Проте, існують аспекти, в яких реалізація може бути суттєво покращена.

Одним із шляхів посилення візуального ефекту групування є колірне кодування вершин відповідно до їх приналежності до кластерів. Це дозволить візуально підтвердити успішність застосованої техніки кластеризації через виразні колірні патерни. Додатково, викиди та хаби можуть бути візуально виділені шляхом присвоєння їм відмінних кольорів, що дозволить глядачеві легко ідентифікувати аномальні або центральні елементи.

Реалізація також потребує обліку особливих випадків для вершин або кластерів, які повністю відокремлені від решти графа. Поточний алгоритм переміщує такі ізольовані частини графа на значну відстань від основної маси, оскільки відсутня відновлювальна сила, що протидіяла б силі відштовхування між вершинами. Однак для візуального розуміння того, що ці елементи є відокремленими, не є обов'язковим їх розміщення на максимально можливій відстані від основної структури графа. Оптимізація їхнього просторового розміщення, при збереженні візуальної диференціації, може покращити загальну естетику та зрозумілість візуалізації.

Крім того, поточна реалізація встановлює фіксовані обмеження як на початкове переміщення вершин, так і на визначення точки рівноваги. Обидва ці параметри можуть суттєво впливати на продуктивність алгоритму залежно від розміру та характеристик набору даних.

Початкове переміщення вершин: Необхідно забезпечити початкове переміщення всіх вершин перед будь-якими обчисленнями сил, щоб уникнути ситуації, коли дві вершини розташовані в ідентичних координатах, що призвело б до ділення на нуль при обчисленні відштовхуючих сил. Поточна реалізація встановлює горизонтальні та вертикальні межі для початкового розташування вершин. Однак, для значно великих наборів даних, може бути виправданим розширення цих початкових меж. Це дозволить уникнути надмірних сил відштовхування на початкових ітераціях, які виникають через надмірну близькість вершин.

Критерій зупинки (точка рівноваги): Для більших наборів даних, що характеризуються більшою кількістю взаємодіючих сил, досягнення стану балансу (де найбільший рух вершини стає меншим за встановлене обмеження) може вимагати значно більшої кількості ітерацій, ніж є дійсно необхідним. Дослідження адаптивних або евристичних критеріїв зупинки може оптимізувати час виконання алгоритму без суттєвої втрати якості візуалізації.

Найбільш корисною роботою для подальшого розвитку цієї програми буде систематичний аналіз різних наборів даних у ширшому спектрі застосувань. Таке дослідження дозволить краще зрозуміти сильні та слабкі сторони даної технології візуалізації, визначити її оптимальні параметри для різних типів даних та розширити її функціональність для вирішення нових, більш складних задач кластеризації та візуалізації.

ВИСНОВКИ

У даній дипломній роботі успішно розроблено та реалізовано метод візуалізації кластеризації даних, що базується на принципах фізичних моделей, зокрема взаємодії електричних зарядів та пружин. Проведене дослідження охопило аналіз предметної області візуалізації кластеризації даних, теоретичні та практичні аспекти кластеризації, а також детальне представлення архітектури та реалізації розробленої системи.

Розроблено методологічну основу та реалізовано метод візуалізації кластеризації даних. Запропонований метод ефективно абстрагує елементи даних як вершини графа, а визначені відношення між ними – як ребра. Моделювання цих елементів як електричних зарядів та пружин відповідно дозволяє використовувати ітераційний процес оптимізації для просторового розташування вершин. Це забезпечує візуальне групування пов'язаних елементів та відокремлення непов'язаних, що є ключовим для розуміння кластерної структури.

Дослідження підкреслило, що візуалізація є не просто альтернативою табличному представленню даних, а потужним інструментом для виявлення прихованих закономірностей, хабів та викидів, які інакше могли б залишитися непоміченими. Використання багатопотоковості для анімації процесу кластеризації значно покращує інтуїтивне розуміння динаміки формування кластерів.

Здійснено глибокий аналіз сучасних алгоритмів кластеризації: Проведено систематичний огляд та класифікацію відомих методів кластеризації (на основі поділу, ієрархічні, на основі щільності, моделі, сітки та графів), що дозволило позиціонувати запропонований метод у контексті існуючих підходів та обґрунтувати його переваги у візуалізації складної структури даних.

Детально представлено алгоритми функціонування системи візуалізації. Робота містить докладний опис математичних моделей (Закон Кулона, Закон Гука), які лежать в основі алгоритму переміщення вершин, а також принципи їх інтеграції в обчислювальний процес. Окрему увагу приділено аспектам багатопотоковості та взаємодії з графічним інтерфейсом користувача (GUI). Виконано програмну імплементацію та експериментальне тестування: Розроблено програмний додаток на базі Java, який демонструє функціональність запропонованого методу. Експерименти на реальних наборах даних, таких як мережі онлайн-блогерів, соціальні мережі друзів та мережі футбольних матчів, наочно продемонстрували здатність алгоритму візуалізувати приховані кластери, ідентифікувати хаби та викиди, а також ефективно масштабуватися для великих даних.

Розроблений метод візуалізації кластеризації даних на основі фізичних моделей демонструє значну цінність для аналізу складних взаємозв'язків у великих наборах даних. Алгоритм не залежить від конкретного типу даних, дозволяючи візуалізувати кластери на основі довільно визначених відносин між елементами, що розширює його застосовність.

Загалом, результати даної дипломної роботи підтверджують ефективність та перспективність застосування методів візуалізації кластеризації даних на основі фізичних моделей. Розроблена система може бути цінним інструментом для аналізу даних у наукових дослідженнях, соціальному аналізі та комерційних застосуваннях, де швидке та інтуїтивне розуміння складної структури даних має вирішальне значення.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Cluster Analysis using Python — Part 1 | by Naina Chaturvedi | DataDrivenInvestor - https://medium.datadriveninvestor.com/cluster-analysis-using-python-part-1-4ceee387d79a
2. Порівняння методів кластеризації для створення цільових груп клієнтів у наборі даних - https://www.researchgate.net/publication/375880613_Porivnanna_metodiv_klasterizacii_dla_stvorenna_cilovih_grup_klientiv_u_nabori_danih
3. Оцінка методів кластеризації різнотипових даних / Evaluation of methods of clusterization of different types of data - otsinka-metodiv-klasterizatsiyi-riznotipovikh-danikh-2uwgsy96.pdf
4. Hierarchical Cluster Analysis · UC Business Analytics R Programming Guide - https://uc-r.github.io/hc_clustering
5. Convert Text Documents to a TF-IDF Matrix with tfidfvectorizer – Kdnuggets - https://www.kdnuggets.com/2022/09/convert-text-documents-tfidf-matrix-tfidfvectorizer.html
6. View of Energy Models for Graph Clustering - https://jgaa.info/index.php/jgaa/article/view/paper154/2814
7. SCAN: a structural clustering algorithm for networks - scan-a-structural-clustering-algorithm-for-networks-1bp86hao97.pdf
8. Interactive Exploration of Fuzzy Clusters Using Neighborgrams - Wiswedel_243746.pdf - https://kops.uni-konstanz.de/server/api/core/bitstreams/0027cb4a-45a3-4f42-b654-e63642b1caa7/content
9. From Abstract Painting to Information Visualization - https://personal.utdallas.edu/~kzhang/Publications/VISV.pdf
10. K-Medoids Clustering (ML). K-Medoids is a partition-based… | by Prasanth babu | Medium -* https://medium.com/@prasanth32888/k-medoids-clustering-cee6042155c6

11. Learn Cluster Centers | K-Medoids Algorithm - https://codefinity.com/courses/v2/138ab9ad-aa37-4310-873f-0f62abafb038/6d525817-4dbf-4817-80bf-68b630220033/61ab8ff3-63c3-4509-9931-b564689ad318
12. A Hybrid Method for Traffic State Classification Using K-Medoids Clustering and Self-Tuning Spectral Clustering - https://www.mdpi.com/2071-1050/14/17/11068
13. Ordering Points To Identify Cluster Structure (OPTICS) using Sklearn – GeeksforGeeks - https://www.geeksforgeeks.org/ordering-points-to-identify-cluster-structure-optics-using-sklearn/
14. Aggarwal, C. C., & Reddy, C. K. (2014). Data clustering: Algorithms and applications. CRC Press.
15. Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), 226–231.
16. Kaufman, L., & Rousseeuw, P. J. (2009). Finding groups in data: An introduction to cluster analysis. Wiley.
17. MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, 281–297.
18. Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: An overview. WIREs Data Mining and Knowledge Discovery, 2(1), 86–97.
19. Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems (NIPS), 14, 849–856.
20. Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3), 645–678.

21. Zhou, Z.-H. (2012). Ensemble methods: Foundations and algorithms. Chapman and Hall/CRC.
22. Bae, S., & Choi, S. (2015). Visualization of clusters using force-directed layout algorithm. Information Sciences, 328, 299–312.
23. Brandes, U. (2001). A faster algorithm for betweenness centrality. Journal of Mathematical Sociology, 25(2), 163–177.
24. Fruchterman, T. M. J., & Reingold, E. M. (1991). Graph drawing by force-directed placement. Software: Practice and Experience, 21(11), 1129–1164.
25. Herman, I., Melancon, G., & Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics, 6(1), 24–43.
26. Karypis, G., Han, E.-H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32(8), 68–75.
27. Kerren, A., Stasko, J. T., & Fekete, J.-D. (Eds.). (2014). Information visualization: Human-centered issues and perspectives. Springer.
28. Kleinberg, J. (2002). An impossibility theorem for clustering. Advances in Neural Information Processing Systems (NIPS), 15, 463–470.
29. Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(3), 231–240.
30. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.
31. Murtagh, F. (2014). Multidimensional clustering algorithms. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(5), 335–345.
32. Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53–65.

33. Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888–905.
34. Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering techniques. Proceedings of the KDD Workshop on Text Mining, 400, 525–526.
35. Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, 267–273.
36. Zimek, A., Schubert, E., & Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. Statistical Analysis and Data Mining, 5(5), 363–387.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Розробка та реалізація методу візуалізації кластеризації даних ”

Обсяг пояснювальної записки: 79 аркушів.

Дата закінчення роботи: 9 червня 2025 р.

Підпис студента _____