Санкт-Петербургский государственный университет

Кафедра системного программирования

Кутуев Владимир Александрович

Кластеризация социального графа с использованием графовой базы данных

Курсовая работа

Научный руководитель:
доц. , к.т.н. Литвинов Ю. В.

Консультант:
рук. отд. раз. ПО, ООО “Белкасофт” Тимофеев Н. М.

Санкт-Петербург
2019

Оглавление

Введение

Визуализация — один из самых эффективных инструментов представления данных для экспертного анализа. Она помогает наглядно отследить основные тенденции, зависимости, отклонения и принять правильные решения на основе имеющейся информации. В ПО для цифровой экспертизы подсистема визуализации данных является одной из важнейших, т.к необходимо наиболее информативно отображать большие объёмы данных.

Граф связей — важный элемент визуализации данных в криминалистическом анализе. Раскладка графа, отражающая его структуру, даст криминалисту больше информации об обнаруженных контактах и их взаимодействиях. Одним из главных методов для достижения этой цели является выделение сообществ в графе взаимодействий. На рисунках 1 и 2 можно видеть, что раскладка графа с выделенными сообществами отражает структуру графа лучше, чем раскладка необработанного графа.

Данная работа является развитием работы [1] Куликова Егора, в которой был разработан и реализован алгоритм выделения сообществ в социальных графах.

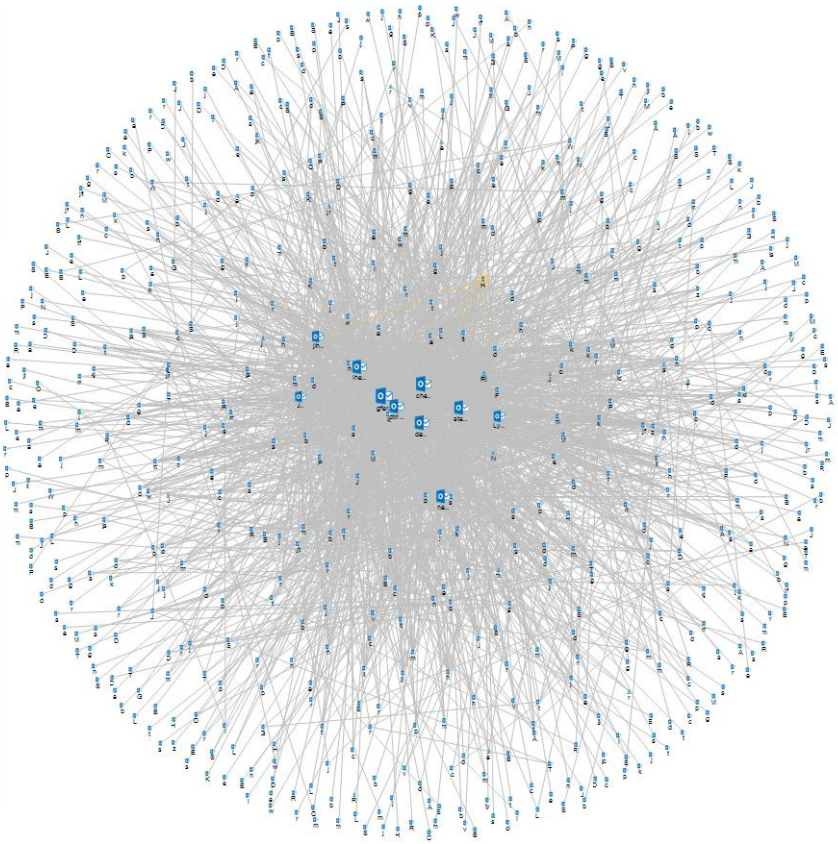

Рисунок 1: Необработанный граф

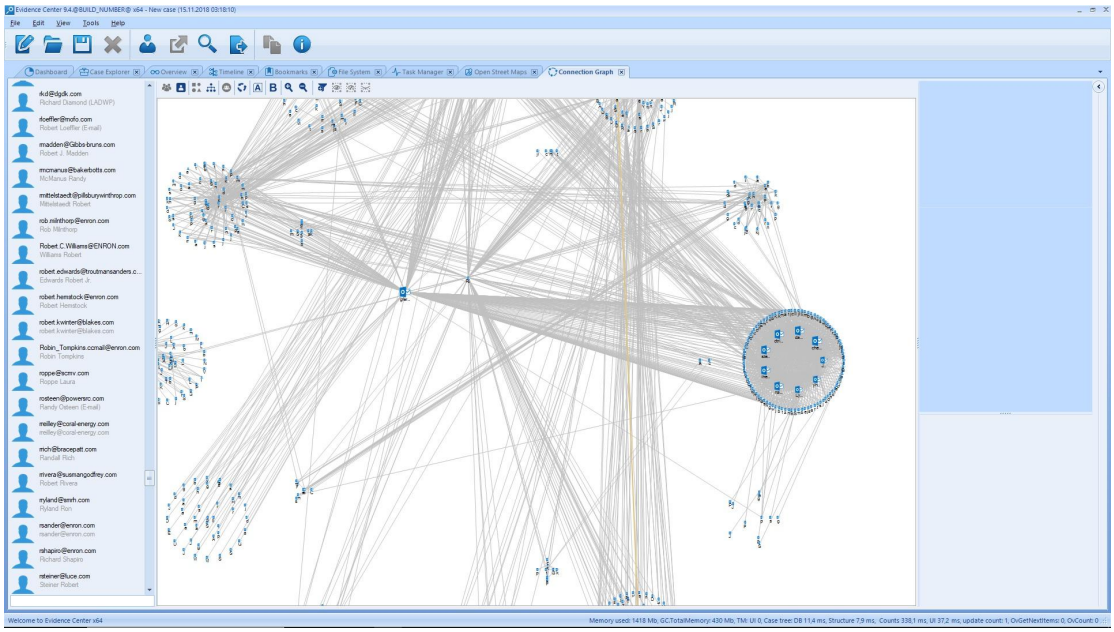

Рисунок 2: Граф с выделенными сообществами

1. Постановка задачи

Цель работы — оптимизировать хранение и работу с графом связей для быстрой фильтрации и выделения сообществ. Для достижения цели были поставлены следующие задачи:

- Выбрать алгоритм выделения сообществ в социальном графе
- Научиться эффективно использовать возможности графовой СУБД. Для этого необходимо:
    1. Выбрать оптимальную для достижения цели работы СУБД;
    2. Исследовать различные подходы к вставке в базу больших графов для выбранной СУБД;
    3. Реализовать их и сравнить.
- Реализовать алгоритм с использованием графовой СУБД

Полученную в результате работы реализацию алгоритма выделения сообществ планируется использовать для визуализации графа связей в продукте Belkasoft Evidence Center[1], предназначенном для проведения цифрового криминалистического анализа.

[1] https://belkasoft.com/ec

2. Обзор существующих решений

- **Nuix Insight Analytics & Intelligence**

  Платформа Nuix Analytics & Intelligence предоставляет граф связей как один из видов визуализации данных. В этом продукте кластеризация графа осуществляется только на основе типа исследуемых данных (люди, объекты, местоположение и события) и источника этих данных. Кластеризация на основе структуры графа не производится. [2]

- **Оксиджен Софтвер «Мобильный криминалист»**

  Программный комплекс «Мобильный криминалист» использует граф связей для демонстрации сложных связей внутри группы исследуемых контактов. Данный инструмент может выделить общие контакты двух пользователей исследуемых устройств, отобразить связи конкретного пользователя. Другой функциональности по автоматизированному анализу данный продукт не предоставляет. [3]

- **Magnet AXIOM**

  Данный продукт может визуализировать связи между контактами, источниками данных и другими артефактами. Однако функциональности по кластеризации, фильтрации графа связей в этом продукте не предусмотрено. [4]

- **Access data “Forensic Toolkit”**

  Один из компонентов этого инструмента цифровой криминалистики отображает общение по электронной почте в виде графа. Этот компонент поддерживает фильтрацию контактов, а также может отображать контакты с наибольшим числом связей. Кластеризацию графа данный продукт производить не может. [5]

- **IBM i2 Analyst's Notebook**

  IBM i2 Analyst's Notebook — мощный инструмент многомерного визуального анализа связей. Он позволяет автоматически выделять наиболее значимые контакты, основываясь на различных подходах к определению

центральности узла в графе. Однако в нём нет функциональности по кластеризации графа. [6]

В результате анализа функциональности инструментов для проведения компьютерной криминалистики выявлено, что из рассмотренных инструментов только IBM i2 Analyst's Notebook обладает функциональностью по автоматизированному анализу графа связей на основе его структуры. Но существенный минус этого приложения — отсутствие возможности выделения кластеров в графе, что не позволяет использовать данный инструмент для многих криминалистических задач. Поэтому было принято решение развивать данную функциональность в Belkasoft Evidence Center.

3. Обзор используемых инструментов

Помимо наглядности визуального представления важным техническим требованием является эффективное хранение графа. Выбранный формат должен позволять проводить фильтрацию вершин и рёбер. Для этого можно хранить его в базе данных. Необходимо описать структуру графа связей в терминах выбранной СУБД и реализовать преобразования полученного графа с помощью её возможностей. В реализации работы [1] была использована СУБД SQLite, что в процессе промышленной эксплуатации оказалось неудобным: моделирование графа в реляционной СУБД не обладало достаточной мощностью для фильтрации и взятия подграфов, а получаемый при разработке код было сложно поддерживать. Было принято решение обратить внимание на БД с поддержкой графов на уровне архитектуры базы. Одной из таких СУБД является Neo4j. Данная система была выбрана из следующих соображений:

- Хранение данные в специальном формате, приспособленном для графов, что оптимальнее моделирования их структуры средствами реляционной СУБД;
- Технология активно развивается и используется. Первый выпуск — 2007 год, клиенты: Adobe, Microsoft, IBM, eBay, NASA и др.;
- Наличие документации для различных аспектов работы;
- Интерфейс программирования приложений для СУБД реализован для многих языков программирования и платформ, в том числе .Net;
- В СУБД используется собственный язык запросов и манипулирования данными — Cypher, который содержит шаблоны для описания вершин и рёбер;
- Наличие открытой Community версии.

Однако вставка в базу используя стандартные запросы (метод вставки Basic) оказалась очень медленной: время вставки полного графа, содержащего 140 вершин, составляет около полутора минут. После изучения различных подходов к вставке, были выделены следующие методы решения этой проблемы:

- LoadCsv — С помощью выражения “LOAD CSV” данные из CSV-файла делаются доступными для использования в запросе к базе. Затем осуществляется их запись с использованием обычных выражений языка запросов Cypher; [7]
- ApocLoadCsv — Использование аналогичного подходу LoadCsv метода “LOAD CSV” библиотеки процедур для Neo4j — Apoc; [8]
- UserProcedures — Написание своих процедур для Neo4j, ориентированных на вставку данных в базу; [9]
- BatchNeo4j — Выполнение вставки с использованием специального метода, описанного в документации к одной из прошлых версий БД; [10]
- Embedded — Использование встраиваемой версии БД; [11]
- BatchEmbedded — Комбинирование подходов BatchNeo4j и Embedded.

4. Сравнительный анализ вставки в базу данных

Так как вставка в базу данных стандартным методом работает неудовлетворительно долго, то было необходимо реализовать и провести сравнительный анализ методов, предложенных для решения этой проблемы. Все они были реализованы на языке Kotlin, так как СУБД Neo4j работает на платформе JVM, а некоторые из методов вставки требуют взаимодействия с API, недоступным для других платформ. Исходный код доступен на GitHub: https://github.com/KutuevVladimir/neo4j-insert-testing

Тестирование проводилось на сгенерированных и реальных графах. Генерация графов производилась утилитой, предоставленной авторами статьи [12]. Реальные социальные графы были предоставлены компанией Belkasoft.

Результаты тестирования (время указано в секундах):

Сгенерированные графы								
Граф		Basic	LoadCsv	ApocLoadCsv	UserProcedures	BatchNeo4j	Embedded	BatchEmbedded
Число вершин	Число рёбер							
10	30	1	1	1	<1	12	<1	<1
50	406	3	1	1	<1	12	<1	1
100	1020	5	2	2	1	12	<1	1
500	9872	35	11	13	4	14	2	2
1000	34532	142	56	18	20	21	10	9

Реальные графы								
Граф		Basic	LoadCsv	ApocLoadCsv	UserProcedures	BatchNeo4j	Embedded	BatchEmbedded
Число вершин	Число рёбер							
458	754	6	2	2	1	12	<1	1
717	3236	16	6	7	2	13	1	2
889	11444	45	17	17	6	15	3	3
3547	14094	107	90	55	30	23	17	12
9298	39155	890	770	168	280	85	117	770

Тестирование проводилось на компьютере с 64-разрядным четырёхъядерным процессором Intel Core i5-8250U частотой 1.6 ГГц, вместимостью оперативного запоминающего устройства 8 Гб и SSD.

На основании проведённого тестирования можно сделать следующие выводы:

- Методы вставки LoadCsv и ApocLoadCsv наиболее удобны при работе с сервером базы данных, так как вставка происходит за 1 запрос, не требует управления транзакциями. Однако она требует создания файла с импортируемыми данными. Такие методы подойдут для работы с большими объёмами данных, когда необходимо выполнять сложные запросы к базе;
- Метод UserProcedures также требует создания файла с импортируемыми данными, также он требует управления транзакциями и контроля за deadlock’ами;
- Метод BatchNeo4j подходит при редких вставках больших объёмов данных, так как требует остановки сервера базы, что делает её недоступной для запросов;
- Метод Embedded удобен при написании приложений под JVM, которым не требуется оперировать большими объёмами данных и выполнять сложные запросы к ним;
- Метод BatchEmbedded не обладает ни хорошей производительностью, ни удобством использования.

5. Реализация алгоритма выделения сообществ

Для кластеризации графа решено было применить совместное использование алгоритмов Scaled community detection (Prat-Perez et al.) и Louvain-метода, описанное Куликовым Егором в статье [1]. Достоинства такого подхода заключаются в том. что он позволяет получать разбиения, достаточно близкие к тем, которые хотели бы использовать в своей работе эксперты-криминалисты, в то же время такой алгоритм является достаточно эффективным с точки зрения производительности для использования в продуктах криминалистического анализа.

На первом шаге алгоритма строится разбиения графа на сообщества алгоритмом Scaled community detection. На основе выделенных сообществ строится мультиграф, вершинами которого являются сообщества, полученные ранее. Ребро между двумя вершинами наделяется весом, равным сумме весов рёбер между сообществами, «стянутыми» в эти вершины. Кроме того, в каждой вершине строится петля с весом, равным удвоенной сумме весов всех рёбер сообщества, если она не равна нулю. В полученном мультиграфе выделяются сообщества с помощью Louvain-метода.

Реализацию выбранного алгоритма было решено делать как расширение библиотеки алгоритмов для Neo4j [13] (набор процедур выполняемых на сервере БД). Такое решение было принято из следующих соображений:

- Выполнение кода на сервере БД, т.е не возникает необходимости получать данные с сервера с помощью запроса, и записи их обратно;
- Такой подход позволяет абстрагироваться от взаимодействия с API СУБД (например, в библиотеке реализована выборка нужного подграфа, т.е множества вершин и рёбер, к которому необходимо применить алгоритм). Библиотека также содержит различные классы представления графов, которые отвечают разным требованиям, так, например, один из классов позволяет взаимодействовать с очень большими графами (до 64 терабайт на рёбра);

- Наличие в библиотеке функциональности по обработке графов, которую можно применить при реализации выбранного алгоритма кластеризации.

6. Заключение

В рамках работы были выполнены следующие задачи:

- Выбран алгоритм выделения сообществ в социальном графе;
- Выбрана СУБД для оптимизации хранения и обработки графа связей;
- Для выбранной СУБД исследованы различные подходы к вставке в базу больших графов;
- Проведено сравнение данных подходов;
- Реализован алгоритм кластеризации графа с использованием графовой СУБД.

Список литературы

[1] Куликов Е.К. Выделение сущностей в криминалистическом анализе источников данных. Бакалаврская работа. Санкт-Петербургский государственный университет, 2016.

[2] Nuix Analytics & Intelligence. NUIX FACT SHEET — URL: http://attain-it.co/sites/default/files/public/Nuix%20insight%20analytics%20and%20intelligence.pdf (online; accessed 21.12.2018).

[3] Оксиджен Софтвер «Мобильный криминалист». Граф связей // web page. — URL: https://www.oxygensoftware.ru/ru/features/main/social-graph (online; accessed 21.12.2018).

[4] Magnet AXIOM — URL: https://www.youtube.com/watch?time_continue=140&v=8T-lJFAzq4g (online; accessed 21.12.2018).

[5] AccessData “Forensic Toolkit”. UserGuide — URL: https://support.accessdata.com/hc/en-us/article_attachments/218021287/FTK_UG_v6.1.0.pdf (online; accessed 21.12.2018).

[6] IBM i2 Analyst’s Notebook. Social Network Analysis — URL: https://onaxion.nl/wp-content/uploads/2017/12/Social-Network-Analysis-IBM-Notebook.pdf (online; accessed 21.12.2018).

[7] Neo4j documentation. Cypher manual. Clauses. LOAD CSV. // web page. — URL: https://neo4j.com/docs/cypher-manual/current/clauses/load-csv/ (online; accessed 04.05.2019).

[8] APOC User Guide. Load CSV and XLS // web page. — URL: https://neo4j-contrib.github.io/neo4j-apoc-procedures/index33.html#_load_csv_and_xls (online; accessed 04.05.2019).

[9] Neo4j documentation. Extending Neo4j. Procedures // web page. — URL: https://neo4j.com/docs/java-reference/3.5/extending-neo4j/procedures/ (online; accessed 04.05.2019).

[10] The Neo4j Manual v2.3.3 // web page. — URL: https://neo4j.com/docs/stable/batchinsert-examples.html (online; accessed 04.05.2019).

[11] Neo4j documentation. Using Neo4j embedded in Java applications. // web page. — URL: https://neo4j.com/docs/java-reference/3.5/tutorials-java-embedded/ (online; accessed 04.05.2019).

[12] Andrea Lancichinetti, Santo Fortunato, Filippo Radicchi. Benchmark graphs for testing community detection algorithms. — Physical review, 2008.

[13] Neo4j documentation. The Neo4j Graph Algorithms User Guide. // web page. — URL: https://neo4j.com/docs/graph-algorithms/current/ (online; accessed 04.05.2019).