САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Щеникова Снежана Алексеевна

Выпускная квалификационная работа

Компромиссное решение в конкурентной модели по выявлению скрытых сообществ в социальной сети YouTube

Направление 02.04.02
«Фундаментальная информатика и информационные технологии»
Магистерская программа «Технологии баз данных» №19/5503/1

Научный руководитель:

доктор физ.-мат. наук, профессор

О.А. Малафеев

Рецензент:

доктор физ.-мат. наук, профессор

институт ИБМП ГУАП

Ю.А. Пичугин

Санкт-Петербург

2021

SAINT-PETERSBURG STATE UNIVERSITY

Shchenikova Snezhana

Graduation project

A compromise solution in a competitive model for detection hidden communities on the social network YouTube

Master level 02.04.02
«Fundamental Informatics and Information Technology»
«Database Technologies» №19/5503/1

Research supervisor:

Doctor of Physics and Mathematics, Professor

O.A. Malafeev

Reviewer:

Doctor of Physics and Mathematics, Professor

IBMP GUAP

U.A. Pichugin

Saint-Petersburg

2021

Содержание

Введение

На сегодняшний день социальная сеть является одним из основных способов организации взаимодействий между людьми. В качестве «взаимодействия» можно понимать как явное обозначение дружбы (например, добавил в список друзей или контактов), обмен сообщениями, так и участие в каких-либо дискуссиях, что образуются благодаря комментариям под записью, видео или фотографией. Можно провести аналогию с СМИ: пользователи доверяют информации, что получают через социальную сеть больше, чем другим источникам новостей, например, телевидению или журналам.

Анализ социальных сетей на сегодняшний день развивается достаточно быстро. Одной из центральных задач анализа социальной сети является задача выявления скрытых сообществ на основе исследования взаимодействий между пользователями. Под скрытыми сообществами понимаются группы людей, которые невозможно определить без применения специальных математических алгоритмов.

Объектом исследования является социальная сеть YouTube. По статистике на 2020 год она охватывает 2 миллиарда пользователей в мире (69.1% населения в России). Это говорит о том, что в YouTube присутствует достаточно большое количество пользователей, из которых формируются крупные сообщества, что интересно исследовать в дальнейшем.

Задача выявления скрытых сообществ относится к классу задач кластеризации. Одним из интересных направлений исследования является определение пересекающихся сообществ (overlapping community), случай, когда один и тот же пользователь может относиться сразу к нескольким скрытым группам.

Актуальность данного направления исследований обусловлена популярностью использования социальной сети YouTube. Если изначально данная социальная сеть была создана с целью распространения видео по Всемирной паутине, то сейчас ее используют для распространения какой-либо информации, начиная с рекламы и спама, заканчивая сообщениями для участников скрытой группы людей. Поэтому считается, что в данном исследовании может быть заинтересовано большое количество физических лиц или организаций, что смогут выбрать конкретный алгоритм для использования его в своих целях в дальнейшем.

Применяя разные алгоритмы по выявлению таких сообществ, мы получаем несколько результатов для дальнейшего исследования со стороны агентов (физическое лицо или организация). Интересно было бы оценивать их эффективность и рассматривать их применение в социологии, политике, экономике. У каждого агента есть собственная оценка алгоритмов, он расставляет эти алгоритмы в порядке приоритета. Таким образом, между множеством агентов возникает противоречие в выборе конкретного алгоритма по поиску скрытых сообществ. В связи с этим в данной работе рассматривается задача поиска компромиссного решения, что основывается на теории игр. Оптимальное решение задачи позволяет найти компромисс для агентов в возникающей конфликтной ситуации.

На сегодняшний день существует большое количество инструментов, направленных на выявление кластеров пользователей в социальной сети с использованием теории графов, однако ни один из них не является универсальным. Обусловлено это тем, что в данных

решениях можно подчеркнуть либо высокую стоимость, либо отсутствие открытого доступа к исходному коду и, как следствие, невозможность направления компонентов программы на решение определенных задач.

Целью исследовательской работы является разработка программного продукта, что позволяет:

1. Выявить скрытые сообщества;
2. Визуализировать результаты;
3. Сохранить полученные результаты для дальнейшей оценки со стороны агентов;
4. Найти компромиссное решение по полученным оценкам;

В данной работе использованы идеи из статьи [1]. Отсюда наследуется способ сбора необходимых для исследования данных и математическая модель.

Сформулируем задачи для достижения поставленной цели:

1. Модифицировать алгоритм сбора данных;
2. Реализовать/включить в проект новые алгоритмы по выявлению скрытых сообществ и протестировать их;
3. Реализовать возможность настройки параметров для алгоритмов;
4. Реализовать визуализацию с отображением сообществ;
5. Реализовать пользовательский интерфейс для знакомства и оценки результатов алгоритмов со стороны агентов;
6. Реализовать алгоритм поиска компромиссного решения;
7. Провести апробацию и тестирование на реальных данных социальной сети YouTube.

В результате разработано программное решение, что предоставляет для ознакомления агентам несколько подходов по выявлению скрытых сообществ и помогает выбрать им совместное оптимальное решение, решить конфликтную ситуацию.

Глава 1. Обзор литературы

Данный раздел посвящен рассмотрению существующих методов выявления скрытых сообществ в социальных сетях. Основной целью главы является знакомство с предлагаемыми подходами, их анализ. Сначала рассматриваются основные методы, что использовались на протяжении всего времени, затем производится анализ статей за последние пять лет.

Прежде всего, необходимо начать с того, что понимается в качестве сообщества. В статье [3] говорится о том, что общепринятого определения сообщества нет, и определение часто зависит от конкретной имеющейся системы и/или приложения. Каждый алгоритм в рассматриваемых работах фокусируется на некоторых свойствах и вводит собственное определение сообщества. Например, в статье [4] упоминаются следующие способы выявления сообществ:

1.По расстоянию между вершинами (Feature Distance)
В одно сообщество входят узлы, что располагаются ближе друг к другу.

2.По внутренней плотности (Internal Density)
Внутренняя плотность в сообществе выше, чем внешняя.

3.По мостам (Bridge Detection)
Сообщества – такие «плотные» части графа, среди которых очень мало ребер, что могут разбить сеть на части, если их удалить. Мосты – те ребра, что удаляются, а компоненты сети, возникающие в результате их удаления, являются желаемыми сообществами.

4.По диффузии (Diffusion)
Сообщества – группы узлов, на которые может влиять распространение определенного свойства или информации внутри сети.

5.По близости узлов (Closeness)
Сообщество как группа узлов, которые могут достичь каждого из своих собственных компаньонов сообщества с очень небольшим количеством прыжков по краям графа, в то время как сущности вне сообщества находятся значительно дальше друг от друга.

6.По структуре ребер (Structure)
Сообщество как почти неизменяемая структура ребер. Определяется некоторый вид структуры, а затем по этому шаблону ищутся сообщества по всему графу.

7.По ссылкам
Группировка в сообщества происходит по ребрам, что рассматриваются в качестве ссылок.

8.Произвольно (No Definition)
Определяются различные операции и алгоритмы для объединения результатов различных подходов к обнаружению сообщества, а затем используется определение сообщества целевого метода для полученных результатов. Аналитик определяет свое собственное понятие сообщества и ищет его в графе.

В статьях [3], [5] определяется три варианта понятия «скрытое сообщество»:

1. Локальный
Сообщество – это подгруппа, члены которой все «друзья» друг для друга – клика.

2. Глобальный
Сообщество – подграф такой, что количество ребер внутри подграфа превышает количество ребер снаружи – понятие модулярности.

3. На основании сходства вершин
Сообщество – множество вершин, мера сходства между которыми минимальна (L_1-norm, L_∞-norm, cosine similarity, Pearson correlation и т.д.). Здесь также предлагаются такие варианты как количество независимых от ребер (или вершин) путей между двумя вершинами или на свойствах случайных блужданий по графу (resistance distance, average first passage time, PageRank).

В упомянутых статьях перечисляются следующие подходы для выявления таких сообществ:

1.Алгоритм Кернигана-Лина
Разбиение графа существует изначально. Далее происходит обмен вершинами между подмножествами имеющегося разбиения графа.

2.Спектральная кластеризация
На основании собственных значений матрицы сходства узлов, что составляется по ребрам.

3.Иерархическая кластеризация
Имеется несколько уровней по разбиению графа на сообщества, выбирается оптимальное разбиение путем анализа.

4.Уровневое разбиение структуры
Геометрический алгоритм, многоуровневые алгоритмы.

5. Ratio-Cut, normalized cut
На исходном графе делается некоторый разрез по конкретному критерию, например, по самому длинному ребру.

6. Минимальная k-кластеризация, k-сумма кластеризации, k-центр, k-медиана, нечеткая k-средняя кластеризация
Сообщества выявляются на основании «близости» вершин по некоторой метрике.

7.Алгоритм разбиения Гирван-Ньюмена
Сообщества образуются на основании центральности ребер.

А также поиск замкнутых непересекающихся путей, алгоритмы вычисления модулярности, динамические алгоритмы (Spin models, Random walk, Synchronization), алгоритмы на основании статистических выводов, методы выявления пересекающихся сообществ (Clique percolation) и другие.

Из этого можно сделать вывод, что возможно не только выявлять скрытые сообщества, но и исследовать их пересечения. Например, в статье [6] рассматриваются методы по выявлению пересекающихся сообществ. Они основываются на таких моментах как:

- Node seeds and local expansion на основании функции качества: Iterative Scan (IS), Rank Removal (RaRe), Moses, DOCS);
- Clique expansion (Clique Percolation Method, EAGLE, Greedy Clique Expansion);
- Link clustering путем разделения набора ссылок, а не набора узлов: MCL, GA-NET+, a hierarchical agglomerative link clustering method by Ahn et al);
- Label propagation: набор узлов, которые группируются в результате распространения одного и того же свойства, действия или информации в сети (COPRA, BMLPA, SLPA);
- Другие подходы:
    - путем объединения концепции модулярности, спектральной релаксации и нечетких c-means;
    - на основании визуального интеллектуального анализа данных: объекты имеют некоторые общие черты, и для оценки близости объектов друг к другу принята метрика отношений;
    - на основании коллективной точки зрения отдельных людей;
    - на основании роевой разведки (SI);
- Dynamic networks: изменения взаимосвязей с течением времени - изменения сетевой структуры на разных временных шагах (Clique Percolation Method Palla et al., iLCD, AFOCS);

По текущим статьям была создана общая таблица всех алгоритмов (Таблица 1), в которой учитываются те факторы, что связаны с исследуемой в дальнейшем моделью.

Алгоритм	**Понятие сообщества**	**Ориентированный Граф**	**Вес**	**Сложность**	**Пересечение сообществ**
Evolutionary	По расстоянию	-	-	$O(n^2)$	-
MSN-BD		-	+	$O(n^2ck)$	+
SocDim		-	+	$O(n^2 \log(n))$	-
PMM		-	+	$O(mn^2)$	-
MRGC		+	?	O(mD)	-
Infinite Relational		-	?	$O(n^{2c}D)$	-
Find-Tribes		-	?	$O(mnK^2)$	-
AutoPart		+	?	$O(mk^2)$	-
Timefall		-	?	O(mk)	-
Context-specific Cluster Tree		-	?	O(mk)	-
Modularity	По внутренней плотности	+	+	O(mk log(n))	+
MetaFac		-	?	O(mnD)	-
Variational Bayes		+	?	O(mk)	-
LA->IS[2]		+	?	O(mk+n)	+
Local Density		+	?	O(nK log(n))	-
Edge Betweenness	По мостам	+	+	$O(m^2n)$	-

CONGO		-	+	O(n log (n))	+
L-SHELL		-	?	$O(n^3)$	+
Internal-External degree		-	?	$O(n^2 \log(n))$	+
Label propagation	По диффузии	-	+	O(m+n)	-
Node Colouring		-	?	$O(ntk^2)$	-
Kirchhoff		+	+	O(m+n)	+
Communication Dynamic		+	?	O(mnt)	+
GuruMine		+	?	$O(T\ An^2)$	-
DegreeDiscount IC		+	?	O(k log(n) + m)	-
MMSB		+	?	O(nk)	+
Walktrap	По близости узлов	-	+	$O(mn^2)$	-
DOCS		-	?	?	+
Infomap		+	+	$O(m \log^2(n))$	-
k-clique	По структуре ребер	-	?	O(nk)	+
S-Plexes Enumeration		-	?	O(kmn)	+
Bi-clique		-	?	$O(m^2)$	+
EAGLE		+	+	$O(3^{n/3})$	+
Link modularity	По ссылке	-	+	O(2mk log(n))	+
HLC		-	+	$O(nK^2)$	+
Link Maximum Likelihood		-	?	O(mk)	+
Hybrid	На основе модулярности	+	+	O(nkK)	+
Multi-relational regression	На основе матрицы силы связей		+	?	-
Hierarchical Bayes	На основе дендрограмм и максимальном правдоподобии	-	?	$O(n^2)$	-
Expectation Maximization	На основе вероятности (алгоритм максимизации математического ожидания)	+	?	?	-
LFM	По соседним узлам	-	?	$O(n^2)$	+
MOSES		-	-	$O(n^2)$	+
CFinder	Поиск клик	-	-	$O(m^{\ln(m)/10})$	+
GCE	Поиск клик	-	-	O(mh)	+
Evans	На основании модулярности	-	+	O(2mk log(n))	+
GA-NET+	По максимизации community score	-	?	O(tp(m + m log(m)))	+
AHN	На основании сходства связей и плотности разбиения	-	?	$O(nd^2_{max})$	+
COPRA	На основании	-	-	O(vm	+

	меток соседних узлов			log(vm/n))	
BMLPA		-	-	O(nlog(n))	+
SLPA		+	+	O(tm)	+
CPM	Динамическое сообщество	CPMd method	CPMw method	?	+
iLCD		-	?	$O(nk^2)$	+
AFOCS		-	-	$O(d_{max}\ m) + O(n^2)$	+
CONGA	На основании кратчайших путей между вершинами	-	+	$O(m^3)$	+
Egonet	На основании эгонета	+	+	$O(n\ (\log(n))^2 + n^2\ \log(n))$	+
Swarm Egonet		+	+	$O(n\ \log^2(n))$	+
Kernighan–Lin	На основании кратчайшего пути	-	+	$O(n^2\ \log(n))$	-
spectral bisection method	На основании собственного вектора (eigenvector of the graph)	-	+	O(mn)	-
closed non-intersecting paths	На основании кратчайших путей	+	+	$O(m^4/n^2)$	-

Таблица 1. Оценка методов выявления скрытых сообществ

За последние пять лет было написано большое количество статей, что связаны с данной темой. На Рис. 1 представлен граф, на котором в качестве узлов отображаются статьи (автор и год издания), в качестве ребер – связь между данными статьями. Это говорит о том, что по сегодняшний день тема является достаточно актуальной.

Большинство статей придерживаются подхода модификации уже существующих методов. В основном говорится о том, что эти методы имеют достаточно высокую вычислительную сложность и их не стоит применять к графам со слабой связностью. Интересно рассмотреть статьи по выявлению как непересекающихся сообществ, так и пересекающихся.

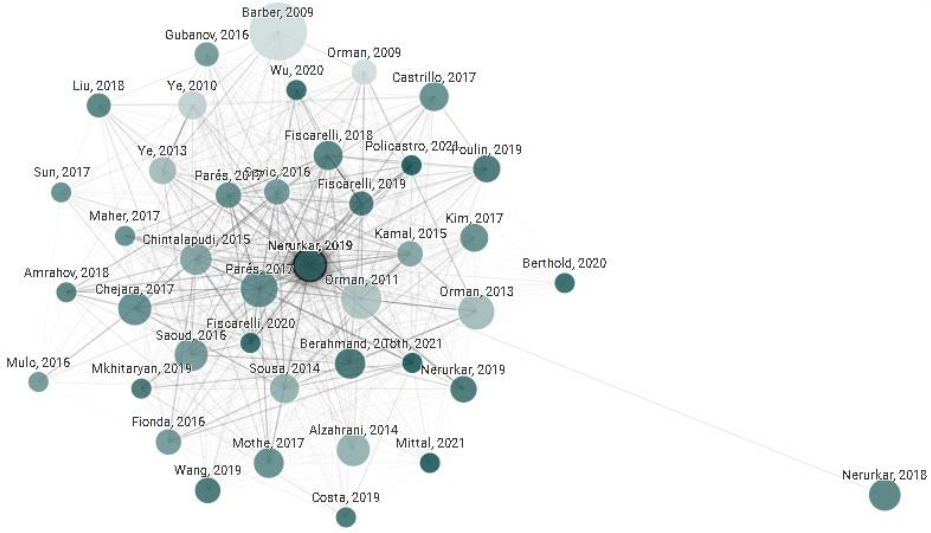


Рис. 1. Статьи, связанные с темой «A Comparative Analysis of Community Detection Algorithms on Social Networks»[1]

В статье [7] производится сравнение следующих алгоритмов:

1. InfoMap

Метод основан на случайном блуждании по графу, которое описывается кодами вершин. Описание может быть достаточно длинным. Используя принцип *географических карт*, где вершины в разных сообществах могут иметь один и тот же код. Происходит минимизация описания – оптимальное разбиение на сообщества.

2. Walktrap

Основан на итеративном формировании сообществ. Изначально каждая вершина относится к одному сообществу, затем создаются кластеры по сходству между вершинами. Сходство вычисляется как вероятность, что при случайном блуждании мы переместимся из вершины i в вершину j.

3. Fastgreedy

Модификация алгоритма разбиения на сообщества с использованием модулярности с использованием алгоритмов аппроксимации для уменьшения временной сложности.

4. Edge-betweenness

Метод, основанный на метрике центральности ребер между вершинами. Подробнее будет описано в Главе 2.

5. Label Propagation algorithm

Алгоритм с метками. Каждая вершина в графе относится к тому сообществу, к которому относится большинство его соседей.

6. Multilevel

Эвристический алгоритм для получения сообществ из графов путем оптимизации меры разбиения модульности.

[1] Источник: https://www.connectedpapers.com

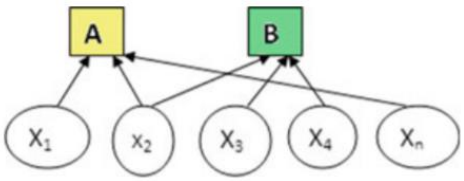
7. Statistical Mechanics of Community Detection

На основании динамики вращений [8].

8. Leading Eigenvector

Строится матрица модулярности, ищутся собственные значения и вектора. Наибольшие собственные значения - максимизация модулярности сети, а значит оптимальный способ разбиения на сообщества.

В статье [9] предлагается алгоритм BIGCLAM. В основе данного алгоритма – модель графа принадлежности и атрибут. Предполагается, что люди объединяются в одно сообщество за счет общих интересов, совпадение атрибутов. Строится двудольный граф, на верхнем уровне которого находятся узлы атрибутов, на нижнем уровне – исходные вершины графа (Рис. 2).



.

Рис. 2. Двудольный граф принадлежности атрибутов

Между узлами долей строится дуга в случае, когда вершина из нижнего уровня соответствует атрибуту из верхнего уровня. Таким образом, по каждому из атрибутов формируются сообщества.

В статье [10] рассматривается модификация алгоритма LPA с использованием памяти – MemLPA («Mem» от слова «memory»). Здесь правило принятия решений учитывает прошлое состояние сети, что берется из памяти. На Рис. 3 представлена логика работы алгоритма. На каждой итерации каждый узел собирает метки своих соседей и обновляет свой список меток в соответствии с весом (для взвешенных сетей) и предпочтениями узла. Если появляется новая метка, в списке создается новая запись, в противном случае счетчик для соответствующей метки увеличивается. Затем каждый узел выбирает метку из списка, используя правило принятия решений, которое основывается на выборе максимального элемента из столбца.

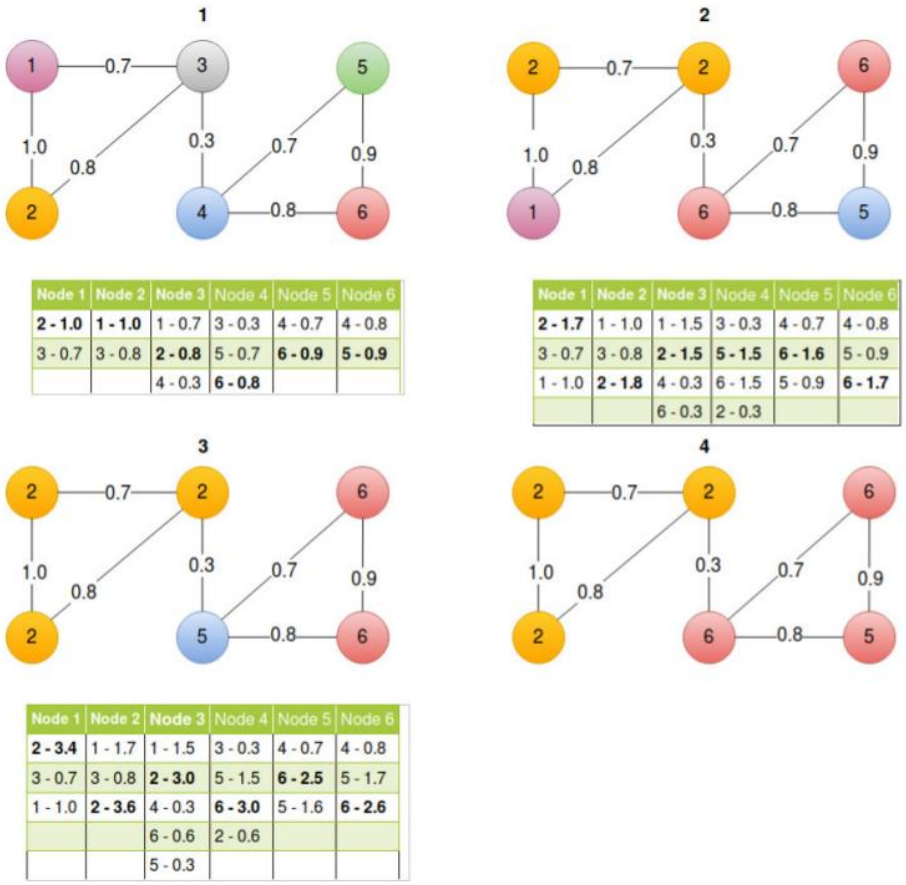


Рис. 3. Итерации MemLPA на взвешенном неориентированном графе

В статье [11] вводится алгоритм DAHCA, основанный на матрице достижимости[2]. Разбиение на кластеры происходит за счет использования идеи, что внутри кластера связь между вершинами сильнее, чем связь с внешними вершинами.

В работе [12] рассматриваются два алгоритма, основанные на максимальной степени вершины и соседних узлах:

1. **SAS-1**

t = 1: Берется вершина с максимальной степенью и объединяется со всеми соседями в одно сообщество.

t = 2: В каждом из сообществ снова ищется узел с максимальной степенью, берутся его соседи и относятся к одному сообществу в случае, если:

$$S_{c_m(1)}^{(j)} = \frac{s_{c_m(1)}^{(j)}}{k_j} \geq \frac{1}{2},$$

где $c_m(1)$ – сообщество, на которое разбили на первом шаге, k_j – степень узла j,

$$s_{c_m(1)}^{(j)} = \left|N_j \cap c_m(1)\right|, \text{ где } N_j - \text{узел } j.$$

[2] В матрице достижимости хранится информация о существовании путей между вершинами орграфа

t > 2: Вершина с максимальной степенью ищется из множества $C_m^{new}(t)$ и из соседей, что не принадлежат этому сообществу по показателю

$$S_{c_m(t-1)}^{(j)} = \frac{s_{c_m(t-1)}^{(j)}}{k_j} \geq \frac{1}{2}$$

отбираются те, что принадлежат этому сообществу в дальнейшем.

2. SAS-2

Аналогичен предыдущему алгоритму, только используются следующие формулы:

$$S_{N_i}^{(j)} \geq \gamma \cdot \max_{j \epsilon N_i} \left\{ s_{N_i}^{(j)} \right\};$$

$$S_{c_m(t-1)}^{(j)} = \max_{j \epsilon N_i} \left\{ S_{c_m(t-1)}^{(j)} \right\};$$

$$s_{c_m(t-1)}^{(j)} = \max_{j \epsilon N_i} \left\{ s_{c_m(t-1)}^{(j)} \right\},$$

где γ – пороговое значение $\gamma_{N_i}^{j}$,

$$\gamma_{N_i}^{j} = \frac{s_{N_i}^{(j)}}{\max_{j \epsilon N_i} \left\{ s_{N_i}^{(j)} \right\}}, j \in N_i$$

По пересечению сообществ предлагаются алгоритмы DNTM [13], LPX [14], Centrality Measure & Local Seed Information [15].

DHTM

Сначала генерируются некоторые непересекающиеся кластеры. Далее находится *Overlapping Candidate Node* $O_{cn}(v)$:

$$O_{cn}(v) = {}_{def} NC(v) \neq \emptyset,$$

где $NC(v) = {}_{def}\{C_i \in C \setminus C_j : \exists y \in C_i \wedge y \in N_r(x)\}$. $N_r(x)$ – все соседние узлы вершины x.

Затем вычисляется порог распределенного соседства:

$$D_t(v) = {}_{def} \left| \frac{|N_r(v)|}{|NC(v)| + 1} \right|$$

Далее происходит обнаружение узла пересечений $ON(v)$:

$$ON(v) = {}_{def} O_{cn}(v) \wedge (D_t(v) \leq |\{y : y \in N_r(v) \wedge y \in C_i\}|)$$

После этого обновляются кластеры по $ON(v)$ и получаются пересекающиеся кластеры.

LPX

В основе данного алгоритма – модификация *X-means* [16], что в свою очередь является модификацией *k-means,* и использование алгоритма LPA. Идея заключается в том, что рассматриваются не только атрибуты, но их подпространства. В своем простом и эффективном алгоритме авторы решают проблему нестабильности разделения и легкого слияния соседних сообществ.

Centrality Measure & Local Seed Information

Изначально вершины разбиваются на непересекающиеся кластеры. Затем для каждого кластера находится вершина с максимальным значением собственного вектора. С использованием алгоритма PageRank [17] к найденным ранее вершинам находится расширение. После расширения получается набор узлов, из которого формируется сообщество по минимальному показателю проводимости сети.

Обобщая вышесказанное, можно подчеркнуть, что в основе выявления скрытых сообществ в последнее время используются и модифицируются следующие подходы:

- Метрика центральности (eigenvector centrality, edge betweenneess);
- Атрибуты узлов;
- Случайное блуждание;
- Распространение меток;
- Метрика модулярности;
- Модификация k-means;
- Гибридный подход: комбинация нескольких методов.

По большому количеству способов выявления скрытых сообществ можно сделать вывод, что данная проблема является достаточно актуальной на сегодняшний день. Однако если рассматривать конкретно социальную сеть YouTube, которая относится к списку популярных социальных сетей, – здесь очень мало исследований, направленных на выявление скрытых сообществ.

Путем анализа существующих подходов по следующим критериям:

- сложность вычисления;
- тип графа;
- сложность реализации;

был сделан вывод, что данные алгоритмы не подходят для крупномасштабных сетей или сетей со слабой связностью.

Глава 2. Алгоритмы выявления скрытых сообществ

Путем анализа сложности реализации и особенностей вычислений было решено рассмотреть следующие алгоритмы:

1. Алгоритм поиска ядер графа;
2. Алгоритм поиска клик графа;
3. Алгоритм поиска кластеров графа на основе MST (**M**inimum **S**panning **T**ree);
4. Алгоритм поиска кластеров графа на основе NAG (**N**ewman **A**nd **G**irvan);

Идея исследовательской работы заключается в том, что ядра, клики и кластеры графа рассматриваются в качестве скрытых сообществ. Данные алгоритмы составляются путем реализации и модификации уже существующих алгоритмов. Перед описанием каждого из алгоритмов необходимо ввести математические обозначения.

2.1. Математические обозначения

Прежде всего, необходимо построить математическую модель, на основе которой далее будет сформулирована задача компромиссного решения. Для решения различных задач, связанных с социальными группами, используется теория графов, так как она предоставляет возможность создавать из данных некоторую структуру, а затем анализировать ее, применяя разные математические алгоритмы. В основе данной работы лежит *теория социальных графов*. В качестве узла $v_j, j = \overline{1,k}$ понимается пользователь (канал в социальной сети YouTube), в качестве дуги $e_{jl} = (v_j, v_l)$ – наличие связи между пользователями j и l. Под «связью» здесь понимается наличие комментария от одного пользователя к другому под каким-либо видео. Для каждой дуги e_{jl} определяется вес w_r как суммарное количество комментариев, адресованных данному пользователю по всем рассматриваемым видео. Обозначим в качестве V множество узлов: $V = \{V_1, V_2, \dots, V_k\}$, в качестве E – множество ребер: $E = \{E_1, E_2, \dots, E_l\}$, $W = \{W_{11}, W_{12}, \dots, W_{rp}\}$ – множество весов. Паре узлов ставится в соответствие ребро, а каждому ребру ставится в соответствие вес. Математическая модель представляет собой упорядоченную тройку множеств $G = (V, E, W)$.

На Рис.4 представлена логика связи между пользователями, что комментируют видео. Пользователи B и C комментируют видео пользователя A, а другие пользователи отвечают на комментарии пользователей B и C.

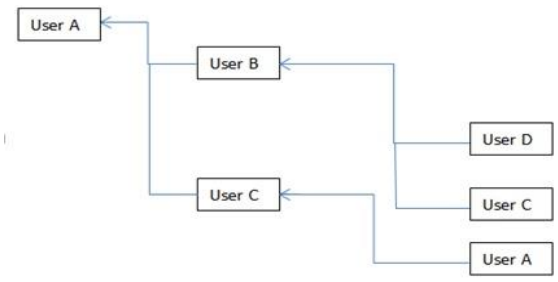


Рис.4. Схема построения слоев комментариев

2.2. Алгоритм поиска ядер графа

В случае выявления скрытых сообществ интересным является такое понятие, как ядро графа. Сначала необходимо ввести понятия *внутренне устойчивого* и *внешне устойчивого множества* вершин графа [18].

Определение 1. Внутренне устойчивое множество вершин – подмножество множества вершин, каждый элемент которого не смежен с любым другим элементом.

Определение 2. Внешне устойчивое множество вершин – множество, в котором либо любая вершина входит в это множество, либо вершина не входит в это множество, но из этой вершины есть дуга в данное множество.

Определение 3. Максимальное внутренне устойчивое множество вершин – множество вершин, которое невозможно расширить путем добавления другой вершины.

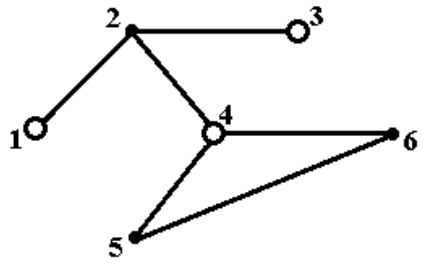


Рис. 5. Максимальное по включению внутренне устойчивое множество {1,3,4}

Определение 4. Минимальное внешне устойчивое множество вершин – множество вершин, при удалении из которого одной вершины устойчивость нарушается.

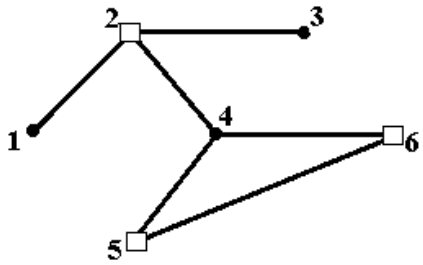


Рис. 6. Внешне устойчивое множество {2,5,6}, не минимальное по включению

Определение 5. Ядро графа – подмножество множества вершин исходного графа, узлы которого являются одновременно внешне и внутренне устойчивыми.

Утверждение 1. Множество вершин Z является ядром графа G тогда и только тогда, когда существуют минимальное по включению внешне устойчивое множество X и максимальное по включению внутренне устойчивое множество Y такие, что:

$$X \subseteq Z \subseteq Y$$

Для поиска текущих множеств используется алгоритм Магу. Алгоритм Магу для поиска минимальных внешне устойчивых множеств вершин состоит из следующих шагов:

1. Составляется матрица смежности;
2. По главной диагонали проставляются единицы;
3. Для каждой строки выписываются дизъюнкции;
4. Сокращаются полученные множители (приводятся к дизъюнктивной нормальной форме);
5. Все вершины, входящие в элементарную конъюнкцию, образуют множество внешней устойчивости.

Алгоритм Магу для поиска максимальных внутренне устойчивых множеств вершин:

1. Составляется матрица смежности;
2. По матрице выписываются все парные дизъюнкции;
3. Сокращаются полученные множители (приводятся к дизъюнктивной нормальной форме);
4. Элементы, которых не хватает в каждой конъюнкции, входят во множество внутренней устойчивости.

Алгоритм поиска ядер графа представляет собой следующие шаги:

1. Находятся внутренне устойчивые множества вершин

$$A = \{A_1, A_{2,} \dots, A_n\};$$

2. Находятся внешне устойчивые множества вершин

$$B = \{B_1, B_{2,} \dots, B_m\};$$

3. Находятся ядра графа как

$$C = A \cap B = \{C_1, C_{2,} \dots, C_k\}.$$

Множество ядер графа может как состоять из нескольких элементов, так и быть пустым (ядер не существует).

В ходе реализации данного алгоритма возникла проблема с производительностью. Обусловлено это квадратичной сложностью алгоритма. Больше всего времени уходит на приведение выражения к дизъюнктивной нормальной форме (перемножение дизъюнкций). В связи с этим было решено оптимизировать вычисления, опираясь на использование битовых шкал [19].

Битовые шкалы

Пусть задано некоторое конечное множество U, называемое универсумом.

$$U = \{u_1, u_2, \dots, u_n\},$$

где n – количество элементов данного множества.

Определение 6. Битовая шкала – последовательность символов $A = \{a[i]\}_{i=1}^{n}$, $a \in [0; 1]$, такая, что

$$a[i] = \begin{cases} 0, \text{если } u_i \notin A \\ 1, \text{если } u_i \in A \end{cases}$$

Рассмотрим применение битовой шкалы к математической модели и вычислениям.

Относительно рассматриваемой модели это множество U интерпретируется как множество рассматриваемых вершин графа $G = G(V, E, W)$, $n = |V|$. Каждая вершина нумеруется от 1 до n. Пусть есть дизъюнкция вида $A \vee B$, где

$$A = \bigwedge_{i=1}^{k} x_i,\ B = \bigwedge_{j=1}^{l} x_j,$$

Тогда A и B можно представить в виде битовой шкалы, где «1» интерпретируется как «присутствует в A (B)», «0» – «отсутствует в A (B)».

Пример. Пусть n = 5.

$$A = x_1 x_2 x_3,$$

$$B = x_2 x_3 x_5$$

Тогда битовое представление $A \vee B$ имеет вид:

$$C = A \vee B = x_1 x_2 x_3 \vee x_2 x_3 x_5 = 11100 \vee 01101$$

Представляя данные в таком виде, логические операции описываются просто:

- пересечение A и B – логическое произведение кодов множеств;
- объединение A и B – логическая сумма кодов множеств.

В результате применения операций мы получаем еще один битовый код.

Такое представление для множеств вершин дает возможность вычислять текущие операции быстрее по сравнению с такими же операциями для списков или массивов.

2.3. Алгоритм поиска клик графа

Прежде чем рассмотреть алгоритм, определим, что понимается в качестве *клики* графа.

Определение 7. Клика графа – подмножество множества вершин графа, в котором любые две вершины являются смежными.

Данное определение относится к случаю неориентированного графа. Необходимо уточнить определение для рассматриваемого ориентированного графа.

Определение 8. Клика ориентированного графа – подмножество множества вершин графа, в котором у любых двух вершин поддерживается связь типа $\boldsymbol{V_i \leftrightarrow V_j}$, что означает наличие дуги в двух направлениях между вершинами.

Помимо условия из определения есть еще одно – клика должна включать в себя максимальное подмножество вершин, то есть клику нельзя расширить.
Смысл выделения клик заключается в том, что мы определяем достаточно сильные связи – такие сообщества, в которых каждый пользователь контактируют с каждым. Задача поиска скрытых сообществ сводится к задаче поиска максимальных по включению клик.

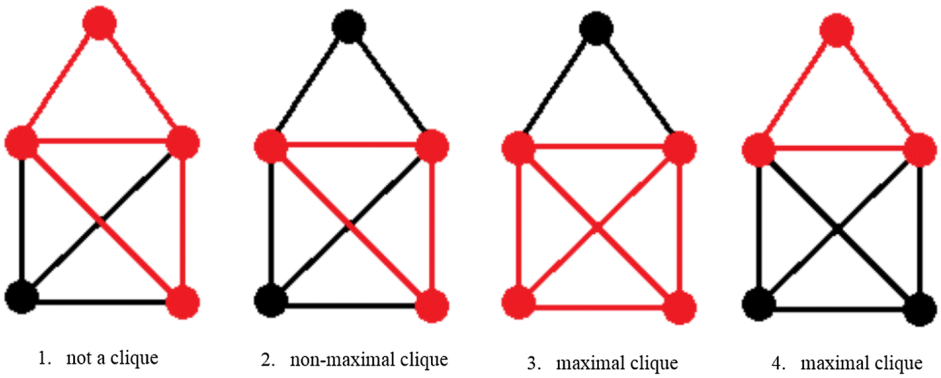


Рис. 7. Разные типы клик

Рассмотрим алгоритм Брона-Кербоша [25]. Алгоритм является простым в вычислении, так как его сложность линейна относительно числа клик в графе. Введем следующие обозначения:

$\boldsymbol{V}$ – множество вершин, $\boldsymbol{v}$ – рассматриваемая вершина;
$\boldsymbol{M}$ – множество вершин клики;
$\boldsymbol{K}$ – множество рассматриваемых кандидатов (тех, кто может быть добавлен в $\boldsymbol{M}$);
$\boldsymbol{P}$ – множество просмотренных вершин, что не добавлены во множество $\boldsymbol{M}$.

Алгоритм Брона-Кербоша имеет вид:

1. В качестве кандидатов рассматривается исходное множество вершин V:
$$K = V$$
2. Пока K не пусто и P не содержит вершины, соединенной со всеми вершинами из K
   2.1. Удаляется первый элемент v из множества K;
   2.2. Заносится v в M;
   2.3. Удаляются из K, P все вершины, несмежные с v;
   2.4. Если K и P пусто: вывод M;
   2.5. Иначе рекурсивно вызывается метод для текущих P и K ;
   2.6. Удаляется последний элемент из множества M;
   2.7. Возвращение к той строке, где M состоит из того множества вершин, что осталось на текущем шаге;
   2.8. Удаление из множества K текущей вершины;
   2.9. Добавление текущей вершины во множество P;

Таким образом, выделяется множество кластеров, что включают в себя максимальное количество вершин. Одна клика рассматривается как одно скрытое сообщество, а одна вершина может принадлежать сразу нескольким кликам.

2.4. Алгоритмы поиска кластеров графа

Данный алгоритм предназначен для разделения исходного графа на фиксированное количество кластеров K – непересекающиеся множества узлов. Рассматривается два подхода – Minimum Spanning Tree и алгоритм Гирван-Ньюмена.

2.4.1. MST

Суть алгоритма [26] заключается в том, что сначала ищется минимальное покрывающее дерево (**M**inimum **S**panning **T**ree), а затем удаляется (K - *1*) ребро с максимальным весом**.**

Определение 9. Под минимальным покрывающим деревом понимается подграф графа, состоящий из того же множества вершин и такого множества ребер, что между любыми двумя вершинами существует маршрут, и отсутствуют циклы.

Для поиска такого дерева существуют алгоритмы Краскала и Прима. Был выбран алгоритм Краскала и проведена его модификация:

1. Исходный ориентированный граф приводится к неориентированному;
2. Формируется список отсортированных по убыванию веса ребер;
3. Для каждого элемента текущего списка;
   3.1. Проверка: если добавить текущее ребро в дерево, не образуется ли цикла. Если нет – добавляется в дерево, иначе – пропуск;
4. В построенном дереве удаляется (*K - 1*) ребро с минимальным весом.

Замечание 1. Ориентированный граф сводится к неориентированному за счет того, что между двумя вершинами строится ребро, весом которого является суммарное значение входящего и исходящего ребра (если нет одного из них, удаляем ребро).

Замечание 2. В рассматриваемой модели важен большой вес дуги, а не маленький. В связи с этим происходит сортировка по убыванию и удаление ребер с минимальным весом. Такие ребра считаются неважными.

Замечание 3. Параметр *K* необходимо задать заранее. Однако можно найти его путем подбора, используя верхнюю и нижнюю границу количества вершин в одном кластере.

2.4.2. NAG

Алгоритм разработан американским математиком Мишель Гирван (Michelle Girvan) и британским физиком Марком Ньюменом (Mark Newman) [27]. Его суть заключается в том, что кластеры находятся путем удаления (K-1) ребер из исходной сети. Однако ребра удаляются не случайно, а по максимальному значению степени посредничества (Edge Betweenness).

Определение 10. Edge Betweenness – количество кратчайших путей, проходящих через текущее ребро:

$$C_B(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}},$$

где $\sigma_{st}(e)$ – количество кратчайших путей от узла s до узла t, что проходят по текущему ребру e, σ_{st} – общее количество кратчайших путей от узла s до узла t.

Алгоритм **NAG** имеет вид:

1. Вычисляются степени посредничества всех рёбер.
2. Ребро с наибольшей степенью посредничества удаляется.
3. Степени посредничества всех затронутых рёбер вычисляются заново.
4. Шаги 1 - 3 повторяются до тех пор, пока остаются рёбра или пока не удалено (K-1) ребро.

Ребро, через которое проходит наибольшее количество кратчайших путей, является «мостом» между кластерами. Каждый раз метрика пересчитывается, так как при удалении ребер меняется значение. Алгоритм является жадным и эвристическим. В итоге получаем множество кластеров и из них выбираем наиболее интересные кластеры.

Замечание 4. В математической модели чем больше вес ребра, тем крепче связь между пользователями. Поэтому здесь в качестве кратчайшего пути понимается путь с большим суммарным значением весов, а не меньшим. В связи с этим при вычислении степени посредничества ребра вес ребра графа заменяется на противоположный.

Замечание 5. Так как в результате вычислений $C_B(e)$ мы получаем отрицательные значения, необходимо поменять их на противоположный знак для поиска из них максимального значения – наибольшей степени посредничества.

Глава 3. Компромиссное решение

Рассматривается математическая модель графа $G = (V, E, W)$ и множество алгоритмов $A = \{A_1, A_2, A_3, A_4\}$ с помощью которых можно выявить скрытые сообщества в социальной сети YouTube.

Пусть имеется некоторое множество агентов, заинтересованных в выявлении скрытых сообществ $I = \{I_1, I_2, \dots, I_n\}$. В качестве агента можно понимать как физическое лицо, так и какую-либо организацию, например, группу программистов. Будем считать, что каждый программист ознакомился с результатами работы алгоритма $A_t, t = \overline{1,p}$ и принял решение о том, насколько качественный результат выдает данный алгоритм (дал некоторую оценку от 1 до 5). Агенты выделяют определенное количество ресурсов для каждого из алгоритмов, то есть, каждому ресурсу ставится в соответствие вес. В качестве ресурса здесь можно понимать, например, сервер или область памяти для хранения данных.

Между множеством агентов возникает конфликт, что основан на разнице в оценках для каждого из алгоритмов. Например, программист I_1 самым оптимальным считает алгоритм A_3, программист I_2 – алгоритм A_1, программист I_3 – алгоритм A_2, программист I_4 – алгоритм A_4, программист I_5 – алгоритм A_1. В таком случае необходимо найти так называемое компромиссное решение – оптимальный алгоритм выявления скрытых сообществ – для группы агентов, чтобы в сумме все остались довольны. Целью выбора конкретного алгоритма является выделение для него необходимого количества ресурсов для получения качественного результата по выявлению скрытых сообществ в социальной сети.

Обозначим множество вариантов за $X = \{x_1, x_2, \dots, x_p\}$, где p – количество всех предлагаемых алгоритмов по выявлению скрытых сообществ, и назовем это *множеством допустимых решений*. За решение здесь принимается выбор конкретного алгоритма $x^* = x_k$, в который все агенты внесут свой вклад.

Введем функцию полезности $H_j(x_l)$ такую, что $H_j(X): X \to R^1, j = \overline{1,n}$, $i = \overline{1,p}$. Из значений данной функции составляется матрица полезности:

$$\begin{pmatrix} H_1(x_1) & \cdots & H_j(x_1) \\ \vdots & \ddots & \vdots \\ H_1(x_p) & \cdots & H_j(x_p) \end{pmatrix}$$

Введем понятие компромиссного решения поставленной задачи. В качестве $M_i, i = \overline{1,p}$ возьмем максимальный доход, который получает агент I_k в результате применения алгоритма A_j:

$$M_i = \max_{x \in X} H_j(x), \qquad j = \overline{1,n}$$

Далее строится идеальный вектор $M = (M_1, M_2, \dots M_p)$, каждый элемент которого определяет максимальный доход для каждого агента. Введем вектор невязки, элементом которого является величина $M_i - H_i(x)$ для $\forall x \in X, i = \overline{1,p}$. Элементы вектора упорядочиваются по возрастанию. Самым первым элементом текущего вектора является величина, которая показывает, что разницы между максимальным доходом и значением

функции полезности нет. Самый последний элемент вектора указывает на так называемого «обиженного» агента, то есть определяется такой агент, у которого разница между максимальным доходом и значением функции полезности максимальна:

$$\max_{i \in I}\left(M_i - H_i(x)\right), \qquad i = \overline{1,p}$$

Под компромиссным множеством понимается C_X^H:

$$\mathrm{C}_X^H = \underset{x \in X}{\operatorname{argmin}} \max_{i \in I}(M_i - H_i(x)), i = \overline{1,p}$$

Сформулируем алгоритм по нахождению компромиссного решения:

1. Формируем таблицу полезности, где каждый *i*-й агент ставит в соответствие *j*-му алгоритму вес w_r.
2. Формируем таблицу, в которой в качестве строк выступает агент, в качестве столбцов – алгоритм – допустимое решение $x_i, i = \overline{1,p}$, а в качестве значений определяется значение функции полезности.
3. По построенной таблице для каждого x_i определяем максимальный выигрыш M_J и формируем идеальный вектор $M = (M_1, \dots, M_n)$.
4. Формируем таблицу невязкости, где в качестве значений берется $M_i - H_i(x)$ для $\forall x \in X, i = \overline{1,p}$.
5. Из данной таблицы для каждого $x_i, i = \overline{1,p}$ определяем максимальное значение отклонения:

$$\max_{i \in I}\left(M_i - H_i(x)\right), \qquad i = \overline{1,p}$$

6. Из полученных значений выбираем минимальное:

$$\mathrm{C}_X^H = \underset{x \in X}{\operatorname{argmin}} \max_{i \in I}(M_i - H_i(x)), i = \overline{1,p}$$

Возможен случай, когда найдено несколько оптимальных решений $x^* = \{x_1^*, \dots, x_q^*\}$. В этом случае применяется тот же алгоритм, но уже к текущим решениям ${}^s\mathrm{C}_X^H$:

$${}^1\mathrm{C}_X^H = \underset{x \in X}{\operatorname{argmin}} \max_{i \in I}(M_i - H_i(x)),\ i = \overline{1,p}$$

$${}^2\mathrm{C}_X^H = \underset{x \in {}^1\mathrm{C}_X^H}{\operatorname{argmin}} \max_{i \in I}(M_i - H_i(x)),\ i = \overline{1,p}$$

…

$${}^k\mathrm{C}_X^H = \underset{x \in {}^{k-1}\mathrm{C}_X^H}{\operatorname{argmin}} \max_{i \in I}(M_i - H_i(x)),\ i = \overline{1,p}$$

Компромиссное решение имеет вид:

$$x^* \epsilon\, X = \arg\left({}^k\mathrm{C}_X^H\right),\ \text{при } \dim\left({}^k\mathrm{C}_X^H\right) = 1$$

Таким образом, мы определили оптимальный вариант выявления скрытых сообществ для всех агентов, то есть нашли *компромиссное решение* исходной задачи.

Рассмотрим пример для лучшего понимания алгоритма поиска компромиссного решения.

Пусть у нас есть n = 5 агентов $I = \{I_1, I_2, I_3, I_4, I_5\}$. Имеется результаты работы p = 4 алгоритмов $A = \{A_1, A_2, A_3, A_4\}$. Каждый агент дал оценку результатам алгоритмов (Таблица 2)

Агент	**Алгоритм 1**	**Алгоритм 2**	**Алгоритм 3**	**Алгоритм 4**
I_1	2	4	5	3
I_2	1	3	4	5
I_3	1	2	3	4
I_4	4	2	3	5
I_5	2	5	5	3

Таблица 2. Оценка алгоритмов агентами

По каждому алгоритму (столбцу) ищем максимальное значение (Таблица 3)

Агент	**Алгоритм 1**	**Алгоритм 2**	**Алгоритм 3**	**Алгоритм 4**
I_1	2	4	**5**	3
I_2	1	3	4	**5**
I_3	1	2	3	4
I_4	**4**	2	3	5
I_5	2	**5**	5	3

Таблица 3. Поиск максимального значения по столбцам

M = {4, 5, 5, 5}. Строим таблицу невязкости (Таблица 4) и из отклонений выбираем максимальное

Агент	$\boldsymbol{M_1 - H_1(x)}$	$\boldsymbol{M_2 - H_2(x)}$	$\boldsymbol{M_3 - H_3(x)}$	$\boldsymbol{M_4 - H_4(x)}$
I_1	4 – 2 = 2	1	0	**2**
I_2	4 – 1 = **3**	2	1	0
I_3	4 – 1 = 3	**3**	**2**	1
I_4	4 – 4 = 0	3	2	0
I_5	4 – 2 = 2	0	0	2

Таблица 4. Построение таблицы невязкости, поиск отклонений

Максимальные отклонения = {3, 3, 2, 2}, из них выбираем минимальное значение, их получается два: 2 и 2 => оптимальное решение состоит из двух алгоритмов: x_3, x_4. Далее работаем с таблицей, состоящей только из этих алгоритмов (Таблица 5).

Агент	Алгоритм **3**	Алгоритм **3**
I_1	**5**	3
I_2	4	**5**
I_3	3	4
I_4	3	5
I_5	5	3

Таблица 5. Новая таблица по оставшимся алгоритмам

Аналогично строим таблицу невязкости (Таблица 6) и ищем максимальные отклонения.

Агент	$\boldsymbol{M_3 - H_3(x)}$	$\boldsymbol{M_4 - H_4(x)}$
I_1	0	**2**
I_2	1	0
I_3	**2**	1
I_4	2	0
I_5	0	2

Таблица 6. Таблица невязкости для оставшихся алгоритмов

Видно, что решение остается прежним. Так как образуется цикл, возможно два варианта решения:

1. Сделать переоценку этих алгоритмов (задать новые значения функции полезности);
2. Выбрать любой из вариантов множества оптимальных решений.

Выбор зависит от точки зрения самих агентов.

Глава 4. Программная реализация

4.1. Архитектура

Для выполнения сформулированных задач исследовательской работы был выбран язык программирования Java, библиотека для визуализации графа *Gephi*[3], библиотека для использования и модификации алгоритма NAG – *JUNG*[4]. В качестве СУБД используется PostgreSQL.

Программа состоит из двух модулей, которые взаимодействуют друг с другом.

Первый модуль отвечает за следующие функции:

1.Регистрация/вход в систему агента;

2.Формирование запроса ко второму модулю на получение результатов работы конкретного алгоритма с параметром;

3.Отправка оценок по каждому алгоритму на сервер (запрос в БД);

4.Вычисление компромиссного решения на основе существующих агентов и оценок.

Второй модуль отвечает за следующие функции:

1.Запуск одного из выбранных алгоритмов по запросу агента;

2.Визуализация полученных результатов;

3.Сохранение/восстановление вычисленных алгоритмами данных;

На Рис. 8 представлена архитектура модуля по сбору и сохранению данных с социальной сети YouTube.

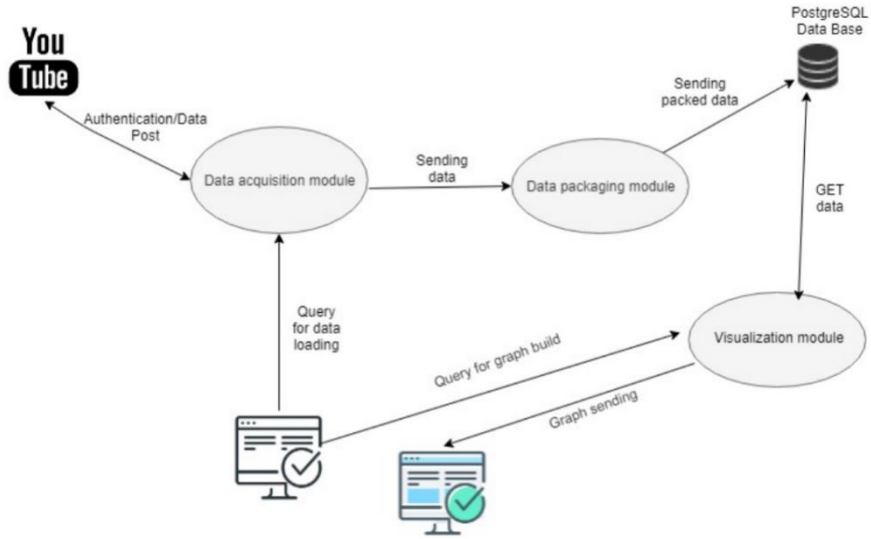


Рис. 8. Архитектура второго модуля

[3] Gephi 0.9.2 API: https://gephi.org/gephi/0.9.2/apidocs/overview-summary.html
[4] Graph framework: http://jung.sourceforge.net

На Рис. 9 представлена архитектура программы, основанная на взаимодействии двух модулей.

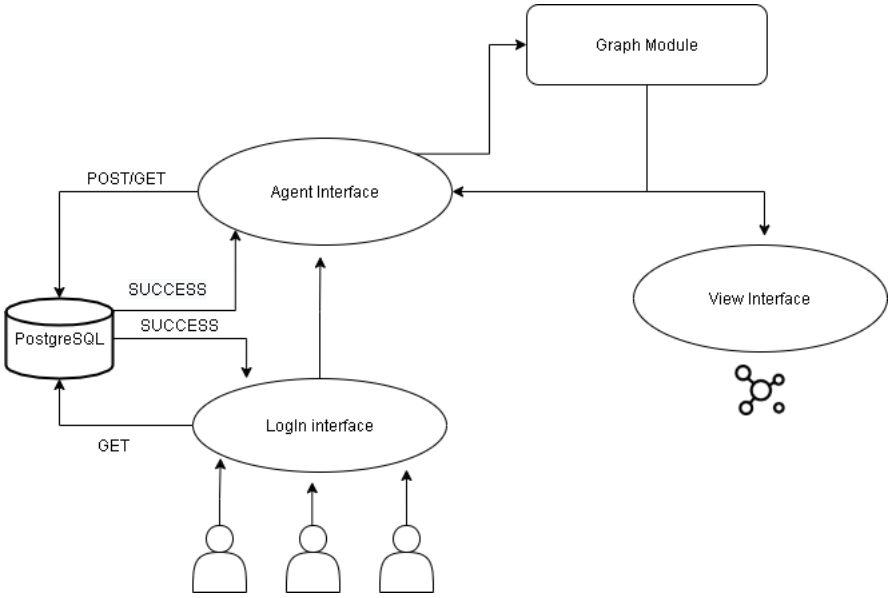


Рис. 9. Архитектура программы

4.2. Обработка данных

Блок программы по сбору данных был модифицирован за счет использования многопоточности, а именно фреймворка *Akka*[5]. Данный фреймворк основан на модели акторов.

Определение 10. Актор – сущность, которая принимает сообщения и может отреагировать на них за счет следующих действий:

- Отправить N сообщений другим акторам (помимо себя);
- Породить M новых акторов;
- Изменить внутреннее состояние;

Многопоточность достигается за счет порождения нескольких одинаковых экземпляров актора.

Модель использует такие понятия как messages, dispatcher и mailboxes. Messages – сообщения, что отправляются от актора асинхронно. Dispatcher – диспетчер, что отвечает за помещение акторов на исполнение на пуле потоков. Mailboxes – очередь сообщений для конкретного актора.

Выделяется главный актор и второстепенные. Главный актор принимает на вход массив и раздает элементы массива для второстепенных акторов, которые выполняют задачу и возвращают главному актору результат.

Данный фреймворк использовался и в реализации алгоритма поиска ядер графа.

[5] https://akka.io/

Помимо базы данных с данными социальной сети YouTube была создана база данных для хранения информации об агентах. Схема БД представлена на Рис. 10.

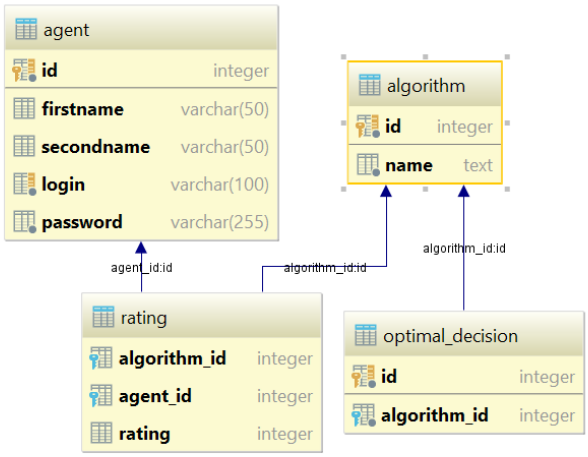


Рис. 10. Схема БД агентов

База данных состоит из следующих таблиц:

- agent – данные об агенте для входа в систему;
- algorithm – данные об алгоритме выявления скрытых сообществ;
- rating – данные об оценке для каждого алгоритма;
- optimal_decision – данные об оптимальном решении для множества агентов.

Текущая база данных используется с целью предоставления возможности идентификации агента, сохранения его оценок для каждого алгоритма и хранения компромиссного решения.

На Рис. 11 представлен вход агента в систему. Если агент не зарегистрирован, ему предоставляется такая возможность.

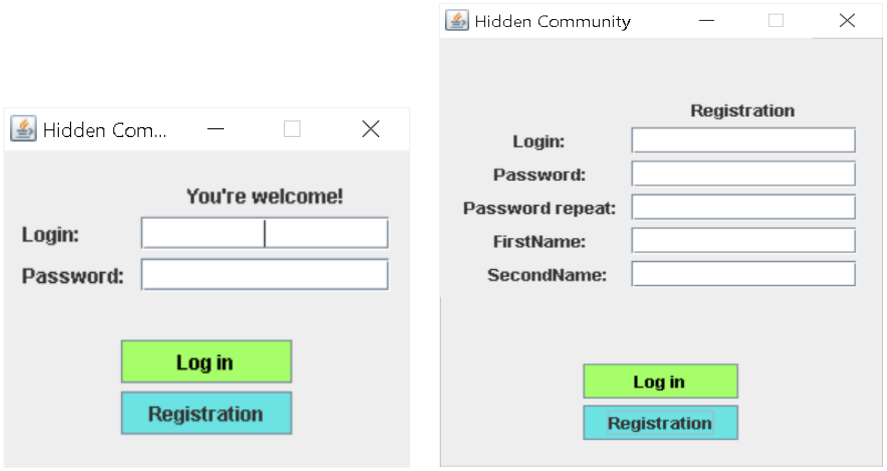


Рис. 11. Реализация входа в систему для агента

На Рис. 12 представлен главный интерфейс продукта. Для каждого алгоритма предоставляется краткое описание и поле для ввода главного параметра – степени вершин k. Агент выбирает интересующий его алгоритм, задает параметр и получает результат в виде графа и таблицы данных (Рис. 13). Если данные вычисления уже происходили, то файл загружается из специальной папки, что ускоряет процесс визуализации.

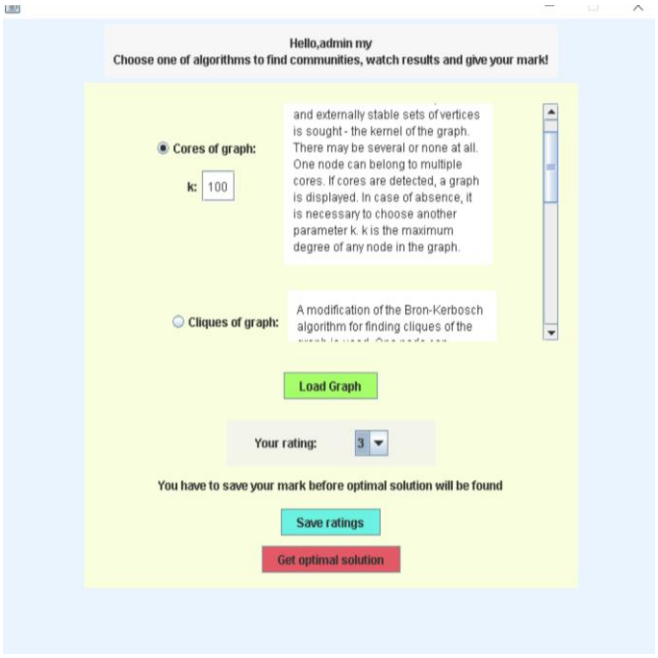


Рис. 12. Главный интерфейс

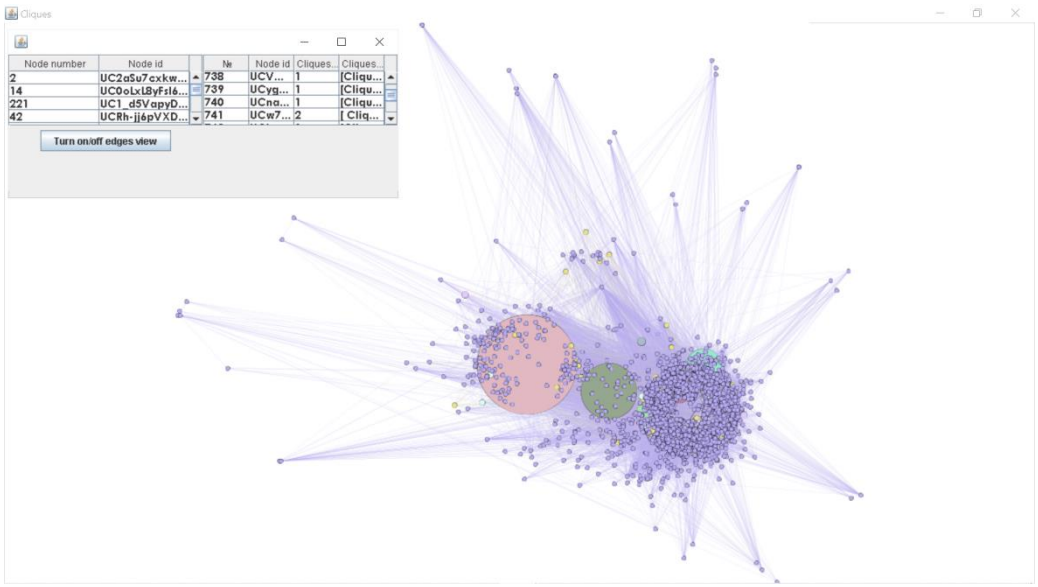


Рис. 13. Пример визуализации результата

У агента есть возможность включать и выключать отображение дуг, что улучшает качество оценки полученного результата. С модулями программного решения можно ознакомиться на сайте github[6] .

[6] Первый модуль: https://github.com/Snezzz/AgentsInterface, второй модуль: https://github.com/Snezzz/YouTubeGetData

Глава 5. Апробация

Для анализа работы программного продукта были собраны данные за 2020 год с социальной сети YouTube по специально подобранным группой социологов белорусским каналам, связанным с политикой:

- «Белсат News»;
- «Гарантий Нет»;
- «NEXTA»;
- «Народный Репортёр»;
- «Покиньте Вагон»;
- «Рудабельская Паказуха».

Для качественной визуализации использовался алгоритм укладки графа OpenOrd, основанный на физических понятиях силы притяжения и отталкивания [28].

В результате получен взвешенный ориентированный граф, состоящий из **75 588** вершин и **215 628** дуг (Рис. 14). Для смысловой начальной визуализации была применена укладка графа OpenOrd и использованы вычисления модулярности. Видно, что выделены некоторые сообщества, и интересно в дальнейшем исследовать расположение их участников.

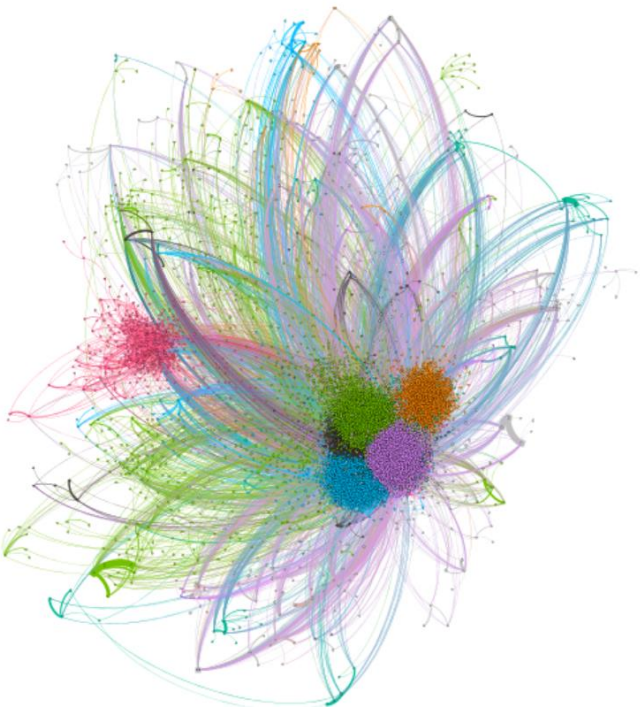


Рис. 14. Исходный ориентированный граф с укладкой OpenOrd
и разбиением на сообщества по модулярности

Теперь рассмотрим результаты работы каждого из предложенных алгоритмов по выявлению скрытых сообществ с использованием разных параметров при настройке визуализации.

5.1. Ядра графа

Так как алгоритм поиска ядер графа имеет квадратичную сложность, в ходе тестирования было решено рассматривать небольшое количество вершин графа. Однако меньшее количество вершин не ослабевает роль поиска ядер графа. Выполняется это при условии, что степень вершины должна быть достаточно большой. Это означает, что у текущей вершины имеется большое количество связей, а значит, ее есть смысл рассматривать.

Таким образом, можно рассмотреть подмножество множества вершин U, у каждой из которых:

- более k соседей (k берется достаточно большое);
- менее k соседей (k берется достаточно малое).

Вершины множества U могут быть смежными или же не иметь связи вообще. Опишем логику визуализации:

1. Из исходного графа по «id» выбираются те вершины, что вошли в список исследуемых вершин – множество C.

2. Для каждой вершины из C определяется количество ядер, в которых она состоит – s, $0 \leq s \leq p$, 0 – не принадлежит ни одному ядру, p – принадлежит всем выявленным ядрам.

3. В зависимости от значения s задается размер для каждой вершины. Чем больше по размеру вершина, тем к большему числу ядер она относится.

В итоге мы получаем граф, на котором видно всех участников ядер графа и их популярность.

На Рис. 15 видно, что в результате задания параметра k = 332 (вариант «более k соседей») удалось определить одно ядро. В список вершин с минимальной степенью равной 332 вошло 24 узла. Однако ядро состоит из 7 вершин. Участники данного ядра обозначены зеленым цветом. Фиолетовым цветом обозначены вершины, которые не относятся к ядру.

Видно, что данные точки располагаются не близко друг к другу, однако близки относительно других вершин. Тем не менее, мы объединили их в одно ядро, что можно интерпретировать как «донесение информации из разных точек».

На Рис. 16 представлен результат для k = 39 (вариант «менее k соседей»). В список вершин с максимальной степенью равной 39 вошло 48 узлов. Ядро состоит из 42 вершин. Участники данного ядра обозначены коричневым цветом. Синим цветом – вершины, которые не относятся к ядру.

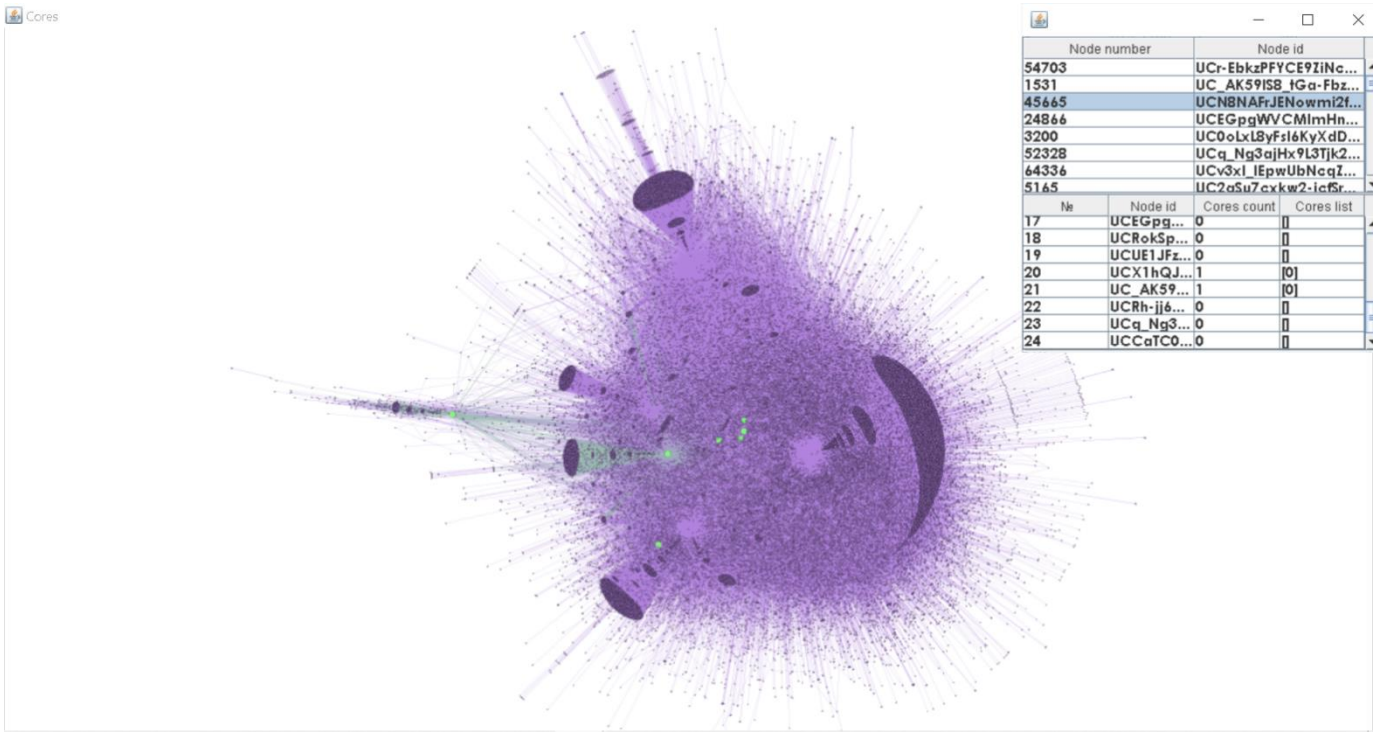


Рис. 15. Ядро графа, k = 332

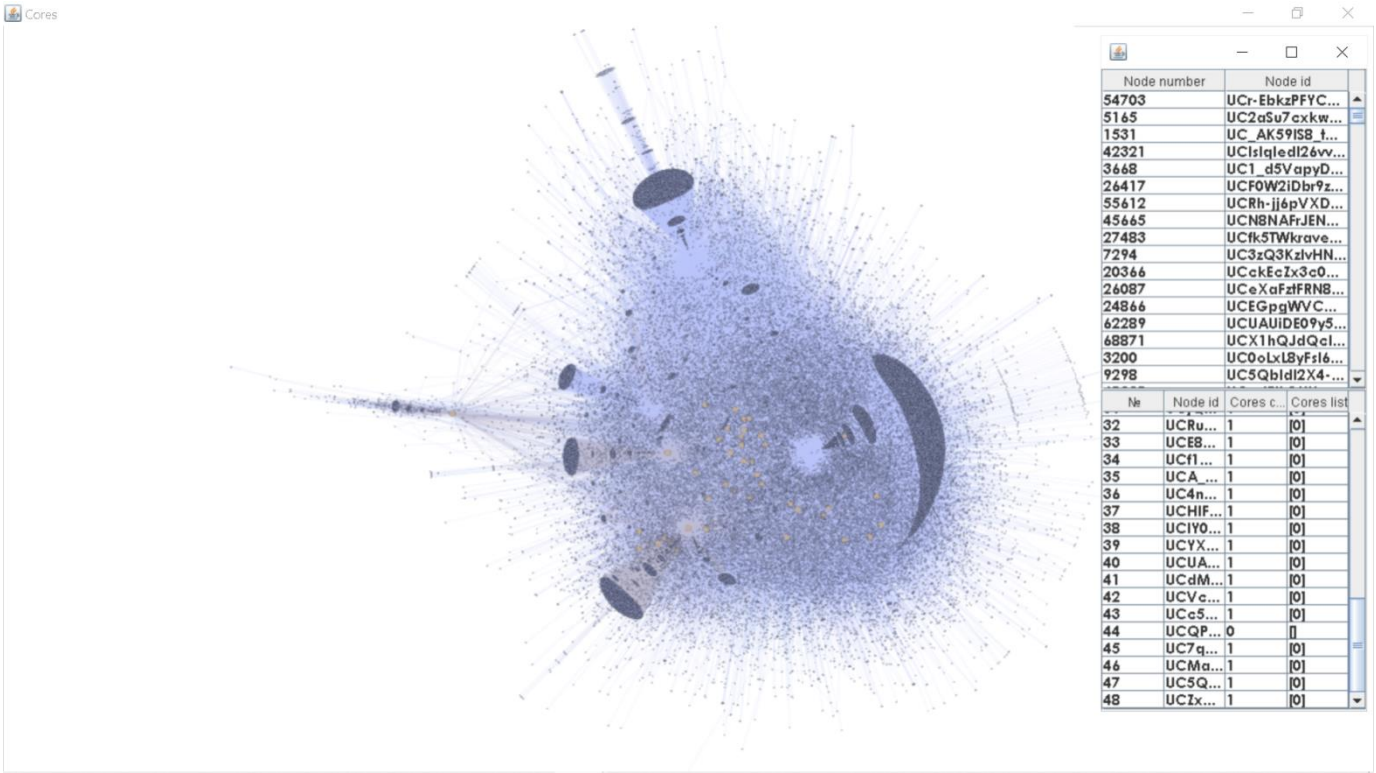


Рис. 16. Ядро графа, k = 39

При подборе разных значений параметра k было замечено, что более одного ядра не выявляется. Это обусловлено тем, что граф не является сильно связным[7].

5.2. Клики графа

Одна и та же вершина может относиться сразу к нескольким кликам. В теории графов это называется пересечением сообществ (overlapping). В результате работы алгоритма для каждой исследуемой вершины возвращается список сообществ, в котором она состоит. Здесь мы не просто можем разделить вершины на сообщества по выделению клик, но и создавать сообщества на основании другой идеи. Она заключается в том, что в одном классе (сообществе) находятся все те вершины, которые принадлежат «X» разным кликам. С помощью размера вершины показывается количество клик, в которые входит данная вершина. Чем больше по размеру вершина, тем в большее количество клик она входит. С помощью цвета идентифицируются участники таких сообществ.

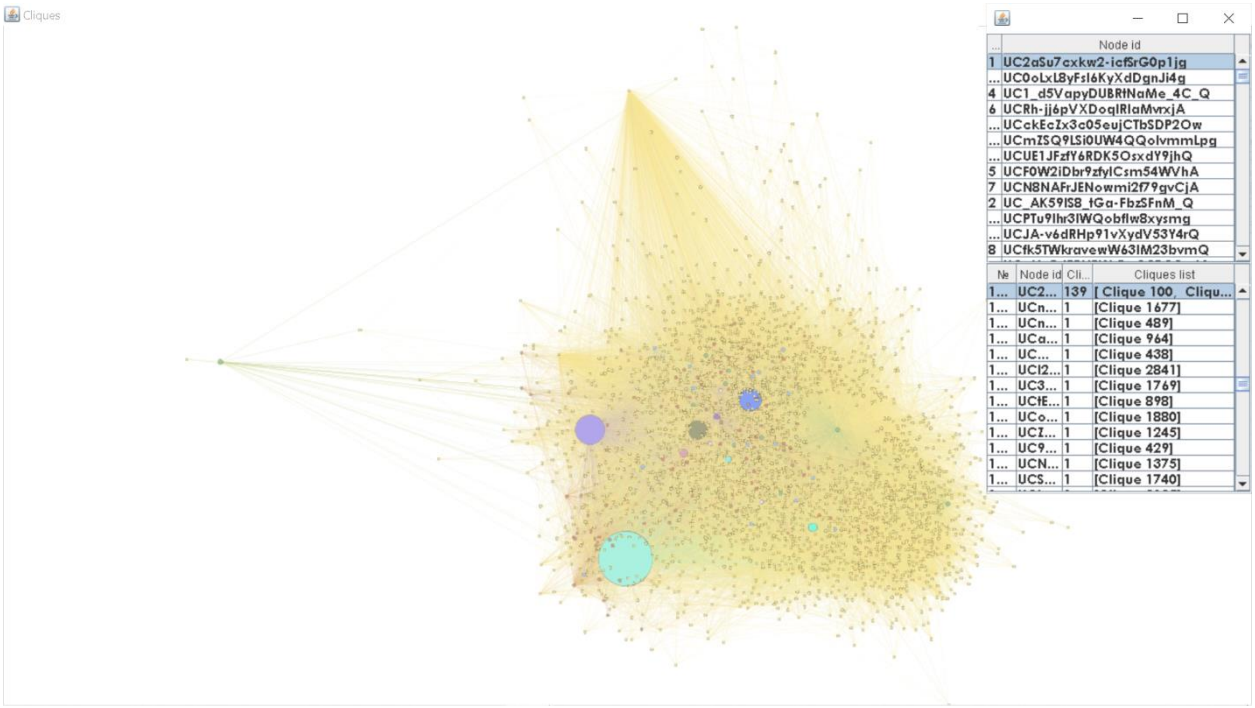


Рис. 17. Разбиение узлов на клики, k = 20

На Рис. 17 представлен результат с параметром k = 20, то есть, у каждой вершины должно быть минимум 20 смежных узлов. Так из исходного графа получилось выделить 2950 вершин и 48166 дуг. Видно, что одна вершина, обозначенная зеленым цветом, входит в 139 разных клик. Вершины, обозначенные желтым цветом, входят только в одну клику.

На Рис. 18 представлен результат с параметром k = 50. Здесь количество вершин – 859, количество дуг – 17412. Светло-фиолетовым цветом выделена вершина, что относится к 60 разным кликам.

[7] Ориентированный граф называется сильно связным, если для любых двух различных вершин существует по крайней мере один путь, соединяющий их. Однако здесь не гарантируется наличие связи между двумя любыми пользователями

Таким образом, используя алгоритм выявления клик, мы можем не только выявить сообщество, но и посмотреть, насколько каждая вершина является центральной. Центральность определяется количеством клик, в которые входит вершина.

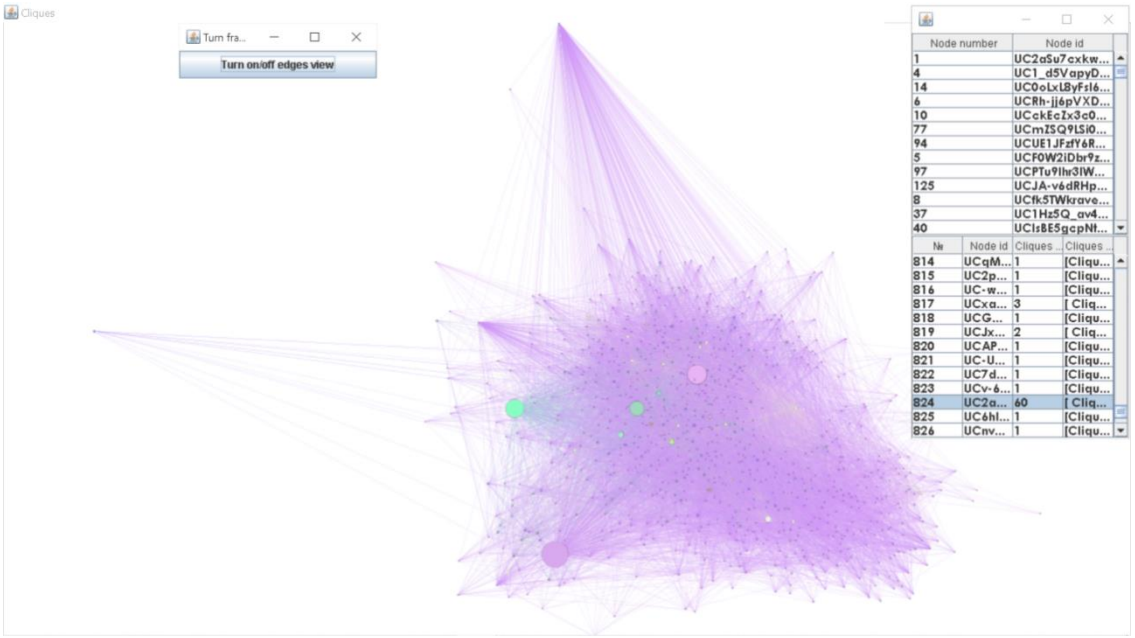


Рис. 18. Разбиение узлов на клики, k = 50

5.3. Кластеры графа (MST)

В отличие от клик, при работе с алгоритмом MST одна вершина может относиться только к одному сообществу (кластеру). В качестве параметров здесь рассматривается нижняя граница – минимальное количество узлов в одном кластере – и верхняя граница – максимальное количество узлов в одном кластере. В зависимости от значений данных параметров получается разное количество кластеров. При работе алгоритма удаляются ребра, в результате чего получается большое количество вершин, что входят в кластер размера 1. Это нам не интересно рассматривать в качестве сообщества, поэтому данные вершины можно отнести к специальной группе «-1», что означает список незначимых вершин.

На Рис. 19 представлен результат работы алгоритма с параметрами нижней границы – 2, верхней границы – 400, k = 12. Было выделено 10 кластеров. С помощью цвета идентифицируется кластер, с помощью размера – количество участников кластера. Видно, что зеленым цветом обозначены вершины самого крупного кластера (302 участника). Серым цветом и маленьким размером обозначен кластер незначимых вершин (по одному участнику в кластере).

На Рис. 20 представлен результат работы алгоритма с параметрами нижней границы – 2, верхней границы – 50, k = 48. В данном примере рассматриваются вершины, у которых достаточно большое количество связей, что интересно исследовать. В результате было выделено 4 кластера. Синим цветом обозначены вершины самого крупного кластера (29 участников). В других кластерах 2, 3 и 5 участников.

Таким образом, используя простой алгоритм MST и настраивая несколько параметров, у нас получается выделять скрытые сообщества и рассматривать расположение их участников относительно друг друга.

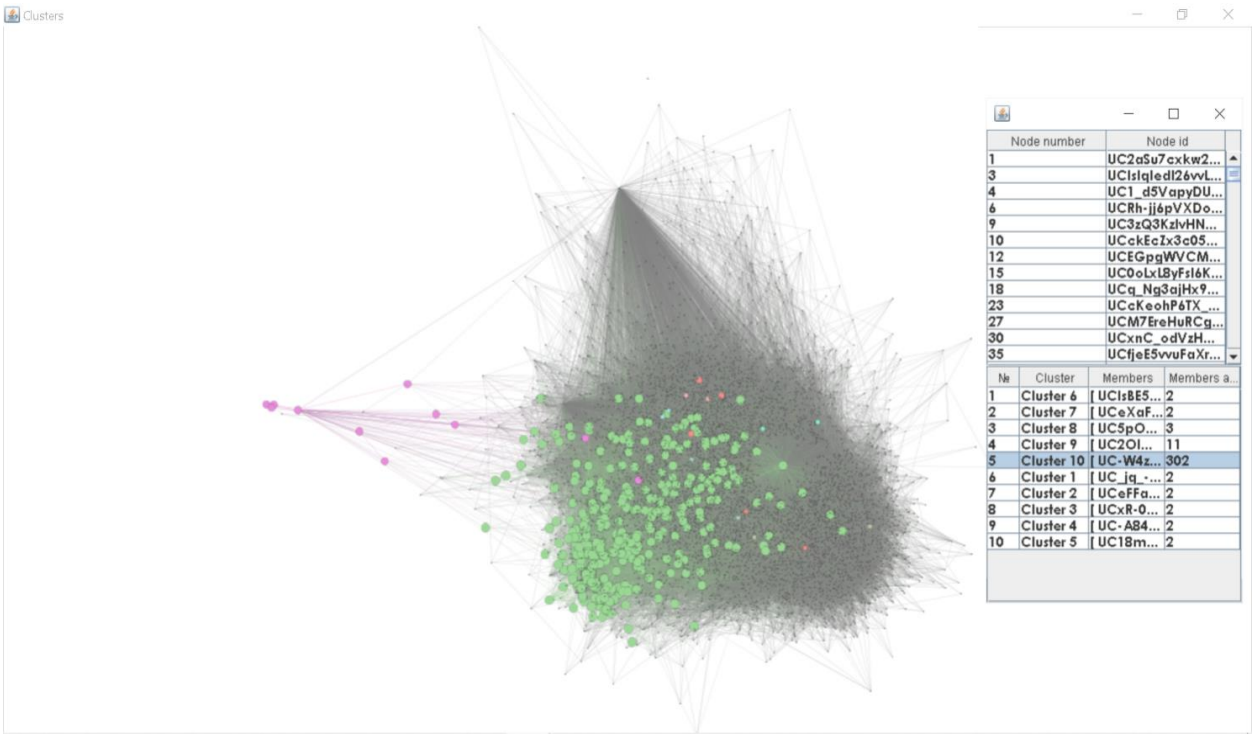


Рис. 19. Укладка Yifan Hu, k = 12, |V| = 5528, |E| = 73755

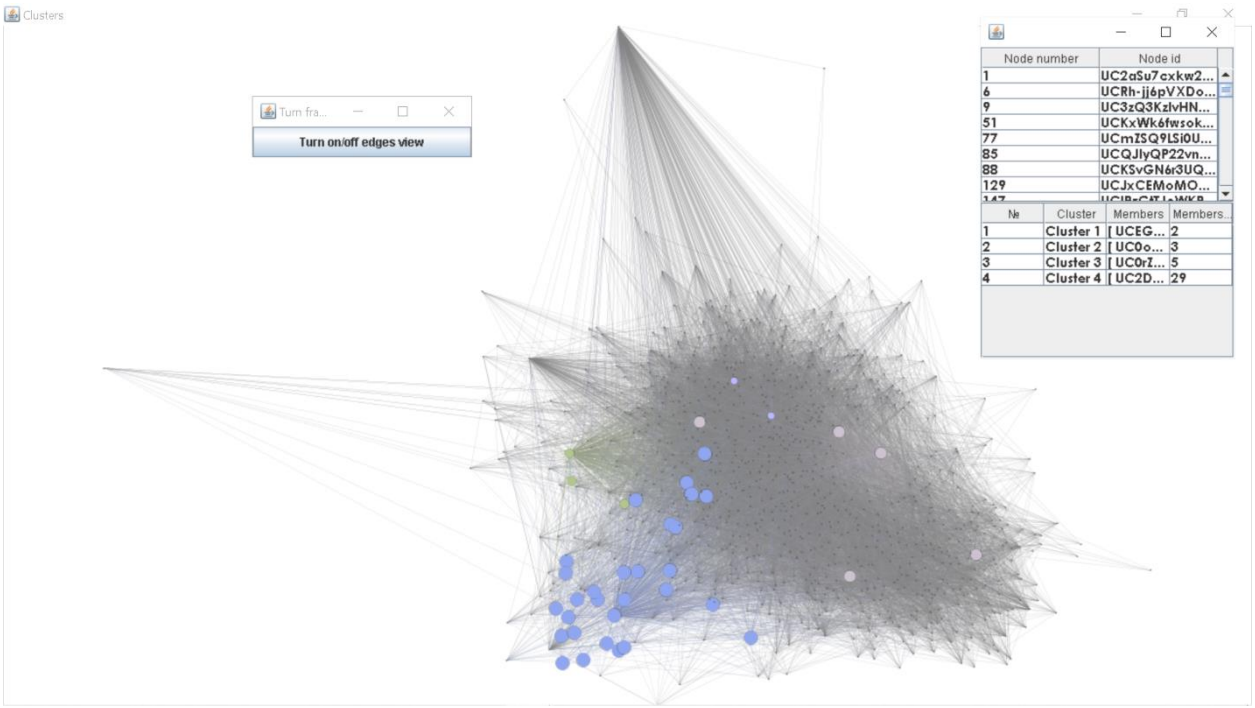


Рис. 20. Укладка Yifan Hu, k = 48, |V| = 911, |E| = 18363

5.4. Кластеры графа (NAG)

Здесь рассматриваются ребра с достаточно большим весом (см. Замечание 4). В связи с этим было решено заранее фильтровать список ребер и оставлять только те, которые имеют вес больше некоторого параметра. Путем эксперимента был подобран данный параметр, а также параметр количества удаляемых ребер. По аналогии с предыдущим алгоритмом, было замечено, что выделяется большое количество кластеров с одним участником, что не очень интересно. В связи с этим есть смысл визуализировать только те сообщества (кластеры), в которых количество участников больше единицы.

Аналогично предыдущим алгоритмам, с помощью цвета визуализируется кластер, с помощью размера – мощность кластера. На Рис. 21 представлен пример с параметрами k = 10, минимальный вес ребер равен 9, количество удаляемых ребер равно 30. Количество рассматриваемых узлов равно 6777, ребер – 83584. Видно, что большое количество сообществ состоит из двух, трех участников и самая крупная из 3251 узла.

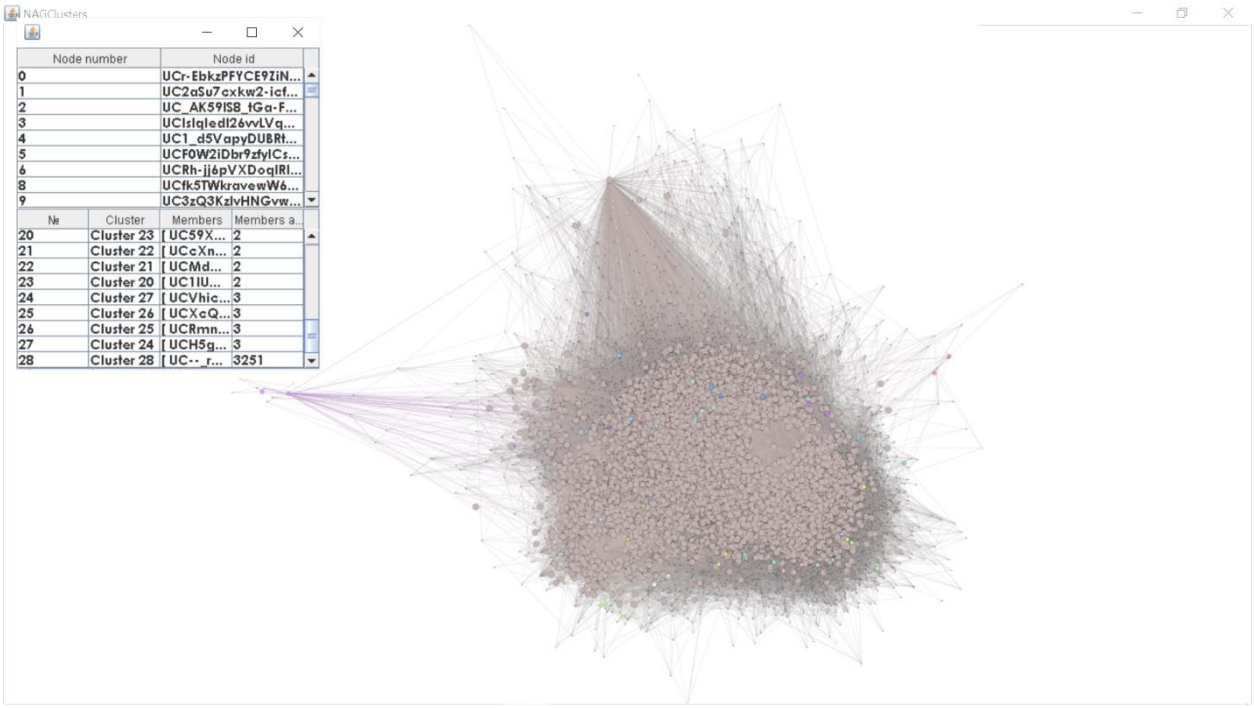


Рис. 21. Разбиение узлов на сообщества, 28 сообществ, k = 10

На Рис. 22 представлен пример с параметрами k = 48, минимальный вес ребер равен 9, количество удаляемых ребер равно 30. Количество рассматриваемых узлов равно 911, ребер – 18363. По аналогии с предыдущим примером, сообщества в основном состоят из двух и трех участников. Самое крупное сообщество – из 756. Интересные результаты можно получать с помощью подбора указанных параметров.

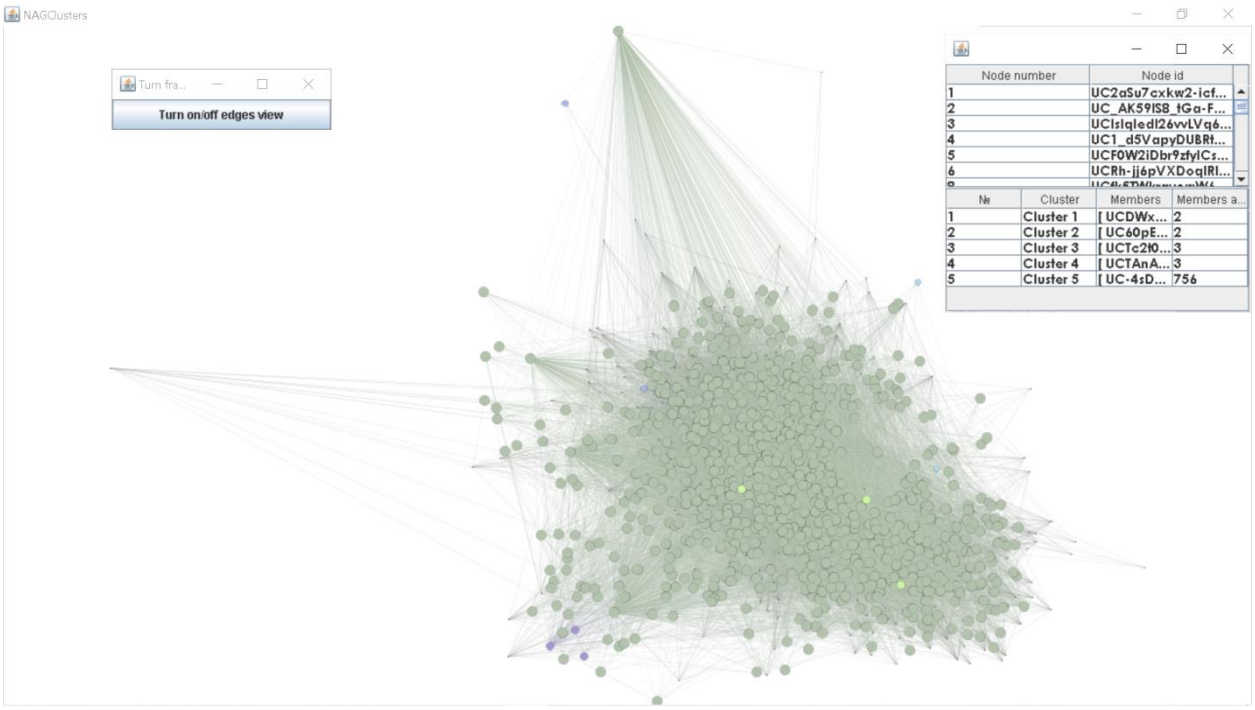


Рис. 22. Разбиение узлов на сообщества, 5 сообществ, k = 48

5.5. Компромиссное решение

Пусть зарегистрировалось пять агентов (Рис. 23), каждый из которых дал оценку результатам работы каждого из алгоритмов (Рис. 24).

```
select id, firstname, secondname from rivalry.agent
```

postgres@localhost [2]_8 - postgres@localhost [2]_8

postgres.rivalry.agent

5 rows | Tab-se...d (TSV) | DDL | View Query

	id	firstname	secondname
1	-738325625	admin	my
2	1630763555	Pedro	Faither
3	311785540	Alex	Raoe
4	-77563336	Michael	Rich
5	1129669250	Evgeniya	Jupiter

Рис. 23. Список агентов

```
select algorithm_id, firstname, secondname, rating, name as algorithm_name from rivalry.rating
 JOIN rivalry.agent on rating.agent_id = agent.id
 JOIN rivalry.algorithm on rating.algorithm_id = algorithm.id
```

postgres@localhost [2]_8 - postgres@localhost [2]_8

Result 14

20 rows

	algorithm_id	firstname	secondname	rating	algorithm_name
1	1	admin	my	5	Ядро графа
2	2	admin	my	1	Клики графа
3	3	admin	my	1	Кластера графа (MST)
4	4	admin	my	4	Кластера графа (NAG)
5	1	Pedro	Faither	5	Ядро графа
6	2	Pedro	Faither	2	Клики графа
7	3	Pedro	Faither	3	Кластера графа (MST)
8	4	Pedro	Faither	4	Кластера графа (NAG)
9	1	Alex	Raoe	2	Ядро графа
10	2	Alex	Raoe	4	Клики графа
11	3	Alex	Raoe	5	Кластера графа (MST)
12	4	Alex	Raoe	5	Кластера графа (NAG)
13	1	Michael	Rich	5	Ядро графа
14	2	Michael	Rich	1	Клики графа
15	3	Michael	Rich	3	Кластера графа (MST)
16	4	Michael	Rich	5	Кластера графа (NAG)
17	1	Evgeniya	Jupiter	5	Ядро графа
18	2	Evgeniya	Jupiter	5	Клики графа
19	3	Evgeniya	Jupiter	5	Кластера графа (MST)
20	4	Evgeniya	Jupiter	1	Кластера графа (NAG)

Рис. 24. Оценка для каждого алгоритма от агентов

В ходе работы алгоритма поиска компромиссного решения в задаче конкуренции получился следующий выбор оптимального алгоритма (Рис. 25).

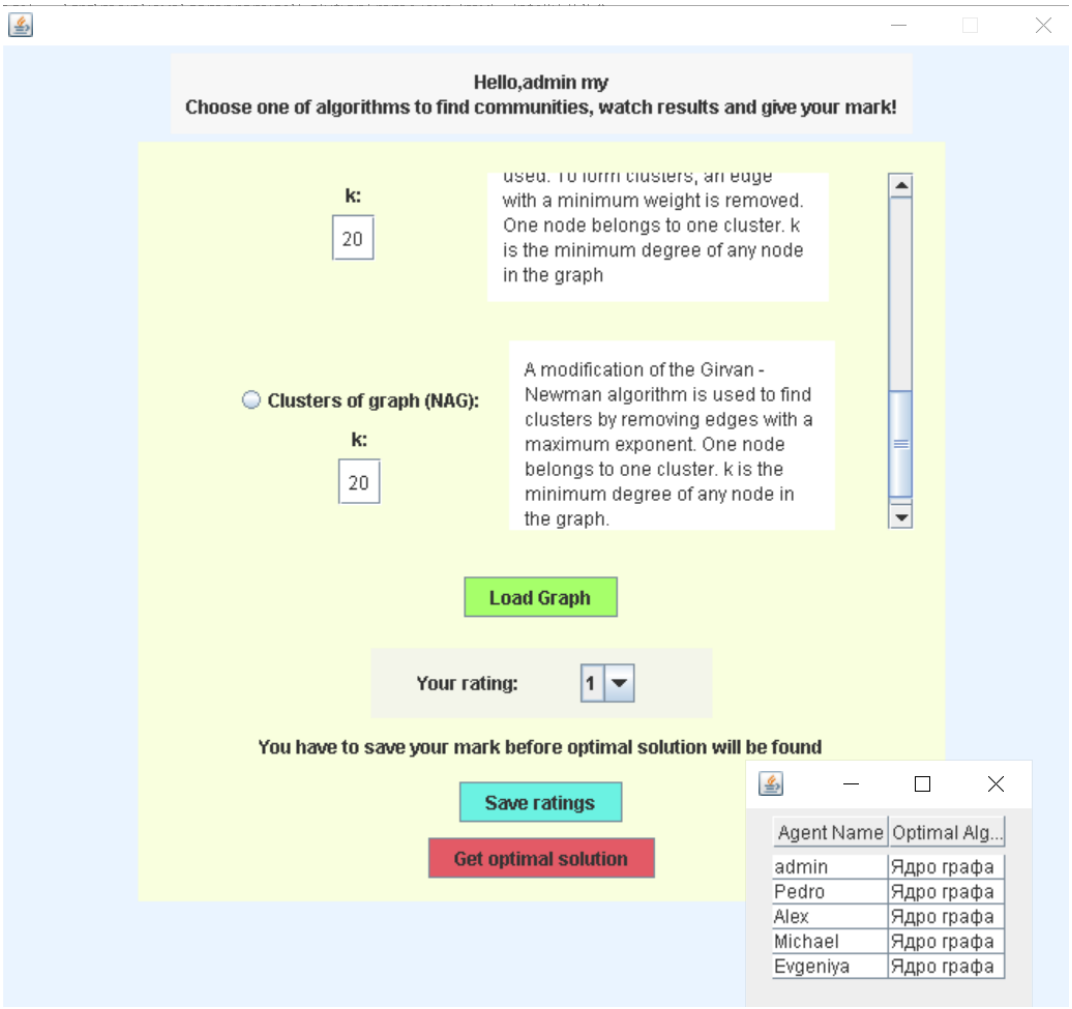


Рис. 25. Интерфейс с компромиссным решением

Когда каждый агент ставит или обновляет оценку одного из алгоритмов, обновляется таблица оценок в БД, и очищается таблица оптимального решения. Затем происходит перерасчет компромиссного решения и возвращается новый результат.

5.6. Тестирование

После реализации алгоритмов было проведено нагрузочное тестирование. Результаты для алгоритмов поиска клик графа, MST и NAG представлены в Таблице 7. Для алгоритма поиска ядер графа была создана отдельная Таблица 8. В первой колонке указано название алгоритма, значения параметров и количество сообществ, которые были обнаружены. Во второй колонке задано значение параметра минимальной (максимальной) степени вершин. В третьей колонке – количество вершин, в четвертой – количество дуг.

Алгоритм	k	|V|	|E|	Время
Поиск клик графа - cliques count = 294 - important cliques: 11	100	275	5446	7.12 с.
- cliques count = 340 - important cliques: 12	90	319	6474	3.59 с.
- cliques count = 494 - important cliques: 13	70	474	9725	4.02с.
- cliques count = 880 - important cliques: 14	50	859	17412	4.54с.
- cliques count = 1761 - important cliques: 14	30	1742	32113	9.153с.
- cliques count = 2229 - important cliques: 14	25	2215	38747	9.405с.
- cliques count = 2963 - important cliques: 15	20	2950	48166	15.59с.
- cliques count = 6782 - important cliques: 16	10	6777	83583	74.325с.
NAG - edges cut = 265 - important clusters: 5	100	275	451	2.54 с.
- edges cut = 441 - important clusters: 6	90	319	514	4.708 с.
- edges cut = 693 - important clusters: 4	70	474	703	16.255 с.
- edges cut = 1148 - important clusters: 2	50	859	1158	106.527 с.
- edges cut = 50 - important clusters: 4	30	1742	1913	11.84 с.
- edges cut = 50 - important clusters: 8	25	2215	2270	20.398 с.
- edges cut = 50 - important clusters: 10	20	2950	2709	32.795 с.
- edges cut = 50 - important clusters: 28	10	6777	4103	371.365 с.
MST - important clusters: 1	100	275	197	0.105 с.
- important clusters: 1	90	319	221	0.043 с.
- important clusters: 1	70	474	278	0.05 с.
- important clusters: 3	50	859	377	0.238 с.
- important clusters: 2	30	1742	479	0.215 с.
- important clusters: 2	25	2215	525	0.133 с.
- important clusters: 2	20	2950	584	0.381 с.
- important clusters: 2	10	6777	782	0.632 с.

Таблица 7. Производительность алгоритмов кластеризации

Алгоритм	k	|V|	|E|	Время
Поиск ядер графа - ядер нет	325	25	166	8.06с
- 1 ядро	331	24	150	11.58с
- ядер нет	310	28	207	4.55с
- 1 ядро	280	35	223	472.36с.
- 1 ядро	260	41	435	828.3с

Таблица 8. Производительность алгоритма поиска ядер графа

На Рис. 26 представлен график, что составлен по текущей таблице. По оси x – минимальная степень вершины, по оси y – затраченное время (в секундах). Голубым цветом обозначены зависимость времени работы алгоритма от параметра k для NAG, оранжевым – для алгоритма поиска клик графа, красным – для MST. Анализируя этот график, можно сделать вывод, что MST выдает результат быстрее всех, NAG тратит больше времени на вычисления.

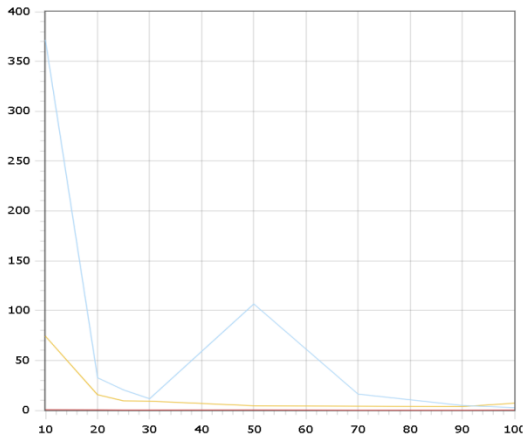


Рис. 26. График производительности алгоритмов MST, NAG и поиска кластеров графа

На Рис. 27 показывается аналогичная зависимость скорости от времени для алгоритма поиска ядер графа. По данному графику видно, что алгоритм является достаточно чувствительным к значению параметра k.

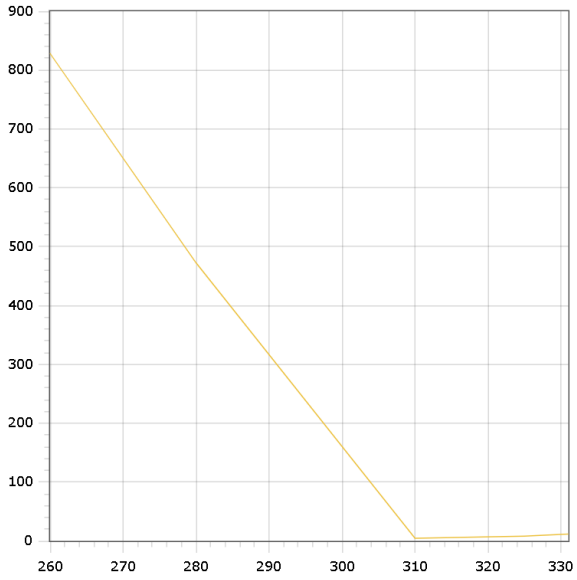


Рис. 27. График производительности алгоритма поиска ядер графа

Помимо нагрузочного тестирования необходимо оценить качество работы алгоритмов.

Оценка качества кластеризации базируется на следующих свойствах:

1. Компактность

Элементы одного кластера должны быть близки друг к другу. Это свойство можно определить как расстояние между элементами или плотность внутри кластера

2. Отделимость

Между разными кластерами расстояние должно быть велико.

3. Концентрация

Есть некоторый центр кластера, вокруг которого сконцентрированы элементы сообщества.

Оценка проводилась на основании первых двух свойств, так как в алгоритмах не использовалось понятия центра кластера.

Качественный анализ результата проводился по двум группам мер:

1.Внешние меры

Идея заключается в том, что сравниваются результаты двух алгоритмов и оценивается схожесть полученных кластеров.

2.Внутренние меры

Результат оценивается по анализу каждого алгоритма по отдельности на основании качества структуры кластеров.

Для оценки алгоритмов MST и NAG было проведено сравнение с использованием следующих внешних мер:

1.NMI
Мера сходства двух разбиений, что основана на теории информации.

Пусть результат работы алгоритма MST определяется как множество меток $\{x_i\}$, алгоритма NAG $-$ $\{y_i\}, i = \overline{1,n}$, где n $-$ количество рассматриваемых вершин. Предполагается, что метки x и y – значения случайных величин X и Y, для которых задано совместное распределение:

$$P(X = x, Y = y) = \frac{n_{xy}}{n},$$

где n_{xy} – количество вершин таких, что $x = x_i, y = y_i$. В качестве меры похожести используется нормированная величина $1 - NMI$:

$$1 - NMI = 1 - I_{norm} = \frac{2I(X,Y)}{H(X) + H(Y)},$$

где $I(X,Y) = H(X) - H(X|Y), H(X) = -\sum_x P(x) log P(x)$ – энтропия,

$H(X|Y) = -\sum_{x,y} P(x,y) log P(x|y)$ – условная энтропия.

Результат представлен на Рис. 28. По оси x расположена минимальная степень вершины графа, по оси y – мера схожести. Видно, что данные алгоритмы не коррелируют (значения близки к 0).

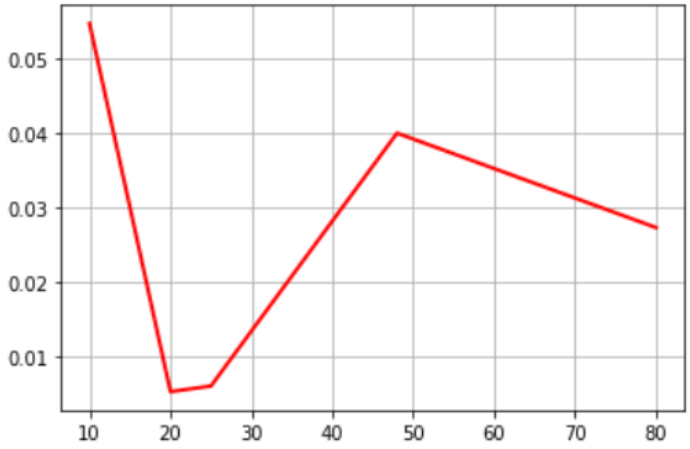


Рис. 28. График зависимости меры схожести NMI от минимальной степени вершин графа

2.Индекс Rand
Мера сходства между двумя кластерам. Рассматриваются все пары выборок и подсчитываются пары, которые назначены в одном и том же или разных кластерах.

$$RI(U,V) = \frac{\text{Количество согласующихся пар}}{\text{Общее количество пар}},$$

где $U, V -$ множества сообществ сравниваемых алгоритмов.

Величина принимает значения от 0 до 1. Чем ближе значение к 1, тем сильнее коррелируют множества U и V.

На Рис. 29 представлен результат, по которому видно, что корреляции практически нет.

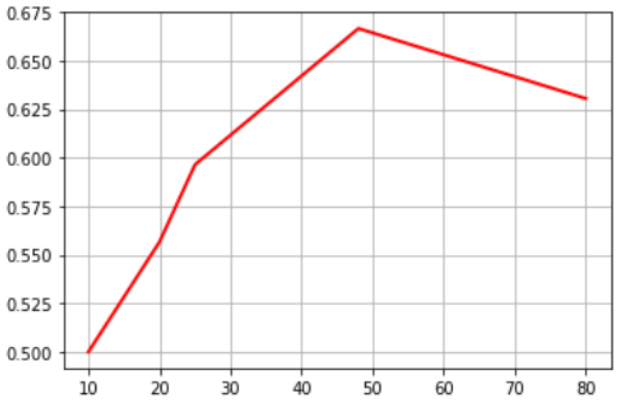


Рис. 29. График зависимости меры схожести Rand от минимальной степени вершин графа

3.Индекс Жаккара

Определятся как количество пересечений результатов двух алгоритмов A и B, деленное на количество объединений результатов:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Чем выше значение индекса, тем больше сходство между кластерами.

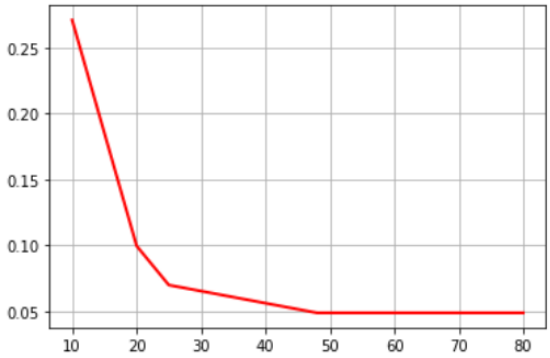


Рис. 30. График зависимости индекса Жаккара от минимальной степени вершин графа

По Рис. 30 видно, что значения индекса для разных по количеству наборов вершин достаточно низкие.

Таким образом, можно отметить, что данные алгоритмы не выдают коррелирующие результаты.

Проведем анализ алгоритмов MST и NAG отдельно по такому показателю как Flake-ODF [40]:

$$F - ODF(C) = \sum_{c \in C} \frac{|\{x : x \in c, |\{(x,y)\} : y \notin c| < D_{out}(x)/2|}{N_c},$$

где $C -$ множество сообществ, $c -$ конкретное сообщество, $N_c -$ количество вершин в данном сообществе, $D_{out}(x)$ – количество исходящих из вершины x дуг.

Идея заключается в том, что связь между вершинами внутри сообщества крепче, чем внешняя связь. Если F-ODF(C) = 1, связь в каждом кластере является достаточно сильной, а значит, разбиение графа на сообщества является оптимальным. Анализ проводился по нескольким графам (с разным значением параметра k). На Рис. 31 представлен результат, где красным цветом обозначены значения показателя для MST, зеленым – для NAG. По оси x располагаются значения минимальной степени каждой вершины графа, по оси y – значения $Flake - ODF(C)$. Видно, что при большом количестве исследуемых вершин лучше использовать MST, так как показатель ближе к 1.

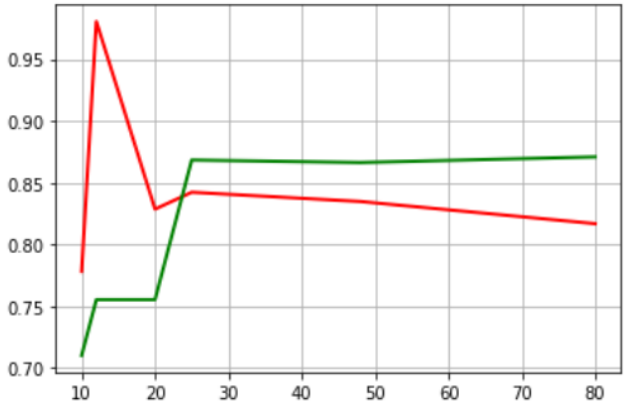


Рис. 31. График оценки алгоритмов MST и NAW по Flake-ODF(C)

Алгоритм поиска пересекающихся сообществ Брона-Кербоша был оценен по такому показателю как *overlap ratio*. Он определяется как среднее число сообществ, к которым принадлежит узел в графе:

$$o(C) = \sum_{v \in V} \frac{|c \in C : v \epsilon c|}{|V|},$$

где $|V| -$ общее количество узлов, $C -$ множество сообществ, $c -$ конкретное сообщество.

Результат представлен на Рис. 32. По оси x располагаются значения минимальной степени каждой вершины графа, по оси y – среднее количество сообществ, в которое входит вершина. Видно, что с увеличением количества связей, пересечение кластеров увеличивается.

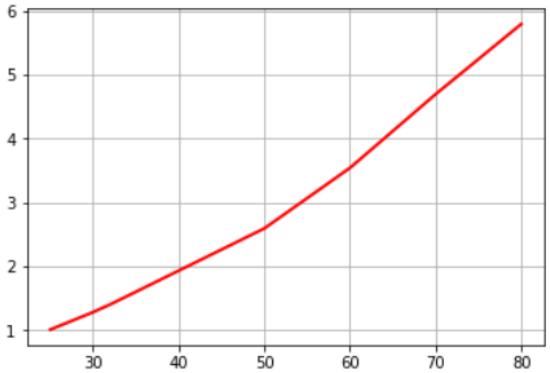


Рис. 32. График оценки алгоритма поиска клик графа по *o-ratio*

Вывод

Таким образом, апробация показывает, что с использованием теории графов непосредственно для выявления скрытых сообществ получаются интересные для дальнейшего аналитического исследования результаты. Используя данный продукт, мы не только определяем скрытые сообщества, но и обращаем внимание на тех пользователей, которых можно отнести сразу к нескольким скрытым сообществам. Другими словами, можно в дальнейшем исследовать не только сообщества, но и смотреть на отдельных пользователей. Самым быстрым и эффективным алгоритмом оказался алгоритм MST, самым сложным – алгоритм поиска ядер графа. Алгоритмы между собой не коррелируют.

Применение теории игр, а именно, поиск компромиссного решения, в свою очередь, позволяет разрешить противоречие между агентами, которые дают разные оценки каждому из алгоритмов при ознакомлении с результатами.

Заключение

В ходе научно-исследовательской работы были выполнены все поставленные задачи, а именно

- реализованы алгоритмы по выявлению скрытых сообществ;
- реализована возможность выбора параметров для конкретного алгоритма;
- модифицирован алгоритм сбора данных;
- реализована визуализация выявленных групп;
- реализован User Interface для знакомства и оценки результатов алгоритмов агентами;
- реализован алгоритм поиска компромиссного решения;
- проведена апробация и тестирование на актуальных данных за 2020 год социальной сети YouTube.

Отметим, что в качестве скрытых сообществ можно считать отдельные ядра, клики, кластеры или же разделять на некоторые классы тех пользователей, которые относятся сразу к нескольким ядрам или кликам. Используя рассмотренные алгоритмы, мы можем разбивать множество вершин на группы, а затем находить среди них узлы, которые входят сразу в несколько сообществ или же относятся к достаточно крупным кластерам.

В перспективе данную работу можно использовать как в поиске террористических группировок в социальной сети YouTube, так и в направлении каких-либо других социальных исследований. Для увеличения спроса на рынке в будущем программный продукт необходимо модифицировать по следующим пунктам:

- производительность;
- параметры алгоритмов выявления скрытых сообществ.

Результаты исследовательской работы частично опубликованы в статье [2].

Список литературы

[1]. Щеникова С.А. Методы анализа AD HOC дискуссий в социальной сети YouTube // Control Processes and Stability (CPS'19). Том 6, раздел 1 (2019). С. 378-382.

[2]. Малафеев О.А., Щеникова С.А., Скворцова О.И. Математическое моделирование задач экономической конкуренции по выявлению скрытых сообществ в социальной сети // Информационные технологии в образовании (2021). С. 167-172.

[3]. Santo F. Community detection in graphs // Physics Reports 486, 75-174 (2010). P.14 – 80.

[4]. Cosciaa M., Giannotti F., Pedreschia D. A Classification for Community Discovery Methods in Complex Networks // Statistical Analysis and Data Mining journal, Special Issue: Networks. Volume 4, Issue 5, pages 512-546 (2011). P.5 – 25.

[5]. Schaeffer S.E. Graph clustering // Computer Science Review 1(1), 27-64 (2007). P.9 – 29.

[6]. Amelio A., Pizzuti C. Overlapping Community Discovery Methods: A Survey. // Social Networks: Analysis and Case Studies. Lecture Notes in Social Networks (2014). P. 3 – 16.

[7]. Nerurkar P., Chandane M., Bhirud S. A Comparative Analysis of Community Detection Algorithms on Social Networks: ICCI-2017. 10.1007/978-981-13-1132-1_23 (2019). P. 4 – 8.

[8]. Reichardt J., Bornholdt S. Statistical mechanics of community detection // Phys. Rev. (2006). P. 113–126.

[9]. Nerurkar, Pranav & Chandane, Madhav & Bhirud, Sunil. Community Detection Using Node Attributes: A Non-negative Matrix Factorization Approach: ICCI-2017. 10.1007/978-981-13-1132-1_22 (2019). P. 5 – 8.

[10]. Fiscarelli A.M., Brust M.R., Danoy G., Bouvry P. A Memory-Based Label Propagation Algorithm for Community Detection // COMPLEX NETWORKS 2018. Studies in Computational Intelligence, vol 812. Springer, (2019).

[11]. Fiscarelli, Antonio, Brust, Matthias, Danoy, Grégoire, Bouvry, Pascal. A vertex-similarity clustering algorithm for community detection // Journal of Information and Telecommunication. 4. 1-15. 10.1080/24751839.2019.1686683 (2019). P. 4 – 6.

[12]. Ning-Ning W, Zhen J., Xiao-Long P. Community Detection with Self-Adapting Switching Based on Affinity // Complexity, vol. 2019, Article ID 6946189 (2019). URL:https://doi.org/10.1155/2019/6946189

[13]. Jaiswal R., Ramanna S. Detecting Overlapping Communities Using Distributed Neighbourhood Threshold in Social Networks // Bello R., Miao D., Falcon R., Nakata M., Rosete A., Ciucci D. (eds) Rough Sets. IJCRS 2020. Lecture Notes in Computer Science, vol 12179. Springer. URL: https://doi.org/10.1007/978-3-030-52705-1_32

[14]. Ge J., Sun H., Xue C., et al. LPX: Overlapping community detection based on X-means and label propagation algorithm in attributed networks // Computational Intelligence; 37: 484–510 (2021). URL: https://doi.org/10.1111/coin.12420.

[15]. Nyunt Nyunt S. Overlapping Community Detection Using Centrality Measure and Local Seed Information //Journal of Computer Applications and Research, Volume 1, No 1 (2020).

[16]. Pelleg D., Moore A. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. Machine Learning // ICML (2000).

[17]. Novak J., Tomkins A., Tomlin J. PageRank computation and the structure of the web: experiments and algorithms (2002)
URL:https://www.researchgate.net/publication/2524595_PageRank_Computation_and_the_Structure_of_the_Web_Experiments_and_Algorithms

[18]. Рубчинский А. Дискретные математические модели. Начальные понятия и стандартные задачи // Директ-Медиа - 267, [1] (2013). С. 75-82.

[19]. Гармашов К.С. Программная реализация алгоритмов вычисления хроматического числа и их сравнение (2018). С. 5, 18 – 20.

[20]. Drineas P., Kannan R., Frieze A., Vempala S., Vinay V. Clustering in large graphs and matrices //ACM-SIAM Symposium on Discrete Algorithms (SODA) (1999). P 10-22.

[21]. Lancichinetti, A, Fortunato S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities // Physical Review E. V. 80, № 1 (2009). P. 22-30.

[22]. Evans T.S. Clique graphs and overlapping communities //Journal of Statistical Mechanics: Theory and Experiment (12) (2010). P.20-22.

[23]. Blonde V. D., Guillaume J., Lambiotte R., Lefebvr R. Fast unfolding of communities in large networks // J. Stat. Mech (2008). P. 12-15.

[24]. Radicchi F, Castellano C., Cecconi F., Loreto V., Parisi D.. Defining and identifying communities in networks // Proceedings of the National Academy of Sciences 101(9):2658-63 (2004). P. 6-12.

[25]. Bron C., Kerbosh J. Algorithm 457 — Finding all cliques of an undirected graph //Comm. of ACM, 16, p. 575—577. (1973). P. 2-10.

[26]. Gower, J. C., Ross, G. J. Minimum spanning trees and single linkage cluster analysis. Applied statistics // Journal of The Royal Statistical Society Series C-applied Statistics (1969). P.54-64.

[27]. Girvan, M. Community structure in social and biological networks // Proceedings of the National Academy of Sciences (2001). P. 5-12.

[28]. Martin S., Brown W.M., Klavans R., Boyack K.W. OpenOrd: an opensource toolbox for large graph layout // SPIE Electronic Imaging. International Society for Optics and Photonics. (2011). P. 786–806.

[29]. AL-Qurishi M., Alrakhami M., Alamri A., Sybil M.A. Defense techniques in online social networks: A Survey // IEEE Access PP. No 99 (2017). P.12–24.

[30]. Freeman L.C, White D. R., Romney A.K. Research methods in social network analysis VA George Mason University // (1989). P.10-15.

[31]. Jebabli M., Cherifi H., Cherifi C., Hamouda A. User and group networks on YouTube: A comparative analysis //ACS 12th International Conference of Computer Systems and Applications (AICCSA) (2015). P.13-19.

[32]. Satu E. S .Graph clustering //Computer Science Review 1(1):27-64 pp. 48-51 (2007). P.9-16.

[33]. Grigorieva X., Malafeev O. A competitive many-period postman problem with varying parameters //Applied Mathematical Sciences 8 (145-148) pp 7249-7258 (2014). P.3-10.

[34]. Malafeev O. On the existence of nash equilibria in a noncoope rative n-person game with measures as coefficients //Communications in Applied Mathematics and Computational Science, no 5(4), pp. 689-701 (1995). P.8-12.

[35]. Pichugin Y., Malafeyev O. Statistical estimation of corruption indicators in the firm //Applied Mathematical Sciences 10 (41-44) pp 2065-2073 (2016). P.5-7.

[36]. Malafeyev O., Saifullina D., Ivaniukovich G., Marakhov V., Zaytseva I. The model of multiagent interaction in a transportation problem with a corruption component //AIP Conference Proceedings 1863 art. No. 170015 (2017). P.7-10.

[37]. Pichugin Y., Malafeyev O., Rylow D., Zaitseva I. A statistical method for corrupt agents detection //AIP Conference Proceedings. No. 100014 (1978). P.6-10.

[38]. Zaitseva I., Malafeyev O., Strekopytov S., Ermakova A., Shlaev D. Game-theoretical model of labour force training // Journal of Theoretical and Applied Information Technology 96 (4) p.978-983 (2018). P.2-7.

[39]. Malafeyev O., Redinskikh N., Nemnyugin S., Kolesin I., Zaitseva I. The optimization problem of preventive equipment repair planning //AIP Conference Proceedings No. 100013 (2018). P. 2-6.

[40]. Leskovec J., Lang K., Mahoney M. Empirical comparison of algorithms for network community detection // Proceedings of the 19th international conference on World wide web, WWW '10, (New York, NY, USA) (2010). P. 631–640, ACM.

[41]. Muhammad A.J., Muhammad S.Y., Siddique L., Junaid Q., Adeel B. Community detection in networks: A multidisciplinary review // J. of Network and Computer Applications, vol. 108 (2018). P. 87-111.

[42]. Azaouzi M., Rhouma D., Ben Romdhane L. Community detection in large-scale social networks: state-of-the-art and future directions //Soc. Netw. Anal. Min., no. 9 (2019).

[43]. Romano S., Bailey J., Nguyen V., Verspoor K. Standardized mutual information for clustering comparisons: one step further in adjustment for chance // Proc. 31st Intern. Conf. on Machine Learning. Beijing, China : PMLR, Vol. 32, № 2. (2014). P. 1143-1151.

[44]. Blondel V., Guil-laume J., Lambiotte R., Lefebvre E. Fast Unfolding of Communities in Large Networks // Journal of Statistical Mechanics: Theory and Experiment № 10 (2008). P. 10008–10020.

[45]. Yang J., Leskovec. J. Overlapping community detection in networks: the state of the art and comparative study // WSDM (2013).

[46]. Kourtellis N., Alahakoon T., Simha R., Iamnitchi A., Tripathi R. Identifying high betweenness centrality nodes in large social networks // Soc. Netw. Anal. Min. 3 (2013). P.899–914.

[47]. Raghavan U., Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks // Physical Review E, vol. 76 (2007).

[48]. Shen H., Cheng X., Cai K., Hu M. Detect overlapping and hierarchical community structure in networks // PHYSICA A, vol. 388 (2009). P. 1706.

[49]. Гусарова Н. Ф. Анализ социальных сетей. Основные понятия и метрики (2016). С. 40-46.

[50]. Зайцева И.В., Казначеева О.Х., Долгополова А.Ф., Резеньков Д.Н. Исследование математической модели компромиссного распределения трудовых ресурсов. // Фундаментальные исследования. № 11 (часть 2) – С. 227-231 (2018). С.1-5.