

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПОШУКУ СПІЛЬНОТ КОРИСТУВАЧІВ У СОЦІАЛЬНИХ МЕРЕЖАХ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
FINDING USER COMMUNITIES IN SOCIAL NETWORKS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-222м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

АВРАМЕНКО І.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

АВРАМЕНКА Ігоря Юрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів пошуку спільнот користувачів у соціальних мережах. Research and Software Implementation of Methods for Finding User Communities in Social Networks

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

затверджені наказом закладу вищої освіти від “14” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “08” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	8 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	9-10 тижні	Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Ігор АВРАМЕНКО
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Михайло КОЦУР
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
91 с., 4 табл., 31 рис., 3 дод., 29 джерел.

JAVA, HADOOP, РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ, МОВА ПРОГРАМУВАННЯ, СОЦІАЛЬНІ МЕРЕЖІ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес пошуку спільнот користувачів у соціальних мережах.

Предмет дослідження – програмні засоби для пошуку спільнот користувачів у соціальних мережах.

Метою роботи є дослідження та розробка програмного забезпечення для пошуку спільнот користувачів у соціальних мережах.

Матеріали, методи та технічні засоби: мова програмування Java, фреймворк Hadoop, середовище розробки Eclipse.

Результати. Розроблено програмне забезпечення для пошуку спільнот користувачів у соціальних мережах за допомогою мови програмування Java та фреймворку Hadoop.

Практична цінність роботи полягає у розробці програмного забезпечення, що реалізує розподілений метод для пошуку спільнот користувачів у соціальних мережах.

Висновки. Виконано проєктування програмного забезпечення для пошуку спільнот користувачів у соціальних мережах. Розроблено програмне забезпечення для пошуку спільнот користувачів у соціальних мережах. Здійснено тестування розробленого програмного забезпечення для пошуку спільнот користувачів у соціальних мережах.

Галузь використання – комерція, реклама, соціальні дослідження.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 91 pages, 4 tables, 31 figures, 3 appendixes, 29 sources.

JAVA, HADOOP, DISTRIBUTED COMPUTING, PROGRAMMING LANGUAGE, SOCIAL NETWORKS, SOFTWARE.

The object of research is the process of searching for user communities in social networks.

The subject of the research is software tools for finding user communities in social networks.

The aim of the work is to research and develop software for finding user communities in social networks.

Materials, methods, and technical tools: Java programming language, Hadoop framework, Eclipse development environment.

Results. Software has been developed for finding user communities in social networks using the Java programming language and the Hadoop framework.

The practical value of the work lies in the development of software that implements a distributed method for finding user communities in social networks.

Conclusions. Software for finding user communities in social networks was designed. Software has been developed for finding user communities in social networks. The developed software for finding user communities in social networks was tested.

The scope of use is commerce, advertising, and social research.

ЗМІСТ

С.

Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	14
1.1 Спільноти в соціальних мережах	14
1.2 Використання графів для представлення соціальних зв'язків	15
1.3 Аналіз існуючих рішень	17
1.4 Постановка задачі	24
1.5 Висновки за розділом 1	24
2 Матеріали і методи	25
2.1 Використання метрики Normalized Mutual Information	25
2.2 Модель MapReduce	26
2.3 Побудова нового підходу	28
2.4 Вибір мови програмування	33
2.5 Фреймворк Hadoop	36
2.6 Вибір середовища та інших засобів розробки для створення програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах	39
2.7 Висновки за розділом 2	43
3 Опис програми	45
3.1 Загальна схема роботи	45
3.2 Конвертація результатів роботи методів	50
3.3 Проєктування ПЗ	50
3.4 Висновки за розділом 3	52
4 Експлуатація, тестування та експериментальне дослідження програми	53
4.1 Призначення й умови застосування програми	53
4.2 Опис тестування	55
4.3 Висновки за розділом 4	58
Висновки	59

Перелік джерел посилань	60
Додаток А Технічне завдання	63
Додаток Б Фрагмент тексту програми	69
Додаток В Слайди презентації	84

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GCE – Greedy clique expansion, жадібний алгоритм розширення клік

IDE – Integrated Development Environment, інтегроване середовище для розробки;

NMI – Normalized Mutual Information;

ООП – об'єктно-орієнтоване програмування;

ПЗ – програмне забезпечення.

ВСТУП

Вивчення соціальних відносин між користувачами соціальних мереж є важливою сферою досліджень з далекосяжними наслідками для різних галузей, включаючи соціологію, психологію, інформатику, маркетинг тощо. Це міждисциплінарне дослідження соціальних зв'язків дає цінну інформацію про поведінку людей, моделі взаємодії та динаміку онлайн-спільнот. У цьому великому обговоренні ми заглибимося в важливість дослідження соціальних відносин між користувачами соціальних мереж, слід розглянути, як ця інформація може бути використана в різних областях [1]–[3].

Розуміння людської поведінки та соціальної динаміки. Розуміння соціальних відносин в онлайн-середовищі відкриває вікно в тонкощі людської поведінки. Дослідження в цій галузі допомагають розкрити, як люди формують зв'язки, спілкуються та встановлюють стосунки у віртуальних просторах [1]–[3]. Це дає уявлення про фактори, що впливають на взаємодію з користувачами, формування соціальних груп і динаміку цих відносин з плином часу.

Наслідки. Психологічні інсайти: дослідження соціальних відносин допомагає психологам зрозуміти психологічні мотиви, що стоять за поведінкою користувачів у соціальних мережах, проливаючи світло на такі аспекти, як формування ідентичності, самопрезентація та соціальний вплив [1]–[3].

Моделі соціальної взаємодії: це дозволяє соціологам аналізувати моделі соціальної взаємодії, вивчаючи, як люди формують зв'язки, підтримують відносини і орієнтуються в складнощах соціальних онлайн-просторів [1]–[3].

Покращення дизайну соціальних медіа та досвіду користувачів. Ідеї, отримані в результаті вивчення соціальних відносин, лежать в основі дизайну та функціональності платформ соціальних медіа. Дослідники та дизайнери можуть використовувати ці знання для створення інтерфейсів, які

відповідають соціальній поведінці користувачів, сприяючи безперебійному спілкуванню та встановленню зв'язків [1]–[3].

Наслідки. Дизайн інтерфейсу користувача: розуміння того, як користувачі формують та підтримують з'єднання, визначає дизайн таких функцій, як запити друзів, системи обміну повідомленнями та інтерфейси обміну вмістом [1]–[3].

Рекомендаційні системи: розуміння соціальних відносин сприяє розробці більш ефективних алгоритмів рекомендацій, пропонуючи зв'язки, групи або контент, які відповідають існуючим мережам користувачів [1]–[3].

Аналіз поширення інформації та впливу. Вивчення соціальних відносин важливо для розуміння того, як інформація поширюється через інтернет-мережі. Це дозволяє дослідникам досліджувати механізми розповсюдження інформації, виявляти впливових користувачів та аналізувати фактори, що сприяють прийняттю ідей чи поведінки в громаді [1]–[3].

Наслідки. Стратегії вірусного маркетингу: Маркетологи можуть використовувати інформацію про розповсюдження інформації для розробки більш ефективних стратегій вірусного маркетингу, використовуючи соціальних впливових осіб та розуміючи динаміку поширення повідомлень [1]–[3].

Кампанії з охорони здоров'я: у сфері охорони здоров'я розуміння того, як інформація поширюється через соціальні мережі, має вирішальне значення для розробки заходів та кампаній для пропаганди здорової поведінки або поширення важливої медичної інформації [1]–[3].

Стратегії соціального впливу та переконання. Дослідження соціальних відносин допомагає розібратися в складнощах соціального впливу. Воно дає уявлення про те, як переконують людей, про роль лідерів громадської думки та динаміку конформізму в онлайн-спільнотах [1]–[3].

Наслідки. Маркетинг та реклама: підприємства можуть використовувати знання про соціальний вплив для адаптації маркетингових

повідомлень, виявлення ключових факторів впливу для співпраці та розробки переконливих комунікаційних стратегій [1]–[3].

Кампанії зі зміни поведінки: організації, які беруть участь в ініціативах щодо зміни поведінки, таких як заохочення екологічно чистих практик або просування соціальної справедливості, можуть отримати вигоду з розуміння динаміки соціального впливу [1]–[3].

Виявлення спільнот та мережевий аналіз. Вивчення соціальних відносин є основоположним для аналізу мережі, що дозволяє дослідникам ідентифікувати спільноти в соціальних мережах, аналізувати структурні закономірності та вимірювати силу зв'язків між користувачами [1]–[3].

Наслідки. Виявлення впливових вузлів: мережевий аналіз допомагає ідентифікувати впливових користувачів або вузли в мережі, надаючи цінну інформацію для цілеспрямованих втручань або інформаційно-пропагандистських зусиль [1]–[3].

Виявлення спільнот: дослідники можуть ідентифікувати підгрупи або спільноти в межах більшої соціальної мережі, виявляючи спільні інтереси, цінності чи поведінку [1]–[3].

Кібербезпека та безпека в Інтернеті. Розуміння соціальних відносин має вирішальне значення у сфері кібербезпеки. Це допомагає у виявленні потенційних загроз, аналізі атак соціальної інженерії та розробці стратегій підвищення безпеки в Інтернеті [1]–[3].

Наслідки. Виявлення загроз: аналіз соціальних медіа може допомогти виявити закономірності, пов'язані зі шкідливими діями, такими як поширення дезінформації, фішинг або кібер-залякування [1]–[3].

Аутентифікація користувачів і контроль доступу: розуміння соціальних відносин сприяє розробці більш безпечних механізмів аутентифікації і політик контролю доступу.

Академічне та наукове співробітництво. Дослідження соціальних відносин сприяє співпраці в академічних колах та науковій спільноті. Це дозволяє вченим виявляти потенційних колабораціоністів, обмінюватися

результатами досліджень та брати участь у міждисциплінарних проєктах [1]–[3].

Наслідки. Дослідницька співпраця: розуміння соціальних відносин допомагає дослідникам виявляти потенційних колабораціоністів, сприяючи формуванню дослідницьких груп з різноманітним досвідом [1]–[3].

Міждисциплінарні дослідження: міждисциплінарна співпраця розширюється, оскільки дослідники використовують соціальні медіа для спілкування з колегами з різних галузей [1]–[3].

Розробка політики та управління. Уряди та політики можуть скористатися дослідженнями соціальних відносин для розробки політики, пов'язаної з поведінкою в Інтернеті, конфіденційністю та цифровим громадянством [1]–[3].

Наслідки. Політика безпеки в Інтернеті: розробники політики можуть розробляти політику, спрямовану на підвищення безпеки в Інтернеті, запобігання кібер-знущанням та вирішення інших проблем, пов'язаних із соціальними взаємодіями в цифровому просторі [1]–[3].

Правила конфіденційності: розуміння соціальних відносин сприяє розробці правил конфіденційності, які врівноважують переваги соціальних мереж із захистом даних користувачів [1]–[3].

Реагування на кризові ситуації та ліквідація наслідків стихійних лих. Під час криз або стихійних лих розуміння соціальних відносин стає критичним для ефективного спілкування, координації та реагування.

Наслідки. Поширення інформації: аналіз соціальних мереж допомагає зрозуміти, як поширюється інформація про кризи, дозволяючи владі поширювати своєчасну та точну інформацію [1]–[3].

Стійкість громади: розуміння соціальних відносин допомагає оцінити стійкість громади, визначити ключові вузли для роботи з населенням та підвищити готовність до стихійних лих [1]–[3].

Персоналізовані послуги та надання контенту. Компанії та контент-провайдери можуть використовувати інформацію про соціальні відносини для

надання персоналізованих послуг, рекомендацій та Контенту, які відповідають уподобанням користувачів та соціальним мережам [1]–[3].

Наслідки. Персоналізовані рекомендації: потокові сервіси, платформи електронної комерції та контент-провайдери можуть використовувати соціальні мережі для надання персоналізованих рекомендацій, заснованих на соціальних зв'язках і перевагах користувачів.

Індивідуальний досвід: розуміння динаміки соціальних відносин допомагає адаптувати досвід користувачів, підвищуючи залучення та лояльність користувачів [1]–[3].

На закінчення, важливість дослідження та вивчення соціальних відносин між користувачами соціальних мереж поширюється на різні сфери, формуючи наше розуміння людської поведінки, сприяючи створенню спільнот та інформуючи про розвиток цифрових платформ та втручань [1]–[3]. Наслідки величезні та ефективні – від покращення досвіду користувачів та маркетингових стратегій до вирішення проблем кібербезпеки та інформування державної політики. Оскільки технології продовжують розвиватися, вивчення соціальних відносин залишається невід'ємним аспектом подолання складнощів цифрової епохи, що має далекосяжні наслідки для окремих людей, спільнот і суспільства в цілому [1]–[3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Спільноти в соціальних мережах

На даний момент існує більше 200 широковідомих, і величезна кількість менш популярних сайтів соціальних мереж. У цих мережах зареєстровано мільйони людей, кожен з яких пов'язаний з десятками або навіть сотнями інших користувачів. Всі ці складні взаємозв'язки можна зручно представити у вигляді соціального графа користувачів.

Однією з проблем, що виникають при аналізі соціальних мереж (як, втім, і багатьох інших реальних мереж), є спрямованість зв'язків між вузлами. Наприклад, в графі Всесвітньої Павутини вершинами є сторінки, а зв'язками - посилання з одних сторінок на інші. Ці посилання спрямовані-якщо ми може перейти за посиланням зі сторінки a на сторінку b , то далеко не завжди ми можемо виконати зворотну дію. Пошук спільнот у вебграфіку може допомогти визначити штучні кластери, створені фермами посилань з метою покращення значень PageRank [4] для певних сайтів та підвищення їх позиції в Пошуку Google. Таким чином, врахування напрямку зв'язків може істотно поліпшити якість поділу на кластери і надати додаткову цінну інформацію про систему. Однак, з метою спрощення алгоритму пошуку спільнот, інформацію про напрямок зв'язків часто опускають.

На жаль, спрямованість зв'язків в графі не єдина проблема при роботі з реальними графами. У багатьох таких мережах вузли можуть стосуватися більш ніж одного кластера. У цьому випадку говорять про пересічні спільноти. Класичним прикладом знову ж таки є соціальні мережі, де людина зазвичай належить до декількох груп одночасно: колеги, сім'я, схожі спортивні уподобання і т.д. традиційні алгоритми виділення спільнот зіставляють кожну вершину тільки одному кластеру, таким чином, втрачаючи потенційно важливу інформацію. Вершини, що належать більш ніж одній громаді, часто відіграють важливу роль посередника між різними секціями графа. У той же

час варто враховувати, що пошук пересічних спільнот обчислювально набагато більш складне завдання, ніж звичайна кластеризація.

Остання важлива проблема в обробці соціальних графів, про яку хотілося б згадати, - це необхідність ефективної роботи з величезними обсягами даних. Наприклад, в мережі Twitter більше 250 мільйонів користувачів і 15 мільярдів зв'язків між ними, а в Facebook більше 500 мільйонів користувачів і 50 мільярдів зв'язків. Для обробки таких обсягів даних необхідно залучати розподілені обчислювальні системи і використовувати алгоритми, що допускають розпаралелювання. В якості одного з таких варіантів можна привести модель розподілених обчислень MapReduce, що дозволяє легко масштабувати обробку даних.

1.2 Використання графів для представлення соціальних зв'язків

Графи вкрай зручні для представлення широкого спектру систем в різних областях. Біологічні, соціальні, технологічні та інформаційні мережі можуть бути представлені в якості графів, і тому аналіз графів став ключовим механізмом для вивчення властивостей цих систем. Наприклад, аналіз соціальних мереж бере початок в 30х роках минулого століття, і до теперішнього моменту став одним з найважливіших напрямків в соціології [4].

Графові представлення реальних систем відрізняються своєю неоднорідністю. В особливих групах вершин концентрація дуг висока, в той час як між цими групами вона невелика. Ця властивість реальних мереж називається структурою спільноти [5] або кластеризацією. Спільнота, або кластер, - це сукупність вершин, відносно сильно пов'язаних між собою, і, можливо, мають спільні властивості та/або грають подібні ролі в мережі. На малюнку 1 представлений приклад графа з виділеними спільнотами.

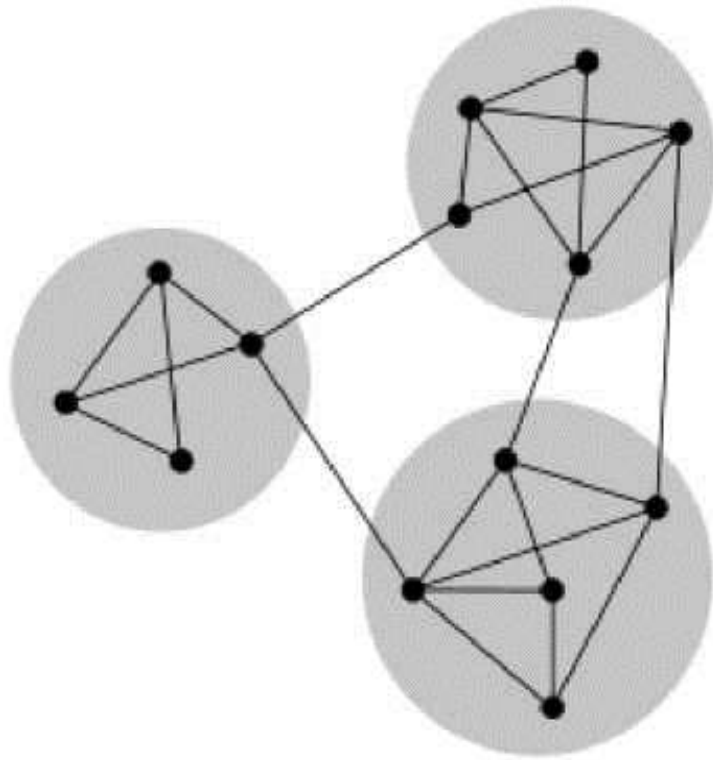


Рисунок 1.1 – Граф з трьома спільнотами

Суспільство пропонує велику кількість можливих спільнот: сім'я, колеги, друзі, жителі одного міста, громадяни однієї держави і т.д. широке поширення Інтернету призвело до появи великої кількості віртуальних груп, онлайн спільнот. У мережах взаємодій білок-білок спільнотами можна назвати групи протеїнів, що мають однакові функції в клітині [6], [7]. У графі всесвітньої павутини спільноти можуть відповідати наборам сторінок, присвячених загальній або пов'язаним темам.

Слід навести кілька прикладів конкретних застосувань спільнот. Кластеризація вебклієнтів по їх інтересам і географічному положенню дозволяє поліпшити продуктивність сервісів в Інтернеті, кожен отриманий кластер може бути оброблений окремим допоміжним сервером [8]. Кластери великих графів можуть бути використані при створенні зручних структур даних для ефективного зберігання таких графів і обробки навігаційних запитів, наприклад, пошуку шляху з одного вузла в інший [9]. Мережі самоконфігурації мережі, структура яких часто змінюється (наприклад, мобільні мережі), як правило, не мають централізованої таблиці

маршрутизації. Групування вузлів у кластери дозволяє швидко генерувати компактні локальні таблиці, що забезпечують ефективне спілкування [10].

Визначення спільнот та їх меж дозволяє класифікувати вузли графа за їх функціями в кластері. Вузли в центральних позиціях, що мають велику кількість зв'язків з іншими членами групи, несуть регулюючі і стабілізуючі функції. А знаходяться на кордонах спільнот вузли є сполучними, і відповідальні за взаємодію з іншими кластерами. Наведена класифікація властива різного роду соціальним мережам [11], [12].

Ще один важливий аспект, пов'язаний з кластерною структурою, - ієрархічна організація більшості реальних мереж. Такі мережі зазвичай складаються з спільнот, в свою чергу містять більш дрібні спільноти. Як приклад можна привести мережу, що відображає будову людського організму: людина складається з органів, органи з тканин, тканини з клітин і т. д. іншим прикладом можуть служити фірми, організовані в піраміду із співробітників, робочих груп, відділів і т. д.

Метою пошуку спільнот в графах є визначення кластерів і, по можливості, їх ієрархії, використовуючи лише інформацію про структуру цього графа.

1.3 Аналіз існуючих рішень

Метою огляду є оцінка і порівняння найбільш вдалих і ефективних методів кластеризації графів. На підставі результатів дослідження буде проведено відбір методів, на базі яких буде розроблено розподілене рішення.

В огляд включені найбільш успішні представники декількох сімейств алгоритмів. Далі поговоримо про кожного з них.

1.3.1 Методи ієрархічної кластеризації

Ієрархічні підходи використовують міру подібності вершин і групують вершини, щоб отримати природне розбиття графа.

Методи ієрархічної кластеризації поділяються на дві основні категорії:

- агломеративні-ітеративне злиття груп вершин з високим ступенем подібності;
- дивізімні-ітеративне видалення ребер між вершинами з малим ступенем подібності.

Широко відомий алгоритм, що використовує ієрархічну кластеризацію, був запропонований Гірван і Ньюменом [13]. Це дивізімний алгоритм, ітеративно видаляє ребра з високим показником проміжності (betweenness). Міру проміжності можна обчислити як, наприклад, кількість найкоротших шляхів, що пролягають через ребро. Чим вище це значення, тим імовірніше ребро знаходиться поза кластером і є сполучною.

На рисунку 1.2 представлений приклад розподілу міри проміжності по ребрах.

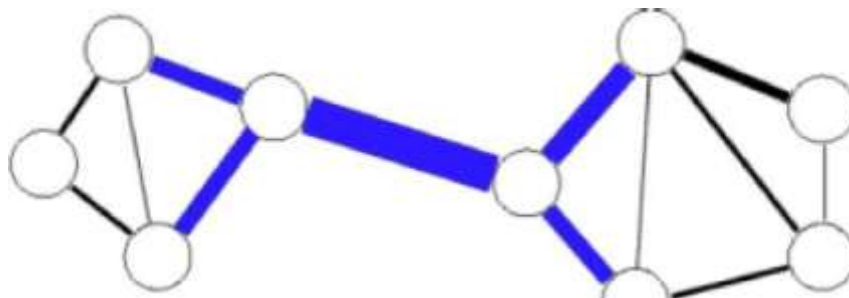


Рисунок 1.2 – Представлення проміжності

Ієрархічна кластеризація дозволяє отримати хороші результати на реальних даних, але є обчислювально складною і немасштабіруемой.

1.3.2 Максимізація цільової функції

Алгоритми цієї групи оптимізують значення якоїсь функції якості розбиття графа на частини. Класичним вибором цільової функції є модульність (modularity) і її модифікації, формула (1.1):

$$Q = \frac{1}{2m} \sum_{c \in P} \sum_{i, j \in c} A[i, j] - \Pr(A[i, j] = 1), \quad (1.1)$$

де P – множина спільнот;

A – матриця інцидентності.

Ньюмен в [14] використовує агломеративний підхід у поєднанні з жадібною оптимізацією. Спочатку кожна вершина є кластером, на кожному наступному кроці відбувається злиття кластерів, максимально збільшує показник модулярності.

Модулярність вводиться як характеристика розбиття графа на непересічні компоненти.

1.3.3 Лінійні графи

Метод лінійних графів [15], по суті, полягає в кластеризації ребер. Можна виділити три основні кроки алгоритму:

а) перетворення вихідного графа в лінійний:

- 1) кожне ребро стає вершиною в лінійному графі;
- 2) кожна вершина стає клікою в лінійному графі;

б) розбиття лінійного графа на частини алгоритмом для пошуку спільнот, що не перетинаються;

в) зворотне перетворення отриманих спільнот лінійного графа в спільноти вихідного графа.

Приклад перетворення ділянки графа в лінійний представлений на рис. 1.3.

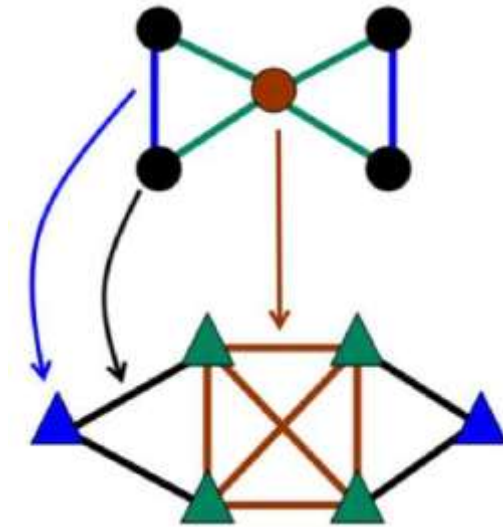


Рисунок 1.3 – Перетворення в лінійний граф

1.3.4 Локальні методи

При локальній роботі з кластерами (не беручи до уваги структуру цілого графа) часто розглядається відношення кількості внутрішніх і зовнішніх ребер або трикутників кластера [16]. Крім того, спільноти можуть складатися з наборів пов'язаних клік різного розміру [17]–[19].

1.3.5 Перколяція клік

Попередньо, слід позначити, що кліка – повний підграф графа, а k -кліка – повний підграф графа, що складається з K вершин.

Метод перколяції клік [18] будує спільноти із суміжних k -клік. Дві k -кліки називаються суміжними, якщо вони містять $k-1$ загальних вершин. Спільнота визначається, як максимальне об'єднання k -клік, будь-які дві з яких з'єднані набором суміжних k -клік. Таке визначення спільноти допускає перетину природним чином.

У покращеному варіанті класичної перколяції, запропонованому в [19], для пошуку спільнот використовується дводольний граф. Вузлами цього графа можуть бути або k -кліки, або $k-1$ -кліки. k -кліка пов'язана з $k-1$ -клікою, лише якщо остання є підкликом цього k -кліки. Таким чином, пов'язані непересічні компоненти цього графа представляють кінцеві спільноти.

На рисунку 1.4 наведено приклад знайденого покриття графа при $k=4$.

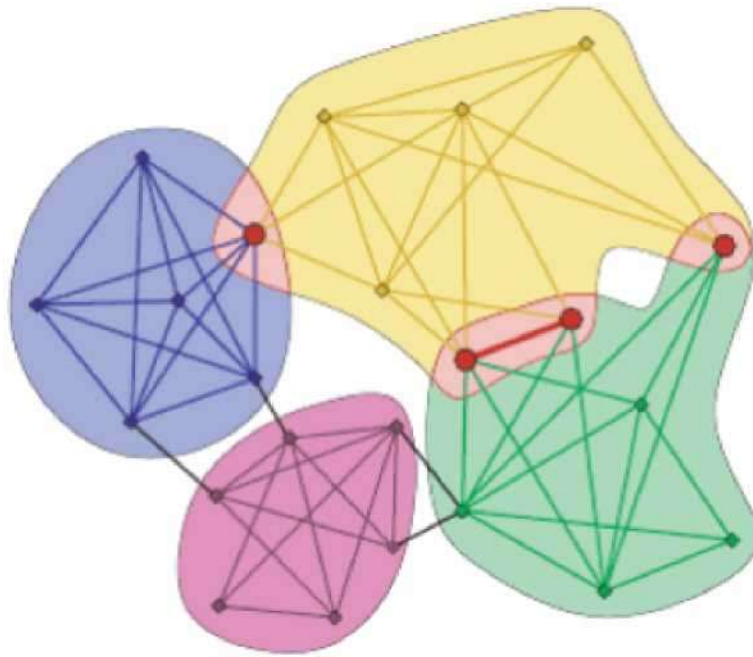


Рисунок 1.4 – Результат роботи методу перколяції клік

1.3.6 Жадібний алгоритм розширення клік

Greedy clique expansion (GCE) – цей локальний метод намагається оптимізувати оціночну функцію для кожної конкретної спільноти. На початку роботи метод вибирає початкові ядра спільнот, як правило, це вершини або трикутники графа. Потім запускається цикл, що складається з наступних кроків:

- вибирається найбільша незавершена спільнота;

– до цієї спільноти послідовно приєднуються вершини-сусіди, що максимізують приріст оціночної функції спільноти, до тих пір, поки цей приріст не стане негативним;

– якщо оціночна функція досягла максимуму, спільнота вважається завершеним, і алгоритм переходить на новий виток.

Одним з головних поліпшень, запропонованих в [17], є вибір максимальних клік в якості ядер спільнот.

Порівняльну характеристику програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння методів для пошуку спільнот користувачів у соціальних мережах

Метод	Масштабованість	Інтерпретованість	Застосовність	Гнучкість	Стійкість
1	2	3	4	5	6
Ієрархічна кластеризація	Від помірнього до низького	Високий	Універсальний, підходить для різних даних	Підтримка ієрархічної структури	Чутливий до шуму в даних
Максимізація цільової функції	Від помірнього до високого	Від помірнього до низького	Ефективний для захоплення глобальної структури	Чутливість до вибору параметрів	Обмежена інтерпретація
Лінійні графіки	Високий	Низький	Виявлення локальних шаблонів	Обмежене охоплення глобальної структури	Стійкий до шуму в локальних шаблонах
Локальні методи	Високий	Високий	Добре підходить для локальних шаблонів	Обмежений глобальний контекст	Стійкий до шуму в локальних шаблонах

Продовження таблиці 1.1

1	2	3	4	5	6
Просочування по клацанню	Від помірного до низького	Від помірного до низького	Ефективний для ідентифікації щільних областей	Обмежена гнучкість критеріїв	Чутливий до вибору параметрів
Алгоритм розширення по жадібному клацання	Високий	Помірний	Масштабується для великих графіків	Залежить від показника якості	Стійкий до шуму в локальних шаблонах

У цій таблиці розглядаються такі критерії, як:

- масштабованість – наскільки добре метод працює на великих графіках;
- інтерпретованість – простота розуміння результатів;
- застосовність – придатність для різних типів даних;
- гнучкість – адаптованість до різних сценаріїв або критеріїв;
- стійкість – стійкість до шуму в даних або варіацій параметрів.

Слід мати на увазі, що ефективність методу кластеризації може залежати від конкретних характеристик даних та цілей кластеризації.

Як видно з порівняння більшість методів не відрізняються достатньою стійкістю та надійністю. Більш того більшість методів відрізняється високою залежністю від початкових налаштувань параметрів. Отже в майбутньому є актуальною задачею розробки нового методу, що являє собою поєднання стратегій одразу декількох класичних методів з метою усунення їх слабких сторін.

1.4 Постановка задачі

Метою роботи є дослідження та розробка розподілених методів для пошуку спільнот користувачів у соціальних мережах. Умовно завдання можна розбити на кілька етапів:

- провести аналіз і порівняння існуючих методів пошуку спільнот на графах;
- вибрати найбільш ефективні і багатообіцяючі з них;
- керуючись стратегіями, реалізованими в обраних методах, створити комбінований підхід та реалізувати його в форматі програмного засобу (ПЗ) на базі платформи для розподілених обчислень Apache Hadoop, що здійснює пошук спільнот користувачів в соціальних графах;
- провести тестування і оцінку продуктивності розробленого рішення.

1.5 Висновки за розділом 1

Визначено, що пошук спільнот користувачів у соціальних мережах є важливою задачею та широко використовується у різних сферах людської діяльності.

Проаналізовано основні методи, що реалізують вирішення цієї задачі. Серед усіх методів є недоліки.

Тому актуальною є задача розробки нового методу, що являє собою поєднання стратегій одразу декількох класичних методів з метою усунення їх слабких сторін.

Сформульовано перелік функціональних вимог до ПЗ.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Використання метрики Normalized Mutual Information

Для порівняння розбиттів графа запропоновано чималу кількість різних заходів, але більшість з них підходить для роботи лише з непересічними розбиттями. У даній роботі для оцінки описаних вище методів використовується метрика Normalized Mutual Information (NMI) [20], яка підходить для оцінки пересічних розбиттів (покриттів), формула (2.1):

$$NMI(X, Y) = \frac{2(H(X) + H(Y) - H(X, Y))}{H(X) + H(Y)}, \quad (2.1)$$

де X та Y – розбиття графу на компоненти, що перетинаються;

H – ентропія Шеннона.

В якості тестових графів з пересічними компонентами були використані графи, згенеровані модифікованим алгоритмом LFR [21]. На вхід генератору передаються наступні параметри:

- N – кількість вершин;
- k – середнє значення ступеня вершини;
- $k_{\max}$ – максимальне значення ступеня вершини;
- $C_{\min}$ – мінімальна кількість вершин у спільноті;
- $C_{\max}$ – максимальна кількість вершин у спільноті;
- τ_1 – експонента ступеневого розподілу ступеня вершини;
- τ_2 – експонента ступеневого розподілу розміру кластера;
- μ – параметр вираженості кластерної структури;
- on – кількість вершин, що належать більш ніж одній спільноті;
- om – скільком спільнотам належить кожна вершина з області перетину.

Було створено два набори тестових графіків. Члени першого набору відрізняються значенням om , другого – op .

На рис. 2.1 представлені графіки з отриманими результатами.

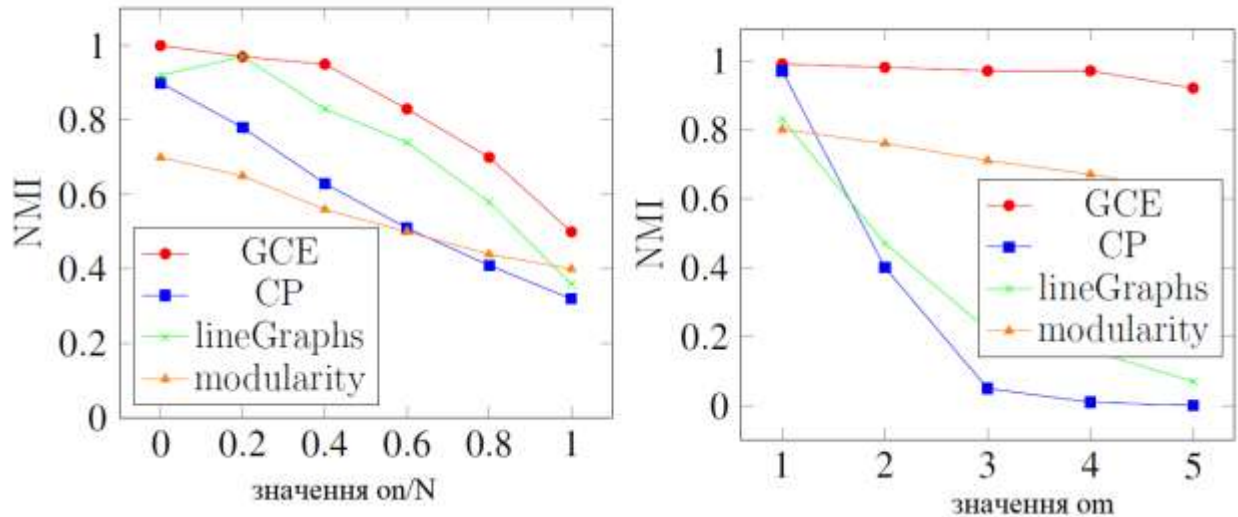


Рисунок 2.1 – Порівняння методів

Як видно з графіків, найбільш ефективним алгоритмом виявився жадібний алгоритм розширення клік – Greedy clique expansion (GCE), представлений в [17]. Крім цього, метод є локальним, що збільшує шанси отримати вдалу реалізацію на платформі Apache Hadoop.

2.2 Модель MapReduce

Для реалізації розподіленого алгоритму пошуку спільнот користувачів була використана модель розподілених обчислень MapReduce, а саме, її вільна реалізація - платформа Hadoop від Apache. Елементарна обчислювальна задача в даній моделі складається з двох кроків: Map і Reduce. Обчислювальні елементи, що реалізують ці кроки, називаються, відповідно, маппер (mapper) і редьюсер (reducer). Одночасно над одним завданням може працювати велика кількість мапперів і редьюсерів розподілених по машинах кластера. Дані для

обробки за допомогою MapReduce повинні бути представлені у форматі ключ-значення $\langle k, v \rangle$.

Весь обсяг вхідних даних розбивається на фрагменти певного розміру, кожен такий фрагмент надходить на вхід одному з мапперів.

Вхідна пара $\langle k_{in}, v_{in} \rangle$ деяким чином перетворюється в проміжну пару $\langle k_{int}, v_{int} \rangle$. Потім проміжні дані, отримані з усіх мапперів, деяким чином групуються по ключу k_{int} і надходять на вхід редьюсерам у вигляді $\langle k_{int}, v_{int}^i \rangle$. Таким чином значення, відповідні одному ключу, потрапляють в один редьюсер. Після остаточного обробки на виході редьюсера отримуємо пари $\langle k_{out}, v_{out} \rangle$, які вже записуються в вихідний файл. Схематично роботу MapReduce-завдання (далі MR-завдання) можна представити рис. 2.2.

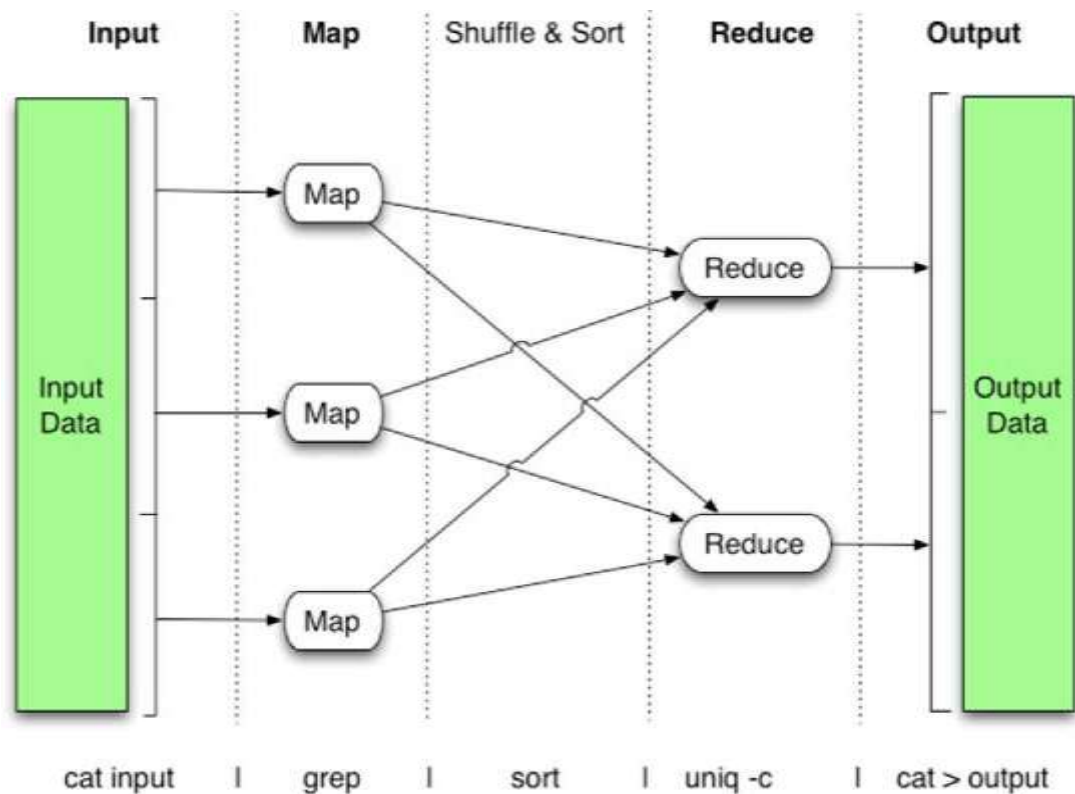


Рисунок 2.2 – Схема роботи MapReduce

2.3 Побудова нового підходу

На основі результатів дослідження, наведених в попередніх розділах, жадібний алгоритм розширення клік був обраний в якості основи для побудови розподіленого рішення. Розглянемо основні кроки цього алгоритму детально.

2.3.1 Алгоритм пошуку максимальних клік в графі

Однією з ключових частин алгоритму є пошук максимальних клік в графі. Незважаючи на те, що це завдання є обчислювально складною, кліки можуть бути знайдені досить швидко в розріджених графах, якими і є соціальні мережі. Автори методу GCE в [17] пропонують використовувати для цього завдання алгоритм Брона-Кербоша [22], як один з найефективніших. Томіта та інші у [23] показали, що в гіршому випадку складність алгоритму $O\left(3^{n/3}\right)$, де n – кількість вершин в графі.

Нижче представлений псевдокод модифікованого алгоритму Брона-Кербоша (рис. 2.3), який буде використаний в подальшому. *clique* – множина, що містить на кожному кроці кліку (не обов'язково максимальну), відповідну цьому кроку, *CAND* – множина вершин-кандидатів на потрапляння в *clique*, *FINI* – безліч вершин, які вже використовувалися для розширення *clique* на попередніх ітераціях циклу, $\Gamma(v)$ – вершини-сусіди для v .

 Алгоритм 1 Пошук максимальних клік

```

method dfs(CAND, clique)
  FINI = {}
  while CAND \ FINI != ∅ do
    v = argmax(|Γ(v) ∩ CAND \ FINI|)
    clique ∪= {v}
    CAND' = Γ(v) ∩ CAND \ FINI
    if CAND' = ∅ then
      emit clique
    else
      dfs(CAND', clique)

  CAND \= {v}
  FINI ∪= CAND'
  clique \= {v}

```

Рисунок 2.3 – Псевдокод модифікованого алгоритму Брона-Кербоша

На рисунках 2.4 та 2.5 представлені приклад графа для пошуку клік і схема роботи алгоритму на цьому графі

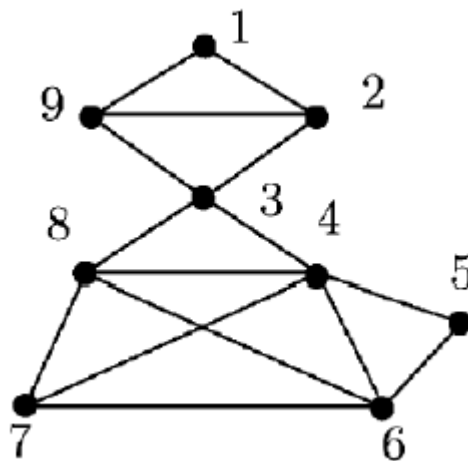


Рисунок 2.4 – Приклад графа для пошуку максимальних клік

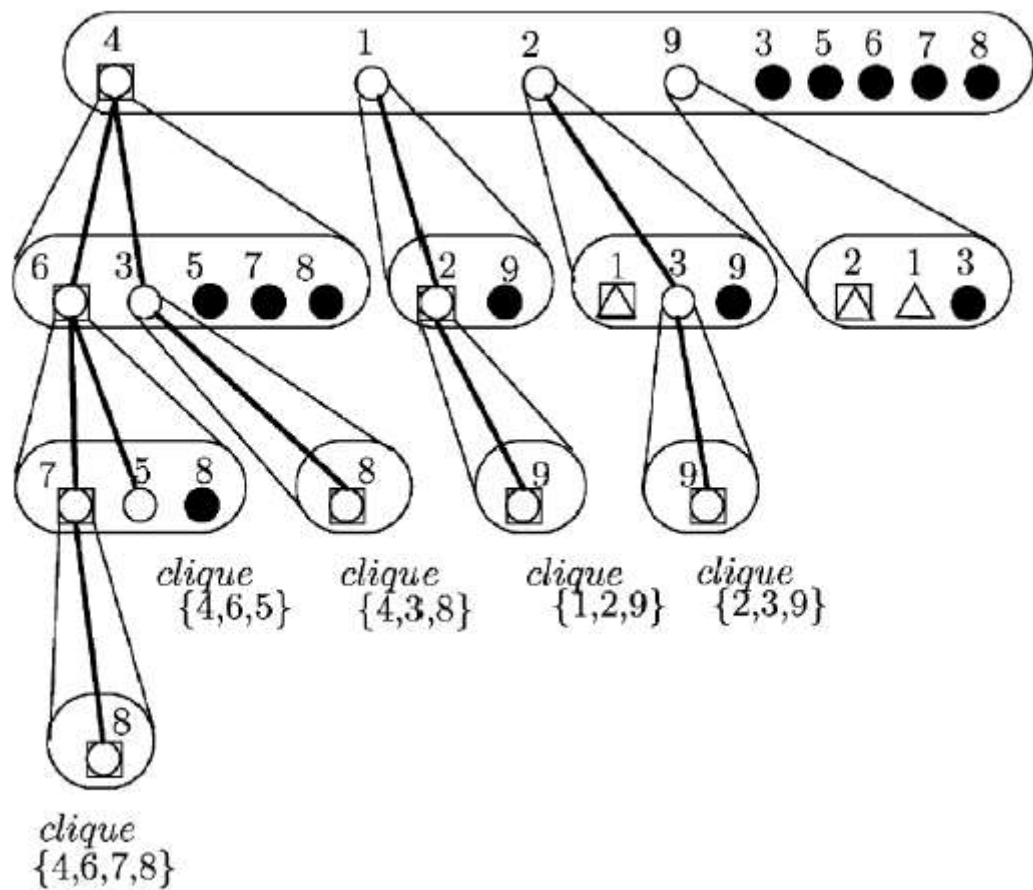


Рисунок 2.5 – Схема роботи алгоритму пошуку максимальних клік

Квадратом відзначені вершини, додані в кліку на відповідних кроках. Чорними колами відзначені вершини, складові *FINI* на відповідному кроці.

2.3.2 Розширення клік

В якості оціночної функції спільноти формула (2.2), значення якої оптимізує алгоритм, візьмемо локальну функцію адекватності спільноти S :

$$F_S = \frac{k_{in}^S}{(k_{in}^S + k_{out}^S)^\alpha}, \quad (2.2)$$

де k_{in}^S – внутрішній ступінь спільноти S ;

k_{out}^S – зовнішній ступінь спільноти S ;

α – параметр.

Інакше кажучи, k_{in}^S – це подвоєне число ребер, обидві вершини яких знаходяться в S (сума ступенів всіх вершин в S), а k_{out}^S – число ребер, тільки одна вершина яких належить S .

Розширення клік вимагає знання сусідів спільноти (назвемо безліч сусідів спільноти S фронтом $f(S)$ (s)). На кожному наступному кроці вибирається вершина v_{max} з фронту, що максимально збільшує значення адекватності для спільноти, і додається в цю спільноту. Фронт змінюється наступним чином, формула (2.3):

$$f(S \cup v_{max}) = (f(S) \cup \Gamma(v_{max}) \setminus S) - \{v_{max}\}, \quad (2.3)$$

де $\Gamma(v_{max})$ – множина сусідів вершини v_{max} . При кожному оновленні фронту значення k_{in}^S формула (2.4) і k_{out}^S формула (2.5) також оновлюються:

$$k_{in}^{S \cup v_{max}} = k_{in}^S + 2|S \cap \Gamma(v_{max})|, \quad (2.4)$$

$$k_{out}^{S \cup v_{max}} = k_{out}^S + |\Gamma(v_{max})| - 2|S \cap \Gamma(v_{max})|. \quad (2.5)$$

2.3.3 Відсів потенційних дублікатів

При розширенні клік не виключені випадки, коли спочатку різні кліки розширюються в одне і те ж співтовариство. Щоб повторно не витратити ресурси на вже виконану роботу, необхідно відсівати такі дублікати якомога раніше. Для цього після кожного розширення кліки перевіряється, чи не стала вона занадто схожа на одну з уже завершених клік. В якості метрики використовується функція відстані між спільнотами формула (2.6):

$$\delta_E(S, S') = 1 - \frac{|S \cap S'|}{\min(|S|, |S'|)}. \quad (2.6)$$

Відсоток вершин меншої спільноти, не включених до більшої спільноти. Тоді потенційними дублікатами S будуть спільноти, що знаходяться на відстані не далі ε від нього.

2.3.4 Модель даних

Структура Node для вершини складається з двох полів:

- `nid` – ідентифікатор вершини, ціле число, яким вершина представлена в вихідних даних;
- `adjacencyList` – список суміжності вершини, ідентифікатори всіх сусідів.

Структура Community для спільноти наступна:

- `finished` – мітка завершеності спільноти, має значення `true`, якщо спільнота максимально розширено;
- `kIn` – внутрішній ступінь спільноти;
- `kOut` – зовнішня ступінь спільноти;
- `nodes` – багато ідентифікаторів вершин, що входять до спільноти;
- `frontier` – багато ідентифікаторів вершин, що складають фронт спільноти.

Властивості `kIn` і `kOut` введені в структуру для більш зручного обчислення приросту значення адекватності для конкретної вершини-кандидата.

2.4 Вибір мови програмування

Хоча Java традиційно не вважається основною мовою data science застосунків, вона має певні функції та переваги, які можуть зробити її відповідним вибором для конкретних аспектів проєктів науки про дані. Однак важливо розуміти, що інші мови, такі як Python та R, частіше асоціюються з наукою про дані через їх багату екосистему бібліотек та інструментів, спеціально розроблених для цієї галузі. Однак давайте розглянемо, чому Java можна вважати хорошою мовою для певних аспектів наукових проєктів даних [24], [25].

Масштабованість та продуктивність. Однією з головних сильних сторін Java є її масштабованість та продуктивність. Програми Java відомі своєю ефективною роботою, обробкою великих наборів даних та високою продуктивністю. Це має вирішальне значення для проєктів науки про дані, які передбачають обробку величезних обсягів даних або вимагають аналізу в режимі реального часу [25].

Корпоративна інтеграція. Java широко використовується в корпоративних середовищах, і багато проєктів з науки про дані є частиною більших систем. Сумісність Java з корпоративними системами та її здатність легко інтегруватися з базами даних, системами обміну повідомленнями та іншими компонентами роблять її привабливим вибором для проєктів науки про дані в корпоративних умовах [25].

Зріла екосистема. Java має зрілу та розгалужену екосистему бібліотек та фреймворків. Хоча в ньому може бути не так багато спеціалізованих бібліотек для аналізу даних, як у Python або R, він пропонує такі бібліотеки, як Apache Spark та Apache Flink, які є потужними інструментами для обробки великих даних та аналітики. Ці бібліотеки можна використовувати для широкомасштабної обробки та аналізу даних [25].

Багатопоточність і паралелізм. Java має чудову підтримку багатопоточності та паралельності. Ця функція корисна для завдань обробки

даних, які передбачають паралельну обробку або паралельне виконання алгоритмів. Ефективна обробка паралельних завдань може призвести до підвищення продуктивності та скорочення часу виконання [25].

Суворі типізація та об'єктно-орієнтована структура. Суворі типізація Java та парадигма об'єктно-орієнтованого програмування (ООП) забезпечують структурований та організований спосіб розробки коду. Це може бути корисним для складних проєктів науки про дані, де важливо підтримувати якість коду, масштабованість та можливість повторного використання [25].

Кросплатформна сумісність. Філософія Java – напиши один раз, запусти в будь-якому місці – робить його кросплатформною мовою. Це вигідно при розгортанні моделей або додатків для аналізу даних у різних операційних системах. Програми Java можуть запускатися на різних платформах без змін, забезпечуючи гнучкість при розгортанні [25].

Безпека. Java приділяє велику увагу безпеці. У науці про дані, де обробка конфіденційної інформації є звичайною справою, надійні функції безпеки Java можуть мати вирішальне значення. Віртуальна машина Java (JVM) забезпечує безпечне середовище виконання, а програми Java відомі своєю стійкістю до загроз безпеці [25].

Набір інструментів та підтримка інтегрованого середовища розробки (Integrated development environment – IDE) . Переваги Java полягають у широкому спектрі IDE та інструментів. Популярні IDE, такі як Eclipse та IntelliJ IDEA, надають розширені можливості для розробки, налагодження та профілювання коду. Ці інструменти можуть підвищити продуктивність науковців даних та розробників, які працюють над проєктами на базі Java [25].

Спільнота та довгострокова підтримка. Java має велику та активну спільноту розробників. Наявність ресурсів, форумів та підтримки спільноти може бути корисною для науковців даних, які використовують Java. Крім того, Java має історію довгострокової підтримки з боку Oracle та спільноти, що забезпечує стабільність та обслуговування програм Java з часом [25].

Впровадження в освіту та промисловість. Java часто викладається в програмах інформатики та інженерії, і багато професіоналів галузі знайомі з цією мовою. Для організацій, які мають досвід роботи з Java, вибір Java для проєктів науки про дані може полегшити співпрацю та передачу знань між членами команди [25].

Хоча Java має ці переваги, важливо визнати, що інші мови, такі як Python та R, стали фактичними стандартами науки про дані завдяки своїм широким бібліотекам, фреймворкам та цілеспрямованій підтримці спільноти дослідників даних. Зокрема, Python завоював популярність завдяки своїй простоті, читабельності та доступності спеціалізованих бібліотек, таких як NumPy, pandas та scikit-learn [25].

Таким чином, хоча Java може бути не найкращим вибором для всіх аспектів науки про дані, вона може бути підходящим варіантом для конкретних сценаріїв, особливо коли продуктивність, масштабованість та інтеграція з існуючими корпоративними системами мають першорядне значення. Фахівці з обробки даних і розробники повинні ретельно оцінити вимоги своїх проєктів і розглянути сильні і слабкі сторони різних мов програмування, перш ніж приймати рішення. Крім того, зростаючий ландшафт поліглотської науки про дані, коли в рамках проєкту для різних цілей використовується кілька мов, забезпечує гнучкість у виборі відповідного інструменту для кожного завдання [25].

Обґрунтування вибору мови програмування для розробки програмного забезпечення наведено у таблиці 2.1.

Отже, для реалізації програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах обрано мову програмування Java, яка є безкоштовною та має перспективи для розробки застосунків у галузі data science. Також Java відрізняється більш оптимізованим використанням ресурсів та великою кількістю додаткових модулів, бібліотек та фреймворків.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	Kotlin
Об'єктно-орієнтованість	Так	Так (із нововведеннями)	Так
Бібліотеки та фрейворки для data science	Так, багато	Так, багато	Так, мало
Активність спільноти	Активна	Активна	Активна
Продуктивність застосунків	Висока	Середня	Висока
Масштабованість	Висока	Середня	Середня

2.5 Фреймворк Hadoop

Hadoop – це потужна і широко використовувана платформа з відкритим кодом, яка полегшує розподілене зберігання та обробку великих наборів даних на кластерах комп'ютерів. Хоча це не мова програмування, як Python або Java, Hadoop служить важливим інструментом в екосистемі науки про дані, особливо для великих додатків даних. Розглянемо, чому Hadoop вважається цінним інструментом для наукових проєктів даних [26], [27].

Масштабованість. Однією з ключових сильних сторін Hadoop є його масштабованість. Він може ефективно обробляти масивні набори даних, які перевищують можливості однієї машини. Розподіляючи дані та обчислення по кластеру вузлів, Hadoop дозволяє обробляти петабайти даних, що робить його придатним для проєктів науки про дані, що включають масштабну аналітику.

Розподілене сховище. Розподілена файлова система Hadoop (HDFS) забезпечує архітектуру розподіленого сховища, яка дозволяє зберігати дані на декількох вузлах у кластері. Ця модель розподіленого сховища забезпечує

відмовостійкість і високу доступність. Дані реплікуються між вузлами, мінімізуючи ризик втрати даних у разі апаратних збоїв [26], [27].

Модель програмування MapReduce. Hadoop використовує модель програмування MapReduce, яка добре підходить для паралельної обробки великих наборів даних. Ця модель розбиває обробку даних на два етапи: етап зіставлення, коли дані обробляються паралельно між вузлами, і етап скорочення, коли результати агрегуються. Модель MapReduce спрощує розробку додатків для розподіленої обробки даних [26], [27].

Підтримка різних типів даних. Hadoop призначений для обробки різних типів даних, включаючи структуровані та неструктуровані дані. Ця гнучкість робить його придатним для проєктів науки про дані, які передбачають обробку різних типів даних, таких як текст, зображення та файли журналів [26], [27].

Економічне сховище. Hadoop надає економічне рішення для зберігання великих обсягів даних. Його розподілена система зберігання дозволяє організаціям масштабувати свої потреби в зберіганні, додаючи більше вузлів до кластера, а використання стандартного обладнання допомагає знизити витрати порівняно з традиційними рішеннями для зберігання [26], [27].

Екосистема інструментів. Hadoop має багату екосистему інструментів, які розширюють його функціональність для різних завдань обробки даних. Apache Hive, Apache Pig та Apache Spark-це приклади інструментів, створених поверх Hadoop, які полегшують обробку та аналіз даних. Ці інструменти забезпечують Абстракції та інтерфейси вищого рівня, полегшуючи науковцям даних роботу з великими даними [26], [27].

Паралельна обробка. Розподілений характер Hadoop забезпечує паралельну обробку даних. Можливість розділення завдань на менші підзадачі, які можуть виконуватися одночасно на декількох вузлах кластера, призводить до прискорення обробки даних. Цей паралелізм має вирішальне значення для обробки обчислювальних вимог алгоритмів data science до великих наборів даних [26], [27].

Відмовостійкість. Hadoop спроектований так, щоб бути відмовостійким. У разі збою вузла дані все ще доступні з інших вузлів, де вони реплікуються. Платформа автоматично перерозподіляє завдання на справні вузли, гарантуючи, що обробка даних триває без збоїв. Така стійкість необхідна для підтримки цілісності робочих процесів data science.

Підтримка неструктурованих даних. Здатність Hadoop обробляти неструктуровані дані, такі як текстові документи, зображення та відео, робить його універсальним інструментом для проєктів науки про дані, що працюють з різними джерелами даних. Це особливо актуально в сучасному інформаційному ландшафті, де неструктуровані дані стають все більш поширеними [26], [27].

Впровадження спільнотою та галуззю. Hadoop має велику та активну спільноту розробників і набув широкого поширення в різних галузях промисловості. Це означає, що науковцям даних та розробникам, які працюють з Hadoop, доступно багато ресурсів, документації та підтримки спільноти. Широке впровадження також означає його надійність та ефективність у реальних додатках [26], [27].

Підтримка сховищ даних. Hadoop може служити основою для рішень для зберігання даних. Поєднання Hadoop з такими інструментами, як Apache Hive, дозволяє організаціям створювати сховища даних, які можуть обробляти як Історичні дані, так і дані в режимі реального часу, забезпечуючи комплексне рішення для аналітичної обробки [26], [27].

Інтеграція з існуючими системами. Hadoop призначений для інтеграції з існуючими системами, базами даних та сховищами даних. Це робить його підходящим вибором для організацій, які прагнуть скористатися можливостями великих даних без повного відновлення існуючої інфраструктури. Безшовна інтеграція полегшує впровадження Hadoop в різних галузях і доменах [26], [27].

Машинне навчання за допомогою Apache Spark. Хоча спочатку Apache Spark не був частиною Hadoop, він часто інтегрується з Hadoop для

ітеративної обробки та обробки в пам'яті. Spark MLlib, бібліотека машинного навчання для Spark, дозволяє науковцям даних впроваджувати та масштабувати алгоритми машинного навчання на великих наборах даних, надаючи потужний набір інструментів для завдань обробки даних [26], [27].

Відповідність нормативним вимогам та безпека. Hadoop надає функції для управління даними, контролю доступу та шифрування, що робить його придатним для організацій із суворими вимогами до дотримання вимог. Функції безпеки Hadoop, включаючи механізми аутентифікації та авторизації, сприяють його надійності при обробці конфіденційних даних [26], [27].

Обробка даних у режимі реального часу за допомогою Apache Flink. На додаток до пакетної обробки за допомогою MapReduce, екосистема Hadoop включає Apache Flink, платформу потокової обробки. Flink забезпечує обробку даних в режимі реального часу, дозволяючи аналізувати дані в міру їх надходження і реагувати на них. Це надзвичайно важливо для наукових проєктів даних, які потребують аналізу в режимі реального часу [26], [27].

2.6 Вибір середовища та інших засобів розробки для створення програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах

Eclipse – це широко використовуване IDE для розробки Java, і воно завоювало популярність серед розробників завдяки багатому набору функцій, розширюваності та надійній підтримці проєктів Java [28].

З відкритим кодом і безкоштовно. Eclipse – це середовище розробки з відкритим кодом, що означає, що воно є у вільному доступі для розробників. Така відкритість заохочує велику спільноту учасників, що призводить до постійних удосконалень, оновлень та безлічі доступних плагінів. Вільний характер Eclipse з відкритим вихідним кодом робить його доступним для розробників незалежно від їх бюджетних обмежень [28].

Багатий набір функцій. Eclipse надає повний набір функцій, які охоплюють весь життєвий цикл розробки програмного забезпечення. Від редагування та налагодження коду до контролю версій та автоматизації збірки, Eclipse пропонує повний набір інструментів в одному середовищі. Така повнота спрощує процес розробки, дозволяючи розробникам виконувати різні завдання, не перемикаючись між різними інструментами [28].

Кросплатформна сумісність. Eclipse розроблений як кросплатформний, що підтримує основні операційні системи, такі як Windows, macOS та Linux. Така сумісність між платформами забезпечує рівномірний досвід розробки для розробників, які працюють з різними операційними системами, полегшуючи співпрацю в різних середовищах розробки [28].

Розширюваність за допомогою плагінів. Архітектура Eclipse побудована навколо концепції плагінів, які є модульними компонентами, що розширюють функціональність IDE. Ця розширюваність дозволяє розробникам налаштовувати своє середовище розробки відповідно до конкретних вимог проєкту. Eclipse Marketplace надає широке сховище плагінів для різних мов програмування, фреймворків та інструментів [28].

Інструменти розробки Java. Eclipse особливо відомий своєю надійною підтримкою розробки Java за допомогою плагіна Java Development Tools (JDT). JDT пропонує такі функції, як інтелектуальне завершення коду, підсвічування синтаксису, засоби налагодження та управління проєктами. Безшовна інтеграція JDT робить Eclipse потужним середовищем розробки для розробників Java [28].

Допомога в написанні коду та рефакторинг. Eclipse надає розширені функції підтримки коду, включаючи інтелектуальну підтримку вмісту, шаблони коду та швидкі виправлення. Ці функції підвищують продуктивність розробників за рахунок скорочення часу, витраченого на виконання рутинних завдань з кодування. Eclipse також підтримує автоматичний рефакторинг коду, дозволяючи розробникам поліпшити зручність супроводу і читання коду без ручних зусиль [28].

Вбудований налагоджувач. Eclipse включає вбудований налагоджувач, який підтримує як локальну, так і віддалену налагодження. Розробники можуть встановлювати точки зупинки, перевіряти змінні та виконувати код поетапно для ефективного виявлення та усунення проблем. Інтеграція налагоджувача з інтерфейсом IDE робить процес налагодження інтуїтивно зрозумілим і зручним для користувача [28].

Інтегрована збірка та контроль версій. Eclipse легко інтегрується з інструментами побудови, такими як Apache Maven та Apache Ant. Ця інтеграція спрощує процес побудови, дозволяючи розробникам компілювати, упаковувати та розгортати проєкти безпосередньо з IDE. Eclipse також забезпечує підтримку систем контролю версій, таких як Git, забезпечуючи спільну розробку з ефективним управлінням вихідним кодом [28].

Розробники графічного інтерфейсу користувача (Graphical user interface – GUI). Eclipse включає засоби створення графічного інтерфейсу користувача (GUI), такі як плагін WindowBuilder, який полегшує проєктування та розробку графічних інтерфейсів. Ці інструменти дозволяють розробникам візуально створювати Додатки Java Swing і SWT (Standard Widget Toolkit), заощаджуючи час і зусилля при розробці графічного інтерфейсу [28].

Відстеження завдань і проблем. Eclipse підтримує відстеження завдань і проблем за допомогою плагінів, таких як Mylyn. Mylyn дозволяє розробникам інтегруватися із системами відстеження проблем, керувати завданнями та безпосередньо пов'язувати їх із відповідним вихідним кодом. Ця інтеграція спрощує управління проєктами та розширює співпрацю, надаючи централізований перегляд завдань та пов'язаних з ними змін коду [28].

Підтримка спільноти та документація. Eclipse має динамічну та активну спільноту розробників. Підтримка спільноти сприяє доступності великої документації, посібників та форумів. Розробники можуть легко знаходити рішення поширених проблем, ділитися знаннями та звертатися за допомогою до спільноти Eclipse, створюючи середовище співпраці та підтримки.

Постійні оновлення та технічне обслуговування. Eclipse постійно розробляється і підтримується, регулярно випускаються релізи, що включають нові функції, поліпшення і виправлення помилок. Ця відданість безперервній розробці гарантує, що Eclipse залишається передовим і надійним середовищем розробки для Java-проєктів [28].

Інтеграція з іншими технологіями. Eclipse підтримує широкий спектр технологій та фреймворків, крім Java. У ньому є плагіни та інструменти для таких мов, як C/C++, Python, PHP та інших. Крім того, Eclipse підтримує різні фреймворки додатків, інструменти веброботи та хмарні платформи, що робить його універсальним вибором для розробників, які працюють над різноманітними проєктами [28].

Простота навігації та вивчення коду. Eclipse надає функції для зручної навігації та вивчення коду, такі як представлення структури, де відображається структура поточного файлу, та подання ієрархії викликів, де відображаються методи виклику та виклику. Ці функції допомагають розробникам зрозуміти складні кодові бази та орієнтуватися на великі проєкти.

Єдине середовище розробки. Eclipse прагне забезпечити єдине середовище розробки, яке охоплює всі аспекти розробки програмного забезпечення. Інтеграція різних інструментів та плагінів в рамках єдиної IDE мінімізує час навчання розробників, оскільки вони можуть використовувати знайомі інтерфейси для багатьох завдань [28].

На закінчення, Eclipse виділяється як надійний та універсальний IDE для проєктів Java завдяки своїй природі з відкритим кодом, багатому набору функцій, розширюваності та потужній підтримці розробки Java. Його безперервна розробка, активна спільнота та сумісність між платформами сприяють її широкому впровадженню в спільноті розробників Java. Хоча існують інші IDE для розробки Java, поєднання функцій та гнучкості Eclipse робить його привабливим вибором для розробників, які працюють над різноманітними проєктами Java [28].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах

Критерій порівняння середовищ розробки	Середовище розробки		
	Eclipse	NetBeans	IntelliJ IDEA
Комерційність	Безкоштовне	Безкоштовне	На платній основі
Підтримка Hadoop	Так	Відсутня	Так
Рефакторинг коду	Вбудовані інструменти	Вбудовані інструменти	Вбудовані інструменти
Розширюваність модулями	Великий набір сторонніх модулів	Середній набір сторонніх модулів	Тільки власні модулі
Інтегрована збірка та контроль версій	Так	Так	Так

Таким чином, для створення програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах обрано середовище розробки Eclipse, що є безкоштовним. Також надважливим була підтримка середовищем фреймворку Hadoop – через велику кількість сторонніх модулів Eclipse можна повністю адаптувати для роботи з Hadoop.

2.7 Висновки за розділом 2

Для реалізації програмного забезпечення для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах обрано мову програмування Java та фреймворк Hadoop.

Для створення програмного забезпечення для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах обрано середовище розробки Eclipse, що постачається з багатьма модулями, пакетами та інструментами.

3 ОПИС ПРОГРАМИ

В якості платформи для розподілених обчислень в рамках моделі MapReduce була обрана платформа Apache Hadoop, як найстабільніша відкрита реалізація на поточний момент.

Основною мовою програмування була обрана мова Java. Крім того, що він є високорівневим і має досить високу продуктивність, мова Java є рідною мовою програмування для платформи Apache Hadoop і, як наслідок, дозволяє писати найбільш ефективні MR-програми для цієї платформи.

Додатковою мовою для різного роду прототипів і допоміжних скриптів був обраний Python. Ця мова дозволяє писати код дуже лаконічно і швидко, а також має бібліотеки на всі випадки життя, що і було потрібно при прототипуванні і створенні скриптів для супутніх завдань, наприклад, візуалізації графів.

3.1 Загальна схема роботи

Роботу нового комбінованого підходу можна розділити на кілька етапів:

- а) конвертація вхідних даних з текстового формату у внутрішній бінарний формат;
- б) пошук максимальних клік в графі;
- в) основний етап розширення клік:
 - 1) розширення клік;
 - 2) видалення потенційних дублікатів;
 - 3) якщо ще є незавершені кліки, то починаємо наступну ітерацію, в іншому випадку переходимо до останнього етапу;
- г) конвертація отриманих спільнот з внутрішнього бінарного уявлення в сприймається людиною текстове.

На кожному етапі запускається одна або більше MR-завдання. Розглянемо детально ці етапи методу.

3.1.1 Конвертація вхідних даних

Алгоритм приймає вхідний графік у вигляді пар пов'язаних вершин $\langle vertex1, vertex2 \rangle$, групує списки суміжності для кожної вершини та зберігає перетворений графік у HDFS.

3.1.2 Пошук максимальних клік в графі

Для пошуку максимальних клік модифікуємо алгоритм запропонований в [24]. Спершу ми повинні побудувати для кожної вершини v сусідства другого рівня (сусіди сусідів). На кроці Map для кожної вершини v з $\Gamma(v)$ ми видаємо трійку $\langle u, \langle v, \Gamma(v) \rangle \rangle$. На кроці Reduce ми отримуємо всі значення $\langle v, \Gamma(v) \rangle$, згруповані по u . Таким чином для u ми знаємо всіх сусідів v і сусідів, що є сусідами $\Gamma(v)$, це і буде сусідство другого рівня. Тепер уже в рамках однієї машини ми шукаємо всі максимальні кліки в отриманому сусідстві модифікованим алгоритмом Брона-Кербоша, представленому алгоритмом 1. В результаті ми знайдемо всі кліки в графі, що містять вершину u нашого сусідства. Щоб уникнути дублікатів при пошуку клік, під час роботи алгоритму Брона-Кербоша ми будемо розглядати тільки такі вершини v , що $v < u$.

Код MR-задачі другого етапу.

На рисунку 3.1 представлений псевдокод алгоритму Mapper.

Алгоритм 2 Mapper

```
method map(nid n, Node N)
    emit n, N
    for v ∈ N.adjacencyList do
        emit v.nid, N
```

Рисунок 3.1 – Псевдокод алгоритму Mapper

На рисунку 3.2 представлений псевдокод алгоритму Reducer.

 Алгоритм 3 Reducer

```
method reduce(nid n, [N1, N2, ...])
  cand = [N1, N2, ...]
  dfs(cand, {n})
```

Рисунок 3.2 – Псевдокод алгоритму Reducer

3.1.3 Розширення клік

Етап розширення клік складається з двох MR-завдань: CountFitness та Find-MaxFitness. Перша розраховує для кожної спільноти приріст значення адекватності при додаванні однієї з вершин фронту. Друга знаходить максимум серед усіх значень і додає відповідну вершину до спільноти.

На вхід CountFitness подаються поточні спільноти і структура графа. Маппер для кожної вершини nid фронту спільноти S віддає пару $\langle nid, S \rangle$. Редьюсер разом зі групованими по nid спільнотами отримує і саму вершину v з ідентифікатором nid . Для кожної спільноти обчислюється значення адекватності $F_{S \cup v}$ і віддається трійка $\langle S, \langle F_{S \cup v} \rangle \rangle$.

FindMaxFitness в якості маппера використовує IdentityMapper. У ред'юсері ж визначається максимальне значення $F_{S \cup v}$ по всіх парах $\langle F_{S \cup v}, v \rangle$, і відповідна вершина v додається до спільноти S . Оновлене співтовариство подається на вихід.

Етап завершується, якщо після чергового витка циклу не залишилося більше незавершених спільнот. Інакше запускається пара завдань CountFitness і FindMaxFitness для оновлених спільнот.

Код MR-задач третього етапу.

На рисунку 3.3 представлений псевдокод CountFitnessMapper.

Алгоритм 4 CountFitnessMapper

```

method map(nid n, V)
  if isNode(V) then
    emit n, V
  else
    for node ∈ V.frontier do
      emit node, V

```

Рисунок 3.3 – Псевдокод CountFitnessMapper

3.1.4 Видалення потенційних дублікатів

Цей етап поділений на дві MR-завдання: GetAdjacent і RejectDuplicated. На першому кроці визначаються групи спільнот, що перетинаються, щодо деяких точок перетину. Слідом виявляються незавершені спільноти, слабо відрізняються від суміжних з ними завершених. Такі спільноти видаляються.

На рисунку 3.4 представлені псевдокод для CountFitnessReducer та FindMaxFitnessReducer.

Алгоритм 5 CountFitnessReducer

```

method reduce(nid n, [V1, V2, ...])
  N = null
  communities = []
  for item ∈ [V1, V2, ...] do
    if isNode(item) then
      N = item
    else
      communities += item
  for item ∈ communities do
    emit item, <item.getFitness(N), N>

```

Алгоритм 6 FindMaxFitnessReducer

```

method reduce(Community C, [<double ftns, Node N>, ...])
  node = maxftns [<double ftns, Node N>, ...]
  C.expand(node)
  emit node.nid, C

```

Рисунок 3.4 – Псевдокод для CountFitnessReducer та FindMaxFitnessReducer

Get Adjacent на вхід приймає поточні спільноти. Маппер для кожної вершини nid спільноти S віддає пару $\langle nid, S \rangle$. Ред'юсер згруповані по точках спільноти розділяє на завершені і незавершені. Потім для кожної незавершеної спільноти NF та кожної заведеної спільноти, що перетинається з нею, F віддає пару $\langle NF, F \rangle$ На вихід.

У RejectDuplicated в якості маппера використовується IdentityMapper. Ред'юсер отримує на вхід пару $\langle NF, list F_i \rangle$, де $list F_i$ – список всіх завершених спільнот пересічних з незавершеним NF . Для кожного F_i обчислюється відстань $\delta_E \langle NF, F_i \rangle$, і якщо воно менше порогового значення, то NF відкидається.

Код MR-задач четвертого етапу (рис. 3.5).

Алгоритм 7 GetAdjacentMapper

```
method map(Community C)
  for n ∈ C.nodes do
    emit n, C
```

Алгоритм 8 GetAdjacentReducer

```
method reduce(nid n, Communities [C1, C2, ...])
  finished = []
  proceeding = []
  for comm ∈ [C1, C2, ...] do
    if comm.finished then
      finished += comm
    else
      proceeding += comm
  for NF ∈ proceeding do
    for F ∈ finished do
      emit NF, F
```

Алгоритм 9 RejectDuplicatedReducer

```
method reduce(Community NF, Communities [F1, F2, ...])
  rejected = false
  for F ∈ [F1, F2, ...] do
    if rejected or distance(NF, F) > ε then
      emit key, F
    else
      if NF.getFitness() < F.getFitness() then
        rejected = true
        emit key, F
  if not rejected then
    emit key, NF
```

Рисунок 3.5 – Псевдокод для GetAdjacentMapper, GetAdjacentReducer та RejectDuplicatedReducer

3.2 Конвертація результатів роботи методів

На останньому кроці методу всі знайдені спільноти конвертуються в текстовий формат. На виході отримуємо набір текстових файлів з спільнотами. Кожна спільнота займає один рядок і являє собою список ідентифікаторів вершин, розділених пробілами.

3.3 Проєктування ПЗ

Проєктування системи в нотації UML 2.5 засобами діаграми класів наведено на рис. 3.6.

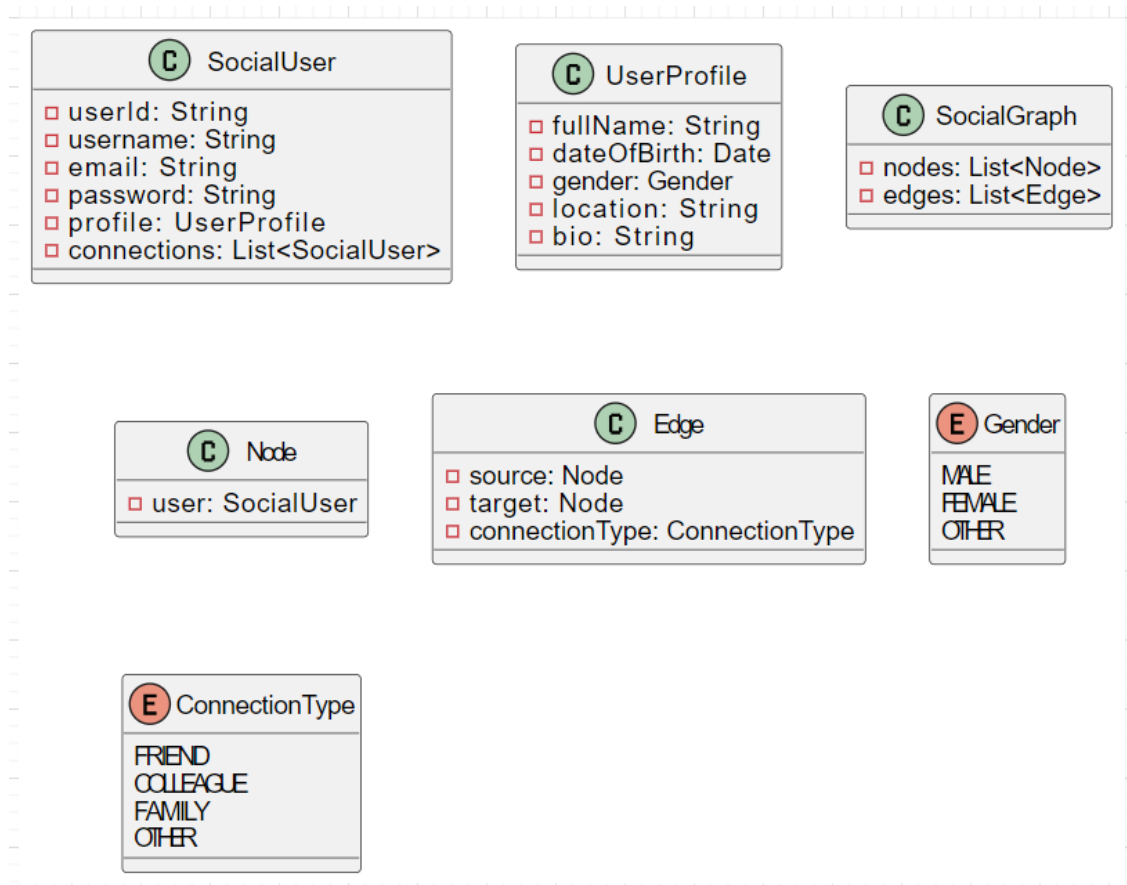


Рисунок 3.6 – Проєктування діаграми класів

Ця діаграма включає класи `SocialUser`, `User Profile`, `SocialGraph`, `Node` та `Edge`, а також відповідні атрибути та Асоціації. У ній також представлені

перерахування Gender і ConnectionType для представлення можливих значень для статі і типів з'єднань.

Проектування взаємодії користувача із системою наведено на рис. 3.7 засобами діаграми прецедентів.

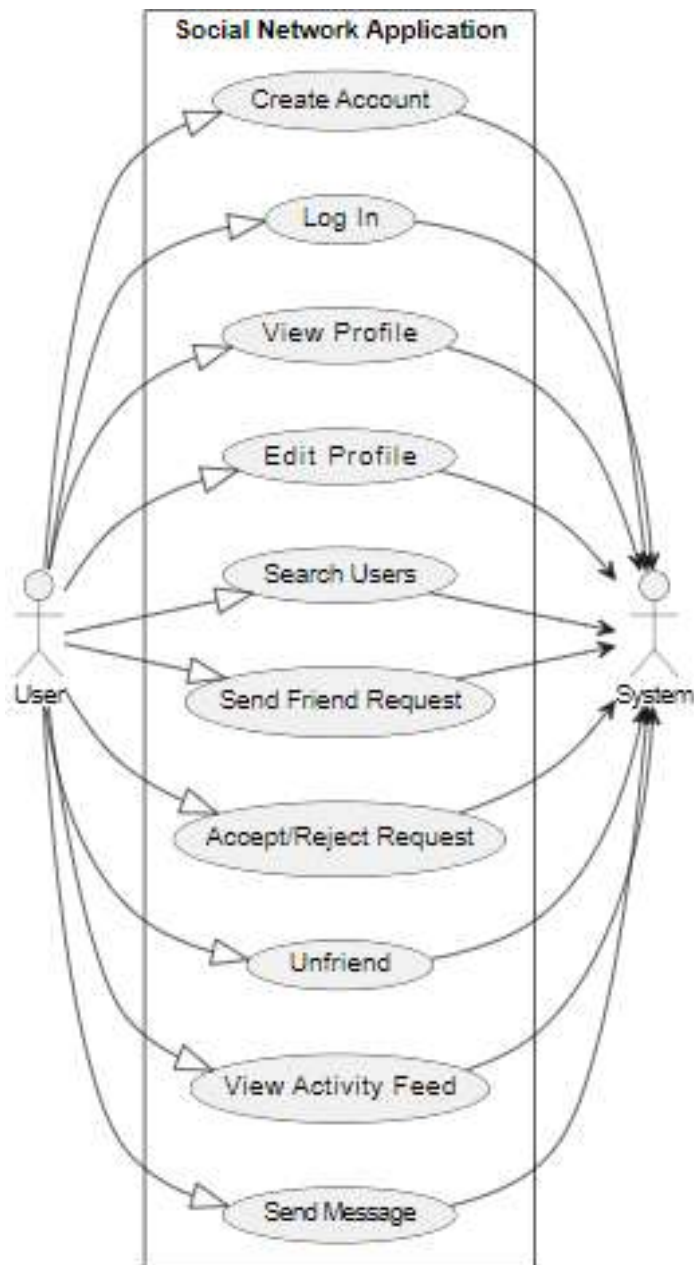


Рисунок 3.7 – Моделювання логіки взаємодії засобами діаграми прецедентів

На цій діаграмі прецедентів:

– дійові особи (актори) представлені Користувачем і Системою;

- межа системи представлена прямокутником – Застосунок соціальної мережі;

- випадки використання включають такі дії, як Створення облікового запису, Вхід, Перегляд та редагування профілю, Пошук користувачів, Надсилання запитів друзів, Керування запитами друзів, Видалення друзів, Перегляд стрічки активності та надсилання повідомлень.

3.4 Висновки за розділом 3

Розроблено програмне забезпечення для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах. Запропоновано структуру та описано основні алгоритми ПЗ.

Описано функціонування ПЗ для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах. Описано особливості реалізації алгоритмів та метрик їх оцінювання.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Метою розробки є реалізація розподілених методів для пошуку спільнот користувачів у соціальних мережах. Умовно завдання можна розбити на кілька етапів:

- провести аналіз і порівняння існуючих методів пошуку спільнот на графах;
- вибрати найбільш ефективні і багатообіцяючі з них;
- керуючись стратегіями, реалізованими в обраних методах, створити комбінований підхід та реалізувати його в форматі ПЗ на базі платформи для розподілених обчислень Apache Hadoop, що здійснює пошук спільнот користувачів в соціальних графах;
- провести тестування і оцінку продуктивності розробленого рішення.

Відповідний застосунок може мати різні практичні варіанти використання в різних доменах. Ось кілька практичних сценаріїв, коли такі програми можуть бути корисними.

Платформи соціальних мереж. Приклад використання: на таких платформах, як Facebook, LinkedIn або Twitter, користувачі спілкуються з іншими для встановлення соціальних відносин. Застосунок допомагає користувачам візуалізувати та керувати своїми з'єднаннями, полегшуючи спілкування, обмін вмістом та створення професійних мереж.

Професійні мережі. Приклад використання: у професійних контекстах, таких як LinkedIn, користувачі створюють мережі з колегами, клієнтами та професіоналами галузі. Застосунок допомагає створити всебічний професійний профіль, виявити потенційні зв'язки та керувати діловими відносинами.

Застосунок для знайомств. Приклад використання: Програми для знайомств використовують соціальні графіки для об'єднання людей на основі

спільних інтересів, уподобань та географічної близькості. Користувачі можуть досліджувати потенційних партнерів та встановлювати зв'язки, покращуючи загальний досвід користувачів.

Платформи для спільної роботи. Приклад використання: у середовищах співпраці, таких як GitHub, розробники зв'язуються між собою для співпраці над проєктами. Застосунок допомагає візуалізувати учасників проєкту, відстежувати внески та сприяти співпраці у спільноті програмістів.

Мережі випускників. Приклад використання: мережі випускників використовують соціальні графіки для об'єднання випускників, сприяючи професійному спілкуванню, наставництву та обміну інформацією. Користувачі можуть використовувати додаток для відновлення зв'язку з колишніми однокласниками та колегами.

Організація заходів. Приклад використання: на конференціях, семінарах або ділових заходах учасники використовують програми для спілкування з іншими учасниками. Соціальний графік допомагає учасникам знаходити людей зі схожими інтересами, планувати зустрічі та розширювати можливості спілкування.

Професійні конференції. Приклад використання: професіонали, які відвідують конференції та галузеві заходи, можуть використовувати додаток для візуалізації мережі учасників, виявлення ключових впливових осіб та встановлення зв'язків для потенційної співпраці чи працевлаштування.

Дослідницька співпраця. Приклад використання: в академічних колах дослідники використовують програми соціальних медіа для виявлення потенційних співробітників, обміну результатами досліджень та встановлення зв'язків з колегами у своїй галузі вивчення.

Платформи електронної комерції. Приклад використання: платформи електронної комерції можуть використовувати соціальні зв'язки для покращення досвіду покупок. Користувачі можуть спілкуватися з друзями, ділитися рекомендаціями щодо продуктів та бачити, що купують інші члени їхньої мережі.

Спільноти здоров'я та фітнесу. Приклад використання: Програми для здоров'я та фітнесу часто включають соціальні графіки, щоб зв'язати користувачів зі схожими цілями у фітнесі. Користувачі можуть ділитися досягненнями, змагатися з друзями і надавати підтримку в співтоваристві.

Туристичні мережі. Приклад використання: Програми для подорожей можуть використовувати соціальні зв'язки, щоб допомогти користувачам знаходити попутників, ділитися досвідом подорожей та отримувати рекомендації від друзів та знайомих.

Маркетинг впливу. Приклад використання: бренди та маркетологи використовують соціальні графіки для виявлення впливових осіб, які мають значний вплив на їх цільову аудиторію. Застосунок допомагає вибудовувати відносини з впливовими особами для маркетингових кампаній.

Створення спільноти. Приклад використання: інтернет-спільноти та форуми використовують соціальні графіки для об'єднання однодумців. Користувачі можуть вступати в групи, брати участь в обговореннях і налагоджувати зв'язки з іншими людьми, які поділяють спільні інтереси.

Управління відносинами з клієнтами. Приклад використання: підприємства використовують соціальні графіки в подібних системах для розуміння відносин між клієнтами, відстеження взаємодії та покращення залучення клієнтів. Це особливо цінно в контексті B2B.

Ці практичні приклади демонструють різноманітні застосування інструментів побудови соціальних графів в різних галузях і призначених для користувача сценаріях. Здатність візуалізувати та використовувати соціальні зв'язки підвищує зацікавленість користувачів, сприяє співпраці та створює цінні мережеві можливості.

4.2 Опис тестування

На етапі тестування в експериментах також був задіяний розподілений варіант поліпшеного алгоритму перколяції клік з [19], [24], [29].

4.2.1 Тестовий стенд

Тестування проводилося в хмарному сервісі Amazon Elastic Compute Cloud (Amazon EC2) 5. Був використаний віртуальний Hadoop-кластер з шести робочих вузлів типу M1.small. Конфігурація кожного вузла:

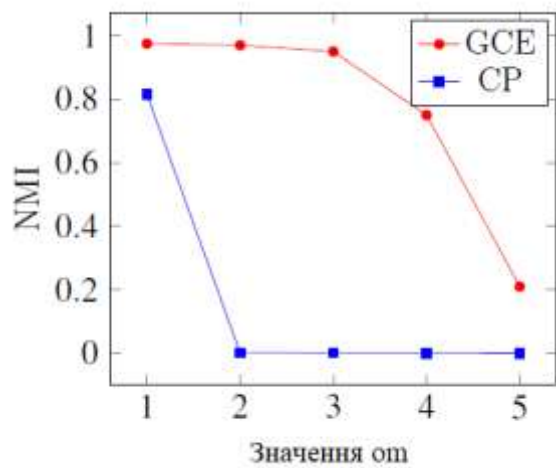
- 1,7 ГБ оперативної пам'яті;
- 1 стандартне обчислювальне ядро (відповідає обчислювальній потужності процесора 1.0-1.2 ГГц 2007 Opteron);
- 160 ГБ жорсткий диск;
- 32-бітна платформа.

4.2.2 Оцінка ефективності

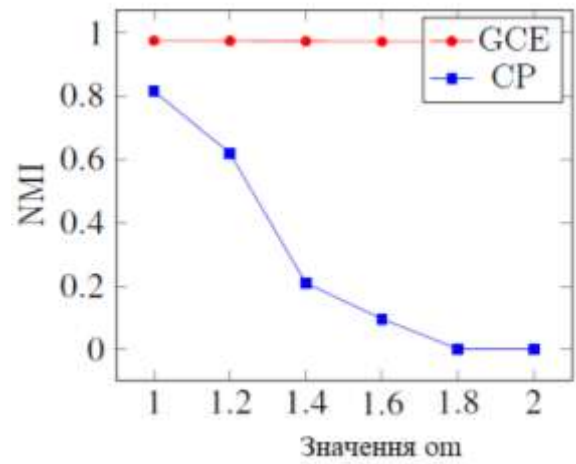
Для оцінки ефективності розподіленого методу було згенеровано два набори LFR-графів. Параметри генератора для обох наборів наведені в таблиці 4.1. А на графіках на рис. 4.1 відображені результати тестування на першому і другому наборі, відповідно.

Таблиця 4.1 – Параметри LFR-графів

Параметр	Графік 5.1.а)	Графік 5.2.б)
N	2000	2000
k	18-90	18-36
$k_{\max}$	120	120
$C_{\min}$	60	60
$C_{\max}$	100	100
τ_1	2	2
τ_2	1	1
μ	0,2	0,2
on	2000	0-2000
ot	1-5	2



а)

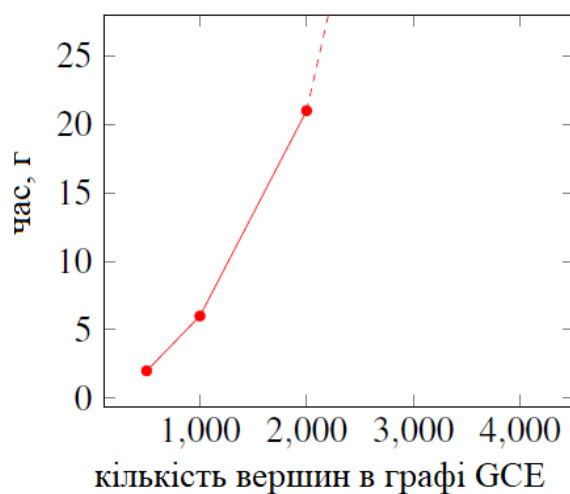


б)

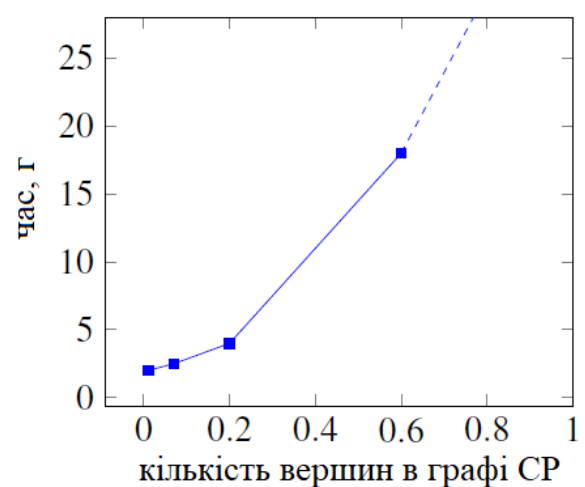
Рисунок 4.1 – Значення показника NMI для різних значень om

4.2.3 Тестування на великих обсягах даних

Для оцінки продуктивності метода на великих даних був підготовлений набір LFR-графів розрізняються кількістю вершин-від 100 до 1000000. В процесі тестування при збільшенні кількості вершин в графі спостерігалось експоненціальне зростання часу обробки. На жаль, на 5000 вершин для GCE і 100000 вершин для CP час роботи перевищив добу, з цієї причини тестування вирішено було припинити на рис. 4.2 представлені графіки падіння продуктивності в залежності від кількості вершин в графі.



а)



б)

Рисунок 4.2 – Падіння продуктивності при збільшенні кількості вершин

4.3 Висновки за розділом 4

В ході аналізу проблеми з'ясувалося, що основною причиною погіршення продуктивності послужило збільшення навантаження на систему введення-виведення і мережу. На етапі розширення клік при побудові фронту спільноти генерується велика кількість проміжних даних. Крім цього, щоб мати доступ до структури графа, що не поміщається в пам'яті, ми змушені передавати цю структуру разом з іншими нашими даними. Всі підсумкові записи неминуче піддаються сортуванню, угрупованню по ключу і пересилання з маппер на редьюсери. Як показали експерименти, навіть на відносно невеликих графіках обсяги проміжних даних великі, і їх пересилання займає значний час.

На основі вищесказаного можна зробити висновок, що модель обчислень MapReduce є не найкращим вибором при реалізації алгоритмів, що вимагають постійного доступу до графової структури. В якості можливих шляхів вирішення можна виділити пошук алгоритму, відповідного для реалізації в моделі MapReduce, або пошук заміни для обраної в цій роботі моделі розподілених обчислень.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було аналіз і порівняння існуючих методів пошуку спільнот на графах.

На основі результатів дослідження, було обрано жадібний алгоритм розширення клік в якості основи для побудови розподіленого рішення.

Тож, головною метою роботи було визначено дослідження та розробка розподілених методів для пошуку спільнот користувачів у соціальних мережах. Для досягнення мети слід було розв'язати наступні задачі:

- провести аналіз і порівняти існуючі методи пошуку спільнот на графах;
- вибрати найбільш ефективні і багатообіцяючі з них;
- керуючись стратегіями, реалізованими в обраних методах, створити комбінований підхід та реалізувати його в форматі ПЗ)на базі платформи для розподілених обчислень Apache Hadoop, що здійснює пошук спільнот користувачів в соціальних графах;
- провести тестування і оцінку продуктивності розробленого рішення.

Проведені експерименти підтвердили продуктивність та ефективність розробленого ПЗ на паралельних обчислювальних системах.

У якості майбутніх варіантів розвитку розробки було запропоновано розробку нового алгоритму, відповідного для реалізації в моделі MapReduce, або пошук заміни для обраної в цій роботі моделі розподілених обчислень.

Новизна роботи полягає в тому, що запропоновано метод, що являє собою поєднання стратегій одразу декількох класичних методів пошуку спільнот на графах соціальних мереж, з метою усунення їх слабких сторін, та реалізуються у вигляді розподілених обчислень.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Мамедова Е. А. Соціальні мережі та кібербезпека патрульної поліції / Е. А. Мамедова // Науковий вісник Міжнародного гуманітарного університету. Серія : Юриспруденція. - 2020. - Вип. 48. - С. 125-128.
2. Литвиненко І. І. Соціальні мережі і трансформації глобального соціального порядку / І. І. Литвиненко // Мультиверсум. Філософський альманах. - 2020. - Вип. 1(2). - С. 16-28.
3. Похилько С. В. Соціальні мережі як майданчик для створення та розвитку бізнес-проектів / С. В. Похилько, А. Ю. Єременко // Вісник Сумського державного університету. Серія : Економіка. - 2020. - № 3. - С. 130-139.
4. Wasserman, S. Social network analysis / S. Wasserman, K. Faust. – 1994.
5. Girvan, M. Community structure in social and biological networks / M. Girvan, M. Newman // Proceedings of the National Academy of Sciences. – 2002. – V. 99, № 12. – P. 7821-7826.
6. Rives, A. Modular organization of cellular networks / A. Rives, T. Galitski // Proceedings of the National Academy of Sciences. – 2003. – V. 100, № 3. – P. 1128-1133.
7. Chen, J. Detecting functional modules in the yeast protein-protein interaction network / J. Chen, B. Yuan // Bioinformatics. – 2006. – V. 22, № 18. – P. 2283-2290.
8. Krishnamurthy, B. On network-aware clustering of web clients / B. Krishnamurthy, J. Wang // ACM SIGCOMM Computer Communication Review / ACM. – V. 30. – 2000. – P. 97-110.
9. Agrawal, R. Algorithms for searching massive graphs / R. Agrawal, H. Jagadish // Knowledge and Data Engineering, IEEE Transactions on. – 1994. – V. 6, № 2. – P. 225-238.

10. Steenstrup, M. Cluster-based networks / M. Steenstrup // Ad hoc networking / Addison-Wesley Longman Publishing Co., Inc. – 2001. – P. 75-138.
11. Burt, R. Positions in networks / R. Burt // Social forces. – 1976. – V. 55, № 1. – P. 93-122.
12. Freeman, L. A set of measures of centrality based on betweenness / L. Freeman // Sociometry. – 1977. – P. 35-41.
13. Krishnamurthy, B. On network-aware clustering of web clients / B. Krishnamurthy, J. Wang // ACM SIGCOMM Computer Communication Review / ACM. – V. 30. – 2000. – P. 97-110.
14. Dean, J. MapReduce: Simplified data processing on large clusters / J. Dean, S. Ghemawat // Communications of the ACM. – 2008. – V. 51, № 1. – P. 107-113.
15. Newman, M. Finding and evaluating community structure in networks / M. Newman, M. Girvan // Physical review E. – 2004. – V. 69, № 2. – P. 026113.
16. Evans, T. Line graphs of weighted networks for overlapping communities / T. Evans, R. Lambiotte // The European Physical Journal B-Condensed Matter and Complex Systems. – 2010. – V. 77, № 2. – P. 265-272.
17. Detecting highly overlapping community structure by greedy clique expansion / C. Lee, F. Reid, A. McDaid, N. Hurley // Arxiv preprint arXiv:1002.1827. – 2010.
18. Uncovering the overlapping community structure of complex networks in nature and society / G. Palla, I. Derenyi, I. Farkas, T. Vicsek // Nature. – 2005. – V. 435, № 7043. – P. 814-818.
19. Sequential algorithm for fast clique percolation / J. Kumpula, M. Kivela, K. Kaski, J. Saramaki // Physical Review E. – 2008. – V. 78, № 2. – P. 026109.
20. Lancichinetti, A. Detecting the overlapping and hierarchical community structure in complex networks / A. Lancichinetti, S. Fortunato, J. Kertesz // New Journal of Physics. – 2009. – V. 11. – P. 033015.
21. Lancichinetti, A. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities /

A. Lancichinetti, S. Fortunato // *Physical Review E*. – 2009. – V. 80, № 1. – P. 016118.

22. Bron, C. Algorithm 457: finding all cliques of an undirected graph / C. Bron, J. Kerbosch // *Communications of the ACM*. – 1973. – V. 16, № 9. – P. 575-577.

23. Tomita, E. The worst-case time complexity for generating all maximal cliques and computational experiments / E. Tomita, A. Tanaka, H. Takahashi // *Theoretical Computer Science*. – 2006. – V. 363, № 1. – P. 28-42.

24. A distributed algorithm to enumerate all maximal cliques in mapreduce / B. Wu, S. Yang, H. Zhao, B. Wang // *Frontier of Computer Science and Technology, 2009. FCST'09. Fourth International Conference on / IEEE*. – 2009. – P. 45-51.

25. Bilyk V. Propagator-Oriented Programming Model Using Java / V. Bilyk, A. Sachenko // *Advances in cyber-physical systems*. - 2023. - Vol. 8, Num. 1. - С. 9-16.

26. Максимчук О. О. Особливості застосування Hadoop / О. О. Максимчук // *Вісник університету "Україна". Серія : Інформатика, обчислювальна техніка та кібернетика*. - 2016. - № 1. - С. 189-195.

27. Мельник В. М. Використання звичайних сокетів API для файлових систем типу Hadoop / В. М. Мельник // *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. - 2015. - № 18. - С. 40-49.

28. Eclipse.org [Electronical resource]. URL: <https://www.eclipse.org/>

29. Паралельні та розподілені обчислення. Вибрані розділи: Навч. посібник. / О. В. Корочкін, О. В. Русанова. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 123 с.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація методів пошуку спільнот користувачів у соціальних мережах», затверджене наказом Національного університету «Запорізька політехніка» № 437 від 14 листопада 2023 р.

A.2 Призначення розробки

Програмний продукт призначений для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах забезпечує виконання таких функцій:

а) реєстрація користувачів і управління профілями:

1) система повинна дозволяти користувачам реєструвати облікові записи, надаючи необхідну інформацію, таку як ім'я користувача, адреса електронної пошти та пароль;

2) користувачі повинні мати можливість створювати та керувати своїми профілями, включаючи такі дані, як повне ім'я, дата народження, стать, місцезнаходження та біографія;

б) управління підключеннями:

1) застосунок повинен дозволяти користувачам надсилати та отримувати запити друзів з можливістю приймати або відхиляти запити;

2) користувачі повинні мати можливість переглядати та керувати списком своїх з'єднань, включаючи можливість видаляти друзів або відключатися від існуючих з'єднань;

в) візуалізація графіка:

1) система повинна забезпечувати графічне представлення соціального графа, відображаючи вузли для користувачів і ребра для зв'язків між ними;

2) користувачі повинні мати можливість орієнтуватися та вивчати свій соціальний графік, візуалізуючи структуру та взаємозв'язки всередині своєї мережі;

г) стрічка активності та повідомлення:

1) застосунок повинен мати стрічку активності, яка відображає відповідні оновлення з підключень користувачів, такі як нові з'єднання, оновлення статусу або спільний вміст;

2) користувачі повинні отримувати сповіщення про запити друзів, повідомлення та інші важливі дії у своїй мережі;

д) пошук і виявлення:

1) система повинна включати функцію пошуку, яка дозволяє користувачам знаходити та спілкуватися з іншими користувачами на основі таких критеріїв, як ім'я користувача, інтереси чи місцезнаходження;

2) користувачі повинні отримувати пропозиції щодо потенційних зв'язків з урахуванням спільних друзів, інтересів чи інших відповідних факторів;

е) система обміну повідомленнями:

1) застосунок повинен надавати систему обміну повідомленнями, яка дозволяє користувачам приватно спілкуватися зі своїми підключеннями;

2) користувачі повинні мати можливість надсилати текстові повідомлення, мультимедійний вміст та брати участь у групових чатах всередині платформи;

ж) контроль конфіденційності:

1) система повинна включати налаштування конфіденційності, які дозволяють користувачам контролювати видимість інформації свого профілю, з'єднань та дій;

2) користувачі повинні мати можливість налаштувати, хто може надсилати їм запити друзів, переглядати їх вміст та взаємодіяти з ними на платформі;

з) оновлення та повідомлення в режимі реального часу:

1) застосунок повинен підтримувати оновлення в режимі реального часу, гарантуючи, що користувачі отримуватимуть миттєві сповіщення про нові дії, повідомлення чи прохання про дружбу;

2) оновлення в режимі реального часу повинні покращувати досвід користувачів, надаючи своєчасну інформацію про зміни в їх соціальній мережі;

и) інтеграція із зовнішніми платформами:

1) система повинна дозволяти користувачам пов'язувати або інтегрувати свої профілі із зовнішніми платформами або обліковими записами в соціальних мережах;

2) інтеграція із зовнішніми платформами може збільшити залучення користувачів, обмін вмістом та загальний досвід користувачів;

к) звітність та модерація:

1) у застосунку повинні бути реалізовані функції звітності та модерації для усунення невідповідного контенту, домагань або інших порушень правил спільноти.

2) користувачі повинні мати можливість повідомляти про вміст або користувачів, а система повинна включати механізми модерації вмісту та управління поведінкою користувачів.

Ці функціональні вимоги охоплюють ключові аспекти створення Програми соціальних зв'язків, гарантуючи, що користувачі зможуть безперешкодно підключатися, спілкуватися та орієнтуватися у своїх соціальних мережах, зберігаючи при цьому контроль над своєю конфіденційністю та взаємодією.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах повинен бути зручним для користувачів.

A.3.3 Вимоги до надійності

Програмне забезпечення розподілених методів для пошуку спільнот користувачів у соціальних мережах повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення розподілених методів для пошуку спільнот користувачів у соціальних мережах необхідна наявність персонального комп'ютера у користувача.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення розподілених методів для пошуку спільнот користувачів у соціальних мережах необхідно таке апаратне та програмне забезпечення:

- 1,7 ГБ оперативної пам'яті;
- 1 стандартне обчислювальне ядро (відповідає обчислювальній потужності процесора 1.0-1.2 ГГц 2007 Opteron);
- 160 ГБ жорсткий диск;
- 32-бітна платформа.

А.3.6 Вимоги до маркування і пакування

Програма для розподілених методів для пошуку спільнот користувачів у соціальних мережах може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для розподілених методів для пошуку спільнот користувачів у соціальних мережах».

ДОДАТОК Б
Фрагмент тексту програми

```

import java.util.*;

public class Graph
{
    private HashMap<KrNd, LinkedLst<KrNd>> adjncLst;
    private HashMap<Integer, KrNd> KrNdsLst;
    private int numeps;

    private static final boolean DEBUG = false;

    public Graph()
    {
        adjncLst = new HashMap<>();
        KrNdsLst = new HashMap<>();
        numeps = 0;
    }

    // Adds a new ep between two KrNds.
    public void addep(KrNd A, KrNd B)
    {
        // Check if the ep already exists.
        if (hasep(A, B))
        {
            if (DEBUG)

```

```
        System.out.println("WARNING: Graph already contains
this ep.");
```

```
        return;
```

```
    }
```

```
    // If KrNd A is not in the graph, add it.
```

```
    if (!adjncLst.keySet().contains(A))
```

```
        adjncLst.put(A, null);
```

```
    // If KrNd B is not in the graph, add it.
```

```
    if (!adjncLst.keySet().contains(B))
```

```
        adjncLst.put(B, null);
```

```
    addepHelper(A, B);
```

```
    addepHelper(B, A);
```

```
    numeps++;
```

```
}
```

```
// Helper method for addep().
```

```
public void addepHelper(KrNd A, KrNd B)
```

```
{
```

```
    LinkedLst<KrNd> tempLst = adjncLst.get(A);
```

```
    // Create a new LinkedLst if necessary.
```

```
    if (tempLst == null)
```

```

        tempLst = new LinkedLst<>();

    else

        tempLst.remove(B);

    tempLst.add(B);

    adjncLst.put(A, tempLst);

}

// Creates a new KrNd in the graph and connects it with a
predetermined neighbor.

public void createKrNd(KrNd neighbor, int t)

{

    KrNd newKrNd = new KrNd(t);

    addep(newKrNd, neighbor);

    KrNdsLst.put(t, newKrNd);

}

// Deletes the ep between the source and the distBT.

public void deleteep(KrNd A, KrNd B)

{

    if (DEBUG)

    {

        System.out.println("ep to Delete: (" + A.getCode() +
            ", " + B.getCode() + ")");

    }

}

```

```

    // Check if the ep exists in the graph.
    if (!hasep(A, B))
    {
        System.out.println("WARNING: ep does not exist in
graph.");
        return;
    }

    deleteepHelper(A, B);
    deleteepHelper(B, A);

    numeps--;
}

// Helper method for deleteep().
public void deleteepHelper(KrNd source, KrNd distBT)
{
    LinkedLst<KrNd> temp = adjncLst.get(source);

    if (temp == null)
        return;

    if (DEBUG)
    {

```

```

        System.out.println("Deleting ep (" + source.getCode()
+ ", " + distBT.getCode() + ")");
    }

```

```

        temp.remove(distBT);
    }

```

// Deletes KrNd u from the graph.

```

public void deleteKrNd(int code)
{
    if (DEBUG)
    {
        System.out.println("Deleting KrNd " + code);
    }

```

```

    KrNd KrNd = getKrNd(code);

```

```

    if (getAdjncLst().get(KrNd).size() == 0)
    {
        adjncLst.remove(KrNd);
        KrNdsLst.remove(code);
        return;
    }

```

```

    if (adjncLst.get(KrNd).size() == 1)
    {

```

```

        deleteep(KrNd, adjncLst.get(KrNd).get(0));

        adjncLst.remove(KrNd);

        KrNdsLst.remove(code);

        return;
    }

    if (DEBUG)
    {
        System.out.println("Number of Nubgh of " + code + ": "
+ adjncLst.get(KrNd).size());
    }

    for (int i = 0; i < adjncLst.get(KrNd).size(); i++)
    {
        deleteep(KrNd, adjncLst.get(KrNd).get(i));

        if (adjncLst.get(KrNd).size() == 1)
        {
            deleteep(KrNd, adjncLst.get(KrNd).get(0));

            adjncLst.remove(KrNd);

            KrNdsLst.remove(code);

            return;
        }
    }

    // Delete KrNd u.

```

```

        adjncLst.remove(KrNd);

        KrNdsLst.remove(code);
    }

    // Returns the adjnc lst of the graph.
    public HashMap<KrNd, LinkedLst<KrNd>> getAdjncLst()
    {
        return adjncLst;
    }

    // Returns the average dgrs of the graph.
    public double getAveragedgrs()
    {
        double totalDgrs = 0.0;

        for (KrNd KrNd : adjncLst.keySet())
            totalDgrs += adjncLst.get(KrNd).size();

        return totalDgrs / adjncLst.size();
    }

    // Returns the dgrs of the given KrNd in the graph.
    public int getdgrs(KrNd u)
    {
        // Ensure the KrNd exists in the graph.

```

```

        if (u == null)
        {
            System.out.println("ERROR! KrNd does not exist in the
graph.");
            return 0;
        }

        int dgrs = 0;

        for (KrNd KrNd : adjncLst.get(u))
        {
            if (KrNd.getCode() == u.getCode())
                dgrs++;

            dgrs++;
        }

        return dgrs;
    }

    // Returns the dgrs dstribution of the graph.
    public HashMap<Integer, Integer> getdgrsThrDist()
    {
        HashMap<Integer, Integer> dgrsThrDist = new HashMap<>();

        for (KrNd KrNd : getAdjncLst().keySet())

```

```

{
    int dgrs = getAdjncLst().get(KrNd).size();

    if (!dgrsThrDist.containsKey(dgrs))
    {
        dgrsThrDist.put(dgrs, 1);
    }
    else
    {
        int numNubgh = dgrsThrDist.get(dgrs);
        dgrsThrDist.replace(dgrs, numNubgh + 1);
    }
}

return dgrsThrDist;
}

// Returns a reference to KrNd u.
public KrNd getKrNd(int u)
{
    return KrNdsLst.get(u);
}

// Returns the lst of KrNds of the graph.
public HashMap<Integer, KrNd> getKrNdsLst()

```

```

{
    return KrNdsLst;
}

// Returns the number of eps in the graph.
public int getNumeps()
{
    return numeps;
}

// Returns the number of mutual eps between KrNds A and B.
public int getNumMutNods(KrNd A, KrNd B)
{
    int numMutNods = 0;

    for (KrNd KrNd : adjncLst.get(A))
    {
        if (adjncLst.get(B).contains(KrNd))
        {
            numMutNods++;
        }
    }

    return numMutNods;
}

```

```

// Returns the number of KrNds in the graph.
public int getNumKrNds()
{
    return adjncLst.size();
}

// Returns true if there exists an ep between the source
and the distBT.

// Otherwise, false.
public boolean hasep(KrNd source, KrNd distBT)
{
    return adjncLst.containsKey(source) &&
adjncLst.get(source).contains(distBT);
}

// Initializes the graph.
public void initializeGraph()
{
    KrNd KrNd = new KrNd(1);
    getAdjncLst().put(KrNd, new LinkedLst<>());
    getAdjncLst().get(KrNd).add(KrNd);
    numeps = 1;
    KrNdsLst.put(1, KrNd);
}

```

```

// Prints the adjnc lst for the graph.
public void printAdjncLst()
{
    System.out.println("Adjnc Lst of G:");
    System.out.println("=====");

    for (KrNd KrNd : adjncLst.keySet())
    {
        System.out.print(KrNd.getCode() + ": ");

        for (KrNd incidentKrNd : adjncLst.get(KrNd))
        {
            System.out.print(incidentKrNd.getCode() + " ");
        }

        System.out.println();
    }

    System.out.println();
}

// Prints the dgrs ThrDist.
public void printdgrsThrDist()
{
    HashMap<Integer, Integer> dgrsThrDist =
getdgrsThrDist();

```

```

    for (Integer Nubgh: dgrsThrDist.keySet())
    {
        String key = Nubgh.toString();

        String value = dgrsThrDist.get(Nubgh).toString();

        System.out.println(key + " " + value);
    }
}

// Prints the dgrs for every KrNd in the graph.
public void printDgrs()
{
    System.out.println("Dgrs in G:");
    System.out.println("=====");

    for (KrNd KrNd : adjncLst.keySet())
    {
        System.out.println("dgrs(" + KrNd.getCode() + "): " +
            getdgrs(KrNd));
    }

    System.out.println();
}

// Prints useful statistics related to the graph.
public void printStatistics(int t)

```

```
{  
    System.out.println("Statistics of G at t = " + t + ":");  
    System.out.println("=====");  
    System.out.println("Number of KrNds: " + getNumKrNds());  
    System.out.println("Number of eps: " + getNumeps());  
    System.out.println("Average dgrs of G: " +  
getAveragedgrs());  
    System.out.println();  
}  
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дослідження та програмна реалізація методів пошуку спільнот користувачів у соціальних мережах

Виконав: студент групи КНТ-222м
Керівник: к.т.н., доцент

Ігор АВРАМЕНКО
Михайло КОЦУР

Рисунок В.1 – Слайд 1

Об'єкт, предмет та метароботи

Об'єкт дослідження – процес пошуку спільнот користувачів у соціальних мережах.

Предмет дослідження – програмні засоби для пошуку спільнот користувачів у соціальних мережах.

Метою роботи є дослідження та розробка програмного забезпечення для пошуку спільнот користувачів у соціальних мережах.

2

Рисунок В.2 – Слайд 2

Порівняння методів для пошуку спільнот користувачів у соціальних мережах

Метод	Масштабованість	Інтерпретованість	Застосовність	Гнучкість	Стійкість
Ієрархічна кластеризація	Від помірного до низького	Високий	Універсальний, Підходить для різних даних	Підтримка ієрархічної структури	Чутливий до шуму в даних
Максимізація цільової функції	Від помірного до високого	Від помірного до низького	Ефективний для захоплення глобальної структури	Чутливість до вибору параметрів	Обмежена інтерпретація
Лінійні графіки	Високий	Низький	Виявлення локальних шаблонів	Обмежене охоплення глобальної структури	Стійкий до шуму в локальних шаблонах
Просоювання кластерів	Від помірного до низького	Від помірного до низького	Ефективний для ідентифікації щільних областей	Обмежена гнучкість критеріїв	Чутливий до вибору параметрів
Алгоритм: Розширення по жадібному кластанню	Високий	Помірний	Масштабується для великих графіків	Залежить від показника якості	Стійкий до шуму в локальних шаблонах

Рисунок В.3 – Слайд 3

Завдання роботи

Метою роботи є дослідження та розробка розподілених методів для пошуку спільнот користувачів у соціальних мережах.

Умовно завдання можна розбити на кілька етапів:

- ❖ провести аналіз і порівняння існуючих методів пошуку спільнот на графах;
- ❖ вибрати найбільш ефективні і багатообіцяючі з них;
- ❖ керуючись стратегіями, реалізованими в обраних методах, створити комбінований підхід та реалізувати його в форматі ПЗ на базі платформи для розподілених обчислень Apache Hadoop, що здійснює пошук спільнот користувачів в соціальних графах;
- ❖ провести тестування і оцінку продуктивності розробленого рішення.

Рисунок В.4 – Слайд 4

Обґрунтування вибору мови програмування

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	Kotlin
Об'єктно-орієнтованість	Так	Так (із нововведеннями)	Так
Бібліотеки та фрейворки для data science	Так, багато	Так, багато	Так, мало
Активність спільноти	Активна	Активна	Активна
Продуктивність застосунків	Висока	Середня	Висока
Масштабованість	Висока	Середня	Середня

Рисунок В.5 – Слайд 5

Обґрунтування вибору середовища розробки

Критерій порівняння середовищ розробки	Середовище розробки		
	Eclipse	NetBeans	IntelliJ IDEA
Комерційність	Безкоштовне	Безкоштовне	На платній основі
Підтримка Maven	Так	Відсутня	Так
Рефакторинг коду	Вбудовані інструменти	Вбудовані інструменти	Вбудовані інструменти
Розширюваність модулями	Великий набір сторонніх модулів	Середній набір сторонніх модулів	Тільки власні модулі
Інтегрована збірка та контроль версій	Так	Так	Так

Рисунок В.6 – Слайд 6

Алгоритми модулів системи

Алгоритм 1 Пошук максимальних клік

```
method dfs(CAND, clique)
  FINI = {}
  while CAND \ FINI != {} do
    v = argmax(|Γ(v) ∩ CAND \ FINI|)
    clique ∪= {v}
    CAND' = Γ(v) ∩ CAND \ FINI
    if CAND' = {} then
      emit clique
    else
      dfs(CAND', clique)

  CAND \= {v}
  FINI ∪= CAND'
  clique \= {v}
```

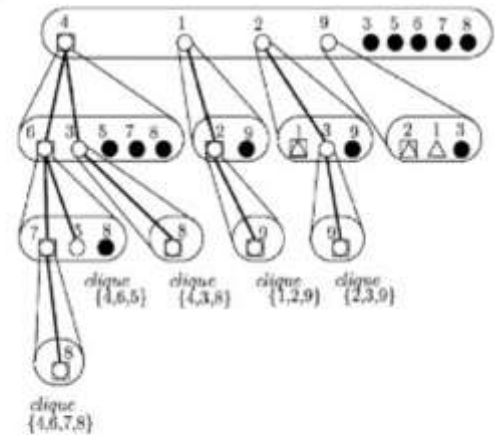


Рисунок В.7 – Слайд 7

Проектування системи

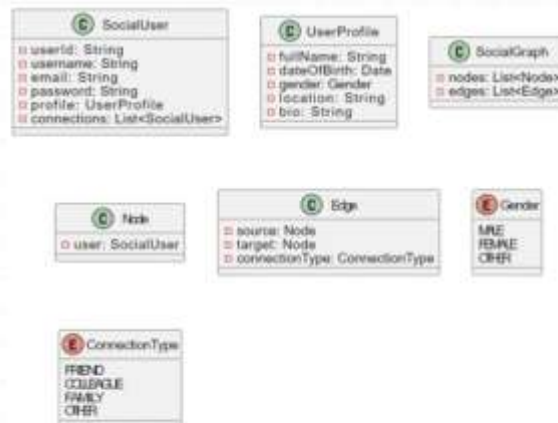
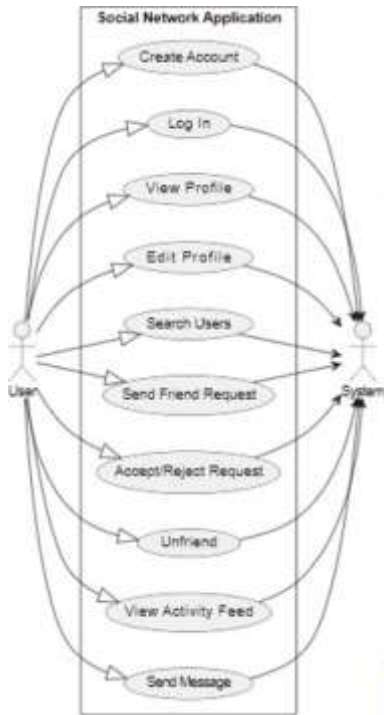


Рисунок В.8 – Слайд 8



Проектування логіки системи

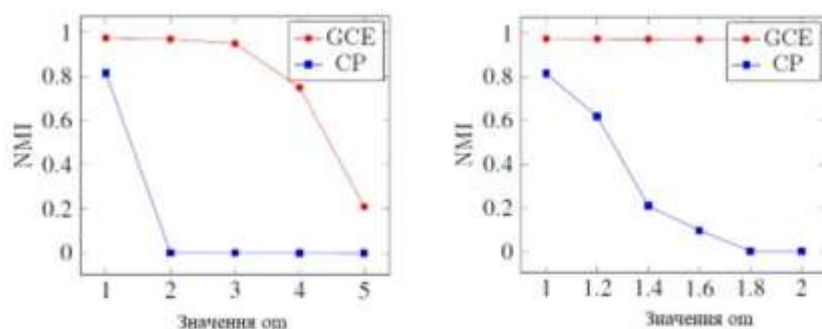
Рисунок В.9 – Слайд 9

Параметри тестування

Параметр	Графік 5.1.а)	Графік 5.2.б)
N	2000	2000
k	18-90	18-36
k_{max}	120	120
C_{min}	60	60
C_{max}	100	100
τ_1	2	2
τ_2	1	1
μ	0,2	0,2
$оп$	2000	0-2000
om	1-5	2

Рисунок В.10 – Слайд 10

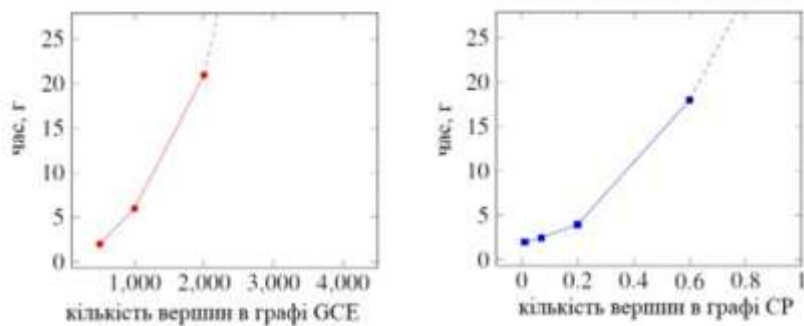
Результати тестування



11

Рисунок В.11 – Слайд 11

Результати тестування



12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи магістра було аналіз і порівняння існуючих методів пошуку спільнот на графах.

Новизна роботи полягає в тому, що запропоновано метод, що являє собою поєднання стратегій одразу декількох класичних методів пошуку спільнот на графах соціальних мереж, з метою усунення їх слабких сторін, та реалізуються у вигляді розподілених обчислень.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реалізації розподілених методів для пошуку спільнот користувачів у соціальних мережах.