Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
УРАЛЬСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
имени первого Президента России Б. Н. Ельцина

Институт математики и компьютерных наук
Кафедра алгебры и дискретной математики

Визуализация и анализ связей пользователей социальных сетей

Допущен к защите

_____

«__»_____ 2017 г.

Квалификационная работа
на степень магистра наук
по направлению «Математика»
студента группы МЕНМ-250107
Галямовой Оксаны Наильевны

Научный руководитель
Авербух Владимир Лазаревич,
кандидат технических наук,
заведующий сектором
ИММ УрО РАН,
доцент кафедры вычислительной математики и компьютерных наук

Екатеринбург
2017

Содержание

Реферат

Галямова О. Н. ВИЗУАЛИЗАЦИЯ И АНАЛИЗ СВЯЗЕЙ ПОЛЬЗОВАТЕЛЕЙ СОЦИАЛЬНЫХ СЕТЕЙ, выпускная квалификационная работа на степень магистра: стр. 78, рис. 41, табл. 3.

Ключевые слова: АЛГОРИТМЫ ВИЗУАЛИЗАЦИИ, FORCE ATLAS 2, MODULARITY, BETWEENNESS CENTRALITY, ВКОНТАКТЕ.API

Объект исследования — социальные графы

Цель работы — анализ и исследование графов связей пользователей социальной сети Вконтакте на основе различных метрик

В процессе работы исследуется структура социальных графов, применяются алгоритмы для разбиения пользователей на сообщества, расчитываются различные показатели для пользователей. На основе получившихся данных делаются выводы применимости подобных приложений в образовательной сфере.

Abstract

Galyamova O. N. VISUALIZATION AND ANALYSIS OF CONNECTIONS OF USERS OF SOCIAL NETWORKS, final qualifying work for a master's degree: p. 78, fig. 41, tab. 3.

Keywords: VISUALIZATION ALGORITHMS, FORCE ATLAS 2, MODULARITY BETWEENNESS CENTRALITY, VKONTAKTA.API

Object of research — social graphs.

Purpose of work — analysis and research of the grafs of users of social network Vkontakte based on various metrics.

The paper investigates the structure of social graphs, algorithms are applied to divided into the user community, calculated various metrics for users. Based on the resulting data conclusions the applicability of similar applications in education.

1. Введение

Графы – это простая, мощная абстракция, применяемая во многих областях науки и техники. Графы позволяют моделировать произвольные системы, представимые в виде набора объектов и связей между ними. За последние десятилетия популярность графов значительно возросла. Объясняется это их универсализмом и независимостью от предметной области, что позволяет обрабатывать и анализировать системы произвольной сложности. Информационный взрыв, вызванный развитием Всемирной Паутины, и развитие компьютерных технологий сделали доступным большое количество информации. Сегодня теория графов используется в биологии при анализе ветвящихся процессов, а именно, размножении бактерий, в радиоэлектронике – при проектировании печатных схем, в химии – для описания кинетики сложных реакций, в экономике – при оптимизации маршрутов и грузоперевозок, в социологии – для изучения связей и закономерностей в обществе и т.п. Графы являются центральным инструментом при разработке программного обеспечения, структурировании бизнес-процессов, проектировании баз данных и используются во многих областях математики и компьютерных наук.

В современном мире существует большое количество информации, которую можно представить в виде объектов и отношений между ними. Например, объектами могут являться научные статьи, тогда, если одна из них ссылается на вторую, то между этими статьями есть связь. Таким образом, все существующие научные работы могут быть представлены в виде графа. Такое же представление возможно для многих других структур из самых разных областей знания: люди и их социальные взаимоотношения, интернет-ресурсы, ссылки.

Для всех подобных структур является естественным их представление в виде графа. Все описанные примеры являются так называемыми социальными графами: они обладают неким набором свойств, специфичных только для такого типа графов. С каждым днем размеры графов увеличиваются, их сложность растет, и даже такая важная задача, как визуализация, становится проблематичной. Один из возможных подходов к ее

решению заключается в более высокоуровневом представлении исходного графа: изображать вместо вершин графа их группы. Однако, важно, чтобы при группировке вершин не потерялась глобальная структура графа. Такие группы можно назвать сообществами в графе.

Методы выделения сообществ в социальных графах активно исследуется в последнее время, и визуализация графа — всего лишь один из практических аспектов этой задачи. В данной работе анализ графов будет идти именно с этой точки зрения.

Целью выпускной квалификационной работы является разработка клиент-серверного веб-приложения. У приложения есть три главных задачи:

1. Получение социально-демографической информации пользователей из социальной сети ВКонтакте.

2. Визуализация данных на плоскости в виде, пригодном для дальнейшего анализа.

3. Расчет характеристик социального графа.

В работе освящены темы касательно математических основ анализа социальных сетей, алгоритмов визуализации графов, а также технических аспектов работы с API ВКонтакте и программной реализацией системы.

2. Основы теории графов

2.1. Базовые понятия теории графов

Граф – система, которая может быть представлена как множество кружков и множество соединяющих их линий (геометрический способ задания графа – см. рис. 1). Кружки называются вершинами графа, линии со стрелками – дугами, без стрелок – ребрами. Граф, в котором направление линий не выделяется (все линии являются ребрами), называется неориентированным; граф, в котором направление линий принципиально (линии являются дугами) называется ориентированным (см. рис. 2). Теория графов может рассматриваться как раздел дискретной математики (точнее – теории множеств), и формальное определение графа таково: задано конечное множество X, состоящее из n элементов ($X = 1, 2, ..., n$), называемых вершинами графа, и подмножество V декартова произведения $X \times X$, то есть $V \subseteq X^2$, называемое множеством дуг. Вводится понятие ориентированного графа G: совокупность (X, V) (неориентированным графом называется совокупность множества X и множества неупорядоченных пар элементов, каждый из которых принадлежит множеству X). Дугу между вершинами i и j, $i, j \in X$, будем обозначать (i, j). Число дуг графа будем обозначать m ($V = (v_1, v_2, ..., v_m)$).

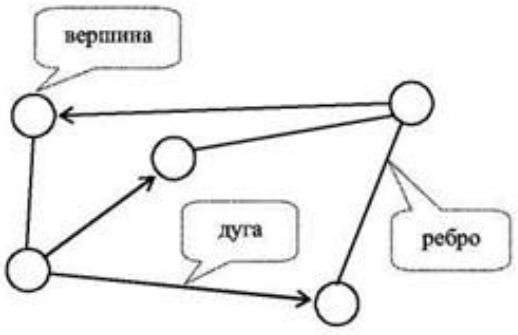


рис. 1

Подграфом называется часть графа, образованная подмножеством вершин вместе со всеми ребрами (дугами), соединяющими вершины из этого множества. Если из графа удалить часть ребер (дуг), то получим частичный граф.

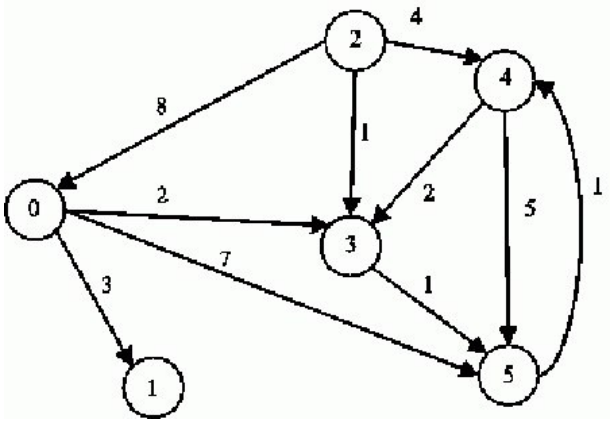


рис. 2

Две вершины называются смежными, если они соединены ребром (дугой). Смежные вершины называются граничными вершинами соответствующего ребра (дуги), а это ребро (дуга) – инцидентным соответствующим вершинам.

Путем называется последовательность дуг (в ориентированном графе), такая, что конец одной дуги является началом другой дуги. Простой путь – путь, в котором ни одна дуга не встречается дважды. Элементарный путь – путь, в котором ни одна вершина не встречается дважды. Контур – путь, у которого конечная вершина совпадает с начальной вершиной.

Граф, для которого из $(i, j) \in V$ следует $(j, i) \in V$, называется симметрическим. Если из $(i, j) \in V$ следует, что $(j, i) \notin V$, то соответствующий граф называется антисимметрическим.

Цепью называется множество ребер (в неориентированном графе), которые можно расположить так, что конец (в этом расположении) одного ребра является началом другого. Другое определение: цепь – последовательность смежных вершин. Замкнутая цепь называется циклом. По аналогии с простым и элементарным путем, можно определить соответственно простые и элементарные цепь и цикл. Любой элементарный цикл является простым, обратное утверждение в общем случае неверно.

Элементарная цепь (цикл, путь, контур), проходящая через все вершины графа называется гамильтоновой цепью (соответственно – циклом, путем, контуром). Простая цепь (цикл, путь, контур), содержащая все ребра (дуги) графа называется эйлеровой цепью (соответственно – циклом, пу-

тем, контуром). Если любые две вершины графа можно соединить цепью, то граф называется связным. Если граф не является связным, то его можно разбить на связные подграфы, называемые компонентами. Связностью графа называется минимальное число ребер, после удаления которых граф становится несвязным. Для ориентированных графов, если любые две вершины графа можно соединить путем, то граф называется сильно связным.

Перечислю известные факты:

связность графа не может быть больше, чем $[2m/n]$, где $[x]$ – целая часть числа x;

существуют графы с n вершинами и m ребрами, имеющие связность $[2m/n]$;

в сильно связном графе через любые две вершины проходит контур.

Связный граф, в котором существует эйлеров цикл, называется эйлеровым графом.

В неориентированном графе степенью вершины i называется число d_i инцидентных ей ребер. Очевидно, $d_i \leq n-1, i \in X$. Граф, степени всех вершин которого равны n – 1, называется полным. Граф, все степени вершин которого равны, называется однородным.

Вершина, для которой не существует инцидентных ей ребер ($d_i = 0$) называется изолированной. Вершина, для которой существует только одно инцидентное ей ребро ($d_i = 1$) называется висячей.

Известно, что: $\sum_{i \in X} d_i = 2m$ (данное выражение называется «леммой о рукопожатиях» – поскольку в каждом рукопожатии участвуют две руки, то при любом числе рукопожатий общее число пожатых рук четно (при условии, что каждая рука учитывается столько раз, в скольких рукопожатиях она участвовала)); в любом графе число вершин нечетной степени четно.

Связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны (теорема Эйлера). Обозначим n_k – число вершин, имеющих степень $k, k = 0, 1, 2, ...$ Известно, что: $\sum_{k:n_k>0} k n_k = 2m$

2.2. Области применения

Среди жителей Кёнигсберга была распространена такая практическая головоломка: можно ли пройти по всем мостам через реку Преголя, не проходя ни по одному из них дважды? В 1736 году выдающийся математик Леонард Эйлер заинтересовался задачей и в письме другу привел строгое доказательство того, что сделать это невозможно. В том же году он доказал формулу, которая связывает число вершин, граней и ребер многогранника в трехмерном пространстве. Формула таинственным образом верна и для графов, которые называются "планарными". Эти два результата заложили основу теории графов и неплохо иллюстрируют направление ее развития по сей день. Граф как математический объект оказался полезным во многих теоретических и практических задачах. Наверное, дело в том, что сложность его структуры хорошо отвечает возможностям нашего мозга: это структура наглядная и понятно устроенная, но, с другой стороны, достаточно богатая, чтобы улавливать многие нетривиальные явления. Если говорить о приложениях, то, конечно, нужно упомянуть большие сети: Интернет, карта дорог, покрытие мобильной связи и т.п. В основах поисковых машин, таких, как Yandex и Google, лежат алгоритмы на графах. Помимо computer science, графы активно используются в биоинформатике, химии, социологии.

С появлением графов, достаточно очевидно, что появилась потребность в визуализации данных.

Типичные области применения графов

- Software engineering: UML диаграммы, диаграммы вызовов
- Биология: геномика, пищевые цепи
- Сети: инструменты управления сетями, Интернет
- Безопасность: сетевые атаки
- Социальные сети: Twitter, Facebook

Пользовательские требования к визуализации данных:

- Читабельность: видны основные структурные особенности графа.
- Конформизм: рисунок должен соответствовать стилевым соглашениям, характерным для конкретной прикладной области.
- Управляемость: пользователь может контролировать параметры укладки.
- Быстродействие соответствует цели (динамический граф на экране / высококачественная диаграмма для печати)

2.3. Примеры приложений

Приведу ряд примеров приложений теории графов.

1. «Транспортные» задачи, в которых вершинами графа являются пункты, а ребрами – дороги (автомобильные, железные и др.) и/или другие транспортные (например, авиационные) маршруты. Другой пример – сети снабжения (энергоснабжения, газоснабжения, снабжения товарами и т.д.), в которых вершинами являются пункты производства и потребления, а ребрами – возможные маршруты перемещения (линии электропередач, газопроводы, дороги и т.д.). Соответствующий класс задач оптимизации потоков грузов, размещения пунктов производства и потребления иногда называется задачами обеспечения или задачами о размещении. Их подклассом являются задачи о грузоперевозках.

2. «Технологические задачи», в которых вершины отражают производственные элементы (заводы, цеха, станки и т.д.), а дуги – потоки сырья, материалов и продукции между ними, заключаются в определении оптимальной загрузки производственных элементов и обеспечивающих эту загрузку потоков.

3. Обменные схемы, являющиеся моделями таких явлений как бартер, взаимозачеты и т.д. Вершины графа при этом описывают участников обменной схемы (цепочки), а дуги – потоки материальных и финансовых ресурсов между ними. Задача заключается в определении цепочки обменов, оптимальной с точки зрения, например, организатора обмена и согласованной с интересами участников цепочки и существующими ограничениями.

4. Управление проектами.

С точки зрения теории графов проект – совокупность операций и зависимостей между ними. Примером является проект строительства некоторого объекта. Совокупность моделей и методов, использующих язык и результаты теории графов и ориентированных на решение задач управления проектами, получила название календарно-сетевого планирования и управления.

5. Модели коллективов и групп, используемые в социологии, основываются на представлении людей или их групп в виде вершин, а отношений между ними (например, отношений знакомства, доверия, симпатии и т.д.) – в виде ребер или дуг. В рамках подобного описания решаются задачи исследования структуры социальных групп, их сравнения, определения агрегированных показателей, отражающих степень напряженности, согласованности взаимодействия и др.

6. Модели организационных структур, в которых вершинами являются элементы организационной системы, а ребрами или дугами – связи (информационные, управляющие, технологические и др.) между ними.

Завершив краткое описание прикладных областей, перейдем к рассмотрению web-графов.

3. Web-графы

Расскажу о том, как зародилась наука о построении моделей веб-графов. Схемы интернета представлена с сайта internet-map.net для стран Россия, Германия, Чехия (см. рис. 3 - 5).

Карта Интернета представляет собой снимок глобальной сети по состоянию на конец 2011 года. Он охватывает более 350 тысяч сайтов из 196 стран и всех доменных зон. Информация о более чем 2 миллионах ссылок между сайтами объединила некоторые из них в тематические кластеры. Как и следовало ожидать, крупнейшие кластеры формируются национальными веб-сайтами, то есть сайтами, принадлежащими одной стране.

Глобальная сеть Интернет - явление технологической цивилизации, и ее исключительная сложность превосходит все, что когда-либо создавало человечество. По существу, здесь мы имеем дело с огромным количеством совершенно неструктурированной информации. Интернет-карта - это попытка заглянуть в скрытую структуру сети, понять ее колоссальный масштаб и исследовать то, что невозможно понять из голых цифр статистики.

Как и любая другая карта, Интернет-карта - это схема, отображающая относительное положение объектов; но в отличие от реальных карт (например, карты Земли) или виртуальных карт (например, карты Мордора), объекты, показанные на ней, не выровнены по поверхности. С математической точки зрения, Интернет-карта представляет собой двумерное представление связей между веб-сайтами в Интернете. Каждый сайт представляет собой круг на карте, и его размер определяется трафиком веб-сайта. Чем больше объем трафика, тем больше круг. Переключение пользователей между сайтами формирует ссылки, и чем сильнее связь, тем ближе сайты к друг другу. Ради удобства все веб-сайты по отношению к определенной стране имеют одинаковый цвет. Например, красная зона вверху соответствует русскому сегменту сети, желтая обозначает китайский сегмент, фиолетовый - японский, большой светло-голубой - американский сегмент, и т.п. В качестве платформы для отображения использовался движок Google Maps.

Концепция карты Интернета также используется в визуализации бо-

лезней во всем мире, информация представлена на сайте: disease-map.net. Электронные медицинские записи стали неотъемлемой частью национальных систем здравоохранения. Во всем мире крайне важно всесторонне использовать информацию, содержащуюся в базе данных. Ученые утверждают, что представленная реализация может помочь в подтверждении уже известных болезней. А также в выявлении новых, ранее не замеченных, связанных с функциональными аспектами медицины и здравоохранения.

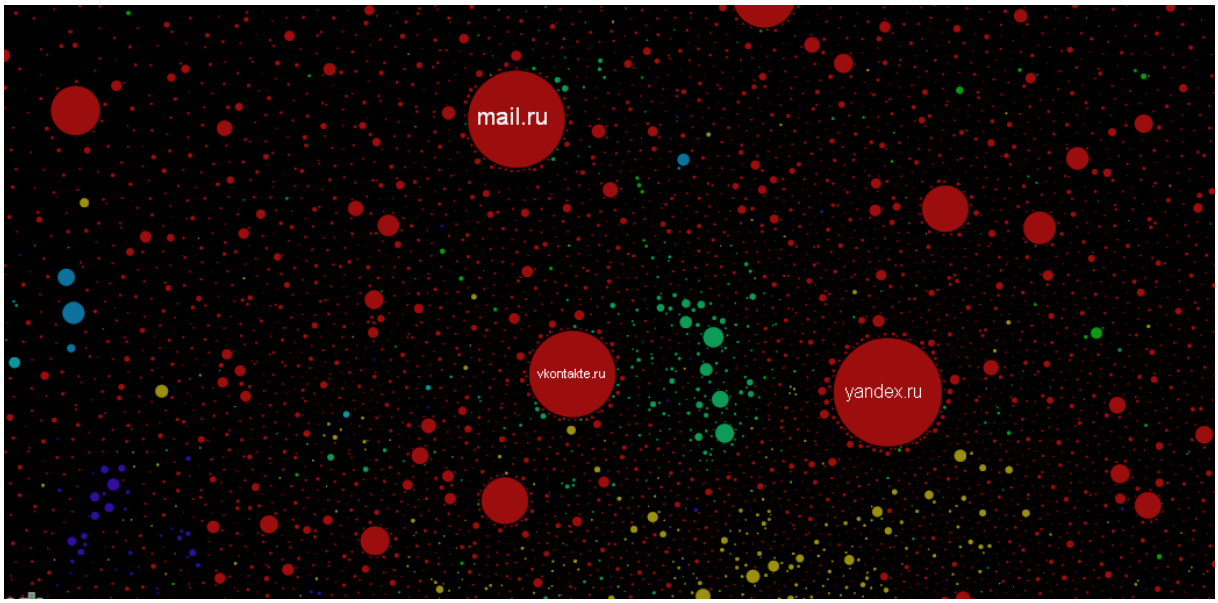


рис. 3

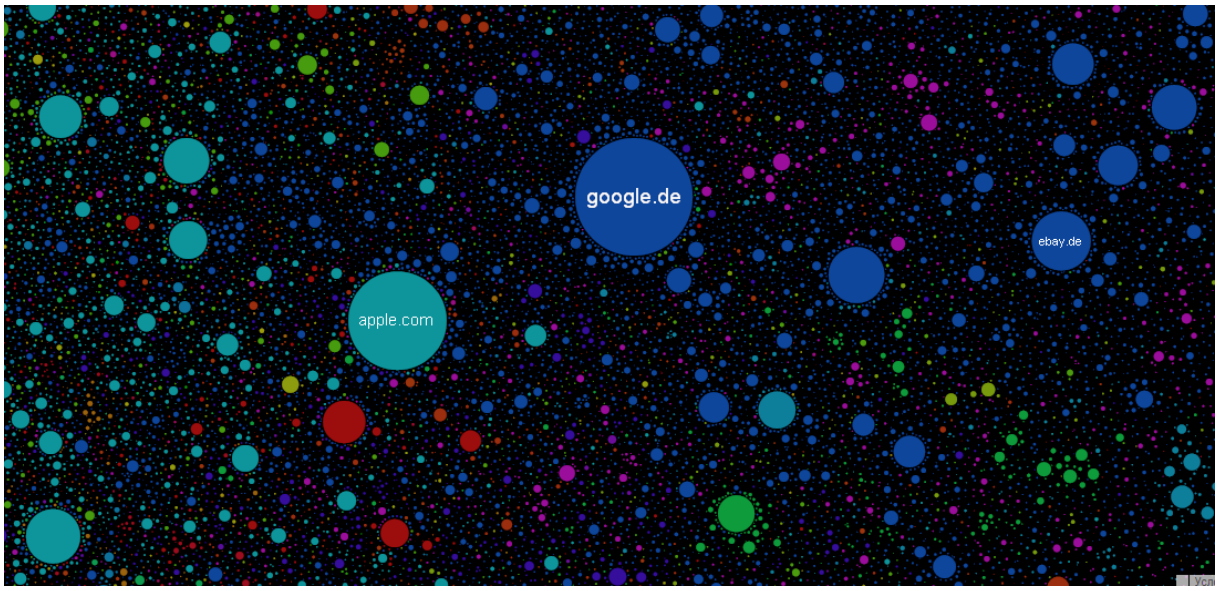


рис. 4

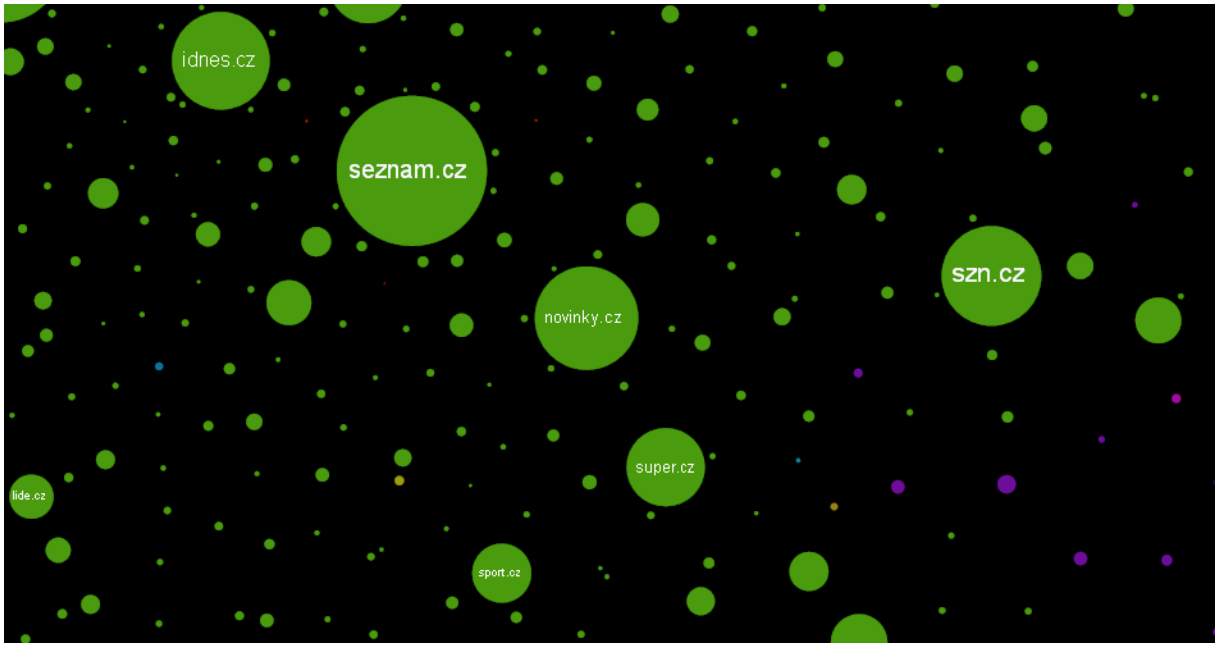


рис. 5

В математике и компьютерных науках графами называют абстрактные структуры данных, описывающие структурные отношения между объектами. Сегодня графы широко используются для моделировании данных в различных предметных областях, требующих выяснения и определения правил и схем отношений объектов, а также возможных отклонений. Такими предметными областями являются Web-графы, социальные сети, семантический Web, базы знаний, сети белок-белковых взаимодействий, библиографические сети и т. д. Постоянно увеличивающиеся объемы графовых данных этих приложений приводят к острой необходимости использовать масштабируемые системы, позволяющие эффективно обрабатывать огромные массивы данных. Введу наиболее популярные определения web-графа.

Web-граф - граф, отображающий страницы Всемирной паутины и прямые ссылки между ними.

Web-граф является ярким примером крупномасштабного графа. По оценкам Google, общее количество web-страниц превышает 1 триллион, экспериментальные графы всемирной паутины содержат более 20 миллиардов вершин (страниц) и 160 миллиардов ребер (гиперссылок). Другой пример – графы социальных сетей. По сообщениям Facebook, в 2012 году эта социальная сеть состояла более чем из миллиарда пользователей (вершины графа) и 140 миллиардов дружеских отношений (ребра графа). Сеть

LinkedIn состоит почти из 8 миллионов вершин и 60 миллионов ребер. При этом графы социальных сетей быстро растут. Так, количество пользователей Facebook увеличилось с 1 миллиона пользователей в 2004 году до 1 миллиарда в 2012. Что касается семантического web, то онтология DBpedia (проект на основе Wikipedia) в настоящий момент состоит из 3,7 миллиона объектов (вершин) и 400 миллионов фактов (ребер).

Еще в середине 90-х годов XX века, около 15–20 лет назад, про интернет практически никто не знал. А если и знал про его существование, то вряд ли имел к нему доступ.

Будем представлять интернет в виде графа. Вершинами этого графа будут сайты, и между двумя вершинами A, B мы проведем столько ребер, направленных от A к B, сколько есть ссылок с сайта A на сайт B, и столько ребер, направленных от B к A, сколько есть ссылок с сайта B на сайт A. Нас будет интересовать устройство этого графа, который по понятным причинам принято называть web-графом. Оказывается, web-граф обладает рядом устойчивых свойств — свойств, которые остаются неизменными на протяжении всей истории исследований интернета. Не претендуя на полноту, опишем несколько таких свойств. Их уже будет достаточно для понимания того, как сильно реальная картина мира противоречит нашей интуиции.

3.1. Свойства веб-графа

Первое свойство веб-графа многим хорошо известно, хотя обычно речь идет о другом. Напишу подробно о законе «шести рукопожатий». А именно, у каждого человека есть знакомые, у этих людей также есть свои знакомые и т. д. Наблюдение состоит в том, что от любого человека до любого другого человека на Земле можно «пройти» по такой цепочке взаимных знакомств и что количество «звеньев» в ней не превзойдет шести. Иными словами, я пожму руку своему другу, он пожмет руку одному из своих приятелей, и через не более чем шесть таких рукопожатий (при правильном выборе их последовательности) очередным знакомым окажется президент страны или какой-нибудь разносчик пиццы из Огайо, — вообще, любой на-

перед заданный человек. Ровно та же история с интернетом. Только здесь рукопожатия заменяются «кликами» (компьютерной мышью по ссылкам) и утверждается, что для перехода с любого сайта на любой другой сайт потребуется не более шести кликов (при правильном выборе их цепочки).

В терминах теории графов речь идет о диаметре графа. Дадим его определение. Расстоянием между вершинами графа называется количество ребер в кратчайшей реберной цепочке, соединяющей эти вершины. Если граф ориентированный (как, например, web-граф), то все рёбра в рассматриваемых цепочках должны следовать друг за другом в одном и том же направлении. Диаметр — это самое большое расстояние между вершинами в графе. Разумеется, бывают несвязные графы. У каждого из них диаметр считается равным бесконечности. Закон шести кликов — это факт, состоящий в том, что диаметр web-графа равен шести. Для обозначения диаметра графа G используют запись $diamG$. Отмечу, что это наблюдение касалось 1999 года, когда величина интернета была соответсвующей.

Второе свойство web-графа состоит в его «разреженности». То есть, если вершин у web-графа n, то ребер у него не более mn с некоторым постоянным $m \geq 1$. В самом деле, даже если пренебречь тем, что в web-графе бывают кратные рёбра и кратные петли (с одних страниц данного сайта вполне могут идти ссылки на другие его же страницы), ему ничто не мешает иметь $C_n^2 = \frac{n(n-1)}{2}$ ребер. Но последняя величина растет квадратично по n, тогда как реальное количество ребер значительно меньше: их, как максимум, mn. В некотором смысле особенно меньше и быть не может: если у графа на n вершинах меньше чем $n-1$ ребер, то этот граф заведомо не связен.

И еще одно, третье, свойство. Давайте смотреть на степени вершин web-графа. Тут, конечно, есть тонкость, связанная с тем, что у ориентированного графа бывают как входящие степени $indegv$ (число ребер, правым концом которых служит данная вершина), так и исходящие степени $outdegv$. Если не оговорено противное, мы будем понимать под степенью вершины сумму ее входящей и исходящей степеней, т.е. число всех ребер, концом которых она является: $degv = indegv + outdegv$. Нас интересует доля вершин web-графа, имеющих данную степень. Иными словами, пусть n — количе-

ство вершин web-графа, а d — некоторое фиксированное число. Обозначим через (n, d) величину $\frac{|\{v:degv=d\}|}{n}$, т.е. мы делим количество вершин степени d (модулем обозначена мощность множества, заключенного в фигурных скобках) на общее количество вершин и получаем искомую долю. Оказывается, что всегда $(n, d) \approx \frac{c}{d^{2.3}}$

Здесь $d \neq 0$, поскольку web-граф связен, а c — константа, которую легко найти, так как мы знаем, что сумма всех величин $|\{v : degv = d\}|$ равна n, откуда $\sum_d (n, d) = 1$. В сущности, (n, d) — это вероятность того, что вершина графа имеет степень d, а сумма всех вероятностей должна равняться единице. Гораздо удивительнее здесь константа 2,3, которая не меняется с течением времени. Описанное свойство называется степенным законом распределения степеней вершин веб-графа.

Несмотря на кажущуюся хаотичность в процессе образования интернета, есть весьма жесткие статистические ограничения, которым он годами подчиняется. Зададимся вопросами: почему это так? Что стоит за всеми свойствами интернета? Каковы законы, управляющие формированием сети? Мало того, что все эти вопросы крайне важны для понимания устройства мира, — ответы на них не могут не принести и серьезную практическую пользу: имея правильную модель интернета, можно пытаться лучше выявлять некоторые виды спама, тестировать алгоритмы обхода интернета поисковым роботом и др. Все эти свойства также применимы к графам социальных сетей. Эти свойства были описаны двумя учеными Боллобаши и Альбертом в 1999 году.

3.2. Модель А. Л. Барабаши и Р. Альберта

В 1999 году двое исследователей А. Л. Барабаши и Р. Альберт предложили крайне простую идею, которая оказалась весьма продуктивной. Идея заключалась в том, что, когда новый сайт появляется на свет, он, скорее всего, будет ссылаться на те сайты, которые и без того уже многими цитированы. Более точно, вероятность, с которой новый сайт ставит ссылку на сайт-предшественник, пропорциональна (входящей) степени вершины веб-графа, отвечающей этому сайту. Один из наиболее удобных и матема-

тически строгих вариантов реализации идеи Барабаши–Альберта (идеи о предпочтительном присоединении) сформулировали в 2000 году математики Б. Боллобаш и О. Риордан.

Зададимся вопросом: как можно придумать вероятностную модель графа, который бы обладам этими свойствами?

Напишу кратко о теории случайных графов. В 1959 году два математика П.Эрдеш и А. Реньи придумали модель случайного графа. Зафиксируем натуральное число $n \in N$ и число $p \in [0; 1]$, которое впоследсвие буду рассматривать как вероятность. Возьмем множество вершин графа (натуральные числа). Пока нет никакой случайности. Случайными будут ребра. Коротко напишу: есть вероятность p, проведу ребра независимо друг от друга. Каждое проводим с вероятностью p, всего $C_n^2 = \frac{n(n-1)}{2}$ ребер. По схеме Бернулли: если выпала решка, то проводим ребро e_1, если выпал орел, то не проводим ребро e_1 и так далее со всеми ребрами. И так делаем C_n^2 раз. Посторили случайный граф по схеме Бернулли.

Схемой Бернулли называется последовательность независимых в совокупности испытаний, в каждом из которых возможны лишь два исхода — «успех» и «неудача», при этом успех в одном испытании происходит с вероятностью $p \in (0, 1)$, а неудача — с вероятностью $q = 1 - p$.

У этого графа среднее число ребер: $p * C_n^2$, где p - это вероятность выпадения ребра. Если $p = 2m/n$, то $p*C_n^2 \sim mn$. Свойству разреженности данная вероятность ребра удовлетворяет. Исслуем распределение степеней вершин в таком графе. Есть какая-то вершина n, остаются $n - 1$ вершин. Вероятность того, что степень данной вершины равна d:

$P(degv = d) = C_{n-1}^d * p^d * (1 - p)^{n-1-d}$. Мы получили вероятность того, что в конкретные d вершин ребра идут, а в оставшиеся вершины $n - 1 - d$ ребра не идут. Перебираем всевозможные наборы таких вершин.

Есть такое вероятностное наблюдение: если вероятность ребра такая, как мы показали, то величина $P(degv = d)$ может быть приближенно записана в другом виде по предельной теореме Пуассона.

Если $p = const/n$, то $C_{n-1}^d * p^d * (1 - p)^{n-1-d} \sim c^d e^{-c}/d!$

Сформулирую теорему Пуассона. Пусть $n \mapsto \infty$ и $p_n \mapsto 0$ так, что $np_n \mapsto \lambda > 0$. Тогда для любого $k \geq 0$ вероятность получить k успехов

в n испытаниях схемы Бернулли с вероятностью успеха p_n стремится к величине :

$C_n^k * p_n^k * (1 - p_n)^{n-k} \sim \frac{\lambda^k}{k!} e^{-k}$

Мы получили, что если мы желаем добиться с помощью модели Эрдаша и Реньи разреженности, то нужно брать $p = const/n$. Но тогда закон распредения степеней вершин будет не как в модели, то есть не степенной закон, а Пуассоновский. Вывод: модель Эрдаша и Реньи не годится для расмотрения нашего случая. Она для классического случая.

Позже А. Л. Барабаши и Р. Альберт предложили идею, связанную со степенным законом распределения. Идея: есть уже какое-то количество сайтов и появляется новый сайт с номером n. Этот сайт с большей вероятностью сошлется на предшественника с большим количеством ссылок. Эта модель назвается моделью предпочтительного соединения. Но эта модель не объяснила, как относительно друг друга будут концы n ребер и не понятно, как вести исследования.

К настоящему времени придумано очень много разных моделей интернета, которые куда более ближе к реальности, чем модели Боллобаша–Риордана. Это огромная увлекательная область теории случайных графов, которую еще предстоит развивать и систематизировать.

4. Визуализация графов

Визуализация информации играет большую роль в жизни человека. Считается, что около 90% всей получаемой информации человек получает через зрение. Человечество за тысячи лет преодолело путь от простейших способов визуализации в виде наскальных рисунков до карт, схем и диаграмм. В настоящее время визуализация – неотъемлемый элемент обработки сложной информации о строении объектов. Многие структуры данных, представляющие практический интерес в математике и информатике, могут быть представлены в виде графов. Одним из основных классов является класс иерархических и/или атрибутированные графов. Преимущества графов во многих случаях становятся ощутимыми только при наличии хороших инструментальных средств для их визуализации и обработки. По-

этому в настоящее время в мире происходит значительный рост интереса к методам и средствам визуализации графов, о чем свидетельствует рост числа публикаций, содержащих описания новых алгоритмов и способов визуализации графов, а также их реализации в программных средствах.

Визуализация или отображение графов, как ответвление теории графов, относящееся к топологии и геометрии — двумерное представление графа. В основном, это графическое представление укладки графа на плоскость (как правило, допускается пересечение рёбер), направленное на удобное отображение некоторых свойств графа или моделируемого объекта.

Проблема визуализации графов встаёт, например, при отображении больших интегральных схем, анализе социальных сетей, в таких областях как картография и биоинформатика.

Далее приведу привер алгоритма визуализации графов.

4.1. Метод связывания ребер

Также данный метод называют метод иерархических жгутов ребер.

Как было сказано ранее, универсальным средством представления абстрактных данных, часто очень больших, являются графы. К сожалению, изображения графов реального мира часто выглядят не эстетично и беспорядочно из-за большого количества пересечений ребер, а также наложений ребер и вершин. Одним из методов уменьшения хаоса и загруженности является техника связывания ребер в жгуты.

Иногда полезно представить граф в графической форме, так чтобы была видна структура. Можно привести десятки примеров, где это может пригодиться: визуализация иерархии классов и пакетов исходного кода какой-нибудь программы, визуализация социального графа (тот же Twitter или Facebook) или графа цитирования (какие публикации на кого ссылаются) и т.д. Но количество ребер в графе зачастую настолько велико, что нарисованный граф просто невозможно разобрать (см. рис. 6). Это граф зависимостей некоторой программной системы. Он представляет собой дерево разбиения на пакеты (серые шарики — пакеты, белые — классы), на которое поверх наложены ребра зависимости одних классов от других.

Рассмотрим метод иерархических жгутов ребер.

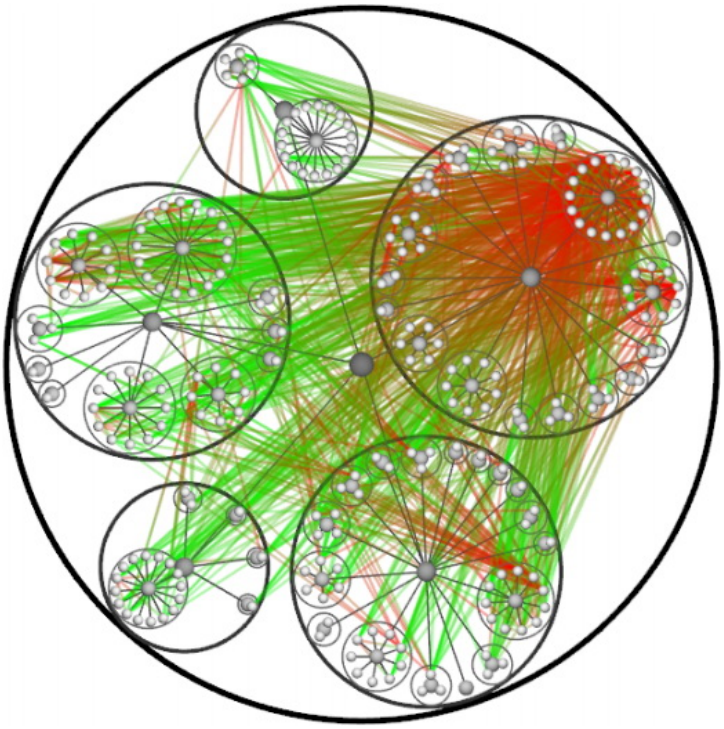


рис. 6

Чтобы не рисовать стрелки направления, ребра нарисованы в виде градиентных линий, где зеленый — это начало, а красный — конец ребра (см. рис. 6). Как видите, граф настолько визуально перегружен, что архитектуру программы невозможно проследить. Методы иерархических жгутов ребер предназначены для построения изображений наборов данных специального вида. С одной стороны, в этих наборах данных имеются иерархические отношения вида «отец-сын» между отдельными элементами, а с другой стороны, существуют и другие отношения, не являющиеся иерархическими. Примером являются социальные сети, в которых может существовать как иерархия индивидуумов, так и другие отношения, например, отношения знакомства. Далее опишу метод, решающий эту проблему.

4.1.1. Описание метода.

Идея метода связана с принципом, что и в кабельных сетях. Известно, какой хаос начинается, когда проводов становится слишком много. Для того, чтобы этот хаос предотвратить, провода объединяют в жгуты. Как эту идею применить к графам и реализовать алгоритмически?

Любая структура, о которой говорили ранее, может быть представлена в виде графа $G(V, E)$, где V- это множество вершин, E - множество ребер графа G, и дерева $T(V, A)$, где V- это множество вершин, A - множество дуг дерева T. При этом множество V является одновременно множеством вершин дерева T и графа G. Для создания иерархических жгутов сначала строится изображение дерева T любым известным алгоритмом визуализации деревьев. Затем координаты вершин построенного дерева используются в качестве опорных точек, через которые проводится каждое ребро графа G. Таким образом, каждому ребру $e = (P_{start}, P_{end})$ сопоставляется последовательность вершин P, состоящая из вершин, расположенных вдоль пути по дереву T от вершины P_{start} через вершину $LCA(P_{start}, P_{end})$ к вершине P_{end}. Вершина LCA (least common ancestor, наименьший общий предок между соединяемыми вершинами) - это минимальный общий предок вершин P_{start} и P_{end}.

Затем из списка P удаляется вершина LCA для того, чтобы связи между вершинами одного уровня не искривлялись через общего родителя этих вершин. Изображение ребра (результирующая кривая, которая и отрисовывается на экран в виде ребра) $e = (P_{start}, P_{end})$ строится в виде кубического β-сплайна по формуле:

$P' = \beta P_i + (1 - \beta)(P_0 + \frac{i}{N-i}(P_{N-1} - P_0)$, где N - количество контрольных точек, i - номер контрольной точки (точки сплайна), $i \in 0, ..., N - 1$, P_i-позиция i-ой контрольной точки, β - коэффициент искривления, $\beta \in [0, 1]$.

С помощью коэффициента искривления β можно управлять формой жгутов.

Рассмотрим пример рисования одного ребра на примере (см. рис. 7). Необходимо провести ребро из вершины P0 в вершину P4:

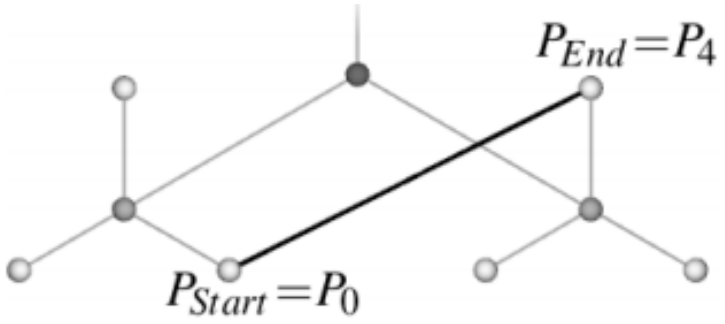


рис. 7

Найдем путь между этими вершинами (см. рис. 8):

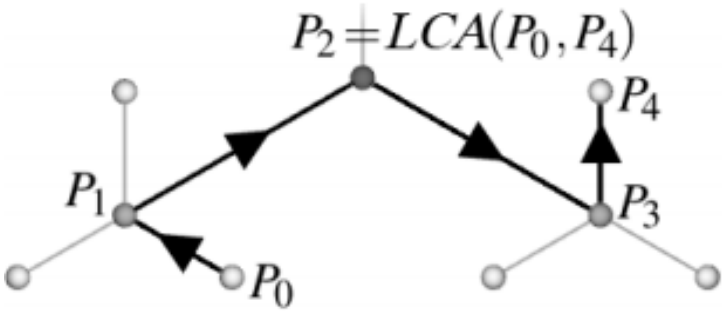


рис. 8

Теперь проведем кривую через полигон, образованный точками P0, P1, P2, P3, P4 (см. рис. 9):

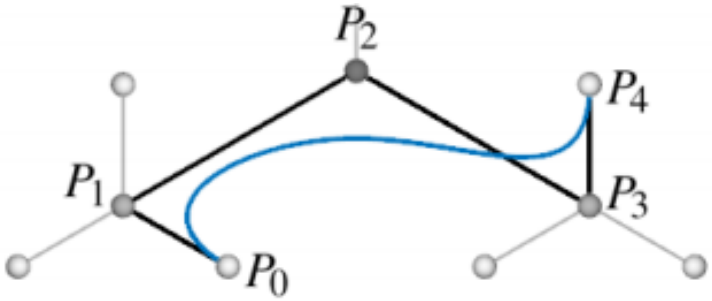


рис. 9

Авторы этого метода в работе [5] проанализировали, что лучше всего для целей визуализации в качестве кривых подходят кусочно-заданные кубические β-сплайны. Задача нахождения таких β-сплайнов является классической и давно решена. β-сплайн — сплайн-функция, имеющая наименьший носитель (замыкание множества, на котором функция отлична от нуля) для заданной степени, порядка гладкости и разбиения области определения. Фундаментальная теорема устанавливает, что любая сплайн-функция

для заданной степени, гладкости и области определения может быть представлена как линейная комбинация β-сплайнов той же степени и гладкости на той же области определения. Термин β-сплайн был введён И. Шёнбергом и является сокращением от словосочетания «базисный сплайн».

Кубический β-сплайн — это просто набор кривых третьего порядка в двухмерном пространстве, для которых выполняется условия сшивки первых и вторых производных на краях. Для того, чтобы управлять степенью связанности ребер, вводится параметр β, который принимает значения от 0 до 1 (0 — ребра не связаны в жгут и представляют собой независимые прямые линии, 1 — ребра максимально связаны друг с другом).

Представим результат с применением введенного β-сплайна (см. рис. 10):

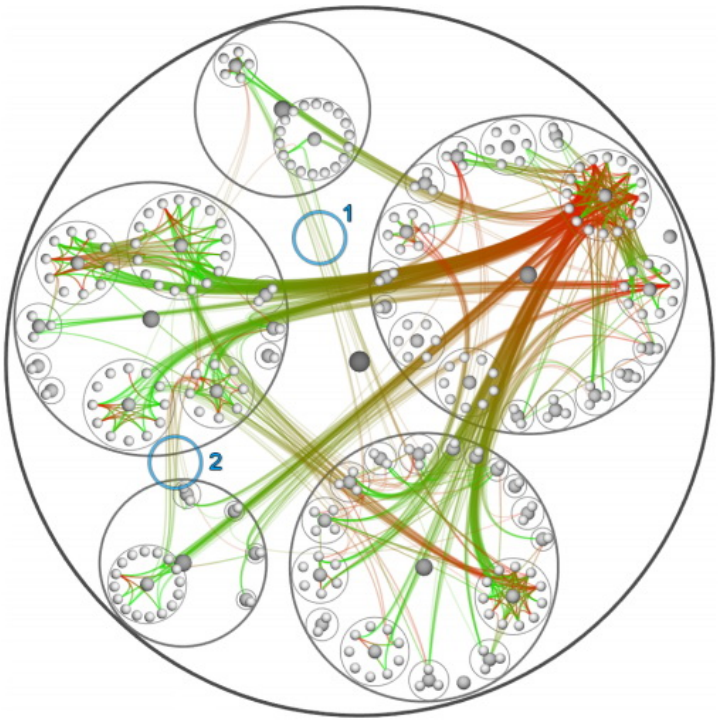


рис. 10

Здесь $\beta = 0.85$. Теперь архитектрура программы представляется значительно яснее. Видны отдельные связки ребер, заметны зависимости между пакетами.

Представим тот же граф, только используется радиальный способ визуализации архитектуры (без применения метода, эквивалентно $\beta = 0$) (см. рис. 11):

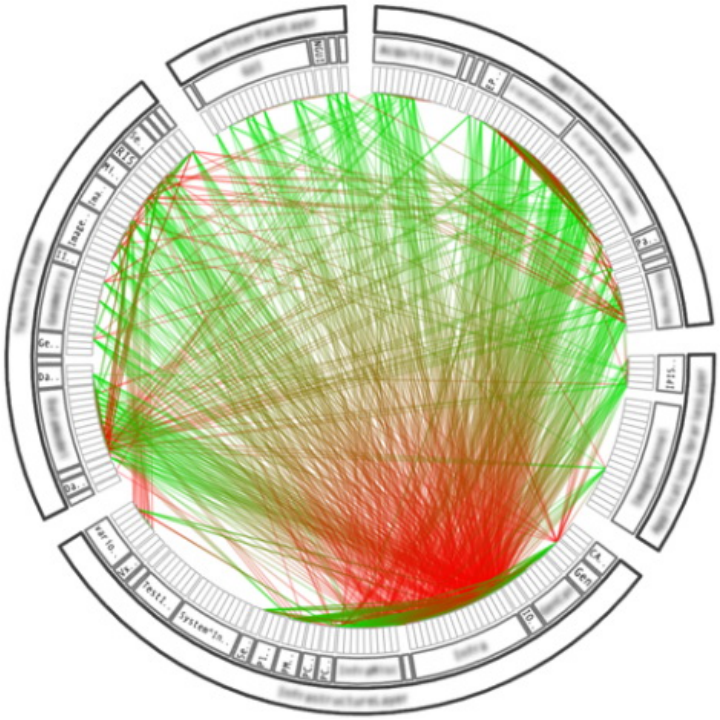


рис. 11

Граф с применением метода ($\beta = 0.85$) (см. рис. 12):

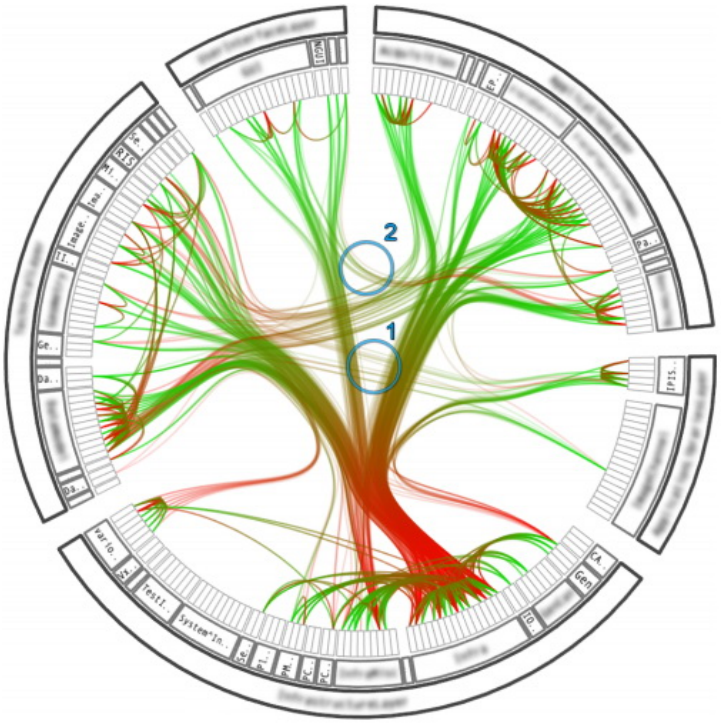


рис. 12

Как видно, метод связывания ребер можно сочитать с различными методами рисования деревьев: в этом его большое преимущество.

С помощью коэффициента искривления β можно управлять формой жгутов. На рис. 13 показаны изображения графа, состоящего из 31 вершины. Дерево T представляет собой трехуровневую иерархию с коэффициентом ветвления 5. Оно изображено при помощи кругового алгоритма. Корень дерева T находится в центре изображения, пять сыновей корня расположены на окружности, центром которой является корень дерева T, а внуки корневой вершины расположены на окружностях, центрами которых являются сыновья. Дуги дерева T изображены при помощи прямых линий. Жгуты ребер, полученные для различных значений коэффициента искривления β (см. рис. 13 - 14):

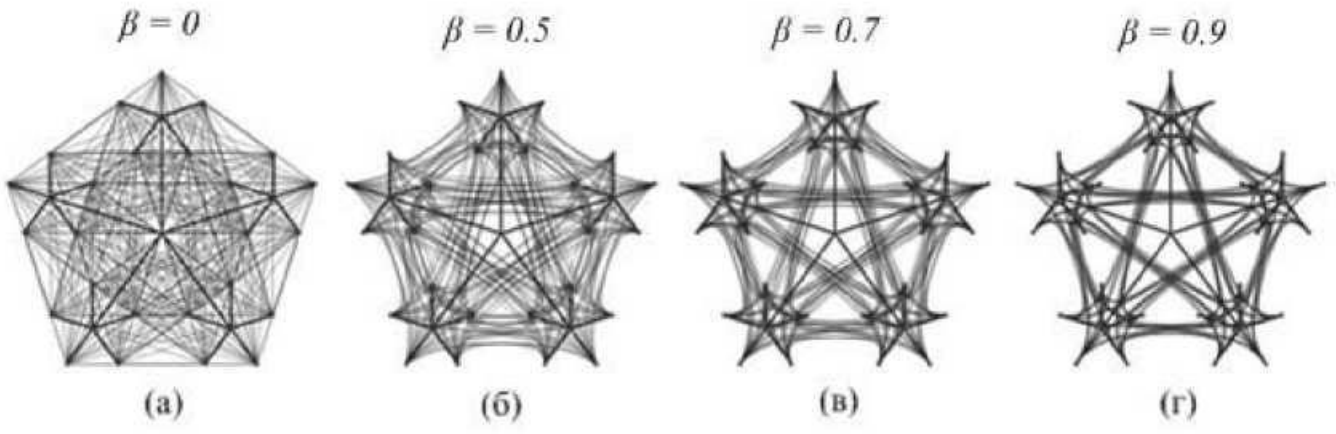


рис. 13

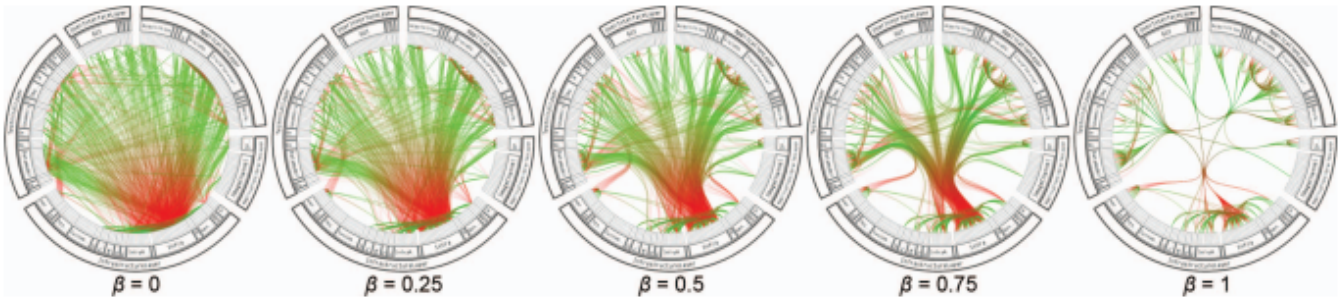


рис. 14

К основным достоинствам метода иерархических жгутов ребер можно отнести то, что он:

- может быть использован в сочетании с любыми технологиями визуализации деревьев и легко интегрирован в существующие программные

средства;

- уменьшает визуальную перегруженность изображения графа с большим количеством смежных ребер за счет того, что кривые объединяются в визуальные «жгуты»;
- позволяет регулировать степень кривизны кубического β-сплайна.

Основной проблемой с применением метода иерархических жгутов ребер является то, что в произвольных графах не всегда можно выделить подходящую иерархическую структуру, вокруг которой возможно построение иерархических жгутов. Поэтому дальнейшие исследования связаны с методами создания различных «опорных структур». В качестве опорной структуры может, например, использоваться собственная геометрия ребер.

4.2. Создание жгутов ребер на основе собственной геометрии ребер

Основная идея метода состоит в том, что жгуты ребер формируются на основе их собственной геометрии, а не привнесенной извне иерархии. Общая схема алгоритма выглядит следующим образом:

1) Сгенерировать прямоугольную сетку размера NxN и наложить ее на изображение графа, построенное любым способом.

2) Для каждой ячейки прямоугольной сетки вычислить основное направление ребер, пересекающих эту ячейку.

3) Объединить соседние ячейки с направлениями, отличающимися не более чем на пороговое значение a, в зоны

4) Вычислить основное направление в каждой зоне, и перпендикуляр к основному направлению зоны.

5) Построить отрезки, проходящие перпендикулярно направлению зоны, до пересечения с границей зоны.

6) Использовать полученные точки пересечения с границей каждой зоны для построения новой сетки.

7) Для каждого построенного ребра найти точки пересечений с ребрами исходного изображения графа. Вычислить центр среди этих точек.

8) Для каждого ребра графа G построить β-сплайн, проходящий через центральные точки ребер контрольной сетки, которые пересекает ребро графа G.

На рис. 15 (а, б) показаны изображения одного и того же графа, имеющего 3000 вершин, построенные методом иерархических жгутов ребер и методом создания жгутов на основе собственной геометрии. В данном случае очевидно преимущество метода построения жгутов на основе собственной геометрии. На рис. 15, б) «жгуты» более четкие, а изображение менее перегружено.

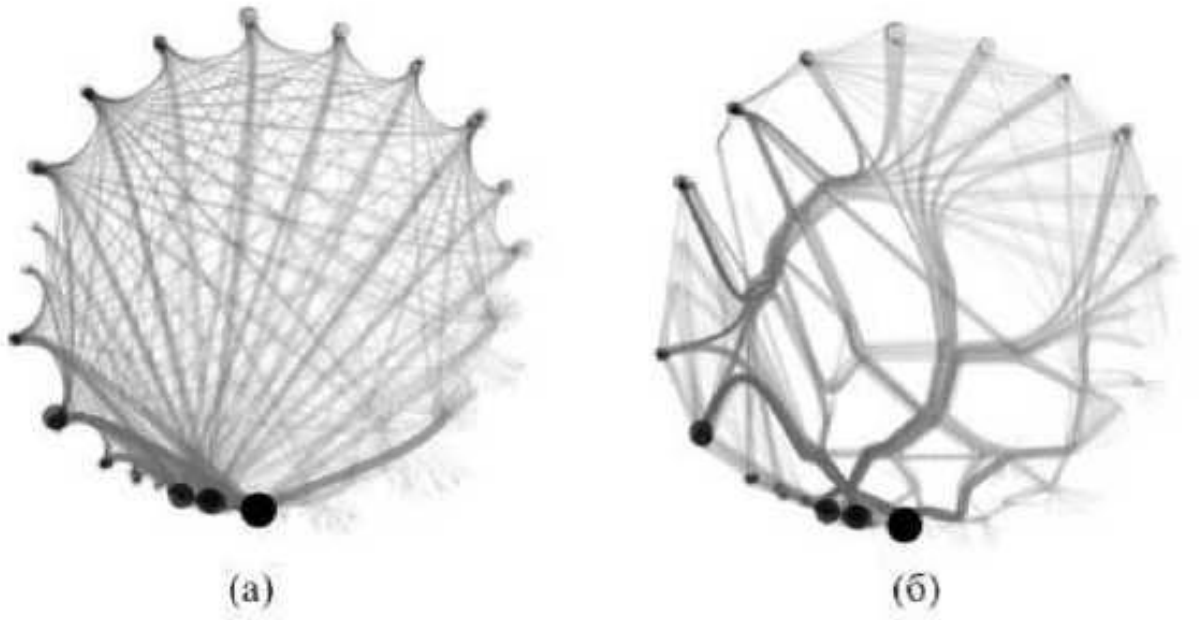


рис. 15

Одним из существенных недостатков этого метода является то, что при уменьшении размера сетки возникает слишком много «зигзагов», требующих последующей обработки. Проблемой является и то, что при создании сетки генерируется слишком много новых вершин, увеличивая временную сложность и затраты по памяти алгоритма вычисления кратчайшего пути. Наконец, на тонкой сетке получается слишком много путей, что уменьшает плотность и количество жгутов для длинных ребер.

Хочу отметить, что несмотря на то, что техника алгоритмов построения жгутов ребер возникла совсем недавно, она доказала свою полезность во многих приложениях и значит, в ближайшем будущем появятся новые

работы в этом направлении.

4.3. Другие примеры

Граф цитирования публикаций. Зеленый круг — это публикации 2008 года, которые ссылаются на публикации предыдущих годов (2007, 2006 и т.д.). Отмечаются только ссылки публикаций 2008 года. Без применения метода связывания ребер граф выглядит так (см. рис. 16):

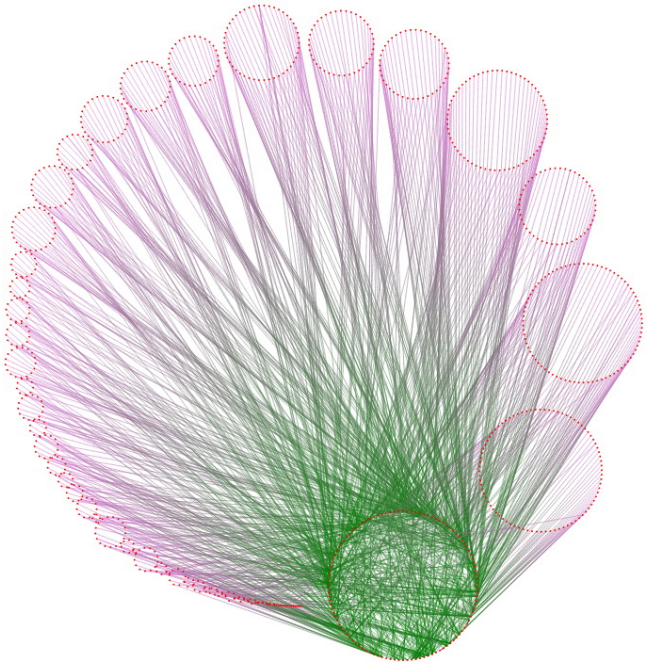


рис. 16

С применением метода ($\beta = 0.99$) см. рис. 17:

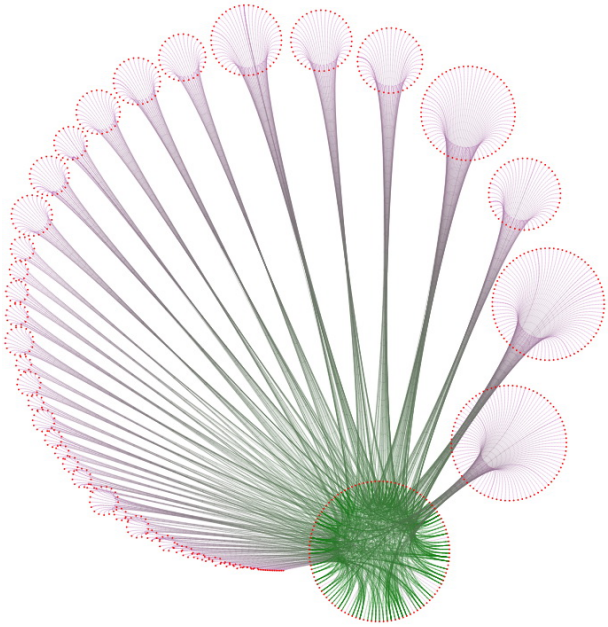


рис. 17

4.4. Графы в современных сервисах

Возвращаясь в современность, рассмотрим граф на миллионе вершин, который представляет собой кусок социальной сети. В зависимости от того, как мы изобразим этот граф, мы либо получим хаотично разбросанные точки и отрезки, либо хорошо просматривающийся набор данных. От алгоритма, который используется для отображения графа, зависит то, увидим ли мы симметрии этого графа, увидим ли сильно связанные части этого графа.

VivaGraphJS является самой быстрой графической библиотекой javascript. Представим некоторые примеры использования библиотеки в реальных проектах. Визуализация Amazon демонстрирует сопутствующие продукты на Amazon.com (см. рис. 18). Визуализация на YouTube наглядно показывает похожие видео с YouTube (см. рис. 19).

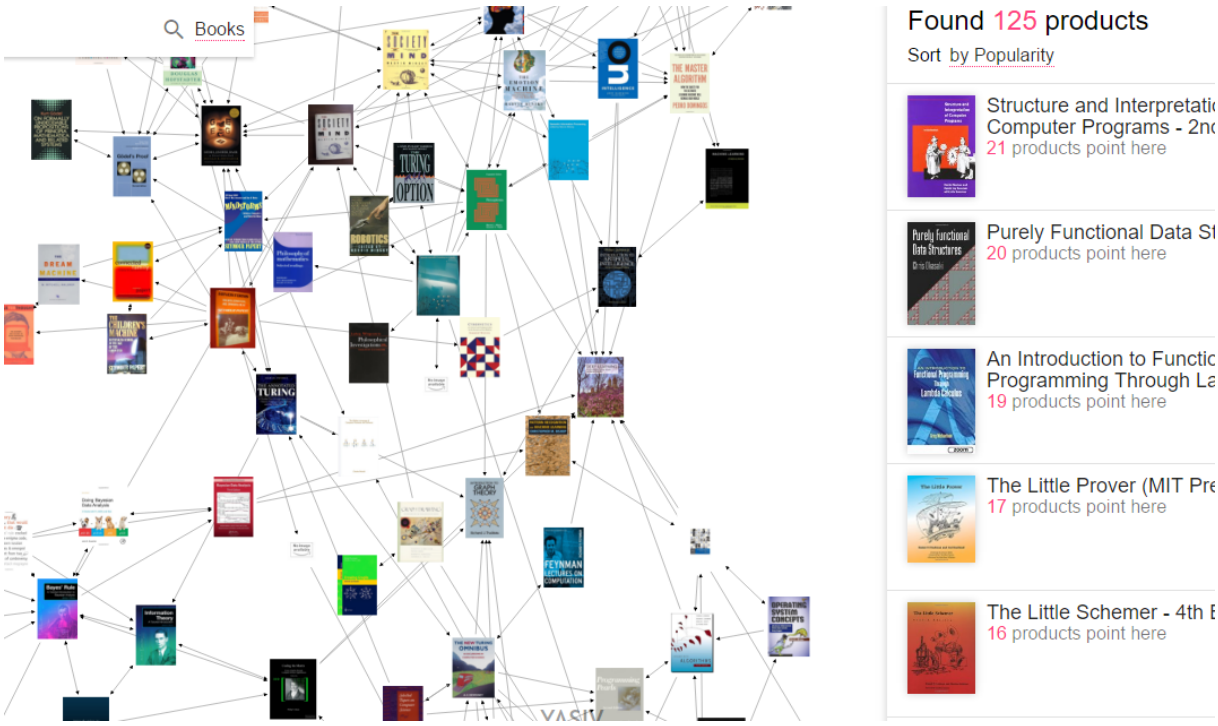

Books
Found 125 products
Sort by Popularity
Structure and Interpretati
Computer Programs - 2n
21 products point here
Purely Functional Data St
20 products point here
An Introduction to Functic
Programming Through La
19 products point here
The Little Prover (MIT Pre
17 products point here
The Little Schemer - 4th E
16 products point here

рис. 18

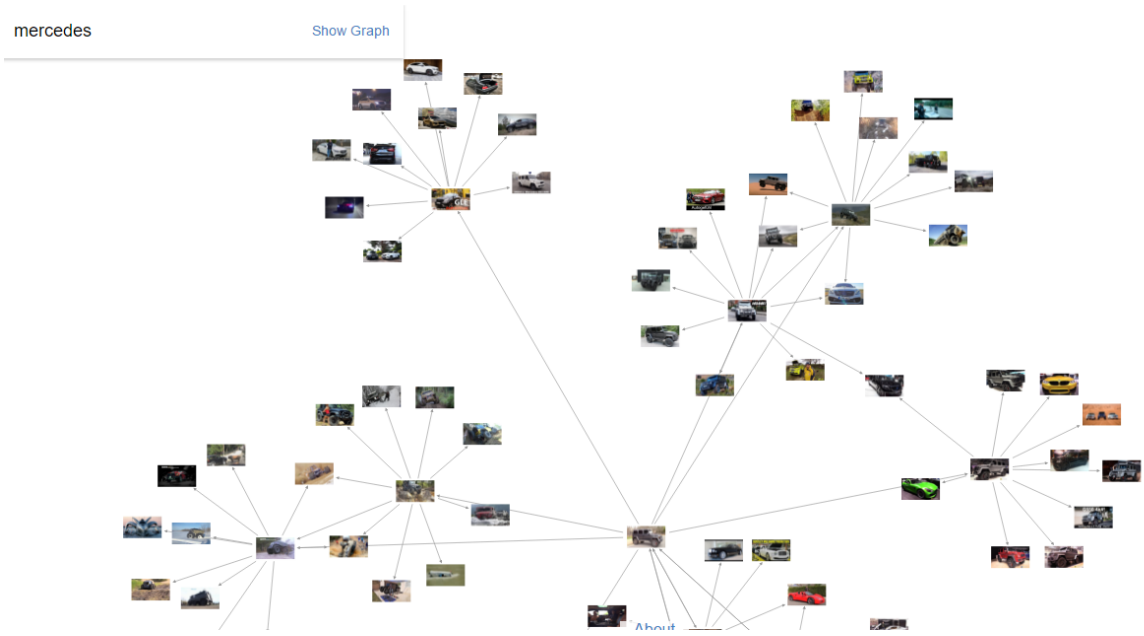

mercedes
Show Graph

рис. 19

5. Инструмент для визуализации графов

В своей работе я буду использовать библиотеку VivaGraph для визуализации графа. VivaGraph, одна из самых быстрых библиотек графических рисунков, построена из модулей ngraph.

Большее семейство модулей можно найти, запросив npm для ngraph. Ngraph - это набор алгоритмов, связанных с графами. Его можно использовать в браузере или на стороне сервера. Этот репозиторий представляет собой набор примеров, которые показывают, как использовать некоторые из них или создавать свои собственные. В настоящее время эта библиотека находится в стадии разработки.

Иногда макет в браузере вычислить невозможно (например, график слишком велик для достижения достойной производительности). В этом случае мы можем вычислить макет в автономном режиме и предоставить статические позиции браузеру. Например, в моей работе для визуализации группы пользователей математико-механичского факультета данные были сохранены, так как группа из большого числа пользователей и каждый раз строить граф затратно по времени.

Ngraph.pagerank - алгоритм, который вычисляет PageRank графа. Поясню, как будет считаться этот показатель. Подробное описание этой меры для социального графа будет в следующем разделе. Данный алгоритм будет делать анализ пользователей социальной сети и присваивать каждому узлу в веб-графе число, известное как PageRank. Идея, лежащая в PageRank, заключается в том, что PageRank страницы пользователя равен сумме показателей PageRank, которые были переданы подписчками пользователю.

6. Метрики для исследования социальной сети

6.1. Современная социальная сеть

Итак, в моей работе представлена визуализация связей всемирно известной сети Вконтакте. В данном случае связь - это дружба между двумя людьми. Четкого формального определения, какой граф можно называть социальным, нет. Ясно, что в основе этого понятия лежит граф знакомств между людьми в обществе. Однако, социальным графом назвается более широкий класс графов. Можно сказать, что социальный граф — это такой специальный вид графов, который характеризуется следующим набором свойств:

1.Одна большая общая компонента связности. В большинстве социальных графов присутствует одна большая компонента связности, которая захватывает большинство вершин. Остальные компоненты гораздо меньшего размера, возможно, они являются даже отдельными одиночными вершинами, как часто бывает в случае с графами социальных сетей. Например, для популярной социальной сети facebook установлено, что 99.91% вершин находятся в одной компоненте связности.

2. Распределение на степенях вершин. Социальные сети относятся к так называемым безмасштабным сетям (англ. scale-free network). Безмасштабная сеть или масштабно-инвариантная сеть — граф, в котором степени вершин распределены по степенному закону, то есть доля вершин со степенью k примерно или асимптотически пропорциональна $k^{-\gamma}$. γ – индивидуальная характеристика сети, которая, как правило, находится в интервале (2,3), но в редких случаях выходит за его границы.

В 1956 году Дерек Джон де Солла Прайс показал, что в сетях, образованных ссылками в научных публикациях, число ссылок на публикацию имеет распределение, похожее на степенной закон. В своих работах он не использовал термин “Безмасштабные сети”. Этот термин появился гораздо позже, в конце 90-х годов, когда Альберт-Лассо Барабаши в своих исследованиях топологии всемирной паутины обнаружил узлы-концентраторы, имеющие большое количество связей. В последствии он обнаружил подобные узлы и в других сетях, после чего дал найденному классу сетей назва-

ние “Безмасштабные сети”. Узлы-концентраторы (англ. hubs) - это узлы, степени которых очень велики по сравнению со степенями остальных узлов.

Эмпирически было установлено, что многие естественно возникающие сети — социальные, коммуникационные, биологические, графы цитирований ссылок во Всемирной паутине (англ. World Wide Web), и другие системы — хорошо моделируются безмасштабными графами. На рис. 20 можно увидеть, как ведет себя распределение в современных социальных сетях.

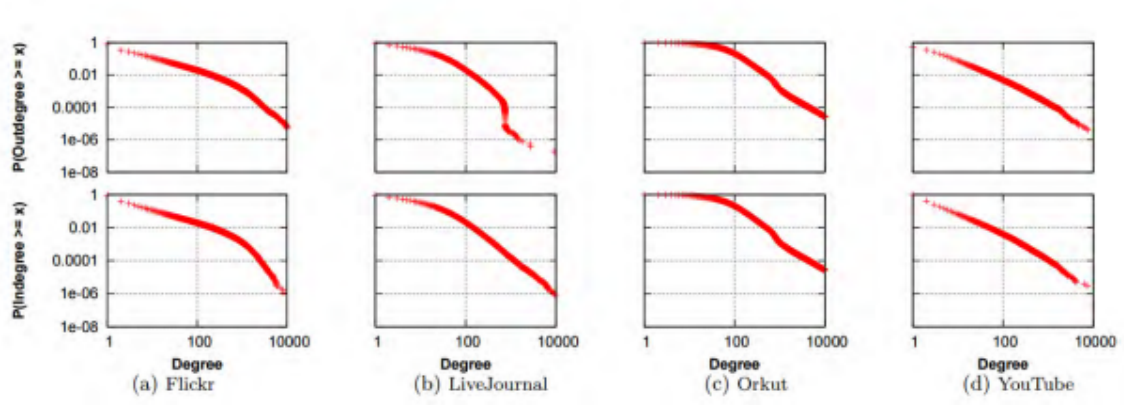


Рис.20 Исследование распределения степеней вершин в современных социальных сетях показывает, что они являются безмасштабными. Т.к. современные социальные сети являются направленными графами, то изучают как входящую (снизу) так и исходящую (сверху) степень вершин

3. Среднее расстояние. Под расстоянием между вершинами понимают минимальную длину цепи в графе, соединяющие эти вершины. Социальные сети в среднем имеют очень маленькое расстояние между двумя случайными вершинами. Это свойство может быть продемонстрированно на модели современного общества. Существует так называемый феномен тесного мира (или теория шести рукопожатий), согласно которому любые два человека на Земле разделены в среднем пятью уровнями общих знакомых и, соответственно, шестью уровнями связей. Позже этот факт неоднократно подтверждался на основе данных современных социальных сетей, таких как facebook (см. рис. 21).

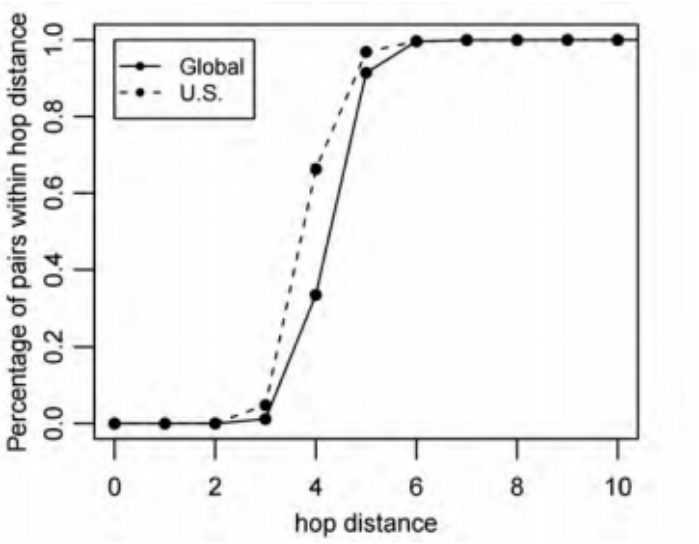


Рис. 21 Изображена функция соседства $N(h)$, которая показывает долю пользователей социальной сети facebook, которые находятся на расстоянии h связей от друг друга. Среднее расстояние между пользователями facebook было равно 4.7

4. Коэффициент кластеризации. Существует множество коэффициентов для графа, которые показывают некоторую его качественную характеристику.

Я буду говорить о задаче кластеризации или поиска разбиения, имея ввиду исходную задачу выделения сообществ в графе. Коэффициент кластеризации показывает, насколько сильно вершины склонны образовывать группы (или сообщества), которые характеризуются тем, что вершины, входящие в одну группу, соединены между собой гораздо плотнее, чем со всем остальным графом. Умение выделять сообщества из графа позволяет многое понять про сам граф. Так, размеры современных графов превышают сотни тысяч вершин и миллионы ребер, поэтому стандартное представление графа теряет свою информативность. Если объединить вершины в сообщества, то можно без потери информации о структуре графа исследовать его на более абстрактном уровне. На рис. 22 задан пример графов на одних и тех же вершинах. Первый явно имеет 3 сообщества, а второй нет.

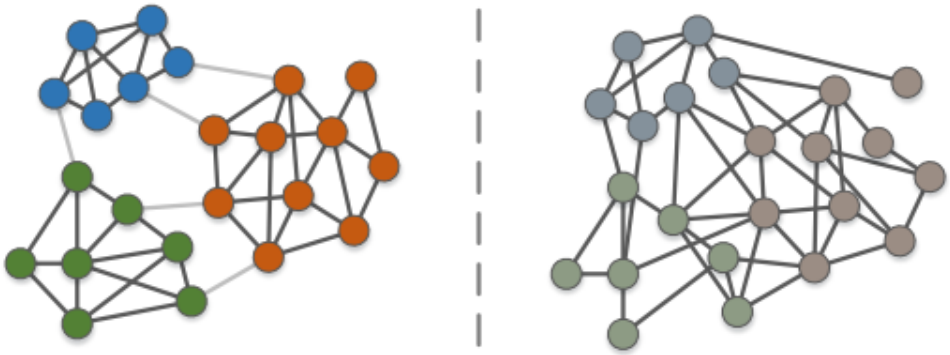


Рис. 22 Пример графа, который имеет структуру из трех сообществ (слева) и графа на тех же вершинах, который ее не имеет (справа).

В своей работе я покажу примеры объединения людей в сообщества в социальной сети Вконтакте.

В моей реализации есть качественные характеристики: modularity, centrality, PageRank. Подробнее опишу меру modularity.

6.2. Модулярность

Каждый человек - это вершина графа. В зависимости от дружбы людей строится граф. И данному графу можно задать меру modularity. Чем эта мера меньше, тем меньше будет сообщество. Она служит так называемым способом разбиения сообществ.

Введем определение понятия сообщество. Сообщество - это определенная группа вершин, причем связи внутри группы гораздо плотнее, чем вне группы. Я буду считать, что каждая вершина может войти только в одно сообщество. После выполнения алгоритма разбиения сообществ нам необходимо оценить качество алгоритма.

Самой популярной и общепризнанной мерой качества для данной задачи является значение модулярности (modularity). Функционал был предложен Ньюманом и Гирваном в ходе разработки алгоритма кластеризации вершин графа. Модулярность — это скалярная величина, принимающая значения на отрезке $[-1, 1]$, которая количественно описывает неформальное определение структуры сообществ, данное выше.

Введу наиболее популярные определения модулярности.

Modularity(модулярность) - оценка качества разбиения графа на подграфы, насколько данное разбиение качественно, т.е. существует много ребер, лежащих внутри сообществ, и мало ребер, лежащих вне сообществ, соединяющих сообщества между собой. Если значение модулярности находится в диапазоне от 0.3 до 0.7, то сеть имеет вполне различимую структуру с сообществами.

Modularity(модулярность) - это отношение суммы разниц между фактическим и ожидаемым количеством ребер (в случайно сгенерированном графе таких же размеров), соединяющих пару вершин из одного кластера. Таким образом, модулярность - это мера отклонения плотности связей в наблюдаемой сети от плотности, распределенной случайным образом.

Modularity(модулярность) - мера качества кластеризации, на основе строится широкий класс алгоритмов, оптимизирующих ее значение. Впервые этот индекс был предложен в [9], где авторы описали новый класс алгоритмов кластеризации на графах. Общая идея заключалась в итеративном вычислении меры, основанной на количестве путей, проходящих через каждое ребро графа и удаление ребер с максимальным значением этой меры.

В работе я рассмотрю алгоритм, основанный на поиске оптимального значения функции модулярности. Методы кластеризации, основанные на максимизации модулярности являются одними из самых популярных среди алгоритмов, позволяющих автоматически определять количество кластеров. Задача таких методов – поиск оптимального разбиения, максимизирующего значение функции модулярности. В целом, задачи такого рода являются NP-трудными, поэтому было разработано множество эвристических подходов.

Запишу формулу для определения модулярности:

$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{d_i d_j}{2m}) \sigma(C_i C_j)$

где A — матрица смежности графа, A_{ij} — (i, j) элемент матрицы, d_i — степень i вершины графа, C_i — метка вершины (номер сообщества, к которому относится вершина), m — общее количество ребер в графе, $\sigma(C_i C_j)$— дельта-функция: равна единице, если $C_i = C_j$, иначе равна нулю. Задача поиска выделения сообществ в графе сводится к поиску таких C_i, которые максимизируют значение модулярности.

Модулярность достаточно просто интерпретируется. Ее значение равно разности между долей ребер внутри сообщества и ожидаемой доли связей, если бы ребра были размещены случайно. Модулярность возможно эффективно пересчитывать при небольших изменениях в кластерах. Например, если добавить изолированную вершину в кластер C, то изменение функционала для взвешенного графа можно посчитать как

$\Delta Q = [\frac{\sum_{in}+d_{i,in}}{2m} - (\frac{\sum_{tot}+d_{in}}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{d_i}{2m})^2]$.

где $\sum_{in}$ — сумма весов ребер внутри сообщества C, $\sum_{tot}$— сумма весов всех ребер инцидентных вершинам из C, d_i — сумма весов ребер инцидентных вершине i, m—сумма весов.

6.3. Индекс значимости узла

PageRank (индекс значимости узла, далее PR) - является распределенной вероятностью и определяется не только за счет числа связей узла (вершины), но и за счет учета значимости самих связей. Принимается во внимание то, как много связей у каждой соседней вершины, связанной с данной.

PR — это числовая величина, характеризующая «важность» веб-страницы. Чем больше ссылок на страницу, тем она становится «важнее». Кроме того, вес страницы A определяется весом ссылки, передаваемой страницей B. Таким образом, PageRank — это метод вычисления веса страницы путём подсчёта важности ссылок на неё.

Само понятие PR является одним из ключевых моментов в работе поисковой машины Google. Наряду с другими параметрами, влияющими на выдачу (сортировку) сайтов в результатах поиска, знание модели PageRank необходимо как для понимания процесса поиска, так и для использования оптимизаторами при продвижении своих сайтов в поисковой системе. PageRank вычисляется при помощи случайного блуждания по графу, где в качестве узлов — страницы, а ребро между узлами — ссылка между страницами. Случайный блуждатель двигается по графу и время от времени перемещается на случайный узел и начинает блуждание заново. PageRank равен доли пребывания на каком-то узле за все время блуждания. Чем он больше, тем узел более авторитетен. PR не является основным, но является

одним из вспомогательных факторов при ранжировании сайтов в результатах поиска. Роль PR в поисковом сервисе Google особенно будет важна при ранжировании страниц по односложному запросу (когда пользователь вводит только одно слово в строке поиска). В этом случае получается, что очень много документов будут релевантны данному запросу. Вот именно в этом случае PR будет играть ключевую роль при определении позиций для этих одинаковых по релевантности (соответствии введенному запросу) документов. PR широко использовался при покупке/продаже ссылок, как главные ориентиры качества сайта.

PR является мерой "важности"страницы в социальной сети. Зависит от числа внешних ссылок на данную страницу и от их веса (важности). Другими словами от количества и качества ссылающихся страниц. А если говорить математическим языком, то PR - это алгоритм расчёта авторитетности страницы.

Классическая формула расчета PageRank:

$PR = (1 - d) + d \sum_{i-1}^{n} \frac{PR_i}{C_i}$, где

PR — PageRank рассматриваемой страницы,

d — коэффициент затухания (означает вероятность того, что пользователь, зашедший на страницу, перейдет по одной из ссылок, содержащейся на этой странице, а не прекратит путешествие по сети),

PR_i — PageRank i-й страницы, ссылающейся на рассматриваемую страницу,

C_i — общее число ссылок на i-й странице.

Основная идея работы с PR заключается в том, что страница передает свой вес, распределяя его на все исходящие ссылки. Чем больше ссылок на странице-доноре, тем меньший вес достанется каждой странице-акцептору.

С помощью PR в полученном социальном графе сделаем анализ авторитетности. Авторитетность в социальном графе можно анализировать разными способами. Самый простой — отсортировать участников по количеству входящих ребер. У кого больше — тот больше авторитетен.

7. Выделение сообществ в больших сетях

7.1. Метод выделения сообществ.

Опишу метод быстрого развертывания сообществ в больших сетях. Данный метод был введен в [10]. Это эвристический метод, основанный на оптимизации показателя модулярности. Показано, что он превосходит любой другой известный метод обнаружения сообществ в терминах времени вычисления. Модулярность используестя как целевая функция для опитимизации. Авторы статьи [11] внедрили этот алгоритм оптимизации модулярности, который позволяет изучать сети большого размера.

Перспективный подход заключается в декомпозиции сетей на подгруппы или сообщества, которые являются наборами узлов с высокой степенью взаимосвязи. Идентификация этих групп имеет важное значение, поскольку так можно раскрыть неизвестные функциональные модули, такие как темы в информационных сетях или кибер-сообщества в социальных сетях.

Точная модульная оптимизация - это задача, которая сложна в вычислительном отношении и поэтому алгоритмы аппроксимации необходимы при работе с большими сетями. Самый быстрый аппроксимационный алгоритм оптимизации модулярности на больших сетях был предложен Клаусетом (англ. Clauset et al). Этот метод состоит в периодическом объединении сообществ, которые оптимизируют модулярность.

Крупнейшими сетями, которые были рассмотрены до сих пор в литературе, являются белковый белок из 30739 узлов, японские социальные сети насчитывают около 5,5 миллионов пользователей. Социальная сеть Facebook имеет около 64 миллиона активных пользователей, оператор мобильной сети Vodaphone насчитывает около 200 миллионов клиентов и Google индексирует несколько миллиардов веб-страниц. Отмечу также, что в большинстве крупных сетей, таких как перечисленные выше, существует несколько естественных уровней организации: общины делятся на субсообщества - и поэтому желательно получить методы обнаружения сообществ, раскрывающие эту иерархическую структуру.

7.2. Описание метода

Опишу алгоритм, который находит высокомодульные разбиения больших сетей. За малое время он разворачивает полную иерархическую структуру сообщества для сети, тем самым предоставляя доступ к различным разрешениям обнаружения сообщества. В статье [11] авторы сообщают, что идентификация сообществ в 118-миллионной сети узлов с помощью этого алгоритма занимает 152 минуты.

Алгоритм разделен на две фазы, которые повторяются итеративно. Предположим, что мы начинаем с взвешенной сети из N узлов.

1) Во-первых, мы назначаем новое другое сообщество для каждого узла сети. Таким образом, в этом первоначальном разделе имеется столько сообществ, сколько есть узлов. Тогда для каждого узла i рассмотрим соседей j и оценим выигрыш модулярности, который будет иметь место путем удаления i из его сообщества и путем помещения его в сообщество j. Узел i помещается в сообщество, для которого существует максимальное усиление (в случае равенства мы используем правило нарушения), но только если это усиление положительно. Если положительный прирост невозможен, узел остается в своем первоначальном сообществе. Этот процесс применяется неоднократно и последовательно для всех узлов до тех пор, пока дальнейшего улучшения не будет. И первый этап завершен. Также отмечтим, что узел может рассматриваться несколько раз. Данная первая фаза останавливается, когда локальный максимум модулярности достигается, то есть когда никакое индивидуальное движение не может улучшить модулярность. Следует также отметить, что алгоритм зависит от порядка, в котором рассматриваются узлы. Предварительные результаты по нескольким тестовым случаям в работе [11] указывают на то, что упорядочивание узлов не оказывает существенного влияния на модулярность, но порядок может влиять на время вычисления. Проблему выбора порядка стоит изучить, поскольку так можно улучшить время вычисления алгоритма.

   Эффективность алгоритма связана с тем, что коэффициент усиления в модулярности ΔQ, полученный перемещением изолированного узла

i в сообщество C, может быть легко вычислен по формуле:

$\Delta Q = [\frac{\sum_{in}+d_{i,in}}{2m} - (\frac{\sum_{tot}+d_{in}}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{d_i}{2m})^2]$

Аналогичное выражение используется для того, чтобы оценить изменение модулярности при удалении узла i из сообщества. На практике оценивается изменение модулярности путем удаления i из его сообщества и затем, переместив его в соседнее сообщество.

2) Вторая фаза алгоритма состоит в построении новой сети, узлы которой будут являться сообществами, обнаруженными на первом этапе. Причем веса связей между новыми узлами задаются суммой весов связей между узлами в соответствующих двух сообществ. Как только эта вторая фаза выполнена, то можно повторно применить первый этап алгоритма к результирующей сети. По мере строительства количество мета-сообществ уменьшается на каждом этапе, и как следствие, большая часть вычислительного времени используется в первой фазе. Этапы повторяется (см. рис. 23) до тех пор, пока не будет больше изменений и максимум модулярности.

Итак, опишу суть этапов алгоритма: на первом этапе модулярность оптимизируется за счет допуска только локальных изменений сообществ; на втором этапе найденные сообщества агрегируются, чтобы построить новую сеть сообщества. Шаги повторяются итеративно до тех пор, пока не произойдет допустимо возможное увеличение модулярности (см. рис. 23).

Авторы в источнике [11] сравнивают данный алгоритм с другими и выясняют, что данный алгоритм выделения сообществ превосходил по времени.

Авторы применили даный алгоритм для большой сети, построенной по записям бельгийской компании мобильной связи, подсчитали общее количество телефонных звонков в течение 6-месячного периода. Сеть состояла из 2,6 млн. клиентов, между которыми есть связи, каждый клиент идентифицировался ключом, с которым связаны несколько записей, например, его возраст, пол, язык и почтовый индекс места проживания. Поременив алгоритм к данной большой сети, было выявлено, что в Бельгии два основных языковых сообществ: французский и голландский. С социологической

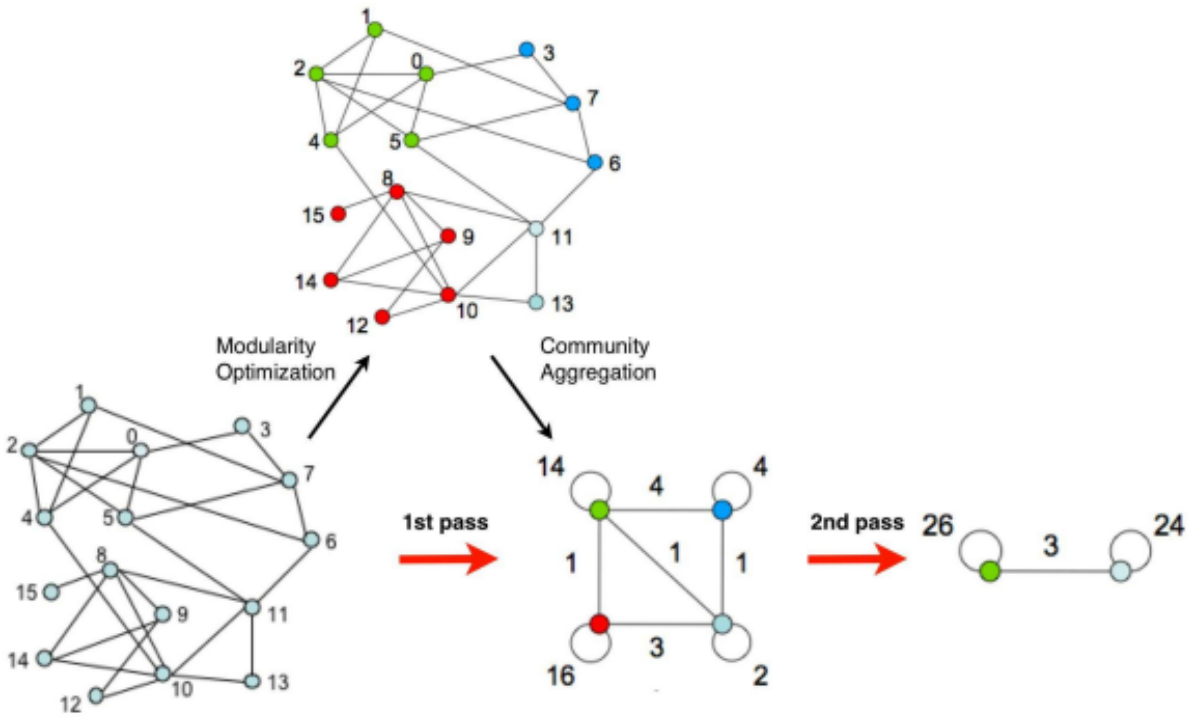


рис. 23

точки зрения, возможность выделить лингвистическую, религиозную или этническую однородность сообществ открывает перспективы для описания социальной сплоченности и потенциальной хрупкости страны.

На рис. 24 показан анализ крупнейших сообществ бельгийской сети мобильной связи. А именно, 261 сообщество, в котором более 100 клиентов. На долю этих сообществ приходится около 75% всех клиентов. Однородность сообщества характеризуется процентом лиц, говорящих на языке доминирующем сообщества; эта величина приближается к 1, когда сообщество стремится быть одноязычным. Есть 36 сообществ с более чем 10000 клиентов. За исключением одного сообщества эти сообщества образуют два языковых кластера, эти сообщества имеют более 85% членов, говорящих на одном языке.

Также авторы статьи заметили еще одно интересное наблюдение, связанное с наличием других языков. На самом деле существуют четыре языка для объединения клиентов этого конкретного оператора мобильной связи: французский, голландский, английский или немецкий. В то время как англоговорящие клиенты распределяются достаточно равномерно во всех

сообщетсвах, более 60% немецкоязычных клиентов сконцентрированы в одном сообществе. Это, вероятно, связано с тем, что говорящие по-немецки люди в основном сосредоточены в небольшом регионе недалеко от Германии, в то время как англоговорящие люди распространены по всей стране.

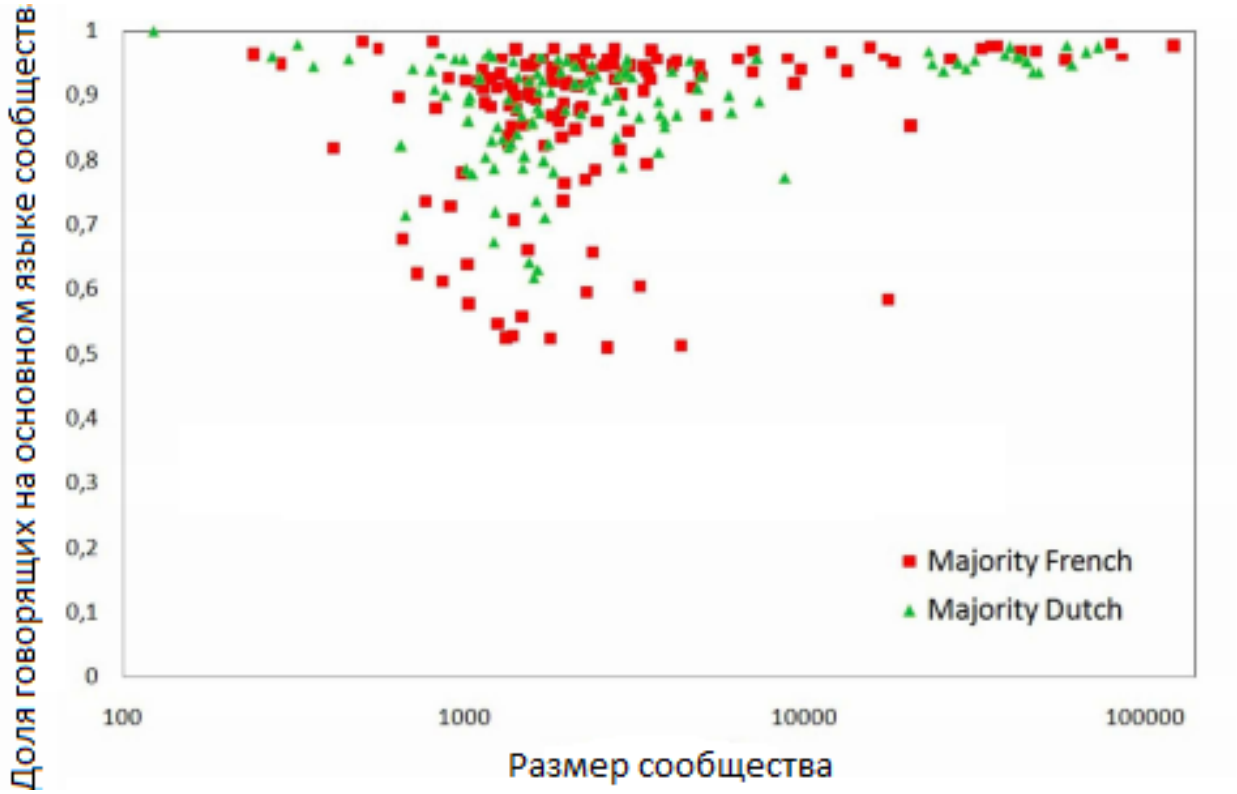


рис. 24

На рис. 25 представлен пример объединения клиентов в сообщество. А именно, представлено графическое представление сети обществ, извлеченных из Бельгийской сети мобильной связи. В этой сети представлено около 2 млн. клиентов. Размер узла пропорционален количеству людей в соответствующем сообществе и его цвет по красно-зеленой шкале представляет основной язык, на котором говорят в сообществе (красный для французского и зеленый для голландского). Если обратить внимание на промежуточное сообщество смешанных цветов между двумя языковыми кластерами, то можно увидеть при более высоком разрешении, что оно состоит из нескольких суб-общин с менее очевидным разделением языков.

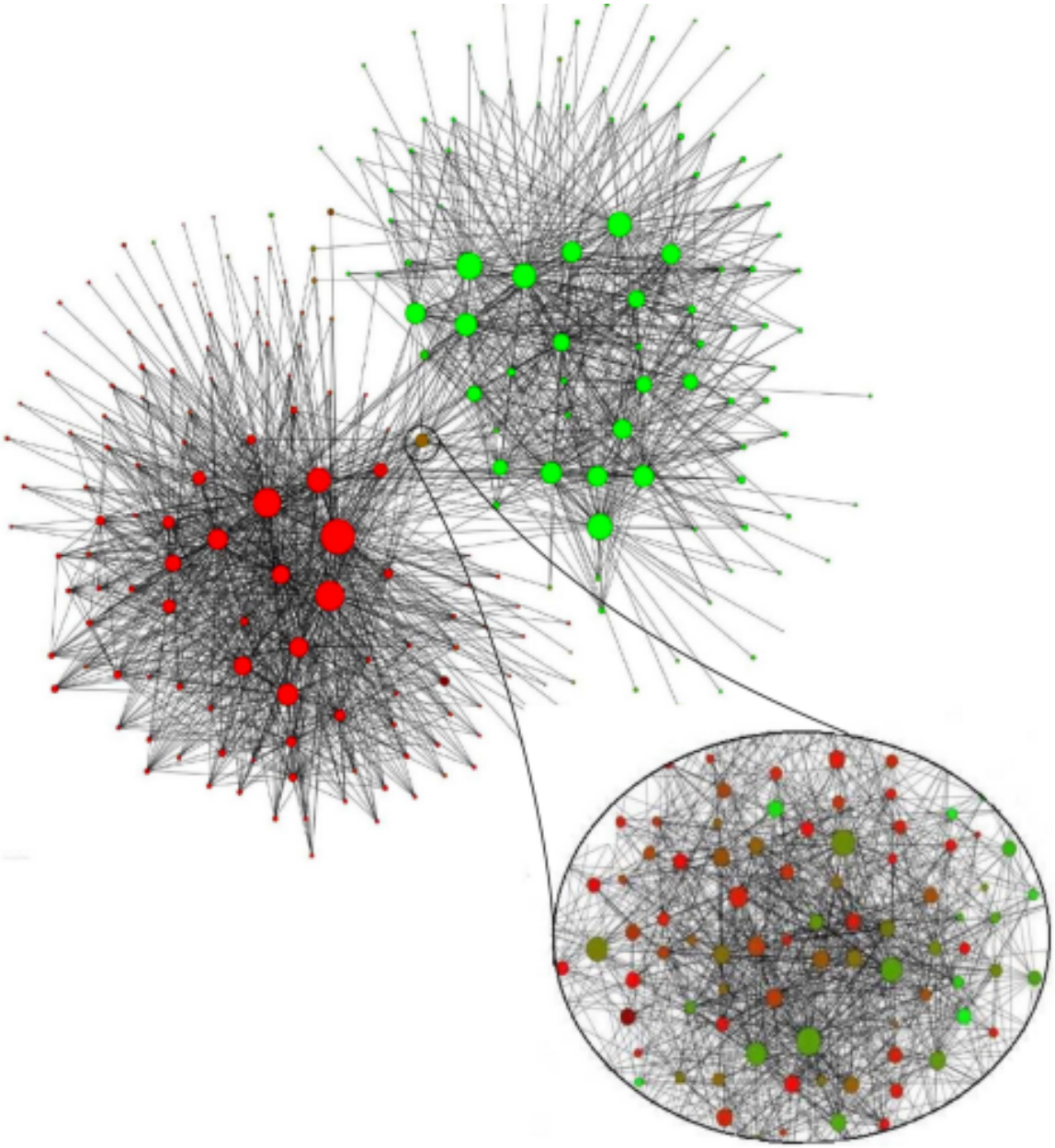


рис. 25

8. Метод подсчета коэффициента Centrality. Betweenness Centrality.

В ходе моей работы использовался метод подсчета коэффициента центральности (centrality). Betweenness Centrality — алгоритм подсчета коэффициента "центральности по посредничеству" (Betweenness), определяемый как количество кратчайших путей между всеми парами вершин, проходящих через данное ребро.

Данный метод разработан Ньюманом и Гирваном. Алгоритм работает по следующей схеме.

1. Подсчет коэффициентов "центральности по посредничеству" на всех ребрах графа.

2. Поочередное удаление ребер с самым большим коэффициентом.

3. Сообществами считаются оставшиеся компоненты связности.

4. Процедура удаления связей завершается, когда достигает максимума модулярность результирующего разбиения.

"Центральность по посредничеству"считается как количество кратчайших путей между всеми парами вершин, проходящих через данное ребро. Если между вершинами N кратчайших путей, то каждому ребру прибавляется $\frac{1}{N}$ к значению коэффициента. Чем больше данная величина, тем более вероятно, что данное ребро соединяет вершины из разных сообществ. Например, в крайней ситуации, когда разные группы соединены одним ребром, все кратчайшие пути из одного сообщества в другое проходят через это ребро. Отмечу, что данный алгоритм имеет множество модификаций, которые сводятся к подсчету других реберных коэффициентов.

На рис. 26 представлен граф, для ребер которого подсчитаны значения коэффициентов centrality. Если удалить ребро с самым большим значением (15), то граф разделится на 2 компоненты связности, которые можно рассматривать как 2 сообщества в исходном графе. Данный алгоритм является одним из первых алгоритмов по выделению сообществ в сетях. Его главный недостаток — время работы. Подсчет коэффициентов на ребрах является вычислительно сложной задачей. Сложность метода $O(m^2n)$, где m — общее количество ребер в графе, n - количество вершин. Это значит,

что данный метод не подходит для графов с большим количеством вершин. Действительно, в ходе работы [10] не на всех тестах удалось применить этот метод из-за его скорости.

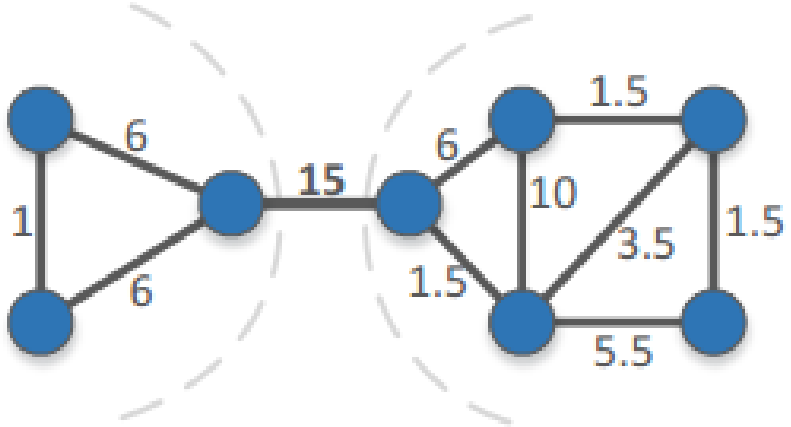


рис. 26

8.1. Математичекое описание метода

В данном алгоритме инструментом для анализа социальных сетей являются индексы центральной значимости, определенные на вершинах графа (Вавелас, 1948, Сабидусси, 1966, Фриман,1979). Они предназначены для ранжирования участников в соответствии с их положением в сети и интерпретируется как известность участникв, включенных в социальная структуру. Многие индексы ценностных ориентаций основаны на связывании кратчайших путей пар участников, измеряющих, например, среднее расстояние от других участников. Алгоритм предложен в работе [10]. Это наиболее эффективный из известных алгоритмов вычисления „центральности по посредничеству". В нем техника аккумулирования значений центральности интегрирована с поиском кратчайших путей.

Пусть исследуемая социльная сеть Вконтакте представлена как граф G = (V, E), где V - множество вершин (пользователей Вконтакте), а E - множество ребер (связи между пользователями). Мы используем n и m для обозначения количества вершин и ребер соответственно. Для простоты предположим, что все графики неориентированы и соединены, хотя они могут иметь петли или множественные ребра. Пусть ω - весовая функция

на ребрах. Предположим, что $\omega(e) > 0$, $e \in E$ для взвешенных графов, и определим $\omega(e) = 1$, $e \in E$, для невзвешенных графов. Весовые коэффициенты используются для измерения, например, прочности связи. Взвешенный граф — граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра).

Определим путь от $s \in V$ до $t \in V$ как чередующуюся последовательность вершин и ребер, начиная с s и заканчивая t, так что каждое ребро соединяется с его предшествующей и с его последующей вершиной. Длина пути - это сумма всех весов его ребер. Мы используем $d_G(s,t)$, чтобы обозначить расстояние между вершинами s и t, т. е. минимальную длина любого пути, соединяющего s и t в G. По определению, $d_G(s,s) = 0$ для любого $s \in V$ и $d_G(s,t) = d_G(t,s)$ для $s,t \in V$. Обозначим через $\sigma_{st} = \sigma_{ts}$ число кратчайших путей из $s \in V$ до $t \in V$, где $\sigma_{ss} = 1$. Обозначим через $\sigma_{st}(v)$ число кратчайших путей от s до t, на которых лежит некоторое $v \in V$. Ниже приведены стандартные меры централизации (C):

$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)}$- closeness centrality (Sabidussi, 1966)

$C_G(v) = \frac{1}{\sum_{t \in V} d_G(v,t)}$- graph centrality (Hage and Harary, 1995)

$C_S(v) = \sum_{s \neq v \neq t \in V} \sigma_{st}(v)$- stress centrality (Shimbel, 1953)

$C_B(v) = \sum_{s \neq v \neq t \in (V)} \frac{\sigma_{st}(v)}{\sigma_{st}}$- betweenness centrality (Freeman, 1977; Anthonisse, 1971)

Для управления размером сети, вышеупомянутые индексы обычно нормализуются, чтобы лежать между нулем и единицей. Индекс центральности является одним из наиболее часто используемых в анализе социальных сетей.

Я буду рассматривать меру „центральность по посредничеству“ (betweenness centrality, далее буду использовать обозначение c_B), говорящая о том, насколько часто рассматриваемая вершина графа лежит на путях между другими вершинами. Ее классическое определение, представленное в работе [12], основано на идее передачи сообщений между вершинами информационной сети по кратчайшему пути, выбранному случайным образом. Вершины, через которые проходит большее количество путей, имеют больший индекс центральности, они могут обладать значительным влиянием, так как в этих вершинах можно, например, контролировать проходящую ин-

формацию. Удаление таких узлов может разрушить коммуникации между другими узлами. Центральность по посредничеству отличается от других определений центральности, поскольку интерес представляет не то, как вершина взаимосвязана с другими вершинами, а то, как часто вершина встречается на пути между другими.

Предметом настоящей работы является вычисление значения c_B для вершин социальной сети Вконтакте, образованной распределенной базой данных, с целью их ранжирования по этой мере. Под термином ранжирование будем понимать установление такого отношения между любыми двумя вершинами, при котором можно утверждать, что "одна вершина имеет ранг выше другой"или "ниже другой"или "их ранги равны"; при этом само множество "рангов"является упорядоченным. Отметим, что с помощью процедуры ранжирования нельзя однозначно выполнить упорядочивание вершин, поскольку сеть может иметь вершины с одинаковым рангом.

Ранее была доказана следующая лемма:

Лемма 1 (критерий Беллмана).

Вершина $v \in V$ лежит на кратчайшем пути между вершинами $s, t \in V$, тогда и только тогда, когда $d_G(s,t) = d_G(s,v) + d_G(v,t)$, где $d_G(s,t)$-расстояние между вершинами s и t, т. е. минимальная длина любого пути, соединяющего s и t.

При заданных попарных расстояниях и кратчайших путях парная зависимость пары $s, t \in V$ на промежуточном $v \in V$ задается как $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$ и кратчайший путь между s и t, на которых лежит v, задается формулой:

$$\sigma_{st}(v) = \begin{cases} 0 & \text{если } d_G(s,t) < d_G(s,v) + d_G(v,t) \\ \sigma_{st}\sigma_{vt} & \text{в противном случае} \end{cases}$$

Итак, чтобы получить индекс центральной вершины v, нам просто нужно суммировать парные зависимости всех пар в этой вершине,

$C_B(v) = \sum_{s \neq v \neq t \in (V)} \delta_{st}(v)$

Таким образом, центральность определяется в два этапа:

1. вычислить длину и количество кратчайших путей между всеми парами

2. суммирование все парных зависимостей.

Итак, centrality – это отношение количества кратчайших путей, проходящих через вершину, к общему числу кратчайших путей в графе.

Центральностью вершины v графа G = (V, E) называется величина:

$C_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, где σ_{st} – число различных кратчайших путей в графе G от вершины s к вершине t, а $\sigma_{st}(v)$– число таких путей, проходящих через вершину v.

Значения индекса C_B иногда удобно нормализовать. Одним из естественных способов нормализации является деление значения центральности на количество пар вершин $|V|^2$, т. е. определять "центральность по посредничеству"следующим образом:

$C_B(v) = \frac{1}{|V|^2} \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$

Тогда значения C_B будут находиться в интервале [0, 1].

Зависимость (dependency) вершины $s \in V$ от единичной вершины $v \in V$ определяется как $\delta_s(v) = \sum_{t \in V} \delta_{st}(v)$

Тогда «центральность по соседничеству» определяется как $C_B(v) = \sum_{s \in V} \delta_s(v)$

Обозначим через $P_s(v)$ множество предшественников (predecessors) вершины $v \in V$ на кратчайшем пути из $s \in V$:

$P_s(v) = u \in V : (u, v) \in , d_G(s, v) = d_G(s, u) + w(u, v)$.

Связь количества кратчайших путей от s до v ($s \neq v \in V$) с количеством путей до предшественников:

$\sigma_{sv} = \sum_{u \in P_{s(v)}} \sigma_{su}$

В работе [10] показано, что для вычисления значения зависимости вершины ($s \in V$) от любой вершины ($v \in V$) можно применять следующую ключевую формулу:

$\delta_s(v) = \sum_{w : v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_s(w))$

Выявление кратчайших путей от начальной вершины s до всех других производится с использованием алгоритмов обхода графа в порядке неубывания расстояния от s: BFS для невзвешенных графов и Дейкстры для взвешенных. В конце каждой итерации, соответствующей вершине $s \in V$, взятой в качестве начальной, рассматриваются все пройденные вершины v в порядке невозрастания расстояния от s, пересчитываются зависимости от предшественников, а зависимость s от v, $\delta_s(v)$ добавляется к значению

индекса $C_B(V)$. Таким образом, для любой вершины $s \in V$ выполняются два шага:

1) производится подсчет длин и количества кратчайших путей от $s \in V$ до всех остальных вершин;

2) вычисляются зависимости s от всех вершин на путях с использованием ранее выведенных формул и суммируются со значениями центральности согласно формуле: $C_B(v) = \sum_{s \in V} \delta_s(v)$

В результате будут рассмотрены все пары вершин и просуммированы парные зависимости. В случае неориентированных графов значение нужно делить на два, так как все кратчайшие пути рассматриваются дважды.

Разберем подсчет числа кратчайших путей.

8.1.1. Базисные алгоритмы нахождения кратчайших путей

Для определения центральности вершины необходимо найти все кратчайшие пути между всеми связанными парами вершин. Предполагаем, что веса ребер неотрицательны.

1. Одним из базисных алгоритмов, составляющим основу многих других, является "алгоритм поиска в ширину"(англ. breadth-first search, BFS). Он применяется к невзвешенным ориентированным и неориентированным графам и относится к числу алгоритмов обхода графа (англ. traversal algorithms). Пусть имеется граф $G = (V, E)$ и зафиксирована начальная вершина s. Алгоритм перечисляет все достижимые из s вершины в порядке возрастания расстояния от s. При этом сначала рассматриваются все соседи, потом соседи соседей и т. д. В процессе поиска из графа выделяется часть, называемая "деревом поиска в ширину"с корнем в s. Для каждой вершины путь из корня в дереве поиска будет одним из кратчайших путей (из начальной вершины) в графе. Поскольку граф невзвешенный, то расстояние от s совпадает с числом ребер кратчайшего пути. Время работы алгоритма оценивается как o(|V| + |E|), т. е. пропорционально размеру представления графа в виде списков смежных вершин.

Суть алгоритма поиска в ширину в том, что мы обходим связный граф таким образом, что сначала мы рассматриваем родителя, потом по очереди рассматриваем его предков, потом рассматриваем предков его предков и

т.д. (см. рис. 27)

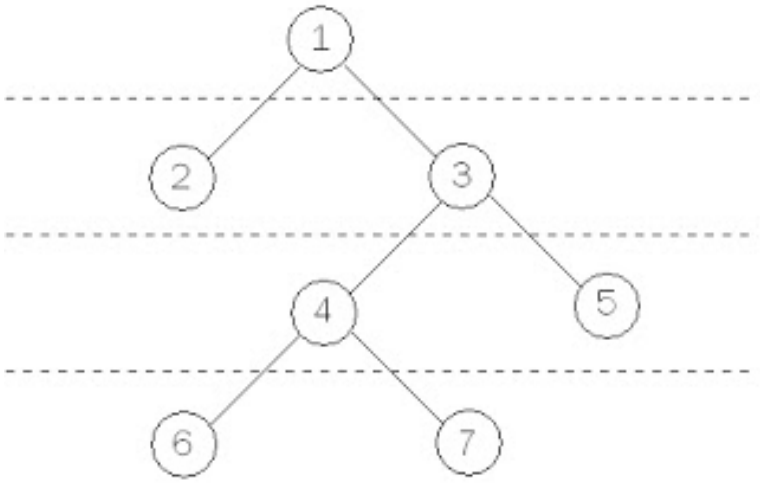


рис. 27

2. Алгоритм Дейкстры решает задачу о кратчайших путях из одной вершины для взвешенного ориентированного графа G = (V,E), в котором веса всех ребер неотрицательны, т. е. $w(u, v) > 0$ для всех ребер $(u, v) \in E$. Алгоритм начинает работу с некоторой вершины $s \in V$ и на каждом шаге добавляет ближайшую вершину к множеству уже обнаруженных вершин, лежащих на кратчайших путях, таким образом добиваясь выявления всех кратчайших путей из источника до всех остальных. Алгоритм основан на ряде свойств кратчайших путей; первое: отрезки кратчайших путей сами являются кратчайшими путями; второе: для всякой вершины $s \in V$ и для всякого ребра $(u, v) \in E$ выполняется неравенство $d_G(s, v) \leq d_G(s, u)+w(u, v)$. При работе алгоритма используется прием, называемый "релаксацией". На начальный момент для всех $v \in V$ верхняя оценка d[v] для расстояния от исходной вершины s устанавливается равной ∞.

Распишу этапы алгоритма.

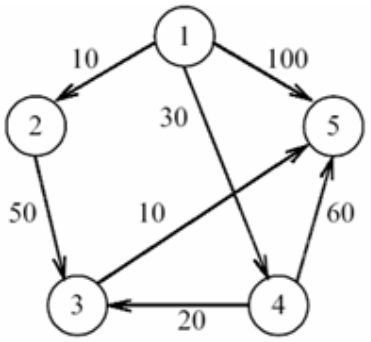
1 шаг. Метка вершины-источника a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые. Первая вершина объявляется текущей.

2 шаг. Вес всех невыделенных вершин пересчитывается по формуле: вес невыделенной вершины есть минимальное число из старого веса данной вершины и суммы веса текущей вершины и веса ребра, соединяющего текущую вершину с невыделенной.

3 шаг. Среди невыделенных вершин ищется вершина с минимальным весом. Если таковая не найдена, то есть вес всех вершин равен бесконечности, то маршрут не существует. Следовательно, выход. Иначе, текущей становится найденная вершина. Она же выделяется.

4 шаг. Среди невыделенных вершин ищется вершина с минимальным весом. Если таковая не найдена, то есть вес всех вершин равен бесконечности, то маршрут не существует. Следовательно, выход. Иначе, текущей становится найденная вершина. Она же выделяется.

Если все вершины посещены, алгоритм завершается. Реализация алгоритма представлена на рис. 28. Вводим массив D, где записываются длины кратчайших путей, множество S - множество пройденных вершин графа, массив P[v] будет содержать вершину, предшествующую вершине v в кратчайшем пути.



Итерация	S	w	**D**[2]	**D**[3]	**D**[4]	**D**[5]
начало	{1}	–	10	∞	30	100
1	{1, 2}	2	10	60	30	100
2	{1, 2, 4}	4	10	50	30	90
3	{1, 2, 4, 3}	3	10	50	30	60
4	{1, 2, 4, 3, 5}	5	10	50	30	60

Массив **P**: | | 1 | 4 | 1 | 3 |

Кратчайший путь из 1 в 5: {1, 4, 3, 5}

рис. 28

9. Постановка задачи

Итак, в моей работе на сайте https://graph-vis.herokuapp.com/ строится граф визуализции друзей какого-то определенного пользователя или список пользователей определенной группы.

С помощью алгоритмов, описанных выше, необходимо разбить пользователей на сообщества и вывести меры центральности, PR для каждого пользователя. Таким образом, мы сможем определить наиболее важных и значимых пользователей.

В работе будет представлена визуализация связей людей социальной сети Вконтакте и для каждого узла (пользователя) будет просчитаны показатели: центральность (C_B) по алгоритму, представленному в работе [10], PageRank.

10. Алгоритм визуализации графа Force Atlas 2

10.1. Описание алгоритма Force Atlas 2

Для построения визуализации мы используем алгоритм Force Atlas2.

Для визуализации социального графа наиболее популярными в настоящее время являются алгоритмы, основанные на аналогиях физических принципов притяжения и отталкивания тел или частиц по закону Гука, Кулона и др. с целью минимизации энергии системы (англ. Force-directed graph drawing). В частности, одним из широко используемых алгоритмов является Force Atlas 2.

В этой работе я буду использовать метод «направленных сил», (англ. «Force-directed method/approach»). В его основе лежит следующий принцип. Во-первых, задается случайное начальное состояние физической системы, состоящей из пружин (ребер) и металлических колец (вершин). Деформированные пружины приводят систему в движение. Происходят колебания, приводящие к изменению деформации пружин и положения колец-вершин. Когда система достигнет минимального энергетического состояния, работа алгоритма будет завершена.

ForceAtlas 2 [14] реализован в программе Gephi. Gephi – это открытый

программный инструмент для визуализации сетей. Алгоритм ForceAtlas 2 основан на минимизации энергии (вершины итерационно притягиваются или отталкиваются друг от друга в пространстве визуализации в зависимости от их взаимного расположения и наличия связей), которая приписывается графу в целом, его вершинам и ребрам (так называемые force-directed алгоритмы укладки). Такого рода алгоритмы хорошо подходят для построения изображений, подчеркивающих структуру группы, а также для визуализации ее подмножеств с высокой степенью взаимодействия. Для нас важна принципиальная картина взаимодействия, которая определяется только отношениями дружбы (двусторонняя связь) и подписки (односторонняя связь). При проектировании «Force Atlas» был сделан акцент на качестве визуализации, что делает раскладку графа, получающуюся на выходе, максимально наглядной. Пока алгоритм работает, узлы отталкиваются и края притягиваются.

ForceAtlas 2 представляет собой силовую направленную компоновку: имитирует физическую систему для сети. Узлы отталкиваются друг от друга подобно магнитам, а ребра притягивают свои узлы, подобно пружинам (см. рис. 29). Эти силы создают движение, которое сходится к сбалансированному состоянию. Эта окончательная конфигурация, как ожидается, поможет анализировать данные. Эта стратегия имеет свои недостатки. Во -первых,этот процесс не является детерминированным, и координаты каждой точки не отражают каких-либо конкретных значений. Во - вторых, результат не может быть прочитан как декартова проекция. Несмотря на эти замечания, стратегия имеет преимущество, позволяющее визуально ин-

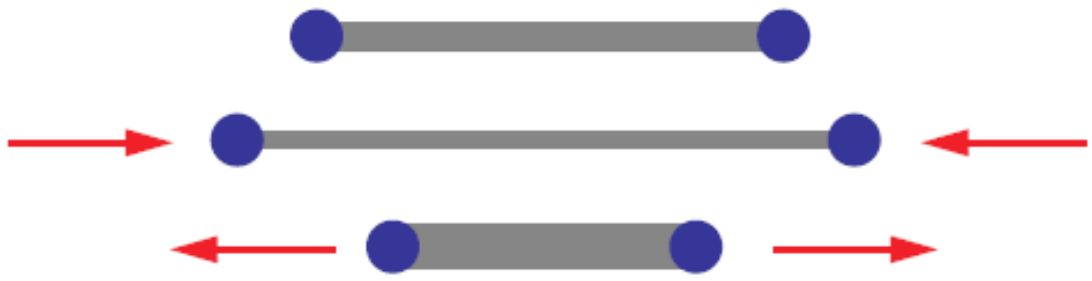


рис. 29

терпретировать структуру. На рис. 29 - 30 показано сжатие/ растягивание пружин и демонстрация работы метода «направленных сил».

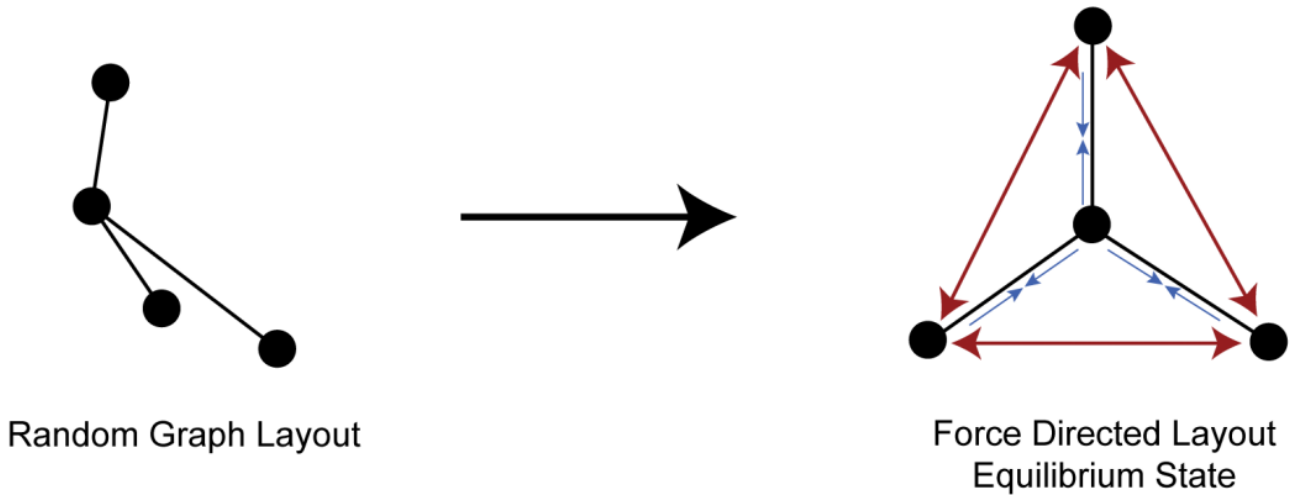


рис. 30

Силовые размещения предполагают создание красивого графа с ребрами одинаковой длины и минимально возможным количеством пересечений. Они представляют граф в виде физической системы. Вершины — электрически заряженные частицы, отталкивающиеся друг от друга при чрезмерном приближении. Ребра ведут себя, как пружины, притягивающие соединенные вершины поближе друг к другу. В результате вершины равномерно распределяются по области графика, а размещение интуитивно в том смысле, что вершины с большим количеством связей находятся ближе друг к другу.

Сложность алгоритма равна $O(N^2)$, поэтому есть возможность обрабатывать графы с числом узлов от 1 до 10000. Поэтому для визацализации связей был выбран данный алгоритм.

10.2. Модель алгоритма Force Atlas 2

Каждый силовой алгоритм опирается на определенную формулу для силы притяжения и силы отталкивания.

«Пружинно-электрическая» компоновка использует формулу отталкивания электрически заряженных частиц ($F_r = \frac{k}{d^2}$) и формулу притяжения пружин ($F_a = -kd$) с геометрическим расстоянием d между двумя узлами.

Фрухтерман и Рейнгольд после создали эффективный алгоритм с использованием пользовательских сил (притяжение: $F_a = \frac{d^2}{k}$, отталкивание:$F_r = \frac{k^2}{d}$ с коэффициентом k, регулирующим масштабирование сети). Шестнадцать лет спустя Ноак объясняет, что самое важное различие между силовыми алгоритмами - это роль, которую играет расстояние в пространстве. В физических системах силы зависят от расстояния между взаимодействующими объектами: более удалённые объекты притягиваются слабее, в то время как более близкие — сильнее. Зависимость между расстоянием и силами может быть линейной, экспоненциальной или логарифмической. Ноак определяет модель энергии или «(Притяжение, отталкивание) - модель» как экспоненту, представленную расстоянием в используемых формулах Для вычисления притяжения и отталкивания (логарифм рассматривается как нулевая степень).

Модель пружинно-электрической компоновки.

1) Классическая сила притяжения: сила притяжения F_a между двумя связанными узлами n_1 и n_2. Она зависит линейно от расстояния $d(n1, n2)$: $F_a(n_1, n_2) = d(n_1, n_2)$

2) Отталкивание по степени: типичным вариантом использования ForceAtlas2 является социальная сеть. Общая особенностью этого типа сети является наличие множества узлов. Узлы, окружающие несколько высокосвязных узлов, являются одним из основные источники визуального загромождения. В алгоритме учитывается степень узлов в отталкивании, так что это визуальное загромождение уменьшается.

Идея состоит в том, чтобы приблизить плохо связанные узлы к очень связанным узлам. Алгоритм настроивает силу отталкивания так, чтобы слабосвязанные узлы и сильно связанные узлы отталкивались меньше. Как следствие, они окажутся ближе в сбалансированном состоянии. Сила отталкивания F_r пропорциональна произведению степеней плюс один (deg +1) двух узлов. Коэффициент k_r определяется настройками. Видно из формулы, что даже узлы со степенью, равной нулю, имеет некоторую силу отталкивания.

$F_r(n_1; n_2) = k_r(deg(n_1) + 1)(deg(n_2) + 1)d(n_1; n_2)$

ForceAtlas2 - практический вклад в сетевые науки. Он не основан на

новой концепции, в нем реализованы многие функции других известных алгоритмов. Однако, по своей конструкции и функциям, он направлен на обеспечение общего и интуитивного способа пространственной организации сетей.

10.3. Модифицированная модель алгоритма Force Atlas 2

Кроме сил притяжения и отталкивания еще введем другую силу, действующую на вершины F_{level}. Данная сила будет притягивать вершину к центру. Таким образом, если вершина изолированная (не взаимодетвует ни с одним пользователем), то на нее будет действовать только сила F_{level}, которая будет ее притягивать к центру. Поэтому все изолированные вершины в конечном состоянии размещаются по окружности.

Также хочу отметить недостаток данного класса алгоритмов. На данный момент пока недостататочно разработаны критерии сходимости таких алгоритмов. Данные алгортмы останавливаются при достижении определенного числа итераций.

10.4. Параметры алгоритма Force Atlas 2

Напишу о параметрах, которые можно изменять в реализованном приложении. В зависимости от них будет видоизменяться граф связей пользователей.

- gravity: увеличивает силы притяжения. Увеличив данный параметр вершины будут находиться ближе друг у другу.
- linLogMode: делает сообщества (кластеры) более плотными
- strongGravityMode: притягивает узлы к центру. Предотвращает отдаление узлов на переферии

11. Математическое приложение

11.1. Вычисление коэффициента центральности (C_B)

Опишу алгоритм вычисления индекса (C_B) для вершин ориентированного взвешенного графа, то есть производим подсчет кратчайших путей, проходящих через вершины графа. Центральность выше у тех пользователй, которые напрямую связаны с большим количеством других.

$C_B[v] \leftarrow 0,\ v \in V;$
for $s \in V$ **do**
  $S \leftarrow$ empty stack;
  $P[w] \leftarrow$ empty list, $w \in V;$
  $\sigma[t] \leftarrow 0,\ t \in V; \quad \sigma[s] \leftarrow 1;$
  $d[t] \leftarrow -1,\ t \in V; \quad d[s] \leftarrow 0;$
  $Q \leftarrow$ empty queue;
  enqueue $s \rightarrow Q;$
  while *Q not empty* **do**
    dequeue $v \leftarrow Q;$
    push $v \rightarrow S;$
    foreach *neighbor w of v* **do**
      // w found for the first time?
      if $d[w] < 0$ **then**
        enqueue $w \rightarrow Q;$
        $d[w] \leftarrow d[v] + 1;$
      end
      // shortest path to w via v?
      if $d[w] = d[v] + 1$ **then**
        $\sigma[w] \leftarrow \sigma[w] + \sigma[v];$
        append $v \rightarrow P[w];$
      end
    end
  end
  $\delta[v] \leftarrow 0,\ v \in V;$
  // S returns vertices in order of non-increasing distance from s
  while *S not empty* **do**
    pop $w \leftarrow S;$
    for $v \in P[w]$ **do** $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w]);$
    if $w \neq s$ **then** $C_B[w] \leftarrow C_B[w] + \delta[w];$
  end
end

Здесь d[t] — верхняя оценка длины кратчайшего пути от начальной вершины до вершины t, Q - очередь вершин с приоритетами, определяемыми значениями функции d; P[w] — список предшественников w на кратчайших путях от начальной вершины до w, $\sigma(w)$ — оценка количества кратчайших путей от начальной вершины до w; $\sigma(v)$ — оценка зависимости начальной вершины от v; Q_1 — очередь, в которой вершины хранятся в порядке неубывания расстояния от начальной вершины; S — стек, в котором вершины хранятся в порядке невозрастания расстояния от начальной вершины.

При реализации вместо использования стека S (приведенного для совместимости с изложением алгоритма Брандеса в части расчета центральности) Q формируется как массив, который в первом цикле (while Q_1) просматривается с начала, во втором случае (while S) — просматривается с конца.

11.2. Вычисление коэффициента PageRank

Опишу реализацию алгоритма PageRank в JavaScript. Этот модуль является частью семейства ngraph.

Покажу пример вычисления PageRank для простого графика с двумя узлами и одним ребром.

.

```
var graph = require('ngraph.graph')();
graph.addLink(1, 2);

var pagerank = require('ngraph.pagerank');
var rank = pagerank(graph);
```

Этот код вычисляет PageRank для двух узлов:

```
{
  "1": 0.350877,
  "2": 0.649123
}
```

В моей работе алгоритм PageRank позволяет указать вероятность на

любом шаге, что человек будет продолжать щелкать исходящие ссылки. Эта вероятность в некоторой литературе называется коэффициентом демпинга и рекомендуется устанавливать между 0.8 и 0.9.

Чтобы настроить эту вероятность, использую второй необязательный аргумент функции pagerank ():

```
// by default this value is 0.85. Bump it to 0.9:
var internalJumpProbability = 0.90;
var rank = pagerank(graph, internalJumpProbability);
```

Текущая реализация использует приближенное решение проблемы собственных векторов. Чтобы указать уровень точности, используется последний необязательный аргумент:

```
var internalJumpProbability = 0.85;
// by default it's set to 0.005, let's increase it:
var precision = 0.00001;
var rank = pagerank(graph, internalJumpProbability, precision);
```

precision повлияет на производительность алгоритма и служит критерием выхода:

```
|r(t) - r(t - 1)| < precision
```

Здесь r (t) - собственный вектор (или pagerank графика) на шаге t времени.

Часть кода для подсчета PR. Идея реализация заключается в том, что считаем PR как сумму всех вкладов от соседних узлов.

```

module.exports = pagerank;

function pagerank(graph, internalJumpProbability, epsilon) {
  if (typeof epsilon !== 'number') epsilon = 0.005;
  if (typeof internalJumpProbability !== 'number') internalJumpProbability

  = 0.85;
  var done = false; //
```

```
10    var distance = 0; //
11    var leakedRank = 0; //
12
13    var nodesCount = graph.getNodesCount();
14    var nodes = initializeNodes(graph);
15    var currentRank;
16    var node;
17
18    do {
19      leakedRank = 0;
20      for (var j = 0; j < nodesCount; ++j) {
21        node = nodes[j];
22        currentRank = 0;
23        if (node.indegree === 0) {
24          node.rank = 0;
25        } else {
26          var neighbors = node.neighbors;
27          for (var i = 0; i < neighbors.length; ++i) {
28            currentRank += neighbors[i].prevRank / neighbors[i].outdegree;
29          }
30          node.rank = internalJumpProbability * currentRank;
31          leakedRank += node.rank;
32        }
33      }
34      // now reinsert the leaked PageRank and compute distance:
35      leakedRank = (1 - leakedRank) / nodesCount;
36      distance = 0;
37      for (j = 0; j < nodesCount; ++j) {
38        node = nodes[j];
39        currentRank = node.rank + leakedRank;
40        distance += Math.abs(currentRank - node.prevRank);
41        node.prevRank = currentRank; // set up for the next iteration
42      }
43      done = distance < epsilon;
44    } while (!done);
45
46    return finalResults(nodes);
47  }
48
49  function finalResults(pageRankNodes) {
50    var result = Object.create(null);
51
52    for (var i = 0; i < pageRankNodes.length; ++i) {
53      var node = pageRankNodes[i];
54      result[node.id] = node.prevRank;
55    }
```

```
56
57    return result;
58  }
59
60  function initializeNodes(graph) {
61    var i = 0;
62    var nodesCount = graph.getNodesCount();
63    var initialRank = (1 / nodesCount);
64    var nodes = new Array(nodesCount);
65    // we want to use integers for faster iteration during computation. This
66    // means we have to map node identifiers to their integers values
67    var idToNumber = Object.create(null);
68
69    // unfortunately we have to do two-pass to initialize both nodes and edges
70    graph.forEachNode(addNode);
71    graph.forEachNode(initLinks);
72
73    return nodes;
74
75    function addNode(node) {
76      nodes[i] = new PageRankNode(initialRank, node.id);
77      idToNumber[node.id] = i;
78      i += 1;
79    }
80
81    function initLinks(node) {
82      var pageRankNode = nodes[idToNumber[node.id]];
83      var inDegree = 0;
84      var outDegree = 0;
85      var neighbors = pageRankNode.neighbors;
86
87      graph.forEachLinkedNode(node.id, initLink);
88
89      pageRankNode.indegree = inDegree;
90      pageRankNode.outdegree = outDegree;
91
92      function initLink(otherNode, link) {
93        if (link.fromId === node.id) {
94          outDegree += 1;
95        }
96        if (link.toId === node.id) {
97          inDegree += 1;
98          // TODO: this needs to be configurable. E.g. use outdegree
99          neighbors.push(nodes[idToNumber[otherNode.id]]);
100       }
101     }
```

```
102     }
103   }
104
105   function PageRankNode(initialRank, id) {
106     this.id = id;
107     this.rank = initialRank;
108     this.prevRank = initialRank;
109     this.outdegree = 0;
110     this.indegree = 0;
111     this.neighbors = [];
112   }
```

12. Вычислительный эксперимент

12.1. Получение данных из базы данных vk.com

Сначала поясню, как же получаются данные из социальной сети. Данные загружаются в приложение по API ВКонтакте. В документации сети Вконтакте написано: Open API — система, которая предоставляет возможность легко авторизовывать пользователей ВКонтакте на Вашем сайте. Кроме этого, с согласия пользователей, Вы сможете получить доступ к информации об их друзьях, фотографиях, видеороликах и других данных ВКонтакте для более глубокой интеграции с Вашим проектом.

В рамках подключения к Open API создается приложение, которое позволяет использовать на моем сайте все текущие методы ВКонтакте API. Помимо этого, Open API упрощает регистрацию новых пользователей на сайте, если у них уже есть учетная запись ВКонтакте.

API (англ. application programming interface) — это посредник между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. API упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными.

API ВКонтакте — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Мне не нужно знать в подробностях, как устроена база, из каких таблиц и полей каких типов она состоит — достаточно того, что API-запрос об этом «знает». Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса.

Сначала происхоит инициализация пользователя, затем авторизация по логину и паролю и далее данные загружаются по API. Информация по API взята из документации социальной сети Вконтакте. Адрес источника: https://vk.com/dev/openapi).

12.2. Сбор и обработка и данных

Поэтапно опишу работу реализованного сервиса.

1.Вводим id пользователя, который нас интересует, или id группы. Далее авторизовываемся по логину и паролю (см. рис. 31 - 32).

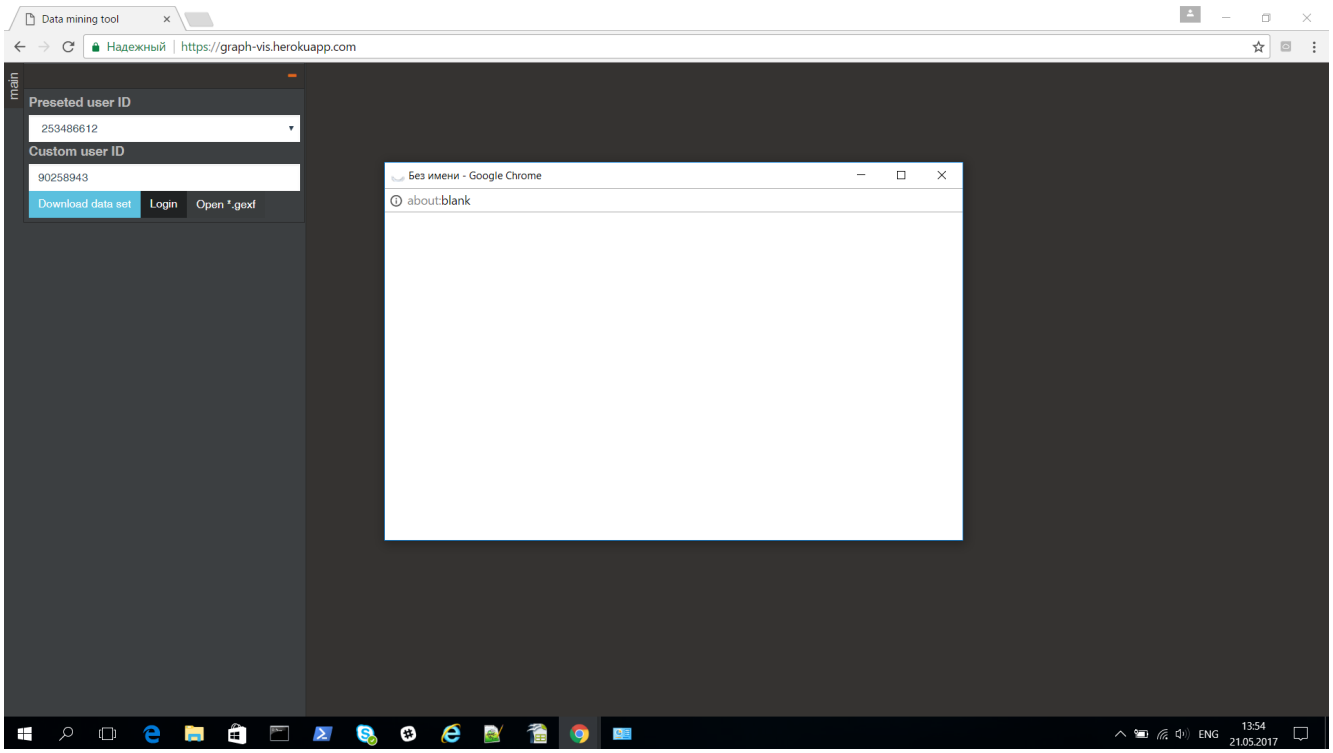


рис. 31

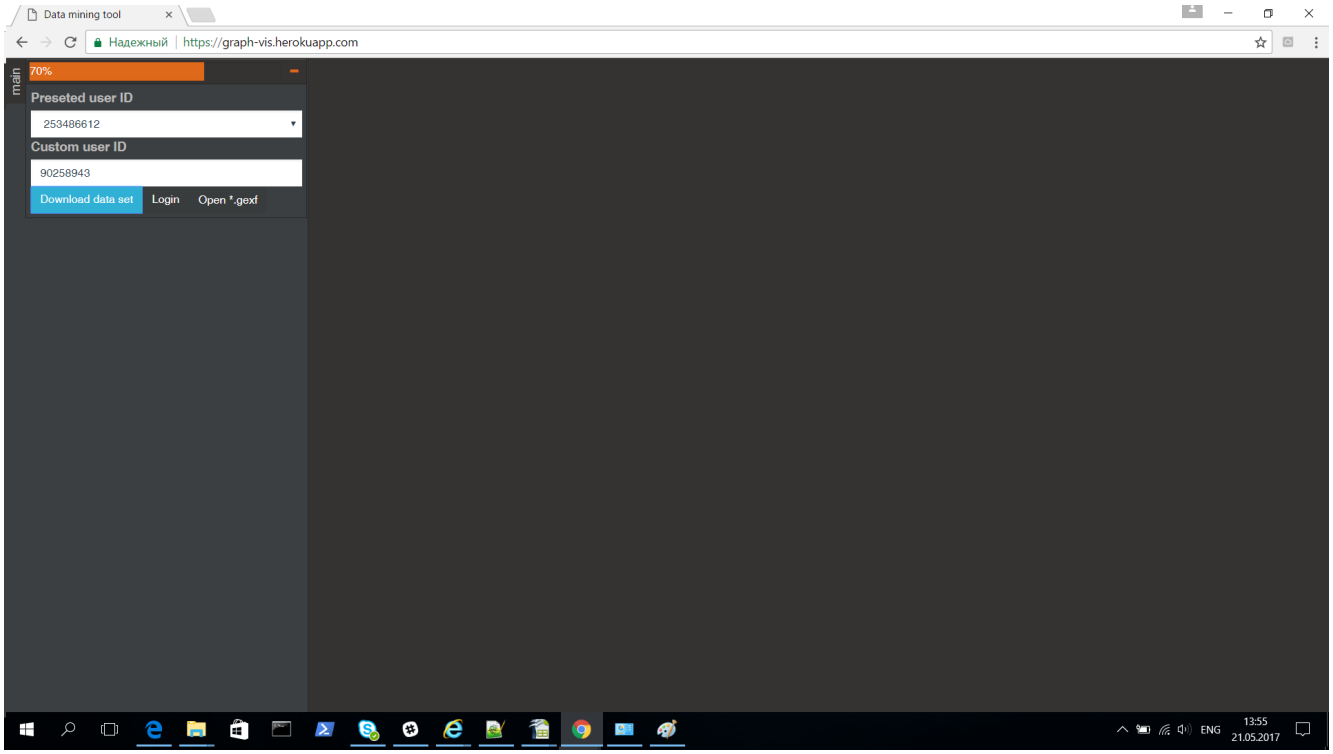


рис. 32

2. Нажимаем на кнопку "Start Layout". После этого сервис начнет загружать данные о друзьях (подписчиках) пользователя и строить граф связей (см. рис. 33).

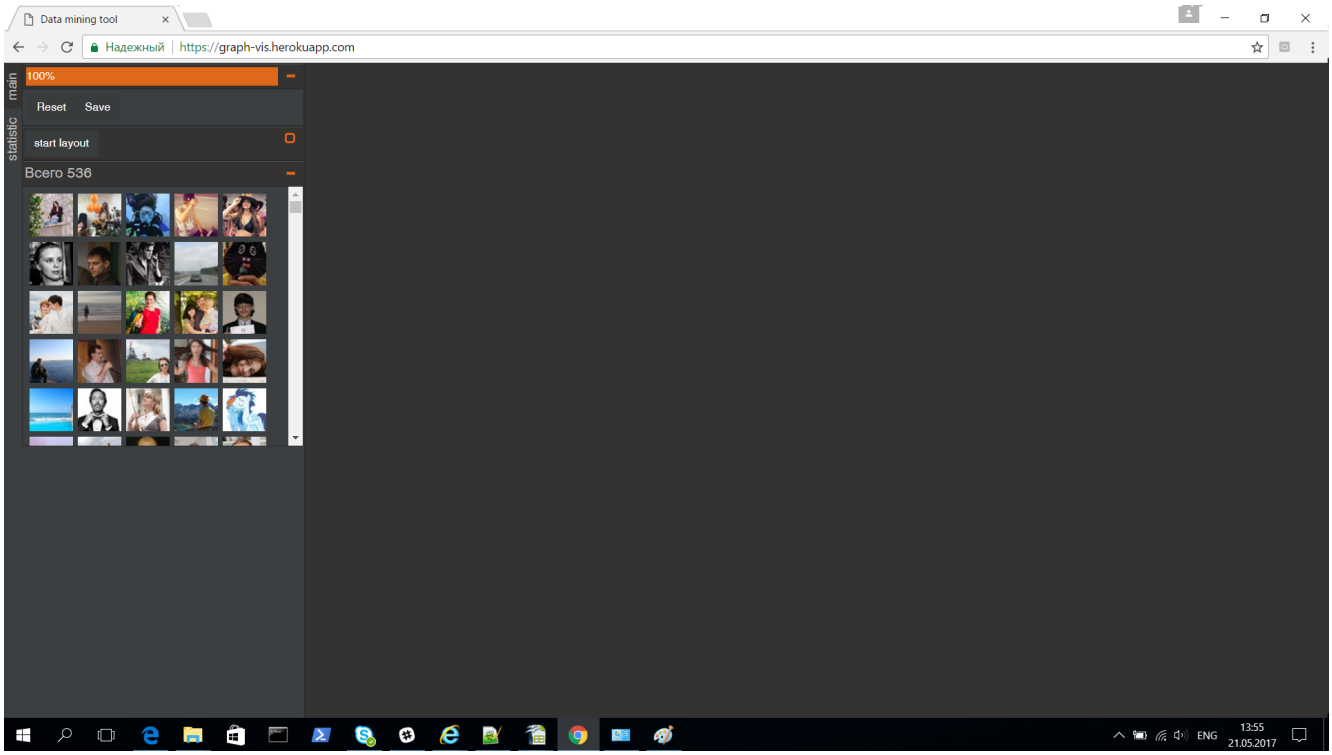


рис. 33

3. Выбираем modularity, чтобы был применен алгоритм разбиения сообществ на основе оптимизации показателя модулярности. Пользователи в режиме реального времени перемещаются в свои группы, причем каждая группа имеет свой цветовой идентификатор. (см. рис. 34)

4. Далее можно выбрать показатель PageRank, Centrality, чтобы данные метрики были рассчитаны для кадого пользователя. Пользователи с наибольшим показателем PageRank будут более жирными и яркими точками на экране. (см. рис 35)

5. В приложении можно детально изучить все получившиеся группы. А именно. сколько пользователей в каждой группе и какие пользователи в ней присутствуют. (см. рис. 36)

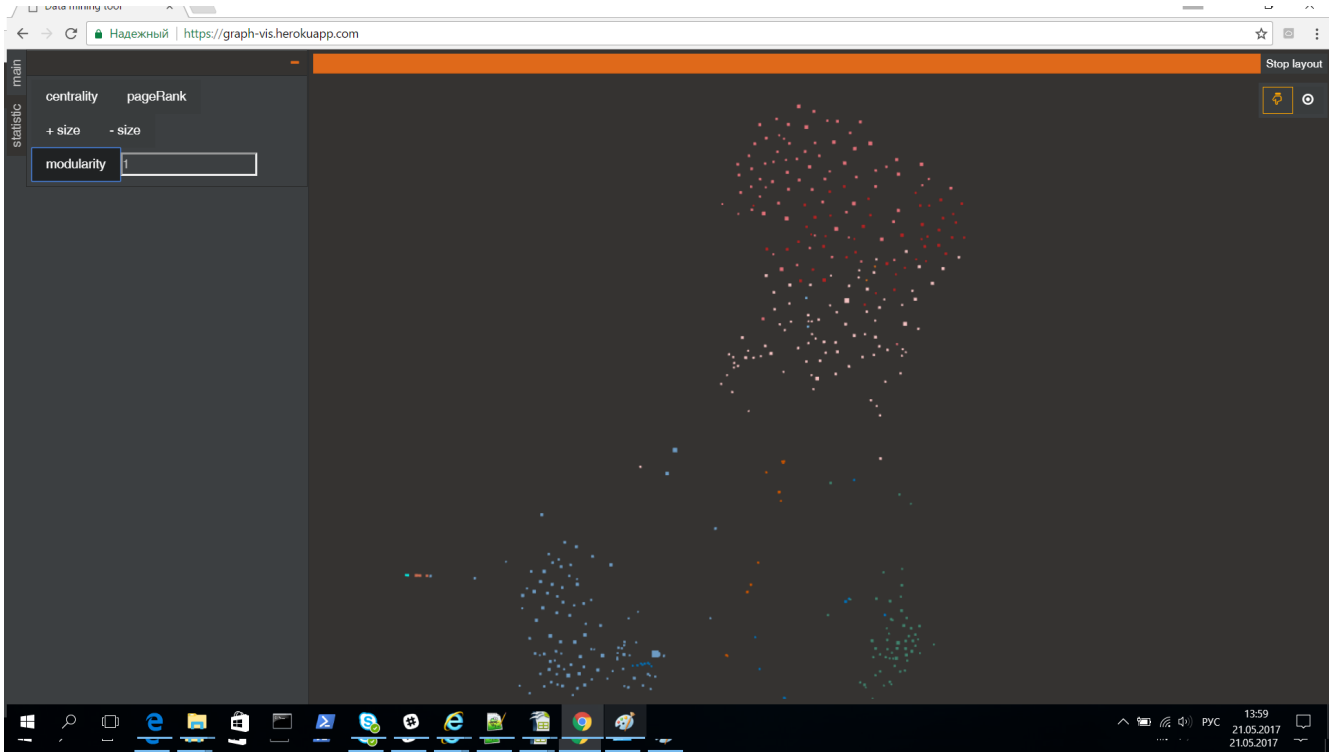


рис. 34

рис. 35

Сделав анализ разбиения подписчиков двух страниц, продемонстрирую получившиеся данные (см. табл. 1 - 2). Пользователи образовали сообще-

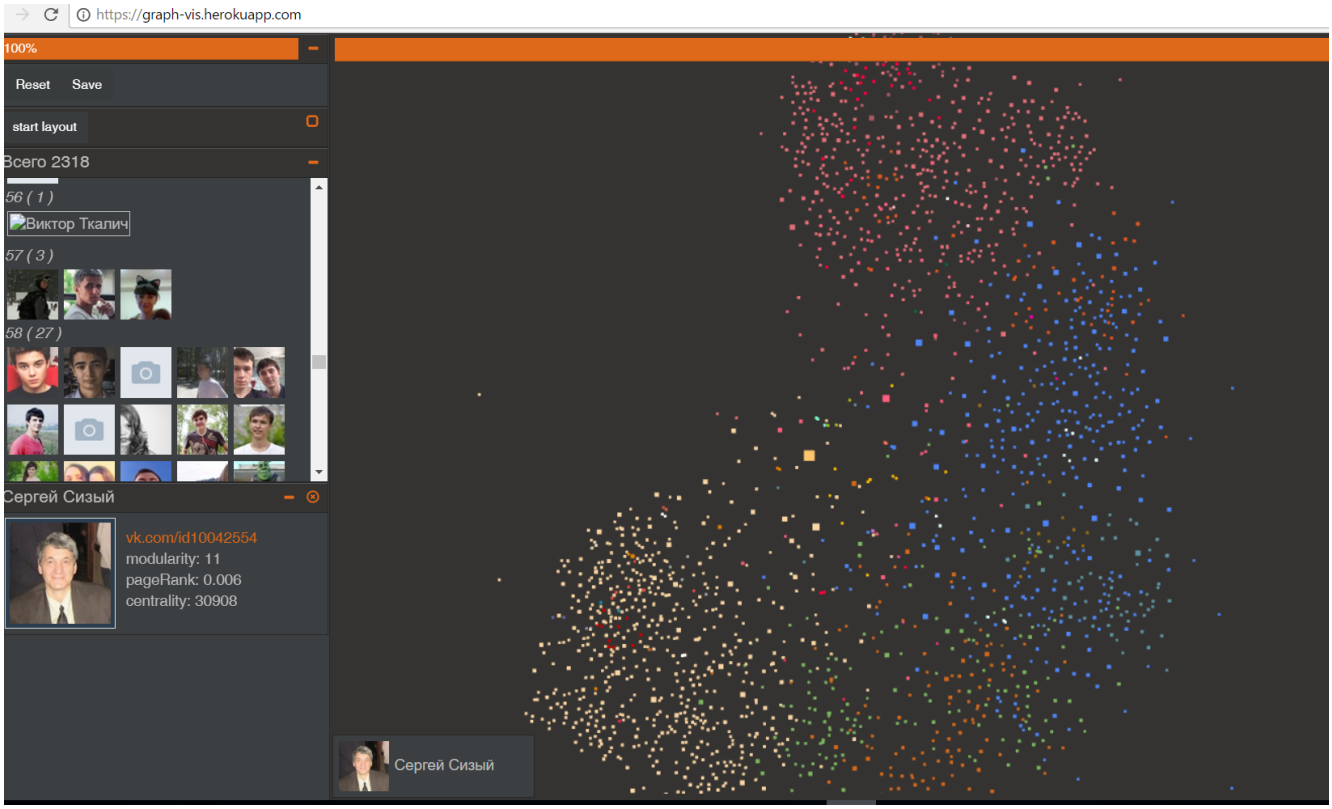


рис. 36

ства путем применения алгоритма оптимизации модулярности. Был проведен анализ и выявлены характерные особенности некоторых групп пользователей.

Номер группы разбиения	**Число пользователей в группе**	**Критерий разбиения**
1	78	Образование: Вуз: Университет МФТИ (ГУ), Факультет: Радиотехники и Кибирнетик
2	87	Город: Москва
3	4	Состоят в группе: Изучение чешского языка
4	2	Одинаковая фамилия и связаны друг с другом через соц. статус
5	11	Работают в танцевальных студиях
6	150	Город: Екатеринбург
7	57	Родной город: Белорецк
8	2	Одинаковая фамилия, Брат и сестра

табл. 1

Номер группы разбиения	Число пользователей в группе	Критерий разбиения
1	151	Место работы: СКБ Контур
2	7	Образование Вуз: УрФУ '11
3	2	Вуз: УрФУ '11 Факультет: Математико-механический Кафедра: Математической физики
4	122	Вуз: УрФУ
5	45	Вуз: УрГПУ
6	37	Город: Белорецк
7	45	Состоят в группе «ЕкаМама»
8	117	Город: Магнитогорск

табл. 2

Проведу анализ зависимости показателей Pagerank, modularity, centrality для подписчиков моей страницы. Причем показатель modularity отвечает за номер группы, в которой находится пользователь. Пользователи взяты случайным образом из разных сообществ разбиения.

Номер пользователя	PageRank	Modularity	Centrality	Число общих пользователей
1	0,006	1	6753	108
2	0,01	9	22349	79
3	0,006	9	19112	81
4	0,006	9	6035	56
5	0,004	9	14073	53
6	0,004	1	4681	52
7	0,007	17	9078	42
8	0,002	13	5392	20
9	0,004	17	2863	19
10	0,002	13	4112	18
11	0,002	17	7076	17

табл. 3

Сделав анализ групп пользователей, разбитых по показателю modularity, прихожу к выводу, что самые крупные узлы с высоким показателем Pagerank и имеют большее число подписчиков, чем другие члены группы. Представлю график, в котором по одной оси будут значения показателя PageRank, а по другой - общее число подписчиков пользователя (см. рис. 37). По графику видно, что число подписчиков не влияет напрямую на показатель PR. Как говорилось ранее, главное, какую долю своего PR подписчики отдают пользователю.

В целом, показатель Pagerank напрямую не зависит от числа подпичсчиков. Если, конечно, предположить, что все пользователи имеют около 300 подписчиков, то каждый пользователь будет отдавать одиноково значение Pagerank, и Pagerank каждого пользователя будет напрямую зависеть от числа подписчиков. Но данная ситуация далека от реальности.

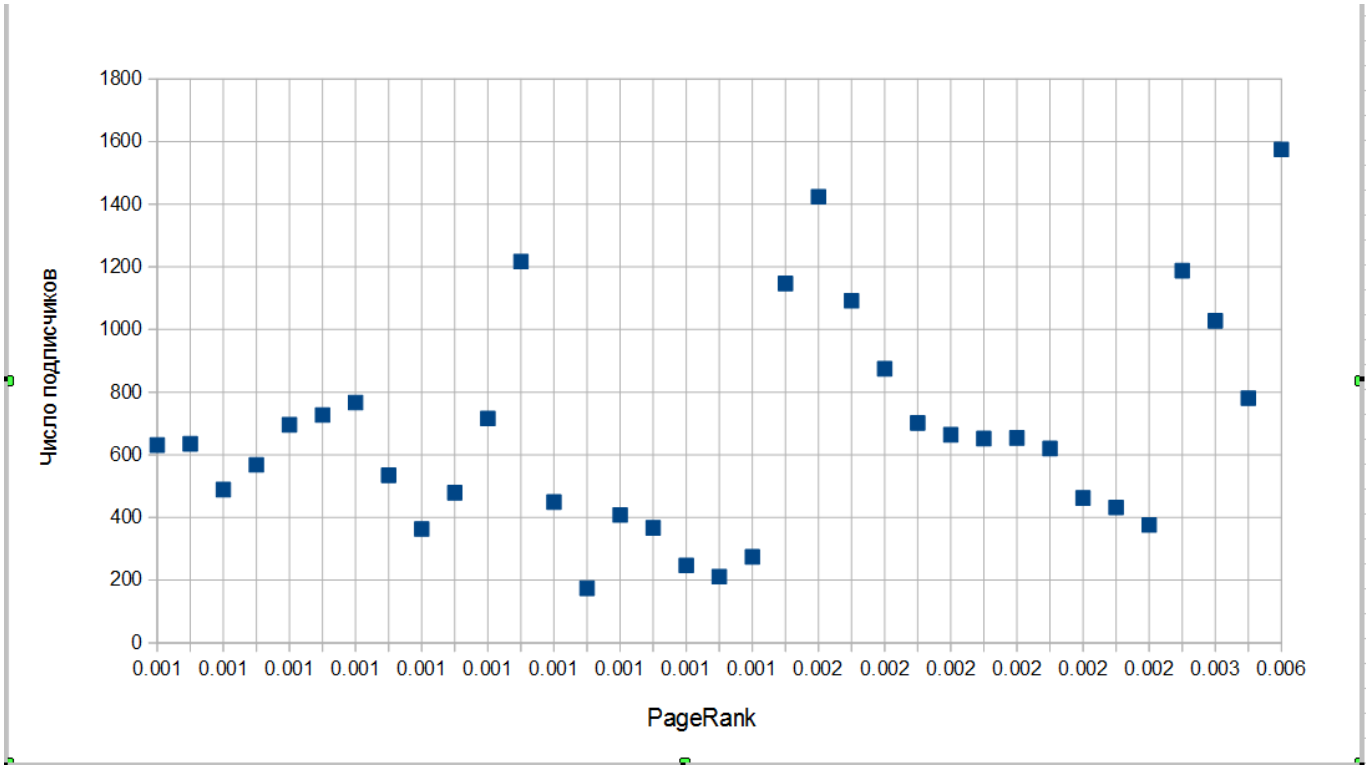


рис. 37

Увеличим масштаб всех узлов, чтобы наглядно показать узлы с самым высоким показателем PR. Эти вершины являютя наиболее важными, то есть связующими звеньями. (см. рис. 38)

Еще один важный параметр, который может охарактеризовать структу-

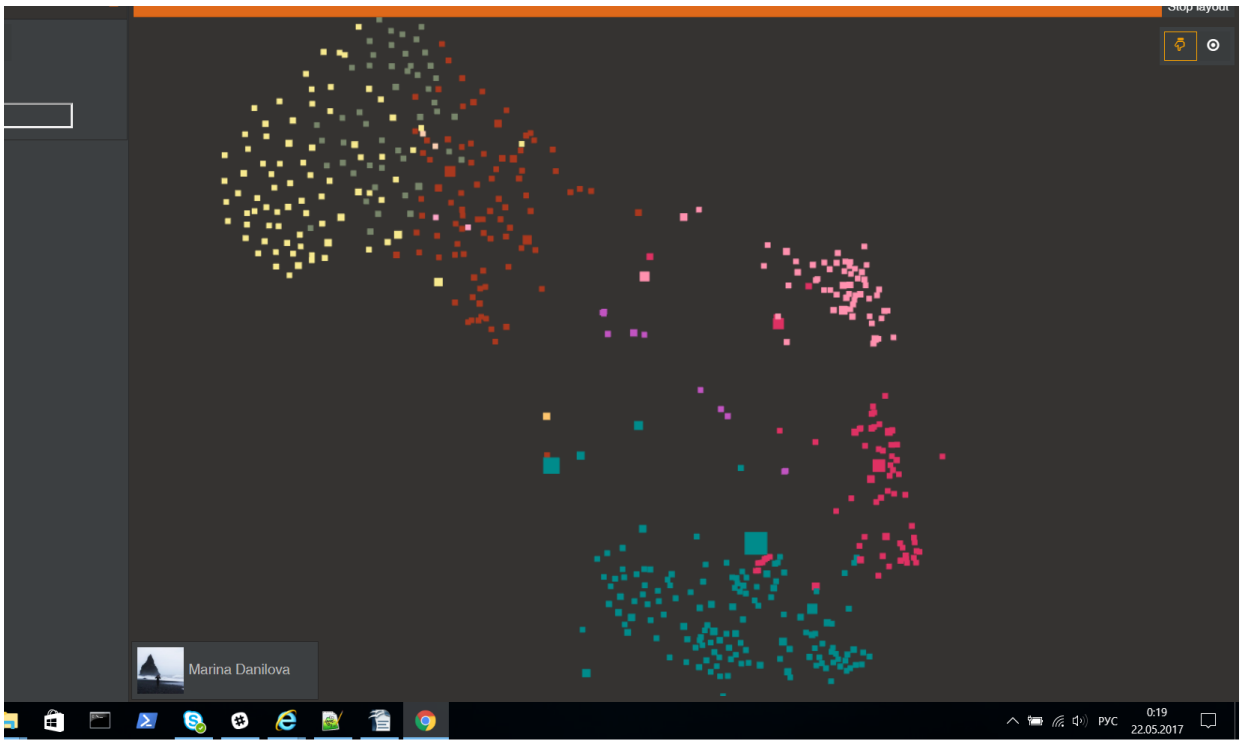


рис. 38

ру визуализации связей подписчиков, – число связных компонент. Это число говорит о численности контекстов, в которых осуществляется взаимодействие. При построении визуализации связей подписчиков первой страницы было выявлено около 16 сообществ, при анализе второй страницы-около 30 сообществ. Это говорит о фрагментированности структуры, большом числе изолированных частей сети, тех контекстов, в которых осуществляется взаимодействие. Сообщества, состоящие из одного пользователя, не учитываются. По количеству сообществ можно судить о сплоченности всех подписчиков.

Теперь на примере группы "Мат-Мех УрГУ (ДММиКН/ИМКН УрФУ)"сделаем анализ характиристик пользователей. В группе всего 2938 участников на момент анализа. Данная группа имеют структуру с относительно небольшим центральным ядром, вокруг которого концентрируются остальные участники. Периферия имеет достаточно низкую плотность взаимодействия (количество пользователей относительно небольшое) и сильно фрагментирована.

У центрального участиника (Сергей Сизый) самый высокий показатель pagerank: 0,006. Это говорит о том, что данный пользователь является связующим звеном, то есть через него проходит большое количество связей, которые повышают его показатель pagerank. Другими словами, у этого пользователя или самое большое число входящих ребер, или данный пользователь имеет подписчиков, которые отдают ему большую долю своего показателя pagerank. Показатель центральности у данного пользователя также высокий: 30908. Это показывает число кратчайших путей, проходящих через данного пользователя, велико. (см. рис 39)

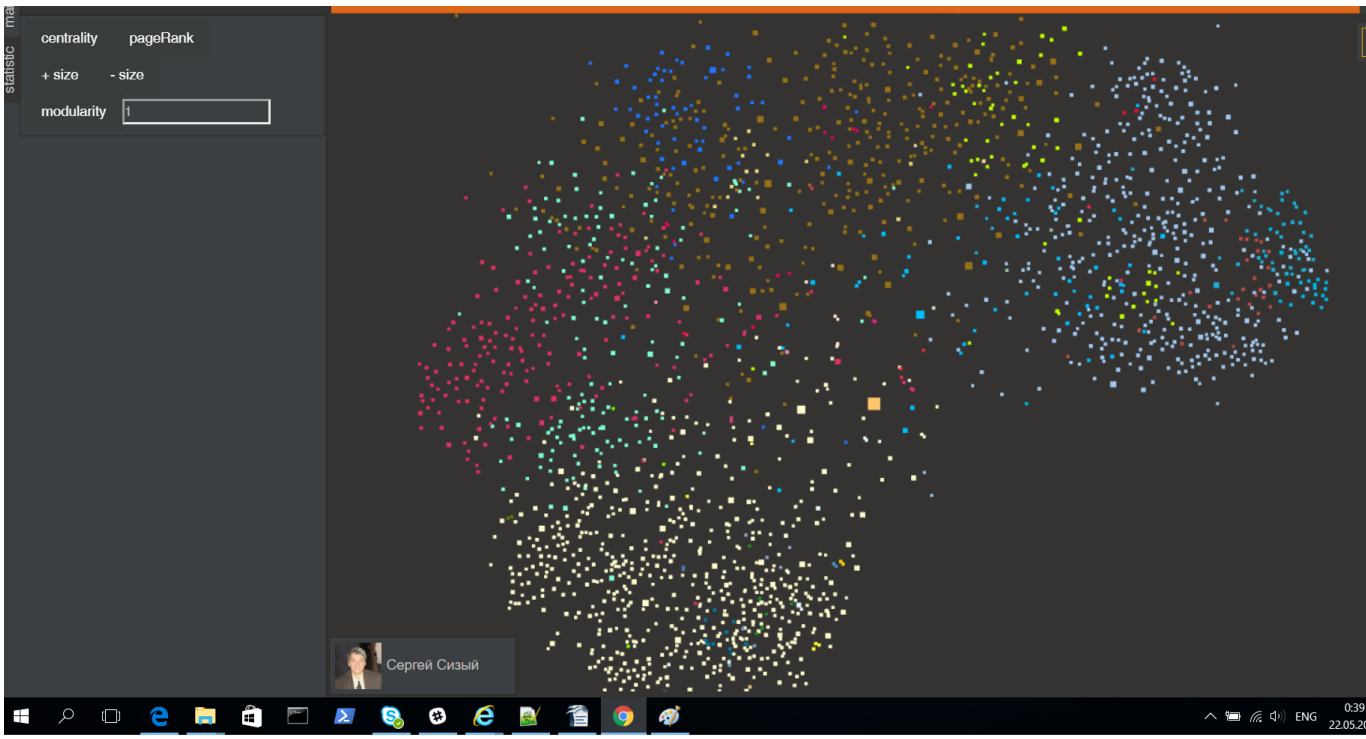


рис. 39

Также есть пользователь (Михаил Рубинчик) с показателями: pageRank

0.003, centrality 96580. Очевидно, что данный пользователь также пользуется авторитетом (см. рис. 40).

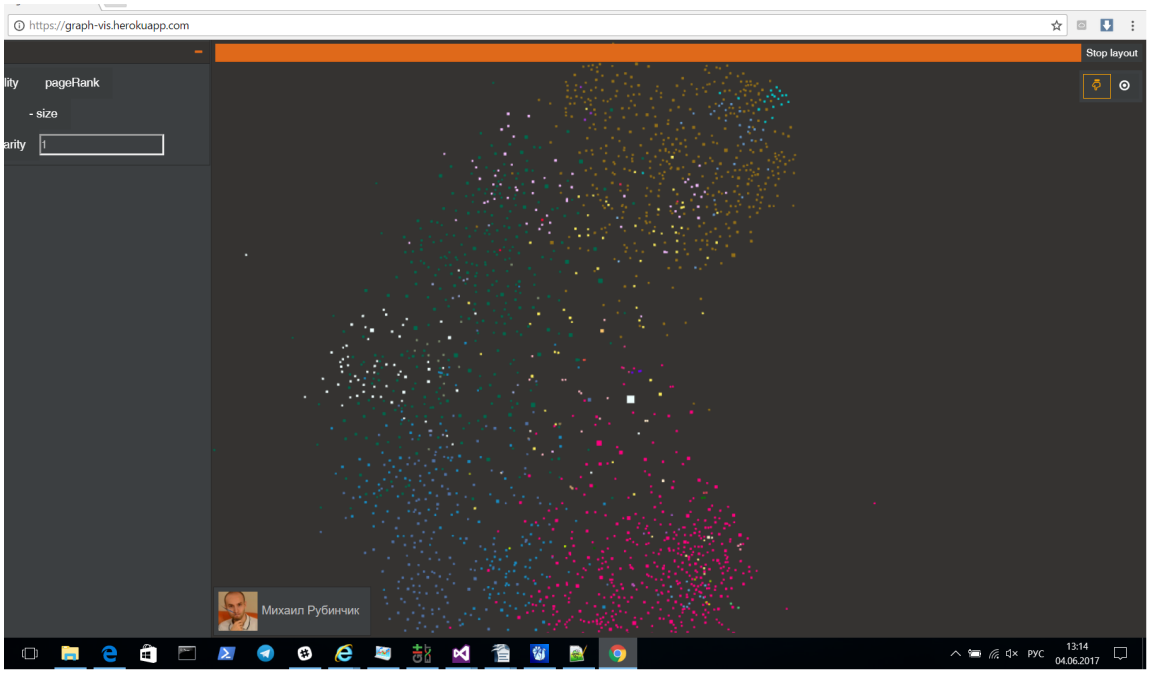


рис. 40

В итоге получилось не так много пользователей с высоким показателем pageRank. Представлю таблицу пользователей с самым высоким показателем pageRank (см. рис. 41).

Имя (ник) пользователя	Pagerank
Сергей Сизый	0.006
Дмитрий Корнев	0.003
Сашулька Жильцова	0.003
Михаил Рубинчик	0.003

рис. 41

В группе "Мат-Мех УрГУ (ДММиКН/ИМКН УрФУ)"если не брать сообщества, состоящих из одного пользователя, то получится около 80 сообществ.

Интерпретация полученных визуальных изображений позволяет делать выводы о распределении плотности взаимодействия, о том какие сообщества более сполоченные, по каким критериям можно разбить пользователей, какой пользователей наиболее популярный.

13. Заключение

В работе рассмотрена задача кластеризации в применении к выделению связанных структур (кластеров) в социальных сетях. Это может свидетельствовать о том, что исследуемые метрики и методы в меньшей мере приспособлены к извлечению неявных знаний о внутренних закономерностях и структуре данных с социальных сетей. Выявленные структурные характеристики достаточно сложно продемонстрировать с помощью формальных математических параметров. Визуализация же системы отношений (дружба, подписка) демонстрирует не только наличие плотно взаимодействующих (сплоченных) подгрупп (ядер в сети), но и связь между характеристиками реального и сетевого информационного пространства. Самый тесный контакт (близкое расположение узлов при визуализации) в виртуальном пространстве наблюдается между пользователями, которые зарегистрированы в одном географическом регионе.

Неочевидным, но визуально представленным выводом стала географическая обусловленность плотности взаимодействия пользователей в группах. Этот вывод, к сожалению, можно ярко продемонстрировать только при наличии цветного изображения социальной сети, где цвет вершины соответствует месту жительства пользователя. Проведен вычислительный эксперимент на реальных данных.

Образовательная организация является достаточно специфическим объектом, в структуре которого происходят периодические изменения с течением времени. Исследование динамики изменений в структуре таких сетей позволит ответственному лицу (в зависимости от типа пользователей им может быть декан факультета, заведующий кафедрой, куратор студенческой группы и др.) принимать соответствующие управленческие решения для эффективного управления учебным процессом. Так, куратор может выделить неформального лидера в студенческой группе, проследить, как будет меняться его роль на протяжении семестра, проанализировать, кто, возможно, «перетянет на себя» роль лидера, или заменит его при удалении из коллектива. Исходя из анализа динамики изменений в структуре сети студентов, можно формировать подгруппы для выполнения различных

образовательных или научных проектов, контролировать степень участия каждого студента в реализации этих проектов. Для заведующего кафедрой или декана факультета важен мониторинг изменений, происходящих в рабочем коллективе с течением времени, например, как перераспределятся роли в коллективе при временном или постоянном отсутствии сотрудника, насколько быстро новый сотрудник адаптируется в коллективе (появятся прямые и косвенные связи с другими пользователями).

14. Список литературы

[1] Бурков В.Н., Горгидзе И.А., Ловецкий С.Е. Прикладные задачи теории графов. Тбилиси: Мецниереба, 1974. – 234 с.

[2] Емеличев В.А,, Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. М.: Наука, 1990. – 384 с.

[3] А. М. Райгородский. Модели случайных графов. — М: МЦНМО, 2011.

[4] B. Bollobás. Random Graphs, Second Edition. — Cambridge Univ. Press, 2001.

[5] З. В. Апанович Методы построения жгутов ребер для улучшения понимаемости информации

[6] Cui W., Zhou H., Qu, H. Wong P. C., X. Li, "Geometry-Based Edge Clustering for Graph Visualization //IEEE Transactions on Visualization and Computer Graphics. — vol. 14, no. 6.— 2008.

[7] «Hierarchical Edge Bundles:Visualization of Adjacency Relations in Hierarchical Data». Danny Holten.

[8] The Five Best Libraries For Building Data Visualizations. Luke Dormehl

[9] Newman, M. E. J. Girvan M. Finding and evaluating community structure in networks // Physical Review E 69, 026113. – 2004.

[10] Brandes, Ulrik. “A Faster Algorithm for Betweenness Centrality.” The Journal of Mathematical Sociology 25, no. 2 (June 2001): 163–77.

[11] Fast unfolding of communities in large networks, Vincent D. Blondel;Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre

[12] FREEMAN L. C. A set of measures of centrality based upon betweenness // Sociometrv. 1977. V. 40. P. 35-41.

[13] С. В. Бредихин, В.М. Ляпунов, Н.Г. Щербакова "МЕРА ВАЖНОСТИ НАУЧНОЙ ПЕРИОДИКИ — ЦЕНТРАЛЬНОСТЬ ПО ПОСРЕДНИЧЕСТВУ".

[14] Jacomy M., Venturini T., Heymann S., Bastian M. ForceAtlas2, a Continuous/ Graph Layout Algorithm for Handy Network Visualization

[15] В.Н. Бурков, Д.А. Новиков, ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ