САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Пешне Кристина Андреевна

Выпускная квалификационная работа бакалавра

Визуализация штатной структуры организации с распределением взаимодействия подразделений друг с другом

Направление 010400

Прикладная математика и информатика

Научный руководитель:

доктор физ.-мат. наук, профессор

Утешев А. Ю.

Научный соруководитель:

старший преподаватель

Давыденко А. А.

Рецензент:

кандидат тех. наук, доцент

Блеканов И. С.

Санкт-Петербург

2022

Содержание

Введение

В нашем мире внедрение системы электронного документооборота является уже не опцией, а необходимостью. Многие компании создают документы в цифровом виде, но из-за недостатков в организации управления документами и их обработки, приходится их распечатывать, перепечатывать, что заставляет терять время и силы на такой рутинный процесс. Системы управления документами – это централизованное хранилище, в которых работники компании могут получать доступ к актуальным документам из одного централизованного хранилища корпорации, а также возможность взаимодействовать с ними. Такая система способствует быстрому созданию, изменению и распространению документов среди сотрудников. Искать в электронных документах гораздо удобнее, чем искать вручную среди бумажных экземпляров. Тем более, чтобы передать документ, достаточно нажать одну кнопку, а не идти в соседний отдел и ждать, пока человек ознакомится и вернет документ с подписью. Однако с такими возможностями нужно быть аккуратными, электронное хранение документов в одном месте может представлять угрозу безопасности данных компании, если не воспользоваться хорошей защитой электронной системы.

Помимо всего этого, внедрение системы электронного документооборота решает главную проблему – «разбросанности» данных, когда разные документы хранятся на разных компьютерах, в разных отделах, а исправления и подтверждения находятся в разных электронных ящиках. Электронная система обеспечивает удобный доступ к документам для всех сотрудников.

Преимущества внедрения автоматизированных систем на сегодняшний день очевидны. Единственное, что сдерживает предприятия от перехода к таким системам – это мнимая сложность перевода бумажного документооборота на электронный и обучение пользования системой сотрудников. Но на самом деле адаптация и перенос проходят быстро и

безболезненно. Главное, что нужно помнить - внедрение системы электронного документооборота всегда преследует основную цель – эффективная работа предприятия в целом.

Деятельность любой компании связана с большим объемом документов, которые необходимо обрабатывать. Но чтобы эффективно работать в большой компании, необходимо понимать структуру компании, связь между подразделениями, выявлять узкие места и с помощью анализа пытаться делать взаимодействие внутри компании максимально эффективным.

Анализ электронного документооборота может позволить выявить взаимосвязь между сотрудниками в подразделениях и оптимизировать их взаимодействие. Поэтому в качестве объекта исследования возьмем набор документов сотрудников Docsvision из системы электронного документооборота.

Постановка задачи

Целью данной работы является разработка функционала для визуализации социального графа организации и анализ полученных результатов для оптимизации взаимодействия сотрудников компании посредством электронного документооборота. Основываясь на данной цели, были сформулированы основные задачи:

- Ввод необходимых понятий и терминов.
- Обзор реализованных методов выделения сообществ.
- Поиск способа объединения результатов работы алгоритмов с целью улучшить результат.
- Сравнение алгоритмов на реальных данных.
- Внедрение полученной реализации в систему электронного документооборота.
- Анализ с помощью полученных результатов структуры организации и взаимодействия между подразделениями.

Обзор литературы

В современном мире существует большое количество информации, которую можно представить в виде объектов и отношений между ними. Например, в [2] изучают структуру социального графа активных пользователей Facebook, крупнейшей из когда-либо проанализированных социальных сетей. В [1] описывают взаимосвязь социального, экономического и технологического миров, уделяя особое внимание некоторым фундаментальным идеям анализа социальных сетей и формулируя ряд теоретико-графовых понятий.

Существует большое число известных методов решение задачи поиска сообществ. Гирван и Ньюман в [3] предложили разделительный алгоритм, который использует коэффициент посредничества как метрику для определения границ сообществ. Этот алгоритм был успешно применен к различным сетям, включая сети сообщений электронной почты, социальные сети людей и животных, сети сотрудничества между учеными и музыкантами, метаболические сети и генные сети. Также в этой работе они ввели понятие модулярности. В [4], [6] предложили простые методы извлечения структуры сообщества крупных сетей, основанные на модульной оптимизации. Показано, что они превосходят другие известные методы обнаружения сообществ с точки зрения времени вычислений. В [8], [10] представлены методы, основанные на случайном блуждании, что дает хорошее отражение структуры сообщества в сети и эффективное вычисление для больших и сложных графов. В статье [7] исследуют простой алгоритм распространения меток, который использует только структуру сети в качестве ориентира и не требует ни оптимизации предварительно определенной целевой функции, ни предварительной информации о сообществах. В работе [5] исследуют четыре общих алгоритма обнаружения сообществ в сетях, а именно агломеративную иерархическую кластеризацию, разделительную иерархическую кластеризацию (Гирвана-Ньюмана), Fastgreedy и метод Лувена. Кроме того,

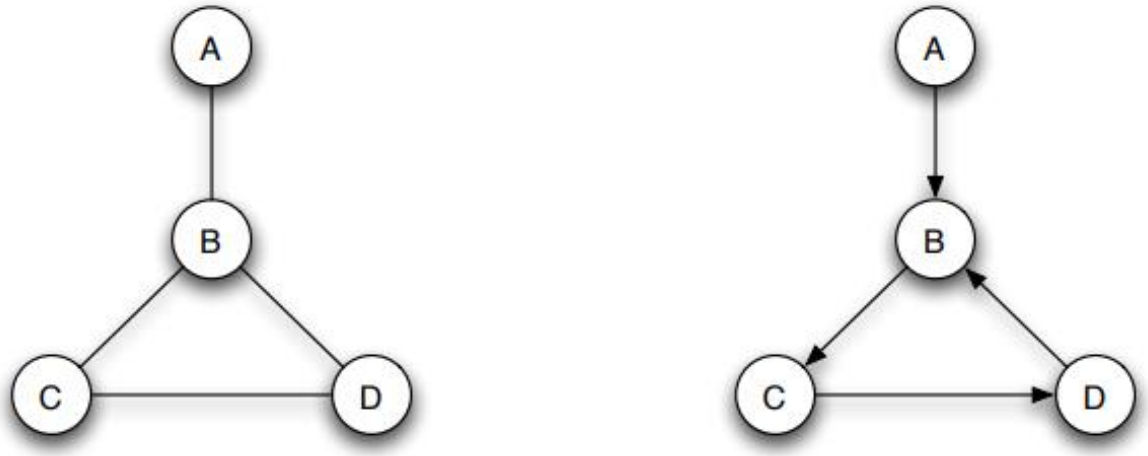
предлагаются некоторые усовершенствования этих алгоритмов, которые показывают многообещающие результаты.

Методы и метрики для выделения непересекающихся сообществ могут быть найдены в библиотеке igraph с подробной документацией [9]. Но несмотря на число известных методов, проблема разработки и исследования новых алгоритмов не теряет актуальности. Дело в том, что не существует алгоритма, который превосходил бы все остальные по всем критериям (быстродействие, нечувствительность к размерам и форме кластеров, количество параметров и т. д.).

Глава 1. Предметная область

1.1 Основные определения

Граф – это способ описания отношений между набором элементов [1]. Граф состоит из набора объектов, называемых узлами, с определенными парами этих объектов, соединенных связями, называемые ребрами. Будем говорить, что два узла являются соседями, если они соединены ребром. На рисунке 1a отношения между двумя концами ребра рассматриваются как симметричные, ребро просто соединяет два узла друг с другом. Если мы хотим выразить ассиметричные отношения, то используем ориентированный граф, состоящий из набора узлов направленных ребер, каждое направленное ребро является ссылкой из одного узла к другому, причем важно направление. Пример такого графа изображен на рисунке 1b, ребра обозначены стрелками. Далее, если не будет указано, что граф является ориентированным, то будем считать, что он неориентированный.



(a) Ненаправленный граф (b) Направленный граф

Рис.1: Два графа: (a) ненаправленный граф и (b) направленный граф

Теперь определим ряд важных понятий, которые пригодятся нам для анализа далее.

Степень вершины – количество ребер графа, инцидентных этой вершине, то есть количество ребер, имеющий начало или конец в этой

вершине.

Путь – это последовательность узлов такая, что каждая последующая пара в последовательности соединена ребром. Иногда удобнее рассматривать путь как последовательность не только узлов, но и ребер, соединяющих эти узлы. Рассматривая граф, мы часто используем понятие пути, чтобы проследить перемещение информации от человека к человеку в социальной сети или поискового робота, посещаемого последовательность веб-страниц по ссылкам [1]. Путь, состоящий не менее чем из трех ребер, в которых первый и последний узлы совпадают, а все остальные узлы различны, называют циклом. Логично ввести сразу понятие расстояния. Под расстоянием между узлами графа понимают минимальную длину пути в графе, соединяющего эти узлы. Если рассматривать граф как сеть дорог, то расстояние между пунктами А и Б — это будет кратчайший путь, по которому можно доехать из пункта А в пункт Б или наоборот.

Рассматривая граф, естественно задаться вопросом, а может ли каждый узел графа достичь каждого другого узла по какому-либо пути. По определению, граф называется связным, если между каждой парой вершин есть путь. Если существует хотя бы одна пара различных вершин, не связанная между собой путем, то такой граф будем называть несвязным. Проще всего представить связный граф как группу людей, в которой всегда найдутся два человека, которые не знакомы друг с другом лично, но существует цепь людей, которые знакомы между собой и соединяют этих двух людей.

Если граф несвязен, то он естественным образом распадается на множество групп узлов, так что каждая группа связана, если ее рассматривать как граф в изоляции, причем никакие две группы не должны пересекаться. Итак, компонента связности графа (или просто компонента графа) – максимальный связный подграф графа. Анализируя граф с точки зрения его связных компонент и границ между ними, мы изучаем структуру сети, и это будет центральной темой данной работы. На рисунке 2 представлен пример несвязного графа, который имеет три компоненты связности: первая

компонента состоит из узлов A, B; вторая – из узлов C, E, D, третья включает в себя все оставшиеся узлы графа.

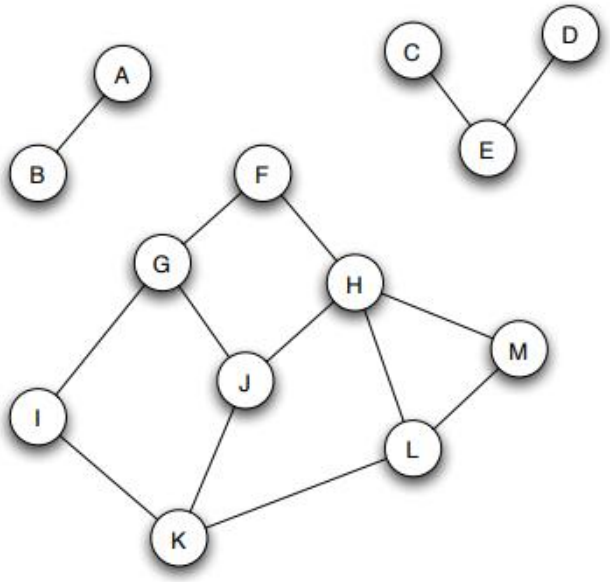


Рис. 2: Граф с тремя компонентами связности

1.2 Социальный граф

В жизни мы часто встречаем информацию, которую можно представить с помощью графов. Например, люди и их социальные связи, города и дороги между ними, веб-пространство и много другое. Однако графы могут быть очень разнообразны по своей структуре, и в данной работе мы будем исследовать, так называемые, социальные графы.

Как определить социальный граф? Первое, что приходит на ум, — это граф, вершинами которого обязательно являются люди. Но это не совсем так. Например, социальным можно также назвать граф научных статей, ссылающихся друг на друга. Строгого определения социального графа нет, скорее, это граф, обладающий некоторым набором свойств:

- Одна большая общая компонента связности. В большинстве случаях социальный граф имеет одну большую компоненту связности, в которую входит большинство вершин. Например, социальный граф активных пользователей Facebook почти полностью связан, при этом 99,91% людей

принадлежат к одному большому связанному компоненту [2].

- Среднее расстояние. Если рассматривать социальный граф, то количество рёбер, необходимых для перехода из одной случайной вершины в другую, невелико. Хорошей иллюстрацией будет известная теория шести рукопожатий, согласно которому любые два человека на Земле разделены в среднем пятью уровнями общих знакомых и, соответственно, шестью уровнями связей. Исследования показали, что среднее расстояние между парами пользователей составляет 4,7 для пользователей Facebook [2].
- Коэффициент кластеризации. Он основывается на том факте, что если вершина A соединена с вершиной B и соединена с вершиной C, то с высокой вероятностью также будут соединены вершины B и C. Если B и C действительно соединены, это называется закрытым триплетом, если нет — открытым. Коэффициент вычисляется как доля закрытых триплетов среди всех триплетов и может интерпретироваться как степень кластеризации данного графа. Пример подсчета такого коэффициента для вершин показан на рисунке 3. Если коэффициент принимает значения близкие к единице, то это значит, что в графе много треугольников и вершины склонны образовывать связь, если они соединены через третью вершину. Соответственно, если значение близко к нулю, то имеет место обратная тенденция. Социальный граф обладает большим коэффициентом кластеризации. 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. Социальный граф имеют структуру сообществ.

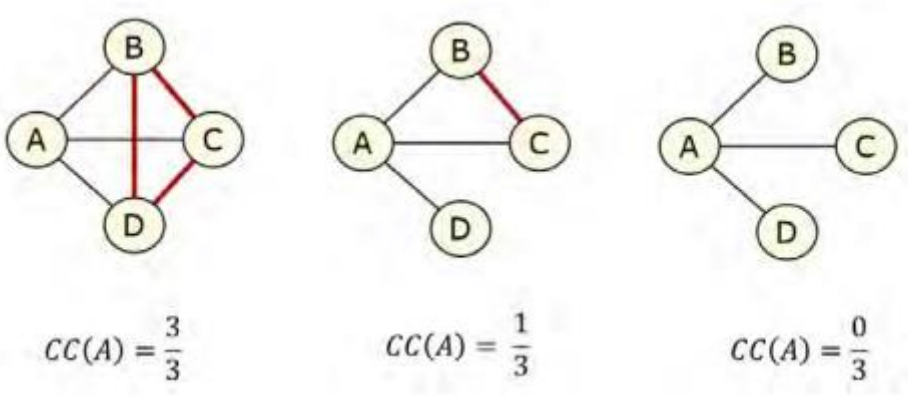


Рис.3: Коэффициенты кластеризации для вершины A для трех случаев.

В нашем случае вершинами социального графа будут сотрудники компании, а ребрами их документы, отправленные друг другу.

1.3 Сообщества

Строгого определения сообщества нет. Сообществом называют множество вершин, внутренние связи которого сильнее, чем внешние. Данное определение хорошо прослеживается на рисунке 4. Слева граф имеет структуру сообществ, справа – нет.

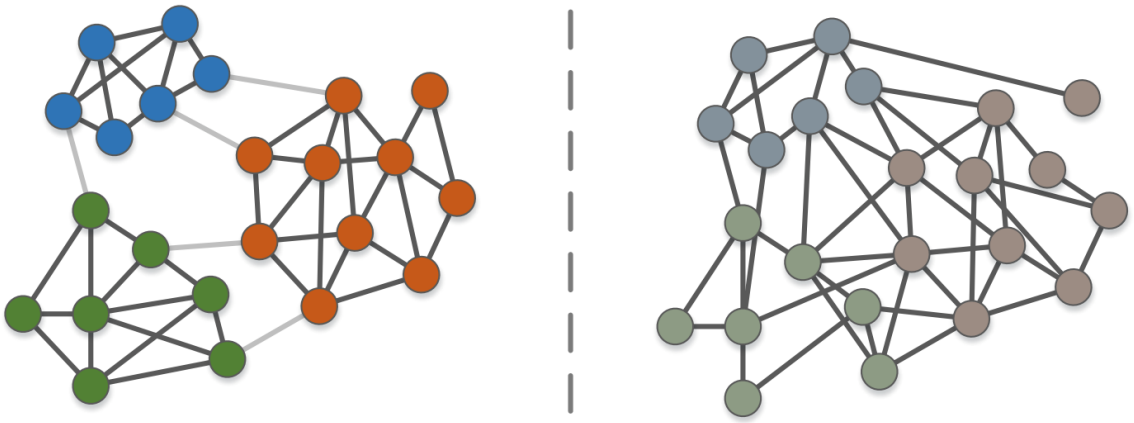


Рис.4: Примеры графов со структурой сообществ и без

В данной работе будет использоваться такое предположение о структуре сообществ в графе, согласно которому каждая вершина графа входит в одно и только одно сообщество. Обычно также делают предположение, что

сообщество состоит из одной компоненты связности. В противном случае его можно разделить на несколько более мелких сообществ. Отметим, что из предположения сразу следует, что сообщества полностью покрывают граф. Тогда можно говорить о поиске сообществ в графе как о задаче поиска разбиения множества вершин на подмножества, которое минимизирует некоторый функционал.

Можно также рассматривать исходную проблему как задачу кластеризации. Действительно, если ввести некоторую меру расстояния на вершинах графа с учетом структурной информации, можно решать задачу кластеризации. Такой подход также позволяет учитывать дополнительную информацию. Далее будем говорить о задаче кластеризации или поиска разбиения, имея ввиду исходную задачу выделения сообществ в графе.

1.4 Модулярность

Также нам потребуется некоторая числовая характеристика, которая описывает выраженность структуры сообществ в данном графе. Так как мы исследуем граф на реальных данных, то нам неизвестно истинное разбиение на сообщества. В этом случае для оценки качества мы будем использовать значение функционала модулярности, что является самой популярной и общепризнанной мерой качества для данной задачи.

Функционал был предложен Ньюманом и Гирваном в ходе разработки алгоритма кластеризации вершин графа [3]. Модулярность – это скалярная величина из отрезка [-1, 1], которая количественно описывает неформальное определение структуры сообществ:

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

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. $\delta(C_i C_j)$ — дельта-

функция: равна единице, если $C_i = C_j$, иначе нулю. Попытаемся понять, что она означает. Возьмём две произвольные вершины i и j. Вероятность появления ребра между ними при генерации случайного графа с таким же количеством вершин и рёбер, как у исходного графа, равна $\frac{d_i d_j}{2m}$. Реальное количество рёбер в сообществе C будет равняться $\sum_{i,j \epsilon C} A_{ij}$.

Таким образом, модулярность равна разности между долей рёбер внутри сообщества при данном разбиении и долей рёбер, если бы они были случайно сгенерированы. Поэтому она показывает выраженность сообществ (случайный граф структуры сообществ не имеет). Также стоит отметить, что модулярность равна 1 для полного графа, в котором все вершины помещены в одно сообщество и равна нулю для разбиения на сообщества, при котором каждой вершине сопоставлено по отдельному сообществу. Для особо неудачных разбиений модулярность может быть отрицательной.

Задача поиска выделения сообществ в графе сводится к поиску таких C_i, которые максимизируют значение модулярности. Формула имеет множество обобщений, например, для взвешенных графов, под A_{ij} понимается вес ребра, соединяющего вершины i и j, а $m = \frac{1}{2}\sum a_{ij}$.

Достоинства. 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

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

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, $d_{i,in}$ — сумма весов ребер соединяющих вершину d_i и вершину из

C, m — сумма весов всех ребер.

Недостатки. Функционал не является непрерывным, и задача его оптимизации — дискретная. Для поиска глобального оптимума используют приближенные схемы. Некоторые из них действительно оптимизируют значение функционала, другие же по значению модулярности выбирают наилучшее решение из найденных, то есть без гарантий глобальной оптимальности решения. У данного функционала существует проблема разрешающей способности (грубо говоря, функционал не видит маленькие сообщества). Эта проблема решается путем использования модифицированного функционала, который сохраняет все достоинства и добавляет параметр масштаба.

Глава 2: Алгоритмы поиска сообществ

2.1 Betweenness

Betweenness — алгоритм на основе коэффициента посредничества, определяемый как количество кратчайших путей между всеми парами вершин, проходящих через данное ребро [3]. В этом методе мы сосредотачиваемся на тех ребрах, которые являются наименее центральными, то есть которые находятся между сообществами. Если существует более одного кратчайшего пути между парой вершин, каждому пути присваивается одинаковый вес такой, что общий вес всех путей равен единице. Если сеть содержит сообщества или группы, слабо связанные между собой несколькими межгрупповыми ребрами, то все кратчайшие пути между разными сообществами должны проходить по одному из этих нескольких ребер. Например, если между вершинами N кратчайших путей, то каждому ребру прибавляется $1/N$ к значению коэффициента. Чем больше данная величина, тем более вероятно, что данное ребро соединяет вершины из разных сообществ. Удалив такие ребра, мы разделим группы друг от друга и таким образом найдем сообщества.

Алгоритм работает по следующей схеме:

1. Подсчет коэффициентов посредничества для всех рёбер графа.
2. Поочерёдное удаление рёбер с самым большим коэффициентом.
3. Обновление промежуточных значений для всех рёбер, затронутых удалением.
4. Процедура удаления связей завершается, когда ребра заканчиваются.

Приведем простейший пример, когда разные группы соединены одним ребром, все кратчайшие пути из одного сообщества в другое проходят через это ребро (рис. 4). Если удалить ребро с самым большим значением, то граф разделится на 2 компоненты связности, которые можно рассматривать как 2 сообщества в исходном графе.

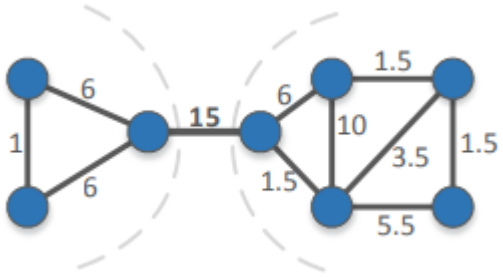


Рис. 4: Пример графа с двумя компонентами связности

Данный алгоритм является одним из первых алгоритмов по выделению сообществ в сетях. Его главный недостаток — время работы. На практике мы вычисляем отношения между сообществами, используя быстрый алгоритм Ньюмена [3], который вычисляет коэффициент посредничества для всех m ребер в графе из n вершин за время O(mn). Поскольку этот расчет должен быть повторен один раз для удаления каждого ребра весь алгоритм работает в худшем случае за время O(m^2n). Это значит, что данный метод не подходит для графов с большим количеством вершин.

2.2 Fastgreedy

Данный метод заключается в жадной оптимизации модулярности. Алгоритм предназначается в первую очередь для анализа больших сетей, он позволяет быстро производить расчеты с сетями до сотен тысяч вершин и миллионов ребер [4]. Работа алгоритма заключается в нахождении изменений модулярности, которые произошли бы в результате объединения каждой пары сообществ, выбирая самое больше значение и выполняя соответствующее объединение. В каждой вершине графа инициализируется отдельное сообщество, а затем объединяются пары сообществ, приводящие к максимальному увеличению модулярности. При этом объединяются обычно не все пары сообществ, а только те, между вершинами которых существуют связи, так как, в противном случае, модулярность не может увеличиться.

Алгоритм работает по следующей схеме.

1. Инициализация сообществ в каждой вершине.
2. Объединение сообществ, в результате которого максимальным образом увеличивается модулярность.
3. Выход, если дальнейшее увеличение значения функционала невозможно.

Метод является вычислительно нетрудоёмким ($O(m\log n)$), легко применим к большим графам и зачастую неплохо справляется с задачей. Однако часто метод порождает одно большое сообщество с большинством вершин графа в нем и множество маленьких.

2.3 Multilevel

Multilevel – простой метод извлечения структуры сообщества из больших графов, основанный на модульной оптимизации [6].

Шаги алгоритма:

1. Инициализация сообществ в каждой вершине, то есть каждой вершине ставится в соответствие по сообществу.
2. Первый этап.

   (а) Для каждой вершины рассматриваем изменение модулярности при перемещении данной вершины в сообщество вершины-соседа. Перемещаем в сообщество соседа, если модулярность увеличивается максимально, в противном случае вершина остаётся в текущем сообществе.

   (б) Выполнять первый шаг до тех пор, пока ни одно отдельное перемещение не сможет улучшить значение модулярности.
3. Второй этап

   (а) Построить новый граф (метаграф), узлами которого являются сообщества, найденные во время первой фазы. При этом рёбра будут иметь веса, равные сумме весов всех рёбер из одного сообщества в другое. Связи между узлами одного и того же сообщества привести к самопетлям для этого сообщества в новой сети.

(б) Перезапустить алгоритм на новом графе и продолжать до тех пор, пока не будет достигнута максимальная модулярность, то есть любое изменение не будет давать улучшение. Все исходные вершины, которые входят в финальную метавершину, принадлежат одному сообществу.

Общая схема алгоритма представлена на рис. 5, где показано 2 итерации. На первой показаны оба этапа: 1) в котором с помощью локальных изменений сообществ оптимизируется модульность и 2) в котором найденные сообщества объединяются, чтобы построить новую сеть сообщества.

Создатели метода утверждают, что алгоритму требуется немного (3-4) перезапусков для завершения и что сложность линейна на типичных и разреженных данных.

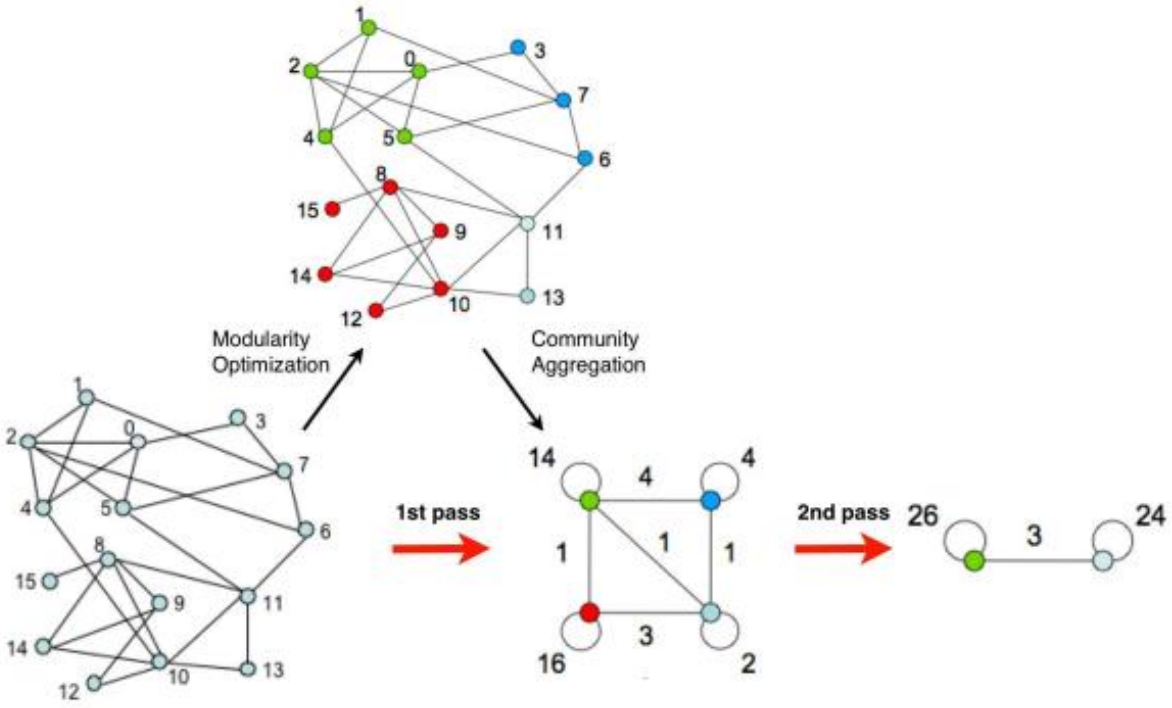


Рис.5: Визуализация шагов алгоритма [6].

2.4 LabelPropogation

LabelPropogation - метод, в основе которого лежит предположение, что вершина относится к тому сообществу, что и большинство ее соседей 35[7]. Если таких сообществ несколько, то выбирается одно из них. Сначала происходит инициализация каждого узла уникальными метками,

соответствующими определенному сообществу. Далее запускается алгоритм распространения меток, то есть для каждой вершины определяем сообщество, к которому принадлежит максимальное число ее соседей. Распространение меток происходит до тех пор, пока происходят изменения. В конце процесса распространения узлы с одинаковыми метками объединяются в одно сообщество. Процесс выполняется итеративно, где на каждом шаге каждый узел обновляет свою метку на основе меток соседей. Порядок, в котором все n узлов в сети обновляются на каждой итерации, выбирается случайным образом, чтобы нейтрализовать зависимость от порядка. По мере итераций количество уникальных меток уменьшается, в результате чего их остается столько, сколько существует сообществ в выбранном графе. На рисунке 6 представлена работа алгоритма для полного графа. Узлы обновляются один за другим по мере движения слева направо. Благодаря высокой плотности ребер все узлы получают одну и ту же метку [7].

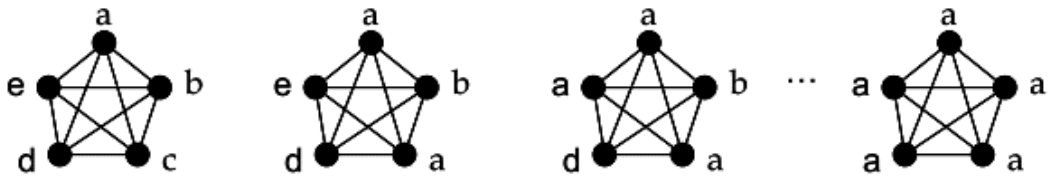


Рис.6: Работа алгоритма для полного графа

Метод характеризуется почти линейной сложностью. Из-за рандомизации алгоритм при нескольких запусках может выдать разные результаты, т.к. критерию останова не обязательно соответствует одна разметка вершин. Авторы метода предлагают агрегировать результаты простым пересечением кластеров. В сложных зашумленных графах метод может объединить все сообщества в одно.

2.5 Walktrap

Walktrap — подход, основанный на случайном блуждании. Использует

идею о том, что короткие случайные блуждания не приводят к выходу из текущего сообщества [8].

Как мы знаем, граф связан со своей матрицей смежности: A_{ij} = 1, если вершины i и j связаны и A_{ij} = 0 в противном случае. Степень вершины i вычисляется $d(i) = \sum_j A_{ij}$.

Рассмотрим дискретный процесс случайных блужданий на графе. На каждом шаге блуждающий объект находится на вершине и перемещается на вершину, выбранную случайным равновероятным образом среди своих соседей. Последовательность посещенных вершин представляет собой цепь Маркова, состояниями которой являются вершинами графа. На каждом шаге вероятность перехода из вершины i в вершину j равна $P_{ij} = \frac{A_{ij}}{d_i}$. Таким образом является матрица перехода P процесса случайного блуждания. Через данную матрицу можно получить вероятность перехода из вершины i в вершину j за t шагов как $P^t{}_{ij}$. Если обозначить за D матрицу, на диагонали которой стоят степени вершин di, то $P = D^{-1}A$.

Чтобы сгруппировать вершины в сообщества, теперь введем расстояние r между вершинами, которые фиксируют структуру сообщества графа. Это расстояние должно быть большим, если две вершины находятся в разных сообществах, и наоборот, если они находятся в то же сообществе.

Теперь мы можем дать определение расстояния между вершинами:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}} = \left\| D^{-\frac{1}{2}} P_{i\bullet}^t - D^{-\frac{1}{2}} P_{j\bullet}^t \right\|$$

Где $\|\cdot\|$ - евклидова норма в $\mathbb{R}^n$, $P_{i\bullet}^t$ – вектор столбец в j позиции которого стоит значение P_{ij}^t.

Метрику можно обобщить на совокупность вершин, т.е. сообщества. Обозначим за P_{Cj}^t вероятность добраться из сообщества C в вершину j за t шагов:

$$P_{Cj}^{t} = \frac{1}{|C|}\sum_{i \epsilon C} P_{ij}^{t}$$

Обозначим за $P_{C}^{t}\bullet$ вектор из вероятностей P_{Cj}^{t}, тогда для двух сообществ C1, C2 ⊂ V расстояние определяется как:

$$r_{C_1C_2} = \left\| D^{-\frac{1}{2}} P_{C_1}^{t}\bullet - D^{-\frac{1}{2}} P_{C_2}^{t}\bullet \right\| = \sqrt{\sum_{k=1}^{n} \frac{(P_{C_1k}^{t} - P_{C_2k}^{t})^2}{d_k}}$$

Теперь, когда задана метрика, задача выделения сообществ сведена к задаче кластеризации верши.

Алгоритм:

1. Инициализация. Каждая вершина содержится в своем кластере, т.е. начальное разбиение выглядит как $\mathcal{P}_1 = \{\{v\}, v \in V\}$

2. Вычисление расстояния между всеми смежными вершинам

3. На каждом шаге k

(а) Выбрать два сообщества C_1 и C_2 из $\mathcal{P}_k$ по следующему критерию. Рассматриваются только те пары сообществ C_1 и C_2, которые инциденты друг другу. Объединяем сообщества, минимизирующие изменение среднего квадратов расстояний между каждой вершиной и их сообществом при объединении этих сообществ:

$$\sigma_k = \frac{1}{n}\sum_{C \epsilon \mathbb{P}_k}\sum_{i \epsilon C} r_{iC}^{2} \to min$$

Такая задача эквивалентна поиску $\mathrm{C}_1, \mathrm{C}_2$, минимизирующих величину

$$\Delta\sigma(\mathrm{C}_1, \mathrm{C}_2) = \frac{1}{n}\left(\sum_{i \in c_3} r_{iC_3}^{2} - \sum_{i \in c_1} r_{iC_1}^{2} - \sum_{i \in c_2} r_{iC_2}^{2}\right) \to \min_{\mathrm{C}_1, \mathrm{C}_2}$$

(б) Объединить эти сообщества в одно новое, $C_3 = C_1 \cup C_2$. Соответствующим образом преобразовать $\mathcal{P}_k$ в $\mathcal{P}_{k+1}$.

(в) Обновить расстояния между сообществами, достаточно

сделать только для смежных с C_1 и C_2 сообществами.

На n−1 шаге мы получаем $\mathcal{P}_n$ = {V} и алгоритм завершается. Таким образом, получена дендрограмма для вершин графа.

Теперь осталось только выбрать наилучшее разбиение $\mathcal{P}_k$, максимизирующее модулярность. Сложность такого метода: O(n^2log n).

2.6 Infomap

Метод, как и предыдущий, использует механизм случайных блужданий. Авторы интерпретируют задачу выделения сообществ в графе, как задачу кодирования пути, который пройдет блуждатель, и пытаются минимизировать длину кода [10].

Каждое сообщество имеет свой уникальный бинарный код. В сообществе каждая вершина имеет свой уникальный внутренний код (вершины из разных сообществ могут иметь одинаковый код). Также есть дополнительный код выхода из сообщества, не совпадающий с кодами вершин в этом сообществе.

Кодирование пути выглядит следующим образом: при попадании в другое сообщество записывается его код и внутренний код вершины, в которую попал объект. Далее при переходе внутри одного сообщества записываются внутренние коды вершин. При переходе в другое сообщество пишется код выхода из данного сообщества и код нового.

Таким образом имеется 2 уровня кодирования: уровень сообществ и вершин. Для каждого из них используются коды Хафмана в соответствии с вероятностью посещения данной вершины или данного сообщества.

В таком виде среднее описание длинны одного перехода равно

$L(M) = q_{\curvearrowright} H(C) + \sum_{i=1}^{m} p_{\circlearrowleft}^{i} H(C_i)$,

Где $q_{\curvearrowright}$ − вероятность покинуть какое-либо сообщество на любом шаге, H(C) - энтропия кодов сообществ, H(C_i) — энтропия кодов внутри сообщества C_i, вес $p_{\circlearrowleft}^{i}$ — это доля перемещений внутри сообщества C_i плюс вероятность

покинуть сообщество.

С помощью матрицы P, описанной в предыдущем методе, вычисляется вероятность посещения той или иной вершины и, используя эту информацию, запускается жадный метод оптимизации функционала L(M), после чего результат работы некоторым образом уточняется.

На практике получается, что при увеличении зашумленности графа метод объединяет все вершины в одно сообщество. Однако, на реальных данных метод работает на уровне всех остальных, что будет видно из дальнейших экспериментов.

На рис. 7 показана схема работы данного метода. На левом рисунке 7(a) показан путь случайного блуждателя. На второй части 7(b) изображены вершины с кодами Хафмана, а ниже закодирован путь, изображенный на левом изображении. На следующей части 7(c) показано кодирование с помощью предложенного двухуровневого метода. Ниже записаны коды сообществ и коды выхода из них. В самом низу закодирован путь. Видно, что длина кода стала меньше. На последнем рисунке 7(d) показана выделенная структура сообществ в графе и коды, соответствующие группам. Ниже выделены коды, касающиеся групп [10].

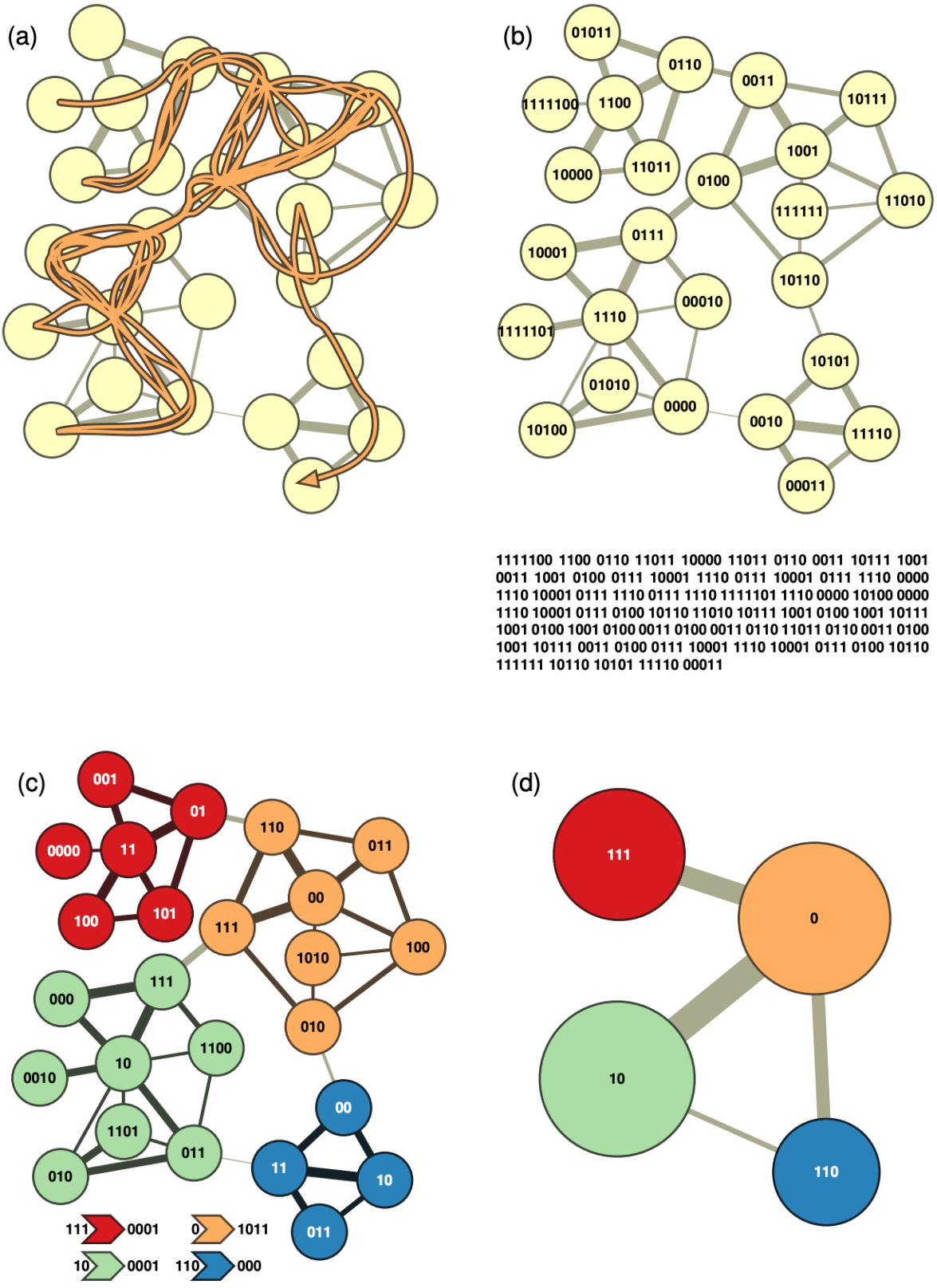


Рис. 7. Схема работы метода infomap [10].

Глава 3: Реализация

3.1 Улучшение методов поиска сообществ

Попробуем улучшить результаты методов поиска сообществ путем объединения результатов их работы. Улучшать будем с помощью агрегирования результатов базовыми методами. Реализовывать базовые и улучшенные методы будем с помощью библиотеки igraph [9].

Каждый алгоритм, который мы используем для поиска сообщества, на выходе дает нам свое разбиение $T^i = \{T_1^i, \dots, T_{m_i}^i\}$, $i = 1, \dots, n$. Мы можем рассмотреть новое разбиение в виде пересечения вершин, которые во всех методах принадлежат одному и тому же сообществу. В результате мы получим разбиение в виде измельчения полученных: $T = \bigcap_i T^i = \{T_1, \dots, T_m\}$.

Таким образом, мы построили новую структуру сообществ, которая объединяет вершины, с большой вероятностью лежащие в одном классе. Теперь для нашего нового разбиения T построим метаграф, вершины которого являются сообществами, вес ребра равен сумме весов всех ребер, идущих от вершин этого сообщества к другому, а вес петли равен сумме весов ребер, соединяющих вершины, которые входят в это сообщество. Вершинам, которые не вошли ни в одно сообщество, сопоставим отдельные метки, то есть проинициализируем каждую вершину своим сообществом. Далее запустим один из рассмотренных методов выделения сообществ и получим новое разбиение $T = \{T_1, \dots, T_k\}$. При этом, вершины x и y будут считаться из одного сообщества, если соответствующие метавершины оказались в одной группе: $x \in T_i, y \in T_j$ из одного сообщества, если $T_i \in T_l, T_j \in T_l$. В результате мы вспоминаем, из чего состоят наши метавершины и формируем сообщества на основе нового разбиение с вершинами из первоначального графа.

На рисунке 8 представлена схема работы метода. В начале мы использовали два базовых алгоритма и получили, соответственно, два

разбиения. Из них создаем новое разбиение путем пересечения исходных, то есть перебираем все вершины, если сообщества в двух разбиениях совпадают, то оставляем текущую метку, если нет, то обновляем. Но основе нового разбиения строим метаграф, объединяя все полученные сообщества в соответствующие вершины. После этого запускаем любой из рассмотренных в главе 2 алгоритмов выделения сообществ. Получаем новое разбиение и возвращаемся к вершинам исходного графа, но с учетом нового разбиения.

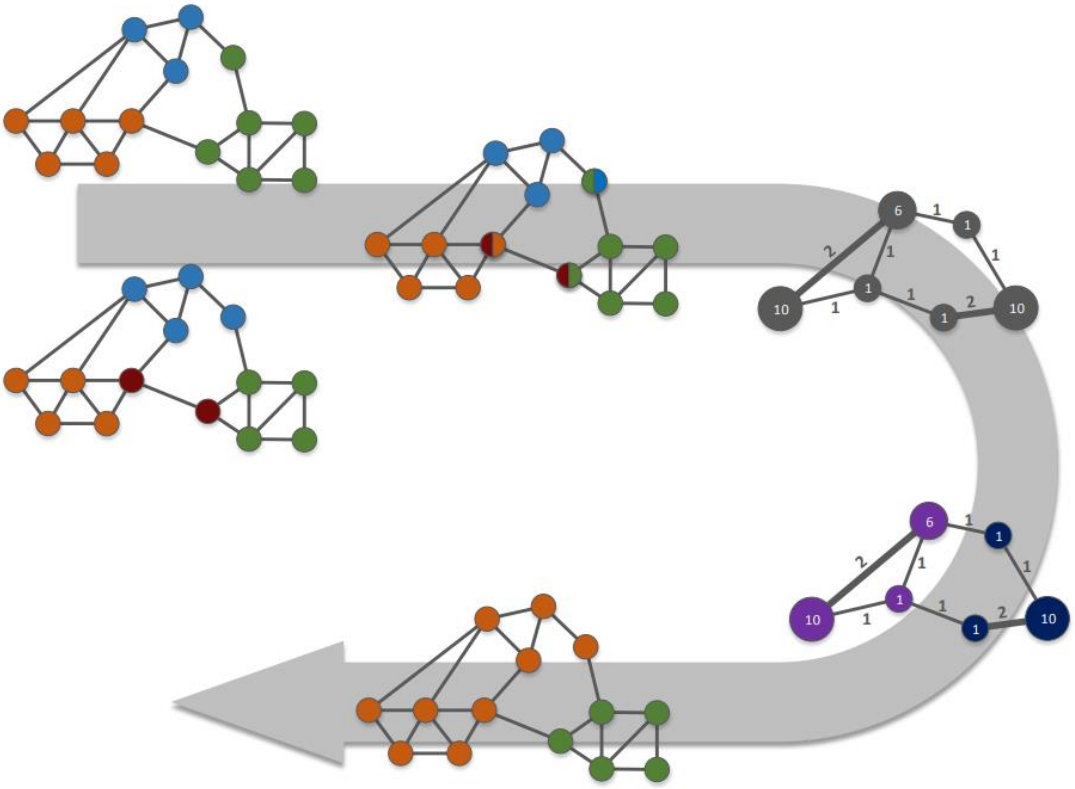


Рис.8: Схема работы алгоритма агрегирования базовыми методами

При получении начальных разбиений с помощью базовых методов при пересечении могут получаться очень маленькие сообщества, поэтому можно принять, что вершины принадлежат одному и тому же сообществу, если они принадлежат одному и тому же сообществу в большинстве исходных разбиений.

3.2 Выявление лучшего алгоритма

Тестировать методы будем на реальных данных, которые предварительно были подготовлены для поиска сообществ и для которых нам и нужно получить наилучший результат. Они представляют собой набор узлов и ребер. Узлами являются сотрудники компании Docsvision, с метаданными, включающими имя, фамилию, должность и подразделение, в котором работает сотрудник, а ребрами два сотрудника, которые отправляли между собой документы с помощью электронного документооборота с весом, равным количеству этих документов.

Для реальных данных единственное, что возможно сделать, это запустить все методы выделения сообществ и исследовать распределение модулярности на них. Необходимо понять, дают ли прирост функционалу описанные методы агрегирования и какой метод лучше всего использовать в качестве агрегирования.

Запустим все методы на реальных данных и посмотрим на распределение значений модулярности для каждого метода.

Для начала рассмотрим базовые методы и выберем 3 наилучших, которые будем использовать в качестве начальных разбиений. Результаты представлены в таблице 1.

Название метода	Значение модулярности
Multilevel	0.845925
Label propagation	0.796742
Infomap	0.755258
Fastgreedy	0.844963
Betweenness	0.770537
Walktrap	0.737904

Таблица 1

Как видно из таблицы 1 лучшие результаты показали Multilevel,

Fastgreedy и Label Propagation. Будем их использовать в качестве методов агрегирования результатов. Чтобы как-то отличить их от базовых методов, добавим в название «Final», то есть если в качестве агрегирования был использован метод Multilevel, то метод будет называться FinalMultilevel. В таблице 2 представлена аналогичная таблица для методов агрегирования результатов, но добавлен еще один столбец, который показывает разность между максимальным значением модулярности среди простых методов и полученным объединением их результатов.

Название метода	Значение модулярности	Прирост
FinalMultilevel	0.845927	0.000002
FinalLabelPropagation	0.829854	-0.016071
FinalFastgreedy	0.861023	0.015098

Таблица 2

Из таблицы видно, что FinalMultilevel практически повторил лучший из имеющихся результатов, метод FinalLabelPropagation худший результат, а FinalFastgreedy показал хороший прирост. Таким образом, самым эффективным оказался FinalFastgreedy, поэтому его мы будем использовать в качестве выделения сообществ в графе сотрудников компании Docsvision.

3.3 Анализ структуры графа

Что именно мы хотим узнать с помощью графа? Во-первых, нам было бы интересно узнать, сформировано ли внутри каждого из подразделений настоящее сообщество и насколько оно разрозненно. Во-вторых, каким образом компания разделяется по группам, как в них взаимодействуют люди из разных подразделений и есть ли там узкие места – люди, на которых держатся большинство связей из подразделений с малым количеством человек, или же все распределено равномерно. В-третьих, насколько выделены отдельно (и выделены ли вообще) такие подразделения, как

бухгалтерия, работа с персоналом, аналитики. В теории все сотрудники компании могут общаться между собой, но нас интересовала практика.

Итак, на рисунке 9 представлена получившаяся структура графа сотрудников компании Docsvision. С целью улучшения эффективности анализа были добавлены метаданные сотрудников на граф. Но по соображениям конфиденциальности, в данную работу имена и фамилии сотрудников я включить не могу, зато могу рассказать, что удалось установить с их помощью. Также был изменен размер узлов так, чтобы большие точки соответствовали людям, получающим особенно много писем.

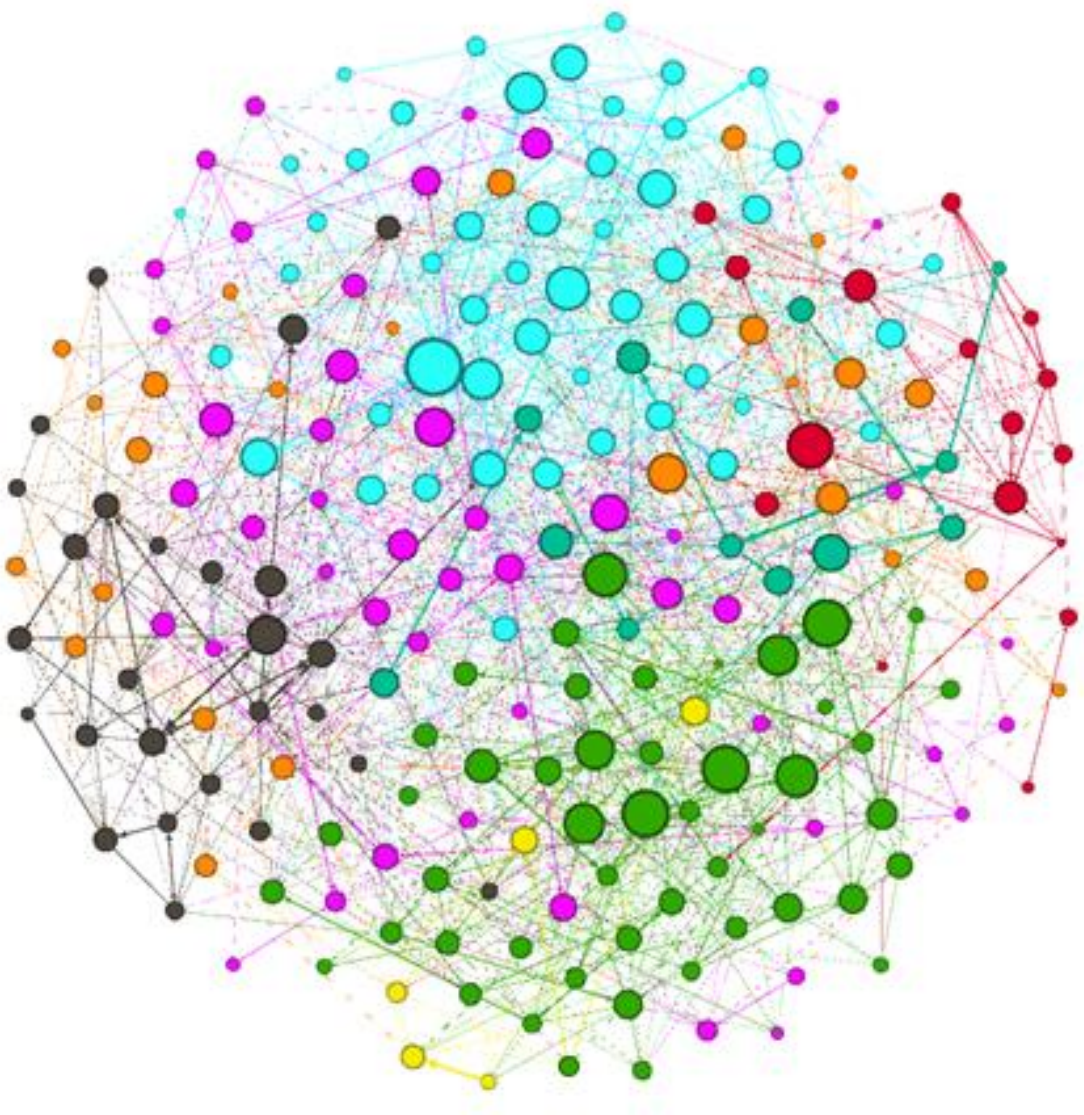


Рис.9: Социальный граф сотрудников компании Docsvision

Голубым цветом представлено подразделение разработки совместно

с HR-менеджерами, бухгалтерами и аналитиками. Между коллегами происходит активная коммуникация с помощью электронного документооборота, даже если разработчики находятся на разных проектах. Лидеры (самые большие кружки) — HR-менеджеры, бухгалтеры, аналитики и ведущие разработчики, которые курируют других своих коллег и активно участвуют в жизни подразделения разработки. Они следят за выполнением задач, релизами, часто присылают ознакомительные документы, связанные с изменением в производстве. Центральный большой узел — главный бухгалтер. С одной стороны, может удивить, что бухгалтер выделен в одно сообщество с разработчиками, потому что его письма равномерно отправляются всем сотрудникам компании, но видимо такое распределение произошло из-за количества людей в этом подразделении разработки, то есть здесь попросту больше связей с этим сообществом. По такому же принципу, скорее всего, сюда вошли HR-менеджеры, которые помогают планировать командировки, рассылают уведомления о новых сотрудниках и значимых мероприятиях. Работа аналитиков тесно связана с отделом разработки, так как именно аналитики производят спецификацию требований, рассылая их по командам.

Зеленым цветом выделено подразделение технической поддержки, туда же вошла часть бухгалтеров и HR-менеджеров по той же причине многочисленности подразделения. Большими кружками отмечены бухгалтеры, HR-менеджеры и руководители группы технической поддержки по понятным причинам.

Фиолетовый цвет — подразделение внедрения и отдел продаж, которые занимаются продвижением компании на рынках труда и общение которых тесно связано между собой, что и отразилось на выделение их в одно сообщество.

Черным цветом обозначено подразделение тестирования, которые достаточно равномерно отправляют друг другу документы. Среди сообщества выделяются естественным образом главные тестировщики.

Красным цветом обозначены системные администраторы. Это те, кто помогает настраивать виртуальные окружения, налаживает работу корпоративных систем и консультирует коллег по поводу различных проблем с инфраструктурой. Все обращения они получают через электронный документооборот. Самые большие точки — два главных системных администратора, которые действительно много времени уделяют разборам обращений и перенаправлением на своих коллег.

Остальные сообщества не представляют собой какие-то конкретные подразделения, в них входят как сотрудники из уже названных отделов, так и сотрудники из отделов по качеству, финансам и так далее.

Думаю, что полученный граф достаточно точно отражает связи внутри компании. Не так уж удивительно, что бухгалтеры и HR-менеджеры не выделены в отдельное сообщество. Их связи с другими подразделениями намного сильнее, чем внутренние. При этом с учетом численности этих сотрудников и количеством документов, которые они отправляют и принимают (заметно по размеру узлов), можно увидеть, что они подвержены большой нагрузке, что является узким местом и может тормозить работу компании. В качестве рекомендаций по устранению узких мест я бы порекомендовала назначить дополнительных заместителей для наиболее загруженных сотрудников и перевести часть менее важных вопросов на них, тем самым снизив нагрузку на ведущих специалистов. В остальных подразделениях взаимодействия достаточно равномерные.

Результаты

В ходе работы были получены следующие результаты:

1. Составлен обзор основных методов и подходов для выделения сообществ в социальном графе.
2. Представлен подход объединения результатов работы методов по выделению сообществ – агрегирование базовыми методами.
3. Были предложены три конкретные реализации методов: FinalFastgreedy, FinalMultilevel и FinalLabelPropagation, а также среди них выявлен лучший метод FinalFastgreedy.
4. Был реализован и внедрен в электронный документооборот функционал визуализации структуры компании с выделенными сообществами. Подтверждение использования кода можно найти в Приложении 1.

Заключение

В работе были рассмотрены методы выделения непересекающихся сообществ в социальных графах. Стоит отметить, что выделение сообществ не является строгой задачей, вследствие чего нет идеального алгоритма. Такое представление организации помогает понять социальную структуру и проследить взаимодействия между подразделениями, а также оценить этих взаимодействий и возможности оптимизации.

Список литературы

[1] David Easley and Jon Kleinberg. Networks, crowds, and markets: Reasoning about a highly connected world. Cambridge University Press, 2010.

[2] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph, 2011.

[3] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 2002.

[4] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. Physical review E, 2004.

[5] Niko Motschnig, Alexander Ramharter, Oliver Schweiger, Philipp Zabka, Klaus-Tycho Foerster. On Comparing and Enhancing Common Approaches to Network Community Detection

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008.

[7] Usha Nandini Raghavan, Reka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. Physical Review, 2007.

[8] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In Computer and Information Sciences-ISCIS 2005

[9] igraph for python, 2015. http://igraph.org/python/ 2015-04-26.

[10] Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. The map equation. TheEuropean Physical Journal-Special Topics, 2009.

Приложение 1

docsvision

Общество с ограниченной ответственностью
«ДоксВижн»
Россия, Санкт-Петербург,
набережная реки Смоленки, д. 33,
лит. А, пом. 144-Н, 199178
Тел. (СПб): +7 (812) 622-16-89
E-mail: info@docsvision.com
docsvision.com
ОКПО 64314109, ОГРН 1107847070638
ИНН/КПП 7801515318/780101001

04.05.2022

Информационное письмо

Общество с ограниченной ответственностью «ДоксВижн», действующее в соответствии с законодательством Российской Федерации, зарегистрированное по адресу 199178, Россия, Санкт-Петербург, наб. реки Смоленки, 33 и являющееся обладателем исключительных прав на программный продукт Docsvision,

настоящим письмом уведомляет о том, что исходный код, полученный в ходе работы студентки Пешне К. А. над выпускной квалификационной работой *"Визуализация штатной структуры организации с распределением взаимодействия подразделений друг с другом"*, полезен и используется в компании ООО «ДоксВижн».

С уважением,
руководитель отдела производства ООО «ДоксВижн»
Елхов Д. С.