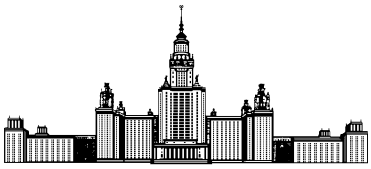

Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Анализ социальных графов»

Выполнил:
студент 4 курса 417 группы
Славнов Константин Анатольевич

Научный руководитель:
д.ф-м.н., профессор
Дьяконов Александр Геннадьевич

Москва, 2015

Содержание

1 -| Вопросы |-

1. заголовки?
2. что выносить на защиту?
3. какие методы/пункты стоит описать подробнее?

2 Введение

В современном мире существует большое количество информации, представимой в виде объектов и отношений между ними. [1]

Например, рассмотрим научные статьи в качестве объектов. Тогда скажем, что между статьями есть связь, если одна из них ссылается на вторую. Таким образом все существующие работы ученых могут быть представленны в виде графа. Такое же представление возможно для многих других структур из самых разных областей знания: люди и отношения знакомства, протеины и их взаимодействия, интернет. 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 ежедневная задача как визуализация становится проблематичной. Один из возможных подходов к ее решению заключается в представлении исходного графа на более высоком уровне. 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. Визуализация графа это всего лишь один из практических аспектов данной задачи.

В этой работе анализ графов будет идти именно с точки зрения данной задачи.

3 Общая постановка задачи

Задача заключается в изучении и сравнении существующих методов выделения сообществ с целью улучшить их результат работы.

В данной работе будут рассматриваться простые графы с неориентированными, невзвешенными, некратными ребрами, хотя большинство рассмотренных методов имеют обобщения и на случай более сложных графов, что будет использоваться в конце работы. Также дополнительная информации, которая может содержаться в вершинах графа не будет рассматриваться.

Глобально работа может быть разделена на несколько частей:

1. Обзор реализованных подходов и методов выделения сообществ,
2. Поиск способов объединения результатов работы разных алгоритмов с целью улучшить результат,
3. Тестирование и сравнение методов на модельных и реальных данных.

Но прежде чем конкретизировать задачи, разберемся с терминологией.

Отметим, что второй пункт является основной точкой данной работы. Многие методы имеют в своей основе разные подходы. Логично попробовать объединить эти методы, чтобы компенсировать их слабые стороны. Тогда можно надеятся на получение более устойчивого и качественного результата.

Цель работы — найти хороший способ агрегирования результатов разных методов выделения сообществ.

4 Обозначения

1. A — Матрица смежности графа, A_{ij} — (i, j) элемент матрицы;
2. d_i — степень i вершины графа;
3. C_i — метка вершины (номер сообщества, к которому относится вершина);

5 Предметная область

§5.1 Социальный граф

Не любой граф является социальным. Например, существует простая модель Эрдёша–Реньи[2], в которой граф выбирается случайным образом из равномерного распределения на всех графах с числом вершин N.

У него нет какой-либо глобальной структуры. Такой граф не будет соответствовать свойствам, которые предъявляются к социальным графам.

Итак, четкого формального определения какой граф можно называть социальным, нет. Это такой специальный вид графов, который характеризуется некоторым набором свойств [1]. Будем называть граф, для которого они выполняются, социальным.

Вот примерный перечень этих 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, отдельные одиночные вершины, как часто бывает в случае с графами социальных сетей.

   Например, для популярной социальной сети fasebook установлено в [3], что 99.91 % вершин находятся в одной компоненте связности.

2. **Распределение на степенях вершин.** Социальные сети относятся к так называемым безмасштабным сетям (Scale-free network). Это значит, что количество вершин со степенью k убывает как $k^{-\gamma k}$ [4]. На рис. 1 можно увидеть как ведет себя распределение в современных социальных сетях

3. **Среднее расстояние.** Под расстоянием между вершинами будем понимать 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 современном обществе. 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 это факт неоднократно подтверждался на основе данных современных социальных сетей, таких как facebook (рис. 2) [3].

4. **Коэффициент кластеризации.** Существует больше количество коэффициентов, которые могут быть расчитаны на всем графе и показывают некую качественную его характеристику. Обычно эта характеристику сравнивают со ее

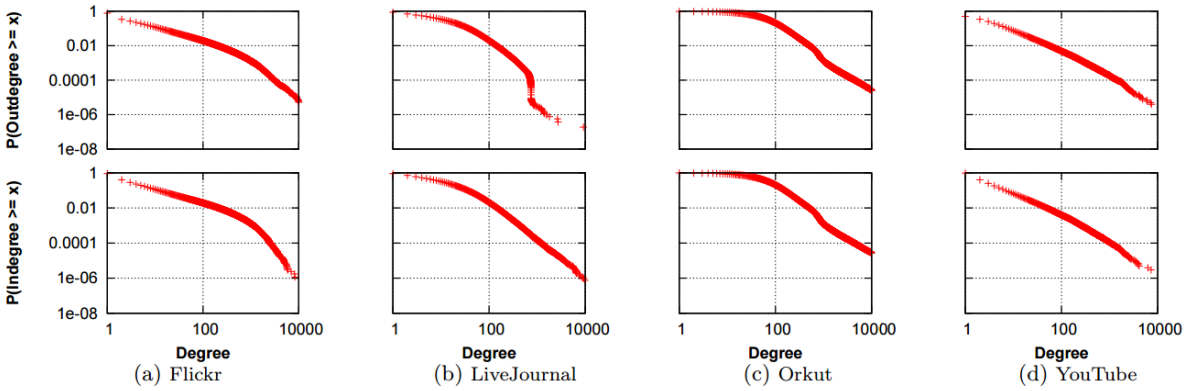

Рис. 1. Исследование распределения степеней вершин в современных социальных сетях, показывает, что они являются безмасштабными. Т.к. современные социальные сети являются направленными графами, то изучают как входящую (снизу) так и исходящую (сверху) степень [4].

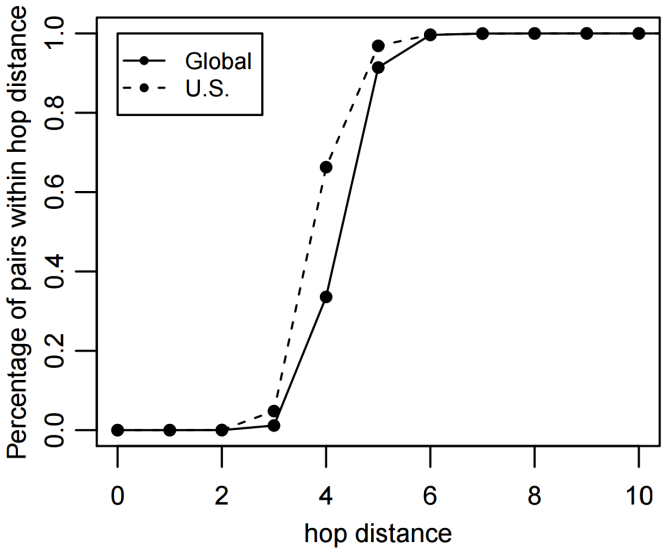

Рис. 2. Изображена функция соседства $N(h)$, которая показывает долю пользователей социальной сети facebook, которые находятся на расстоянии h знакомств от друг друга. Среднее расстояние между пользователями facebook было равно 4.7 [3].

средним значением на случайном графе (например, уже упомянутой модели Эрдёша–Реньи) и выясняется, что она значимо отличается в социальных графах.

Так, данный коэффициент основывается на том факте, что если вершины(люди) A, B и A, C соединены(знакомы) в социальном графе, то с большой вероятностью B и C тоже соединены (знакомы). Если B и C действительно соединены, назовем это закрытым триплетом, если нет, открытым. Коэффициент вычисляется он как доля закрытых триплетов среди всех триплетов и может интерпретироваться как степень кластеризации данного графа (рис. 3). Если коэффициент принимает значения близкие к единице, значит в графе много треугольников и вершины склонны образовывать связь, если соединены

через 3 вершину. Соответственно, если значение близко к нулю, вершины имеют обратную тенденцию. Социальный граф обладает большим коэффициентом кластеризации.

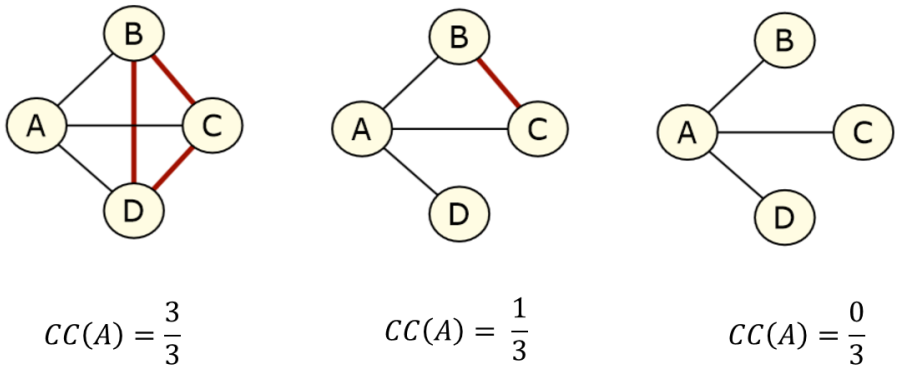

Рис. 3. Коэффициенты кластеризации для трех случаев.

5. **Структура сообществ.** Коэффициент кластеризации показывает насколько сильно вершины склонны образовываться в группы(или сообщества). Вершины входящие в одну и туже группу соединены гораздо плотнее, чем со всем остальным графом.

   Умение выделять сообщества из графа позволяет многое понять про сам граф. Размеры современных графов превышают сотни тысяч вершин и миллионы ребер. Стандартное представление графа теряет свою информативность. Если объединить вершины в сообщества, можно без потери информации об структуре графа исследовать его на абстрактном уровне. На рис. 4 дан пример графов на одних и тех же вершинах. Первый явно имеет 3 сообщества, а второй нет.

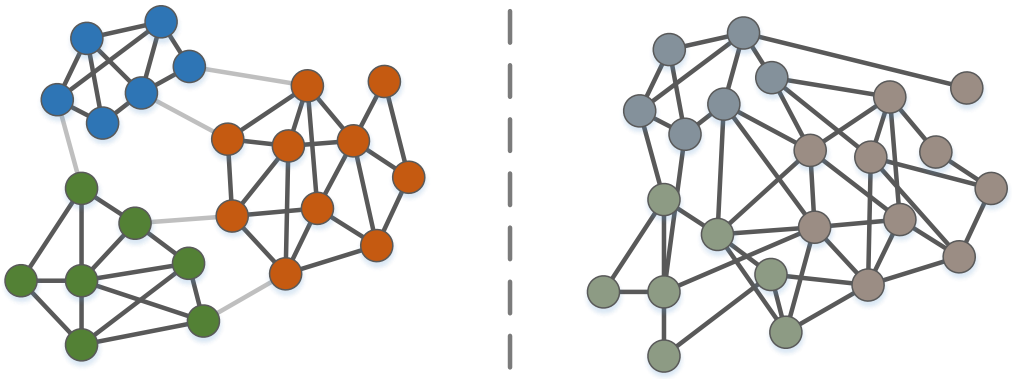

Рис. 4. Пример графа, который имеет структуру из трех сообществ (слева) и графа на тех же вершинах, который ее не имеет (справа).

Этот список не является полным, однако, на нем видна специфичность именно социальных графов, что позволяет отделять их от остальных типов. Далее термины граф и сеть будут употребляться как синонимы.

§5.2 Сообщество

Также как и социальный граф, понятие сообщества сложно формализовать. На концептуальном уровне сообществом называется группа вершин, такая что внутригрупповые связи гораздо плотнее межгрупповых [2], что хорошо видно на рис. 4.

Обычно также делают логичное предположение, что сообщество состоит из одной компоненты связности. В противном случае его можно разделить на несколько по компонентам связности.

В этой работе будет использоваться следующее предположение о структуре сообществ в графе.

Каждая вершина графа входит в одно и только одно сообщество. Вообще говоря, часто рассматривают случай перекрывающихся сообществ, что часто встречается в реальных примерах. Но для упрощения, этот усложненный вариант пока останется в стороне.

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.

Есть третий взгляд на исходную проблему как на задачу кластеризации. Действительно, если ввести некоторую меру расстояния на вершинах графа с учетом структурной информации, можно решать задачу кластеризации. Кстати, такой подход позволяет учитывать дополнительную информацию.

Далее будем говорить о задаче кластеризации или поиска разбиения имея ввиду исходную задачу выделения сообществ в графе.

§5.3 Критерии качества

После того как отработал алгоритм выделения сообществ, необходимо оценить качество получившегося результата. Существуют две принципиально разные ситуации.

В первой мы не знаем истинное разбиение на сообщества. Такая ситуация встречается чаще всего, особенно для графов большого размера и реальных данных. В таком случае для оценки качества используется значение функционала Модулярности [2].

Во второй мы знаем истинное разбиение. Такое возможно в случае модельных данных или для графа знакомств друзей пользователя социальной сети (так называемого эго-графа), где он самостоятельно разметил всех друзей на группы. В таком случае мы можем ввести метрику на разбиениях вершин и посчитать расстояние между истинным и полученным разбиениями. Для таких целей используется split-join distance [5] или нормализованная взаимная информация [2] — информационный критерий для сравнения двух разбиений.

5.3.1 Модулярность

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 [6]. Модулярность — это скалярная величина из отрезка $[-1, 1]$, которая пытается количественно

описать неформальное определение сообщества, данное выше:

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

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.

Формула имеет множество обобщений, например, для взвешенных графов, под A_{ij} понимается вес ребра соединяющий вершины i и j, а $m = \frac{1}{2} \sum a_{ij}$.

Достоинства Модулярности:

Модулярность достаточно просто интерпретируется. Ее значение равно разности между долей ребер внутри сообщества и ожидаемой доли связей, если бы ребра 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

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

где Σ_{in} — сумма весов ребер внутри сообщества C, Σ_{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 [7].

Недостатки:

Функционал не является непрерывным, а задача его оптимизации — дискретная. Соответственно, поиск глобального оптимума является трудной задачей. Поэтому используются приближенные схемы. Некоторые из них действительно оптимизируют значение функционала, другие же по значению модулярности выбирают наилучшее решение из найденных, т.е. нет гарантии того, что это локальный оптимум.

У данного функционала существует проблема разрешающей способности (грубо говоря, функционал не видит маленькие сообщества). Она решается, путем использования модифицированного функционала, который сохраняет все достоинства и добавляет параметр масштаба [8].

5.3.2 split-join distance

Split-join расстояние равняется минимальному количеству операций, необходимое для преобразования одного разбиения в другое. Каждая операция стоит 1 балл:

1. Добавить вершину в существующее сообщество,
2. Удалить вершину из существующего сообщества,
3. Создать новое сообщество с 1 вершиной,

4. Удалить сообщество с 1 вершиной.

Такое расстояние похоже на редакторское и имеет эффективные способы вычисления [5]. Его интерпретация ясна: чем больше сообщества не похожи друг на друга, тем больше преобразований необходимо произвести с одним из них для получения второго. А если алгоритм выделения сообществ отнес в другое сообщество всего пару вершин, то необходимо 4 операции, для того чтобы поправить это.

5.3.3 Нормализованная взаимная информация

В последнее время популярным способом сравнения разных разбиений является подход, основанный на теории информации.

Логично предположить, что если одно разбиение похоже на другое, то необходимо малое количество информации, чтобы восстановить одно из разбиений по другому. Это количество можно интерпретировать как меру непохожести разбиений.

Рассмотрим два разбиения $\{x_i\}$ и $\{y_i\}$, где i это номер вершины, а x_i и y_i метки сообществ из разбиений. Предполагается, что метки x и y являются значениями двух случайных величин X и Y, которые имеют совместное распределение $P(x, y) = = P(X = x, Y = y) = \frac{n_{xy}}{n}$, где n — общее количество вершин, n_{xy} - количество вершин, которым в разбиениях $\{x_i\}$ и $\{y_i\}$ сопоставлены метки x и y. Аналогично $P(X = x) = \frac{n_x}{n}$, $P(Y = y) = \frac{n_y}{n}$. Тогда общая информация определяется как

$$I(X, Y) = H(X) - H(X|Y),$$

где $H(X)$ — энтропия распределения X, а $H(X|Y)$ — условная энтропия:

$$H(X) = -\sum_x P(x) \log x, \quad H(X|Y) = -\sum_{x,y} P(x, y) \log P(x|y).$$

А в качестве меры различия используют нормализованную общую информацию (normalized mutual information):

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

I_{norm} равен единице, когда разбиения одинаковые, и нулю, если они независимы. Для того, чтобы использовать эту величину, как меру сходства, достаточно вычитать ее значение из единицы: $1 - I_{norm}$.

6 Методы выделения сообществ

В ходе работы использовались многие популярные методы выделения сообществ. Их реализация была найдена в библиотеке **igraph** [9] для языка программирования Python 2.7. Перечислим их с указанием основных идей метода, после чего дадим более полное и подробное описание их работы. Алгоритмы будем обозначать следующими ключевыми словами:

1. **Betweenness** — алгоритм на основе коэффициента "центральности по посредничеству" (Betweenness), определяемый как количество кратчайших путей между всеми парами вершин, проходящих через данное ребро [6].

2. **Fastgreedy** — метод, основанный на жадной оптимизации функции модулярности. [10]

3. **Multilevel** — метод, основанный на оптимизации функции модулярности, используя эвристику, предложенную в статье [7].

4. **LabelPropogation** — метод, основанный на присвоении меток к каждой вершине. Каждый раз выбирается метка с максимальной встречаемостью среди смежных вершин [11].

5. **Walktrap** — подход, основанный на случайном блуждании. Использует идею о том, что короткие случайные блуждания не приводят к выходу из текущего сообщества [12].

6. **Infomap** — метод случайного блуждания, основанный на понятии информационных потоков в сетях, кодирования и сжатия информации [13].

7. **Eigenvector** — метод, основанный на собственных векторах матрицы модулярности, которая получается из матрицы смежности [14].

8. **OptModularity** — точная оптимизация функционала модулярности. Применима только для небольших сетей.

Опишем базовые принципы работы данных методов подробнее.

§6.1 Betweenness

Метод, разработанный 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 несколько N кратчайших путей, то каждому ребру достается $\frac{1}{N}$. 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 (рис. 5). Отметим, что данный алгоритм имеет множество модификаций, которые сводятся к подсчету других реберных коэффициентов, либо замены модулярности другим сходим функционалом.

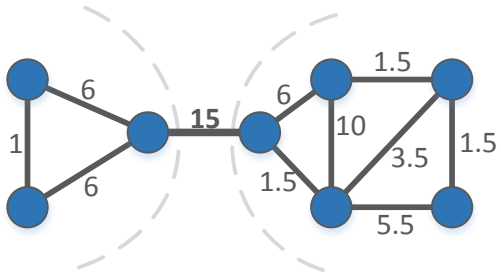

Рис. 5. Граф, для ребер которого подсчитаны значения коэффициентов betweenness. Если удалить ребро с самым большим значением (15), то граф разделится на 2 компоненты связности, которые можно рассматривать как 2 сообщества в исходном графе.

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 $O(m^2n)$. А значит этот метод не подходит для графов с большим количеством вершин. Действительно, в ходе работы не на всех тестах удалось применить этот метод из-за его скорости.

§6.2 Fastgreedy

Данный метод заключается в жадной оптимизации модулярности. 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.

Обычно на втором шаге перебираются не все пары сообществ, а только те, между вершинами которых существуют связи.

Данный алгоритм является быстрым и вычислительно простым (время около линейного), что позволяет применять его для больших графов, таких как граф интернет-ресурсов. Так же в него легко добавить априорную информацию о составе кластеров. Например, если мы знаем, что какие-то конкретные вершины должны лежать в одном кластере.

Однако, метод содержит в себе все недостатки свойственные жадным методам и сходится не к самому лучшему решению. Часто метод порождает 1 большое сообщество, с большинством вершин графа в нем, и множество маленьких [10]. Сложность метода $O(mn)$.

§6.3 Multilevel

Более продвинутая схема оптимизации того же функционала, которая, судя по экспериментам, выигрывает в качестве у алгоритма жадной оптимизации. Алгоритм состоит из двух этапов:

1. Инициализация сообществ в каждой вершине.
2. **Первый Этап**
   (а) Для каждой вершины рассматриваем изменение модулярности при перемещении данной вершины в сообщество вершины-соседа. Выбираем самое выгодное перемещение, если оно увеличивает значение функционала.
   (б) Выполнять первый шаг до тех пор, пока дальнейшее увеличение значения функционала невозможно.
   (в) В случае если никаких перемещении вершин не произошло (т.е. все вершины лежат в своих собственных кластерах) — выход.
3. **Второй Этап**
   (а) Создать новый граф с метавершинами в виде найденных сообществ и ребрами с суммарным весом всех ребер, идущих от одного сообщества к другому (так же появятся петли с суммарным весами связей внутри сообщества). Такой граф будем называть метаграфом.
   (б) Перезапустить Алгоритм на новом графе.

Общая схема алгоритма представлена на рис. 6.

В ходе данной процедуры Модулярность увеличивается. Т.е. в тот момент, когда не произошло объединение вершин на первом этапе достигнут максимум функционала. Все те исходные вершины, которые входят в финальную метавершину принадлежат одному кластеру.

По словам авторов метода алгоритму требуется немного (3-4) перезапусков для выхода. При чем основная работа происходит на первом этапе с исходным графом.

Так же отметим, что алгоритм зависит от порядка перебора вершин на его первом этапе. Эксперименты авторов метода позволяют говорить о том, что порядок не сильно влияет на результат работы метода (если точнее то на значение функционала), но может значительно влиять на время вычисления.

Как уже было сказано, данный алгоритм быстрый и позволяет размечать сети огромного размера. За 152 минуты алгоритм выделил сообщества в веб-базе за 2001 год с сотней миллионов вершин и миллиардом связей. При сравнении данного метода с другими, они просто не справились с таким объемом за сутки [7].

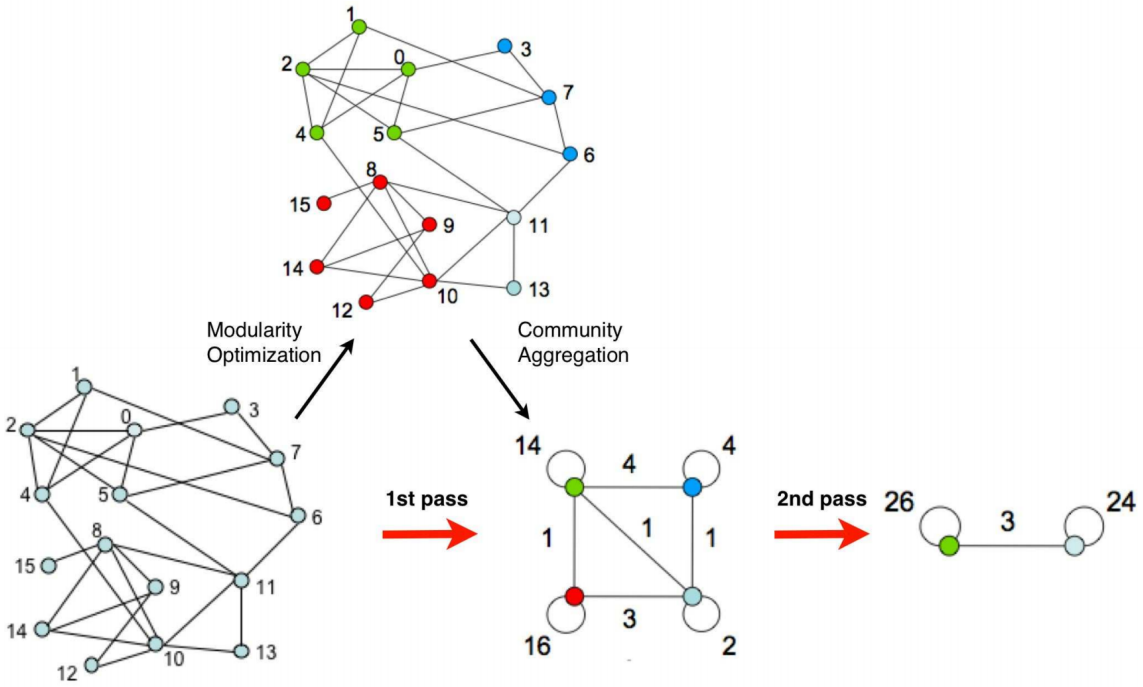

Рис. 6. Общая схема работы алгоритма **Multilevel**. Показано 2 итерации алгоритма. На первой итерации показаны оба этапа: Оптимизация функционала модулярности и создание метаграфа. [7].

§6.4 LabelPropogation

Метод, основанный на эвристике, что вершина относится к тому сообществу, что и большинство ее соседей. У метода почти линейная сложность. Инициализация такая же как и в предыдущих методах: каждой вершине по своему кластеру. Далее, перебирая все вершины, мы переопределяем ее кластер, как кластер, в который входит большинство соседей вершины. Так мы поступаем до тех пор, пока происходят изменения.

Очень важно в данном алгоритме порядок перебора вершин. Для того, чтобы нейтрализовать зависимость от порядка, на каждой итерации алгоритма выбирается новый случайный порядок перебора.

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.

Метод хорош своей простотой и интуитивностью. Также алгоритм является вычислительно эффективным. Однако, метод склонен выдавать разные результаты и является в некотором смысле неустойчивым. На практике и модельных данных это заметно. В сложных зашумленных графах метод может объединить все сообщества в одно сообщество. Данный метод работает в среднем хуже остальных.

§6.5 Walktrap

Для того, чтобы объяснить принцип работы данного метода, сначала опишем что такое дискретный процесс случайного блуждания на графе G. На каждом шаге процесса блуждающий объект находится в вершине и перемещается в другую вершину, выбранную случайным равновероятным образом из соседних вершин. Последовательность посещенных вершин является марковской цепью, состояния которой являются вершинами графа. На каждом шаге вероятность перехода от вершины i к вершине j равна $P_{ij} = \frac{A_{ij}}{d_i}$, где d_i — степень i вершины. Таким образом определется матрица перехода P процесса случайного блуждания. Через данную матрицу можно получить вероятность перехода из вершины i в вершину j за t шагов как $(P^t)_{ij}$. Далее скобки в записи будем опускать подразумевая именно это выражение. Если обозначить за D матрицу, на диагонали которой стоят степени вершин d_i, то $P = D^{-1}A$

С помощью данного процесса происходит сравнение похожести вершин. Т.е. на вершинах вводится метрика, такая, что расстояние между вершинами велико, если вершины из разных сообществ и мало, если из одинаковых.

Метрика вводится следующим образом:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{\left(P^t_{ik} - P^t_{jk}\right)^2}{d_k}} = \left\| D^{-\frac{1}{2}} P^t_{i\bullet} - D^{-\frac{1}{2}} P^t_{j\bullet} \right\|,$$

где $\|\cdot\|$ — евклидова норма в $\mathbb{R}^n$. Вообще, метрика зависит от t, но для упрощения записи будем использовать введенное обозначение. В статье использовались $t \in \{2, 5\}$.

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

$$P^t_{Cj} = \frac{1}{|C|} \sum_{i \in C} P^t_{ij}$$

Обозначим за $P^t_{C\bullet}$ вектор из вероятностей P^t_{Cj}, тогда для двух сообществ $C_1, C_2 \subset V$ расстояние $r_{C_1C_2}$ определяется как

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

Данное определение является обобщением предыдущего.

$r_{C_1C_2}$ связана со спектральными свойствами матрицы P.

Теперь, когда задана метрика задача выделения сообществ сведена к задаче кластеризации вершин. Используется подход основанный на методе Варда [15].

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 вершинами.

3. на каждом шаге k:

(а) выбрать два сообщества C_1 и C_2 из $\mathcal{P}_k$ по критерию, основанному на метрике, который будет описан ниже.

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

(в) Обновить расстояния между сообществами (вообще это необходимо 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.

Этап (а) алгоритма осуществляется по следующему правилу. 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:

$$\Delta\sigma(C_1, C_2) = \frac{1}{n}\left(\sum_{i\in C_3} r_{iC_3}^2 - \sum_{i\in C_1} r_{iC_1}^2 - \sum_{i\in C_2} r_{iC_2}^2\right).$$

Таким образом пытаемся жадным способом решить NP-трудную задачу “K-Median clustering problem”.

Теперь осталось только выбрать наилучшее разбиение $\mathcal{P}_k$, максимизирующее модулярность.

Однако, авторы предложили еще один способ, специфичный для данного метода, о котором можно прочитать в [12].

Сложность такого метода: $O(n^2 \log n)$. Алгоритм себя хорошо проявил на тестах.

§6.6 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.

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 кода 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

$$L(M) = q \curvearrowright H(\mathcal{C}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(C_i),$$

где $q \curvearrowright$ — вероятность покинуть какое либо сообщество на любом шаге, $H(\mathcal{C})$ — энтропия кодов сообществ, $H(C_i)$ — энтропия кодов внутри сообщества C_i, вес $p^i_{\circlearrowright}$ 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 так получается, что при увеличении зашумленности графа метод объединяет все вершины в одно сообщество. Однако, на реальных данных метод работает на уровне всех остальных.

На рис. 7 показана принципиальная схема работы данного метода.

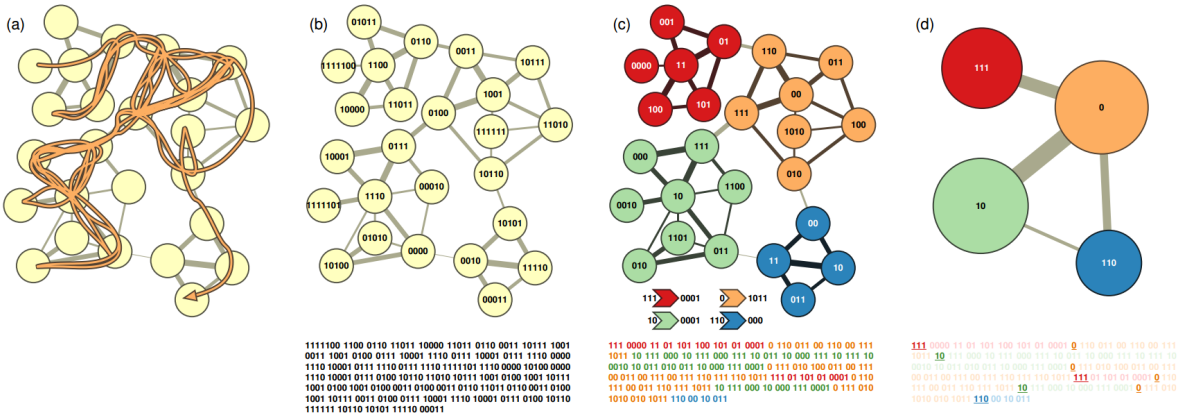

Рис. 7. Принципиальная схема работы метода **infomap** [13]. На самом левом рисунке показан путь случайного блуждателя. На второй части изображены вершины с кодами Хафмана, а ниже закодирован путь в левого изображения. на следующей части показано кодирование с помощью предложенного двухуровневого метода. Ниже записаны коды сообществ и коды выхода их них. А еще ниже закодирован путь. Видно, что длина кода стала меньше. На последнем рисунке показана выделенная структура сообществ в графе и коды, соответствующие группам. Ниже выделены коды, касающиеся групп.

§6.7 Eigenvector

Спектральный метод, основанный на собственных векторах матрицы модулярности, которая определяется следующим образом:

$$B = A - \frac{dd^T}{2m},$$

где d — вектор, в i-ой позиции которого стоит степень i-ой вершины d_i.

Для данной матрицы находится первый собственный вектор (с максимальным собственным числом). Для тех вершин, для которых соответствующее значение меньше нуля, принадлежат одному сообществу, больше нуля — другому.

Подобным образом возможно разделение на большее количество сообществ.

7 Методы агрегирования результатов

В данном параграфе будут описаны основные подходы, с помощью которых возможно объединить результаты работы нескольких методов выделения сообществ. Далее будут приведены конкретные методы, которые будут протестированны на модельных и реальных данных. Будем считать, что метод можно назвать “хорошим”, если он не ухудшает лучший результат среди агрегируемых методов.

§7.1 Агрегирование кластеризацией

В результате работы n алгоритмов по выделению сообществ каждой вершине можно сопоставить набор чисел $(t_1, \dots, t_n)$, которые соответствуют номерам сообществ, к которым отнесли алгоритмы данную вершину. Тогда задача превращается в задачу кластеризации с категориальными признаками, где можно использовать любой подходящий метод.

§7.2 Агрегирование базовыми методами

Реализация всех методов в библиотеке **igraph** предполагает возможность запустить их на взвешенных графах с петлями.

Напомним, что выделение сообществ можно рассматривать как задачу поиска оптимального разбиения на множестве вершин. Тогда каждый из алгоритмов, результаты которых объединяются выдает свое разбиение $T^i = \{T^i_1, \dots, T^i_{m_i}\}$, $(i = = 1, \dots, n)$. Рассмотрим новое разбиение $T = \cap_i T^i = \{T_1, \dots T_m\}$ в виде измельчения полученных разбиений. Т.е. вершины в таком разбиении принадлежат одному и тому же сообществу, только если они принадлежат одному и тому же сообществу во всех разбиениях T^i.

Грубо говоря, мы построили новую структуру сообществ, которая объединяет вершины с большой вероятностью лежащие действительно в одном классе. Теперь для T построим метаграф, как это делается в методе **multilevel**. Напомним, что метаграфом называется граф, вершины которого являются сообществами (метавершинами), а вес ребра равен сумме весов всех ребер, идущих от одного сообщества к другому. Запустим любой обычный метод для выделения сообществ и получим новое разбиение на вершинах-сообществах $\mathbb{T} = \{\mathbb{T}_1, \dots, \mathbb{T}_k\}$. Тогда скажем, что вершины x и y из одного сообщества, если соответствующие метавершины оказались в одной группе: $x \in T_j$, $y \in T_{j'}$ и $T_j \in \mathbb{T}_l$, $T_{j'} \in \mathbb{T}_l$.

На рис. 8 продемонстрированна описанная схема работы метода.

Таким образом предложенная схема является универсальной для объединения результатов с помощью существующих методов.

Отметим, что способ формирования нового разбиения может быть модифицирован: принадлежат одному и тому же сообществу, если они принадлежат одному и тому же сообществу в большинстве исходных разбиений.

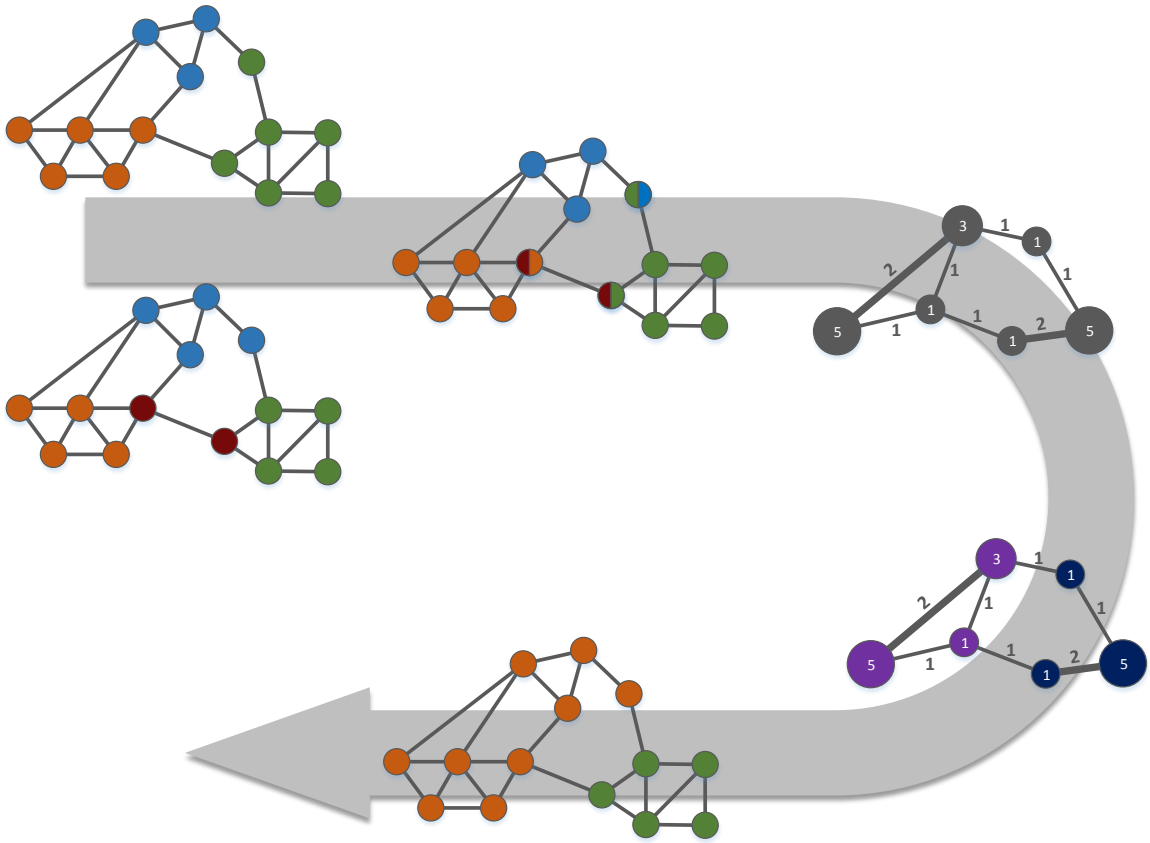

Рис. 8. Принципиальная схема работы метода агрегирования с помощью базовых методов. В начале имеется несколько разбиений. Из них делается новое путем пересечения всех исходных. На его основе строится метаграф (цифрами и размером обозначен вес петли у данной вершины), после чего запускается алгоритм выделения сообществ. В заключительном этапе необходимо вернуться к исходному графу.

8 Тестирование методов

В данном параграфе будут описаны данные на которых далее будет производиться тестирование методов выделения сообществ и агрегирования их результатов.

§8.1 Модельные Данные

8.1.1 l-partition benchmark

Самая простая модель графа, который имеет структуру сообществ [2]. Модель (Planted l-partition model) генерирует граф из $n = g \cdot l$ вершин с l сообществами по g вершин в каждом. Вероятность того, что две вершины из одной группы будут соединены друг с другом p_{in}. Если вершины из разных групп — p_{out}. Таким образом, подграф каждого из сообществ определяется моделью Эрдёша–Реньи с вероятностью появления ребра $p = p_{in}$. Если $p_{in} > p_{out}$, то плотность ребер внутри группы меньше, чем вне его и граф имеет структуру сообществ.

На этой модели основывается тест Гирвана-Ньюмана, уже ставший классическим со следующими параметрами: $l = 4, g = 32$ со средней степенью вершин 16, также предполагается, что $p_{in} + 3p_{out} \approx 0.5$.

Такой подход имеет ряд недостатков.

Т.к. граф по сути является объединением случайных графов он не имеет многих свойств социального графа, которые описывались ранее. Например, распределения степеней его вершин не является экспоненциальным. Все сообщества одинакового размера. Эти факторы могут сказаться на результатах тестирования методов.

Однако, модель легко интерпретируется и требует сложных вычислений.

Далее для краткости такую модель генерации данных будет обозначаться как l-partition. В экспериментах не будем использовать соотношение $p_{in} + 3p_{out} \approx 0.5$, для того, чтобы варьировать степень выраженности структуры графа.

8.1.2 Lancichinetti benchmark

Существует более сложная модель, которая позволяет избавится от проблем, которые возникают у l-partition [16]. Модель определяется следующим набором параметров:

1. Количество вершин,
2. Средняя и максимальная степень вершин,
3. Степень экспоненциального распределения степеней вершин,
4. Степень экспоненциального распределения размеров сообществ,
5. Параметр смешивания. Доля инцидентных данной вершине вершин из других сообществ,
6. Минимальный и максимальный размер сообществ.

Алгоритм генерации:

1. Для каждой вершины определяется ее степень d исходя из экспоненциального распределения. Для того, чтобы соединить вершины и оставить те же степени используется специальная конфигурационная модель [16].
2. Каждая вершина тратит $d(1-\mu)$ связей на соединения с вершинами из того же сообщества. И $d(\mu)$ связей на соединение с остальными вершинами.
3. Размеры сообществ генерируются из соответствующего экспоненциального распределения таким образом, чтобы их сумма давала количество вершин N.
4. Сначала все вершины не имеют привязки к какому-либо сообществу. На первой итерации вершина приписывается случайному сообществу. Если размер сообщества превышает внутреннюю степень вершины (количество вершин из того же сообщества, соединенных с данной), то вершина включается в сообщество, иначе нет. В последнем случае вершина приписывается случайному сообществу. Если оно оказалось полным, одна случановыбранная вершина исключается из сообщества. Процесс останавливается, когда не остается бездомных вершин.

5. Для того, чтобы соблюсти степень смешивания сообществ μ производится поправка.

Такая модель является обобщением l-partition и с помощью нее можно сгенерировать тест Гирвана-Ньюмана. С помощью такой модели можно сгенерировать граф по свойствам близким к социальному. На рис. 9 в левом верхнем углу можно увидеть пример простого графа, сгенерированного с помощью этой модели.

§8.2 Зашумление данных

Для того, чтобы сравнивать методы на тестовых данных недостаточно одной генерации графа с заданными параметрами. Хочется увидеть настолько устойчиво метод распознает сообщества в зависимости от выраженности структуры графа.

Можно сгенерировать большое количество графов для каждого значения параметра, и усреднить получаемое качество. Однако, из-за того что графы могут сильно отличатся друг от друга придется генерировать и обсчитывать очень большое количество графов, что может занимать много времени.

Предлагается альтернативный подход.

Для любой модели можно определить p_{in} как вероятность того, что 2 случайные вершины из одного сообщества соединены и p_{out}, как вероятность связи двух вершин из разных сообществ. В случае модели l-partition, это просто параметры метода. Структура сообществ тем более выражена, чем меньше отношение $\gamma = \frac{p_{out}}{p_{in}}$.

Тогда, для любого графа можно добавлять ребра между вершинами из разных сообществ с вероятностью p_{step}, тем самым увеличивая значения γ на $\frac{p_{step}}{p_{in}}$. Зашумляя исходный граф таким способом, можно контролируемым образом изменять степень выраженности структуры сообществ в графе без генерации нового графа.

Тогда, сгенерировав много графов с выраженной структурой и постепенно зашумляя их описанным способом, можно получить более качественные результаты сравнения за меньшее число генераций данных.

На рис. 9 можно увидеть результат работы такого подхода.

§8.3 Реальные Данные

Реальные данные были взяты из соревнования Learning Social Circles in Networks на платформе kaggle [17]. Это 110 так называемых эго-графов социальной сети, которые строятся следующим образом. Рассматриваются все друзья конкретного пользователя как вершины графа. Часть из них знакома друг с другом и тогда между ними есть ребро. Т.е. его-граф является графом знакомств друзей конкретного человека.

Такие графы являются социальными. На рис. 10 можно увидеть примеры реальных данных.

Для графов дана частичная разметка на пересекающиеся сообщества, по данным которые предоставил пользователь, однако, так как информация не полная, она не будет использоваться.

Абсолютно аналогичный набор данных был собран из социальной сети “В Контакте”. В него включены 357 эгосетей друзей в соцсети автора данной работы.

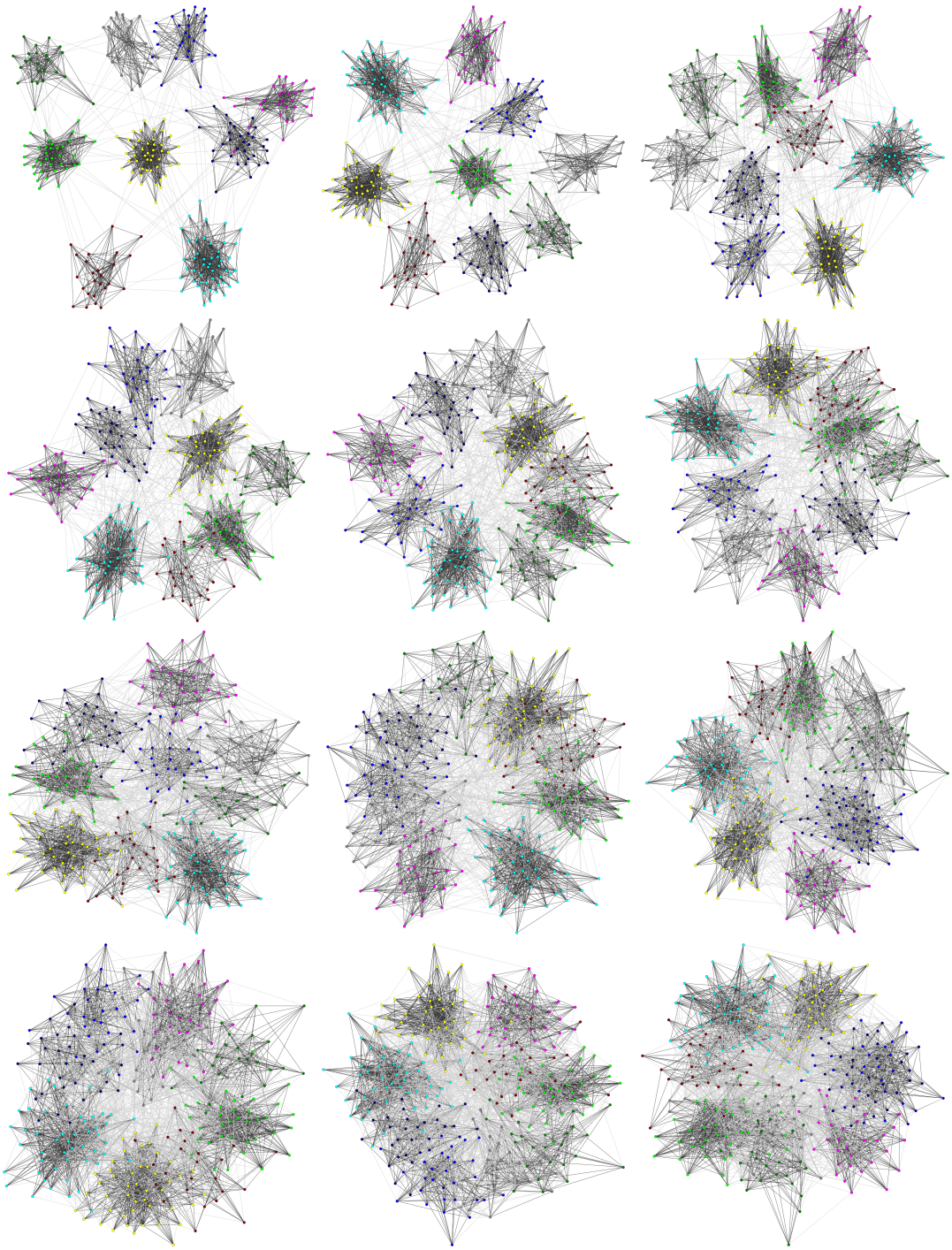

Рис. 9. Пример зашумления тестовых данных. Граф с ярковыраженной структурой сообществ (в верхнем левом углу) пошагово зашумляется до графа с почти отсутствующей структурой (в нижнем правом углу). p_{out} увеличивается в каждой строчке слева направо и на всем рисунке сверху вниз. Первый граф сгенерирован с помощью модели Lancichinetti benchmark.

9 Эксперименты

Итак, есть 3 тестовых набора данных:

1. Тест Гирвана-Ньюмана на основе модели l-partition,

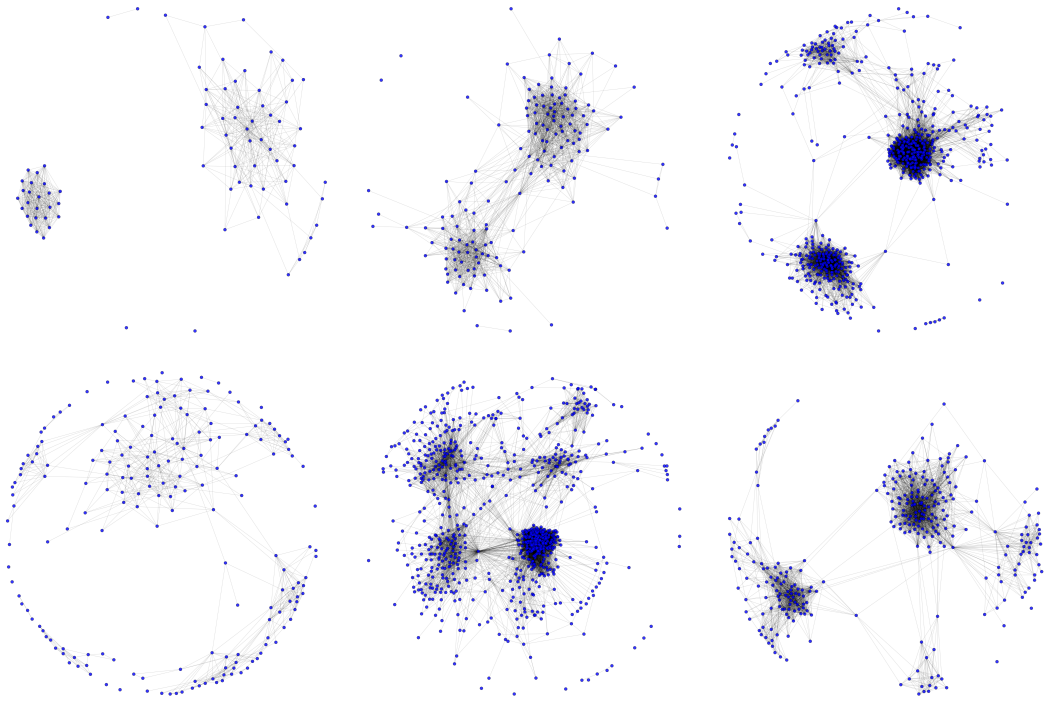

Рис. 10. Примеры тестовых данных: эго-графов из данных для соревнования Learning Social Circles in Networks на платформе kaggle.

2. Тест Lancichinetti,

3. Тест на реальных данных, 110 эго-сетях из совернования Learning Social Circles in Networks.

4. Тест на реальных данных, 357 эго-сетях социальной сети “В Контакте”.

Для модельных данных известна искомая разметка. Поэтому предлагается следить за такими функционалами как split-join distance (sjd) и 1−normalized mutal information (1-nmi), а также использовать зашумление для того, чтобы получить зависимость функционала от параметра γ, как описано в пункте “Зашумление данных”. Для того, чтобы получить более гладкие графики будем усреднять по нескольким запускам.

Для реальных данных, единственное, что возможно сделать, это запустить все методы выделения сообществ и исследовать распределение модулярности на них. Понять дает ли прирост функционалу описанные методы агрегирования.

§9.1 Тест Гирвана-Ньюмана

Для начала запустим тест на всех простых методах для того, чтобы сравнить методы и выбрать те, которые выдают адекватный результат: возможно, часть методов можно исключить из дальнейшего анализа, так как их результат не достаточно хорош остностительно остальных методов.

Напомним, что если два разбиения совпадают, то 1−nmi и sjd равны нулю. Чем значения функционалов больше, тем больше несоответствие между разбиениями. На рис. 11 видно, что с некоторого момента 2 метода **infomap** и **labelprop** выходят к

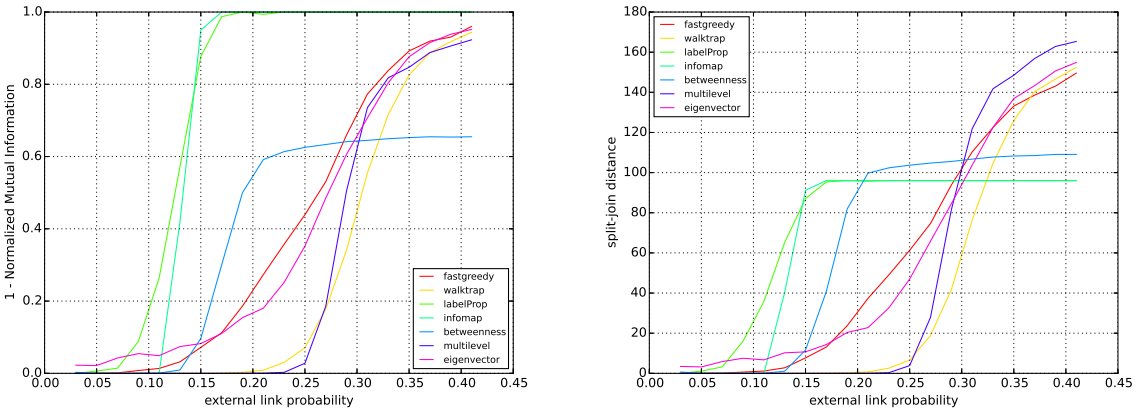

Рис. 11. Зависимость функционалов 1−nmi (слева) и sjd (справа) от вероятности соединенности вершин из разных сообществ для всех базовых методов выделения сообществ. Усреднение по 100 запускам на тесте Гирвана-Ньюмана. $p_{in} = 0.5$

единице для 1−nmi и на константу для sjd. Это происходит в тот момент, когда методы объединяют все вершины в одно сообщество. Такое поведение характерно для этих методов на модельных данных, поэтому в дальнейшем они не будут рассматриваться. Метод **betweenness** имеет похожее поведение (с некоторого момента он выделяет очень большое количество маленьких сообществ) и обладает большой вычислительной сложностью, особенно с ростом количества ребер в графе, поэтому тоже не будет учавствовать в дальнейших экспериментах.

Наиболее важная область для функционалов является средний диапазон значений параметров: 0.15 — 0.30. При меньших значениях структура сообществ в графе сильно выряжена и не составляет труда ее выделить, а при больших значения $p_{in} \approx p_{out}$ граф начинает терять свою структуру, задача выделения сообществ становится бессмысленной.

Лучше всего с задачей справляются методы **multilevel** и **walktrap**.

Посмотрим на то, улучшают ли получаемый результат предложенные методы агрегирования результатов базовых методов на рис. 12. Конечно, в первую очередь интересуют такие методы, которые не ухудшают изначальный результат для лучшего метода из тех, результаты которых объединяются. На интересующем нас диапазоне 0.15 — 0.30, методы **aggMultilevel** и **aggFastgreedy** показывают в среднем лучший результат, чем базовые методы.

Отметим, что в ходе работы проводились эксперименты со всеми базовыми методами, однако, качество их агрегации оставляло желать лучшего. В работе продемонстрированны самые лучшие методы.

Что касается метода **aggClust** (в дальнейшем также упоминается **aggAggClust**), то его результат похож на два других метода. Отметим, что такой способ оказался очень чувствителен к выбору метрики, самого метода кластеризации и количества объединяемых алгоритмов. В данном случае используется метод Agglomerative Clustering с евклидовой метрикой над бинаризованными признаками. Также многим методам необходимо задавать количество кластеров. В таких случаях бралась медиана от количества сообществ в исходных разбиениях.

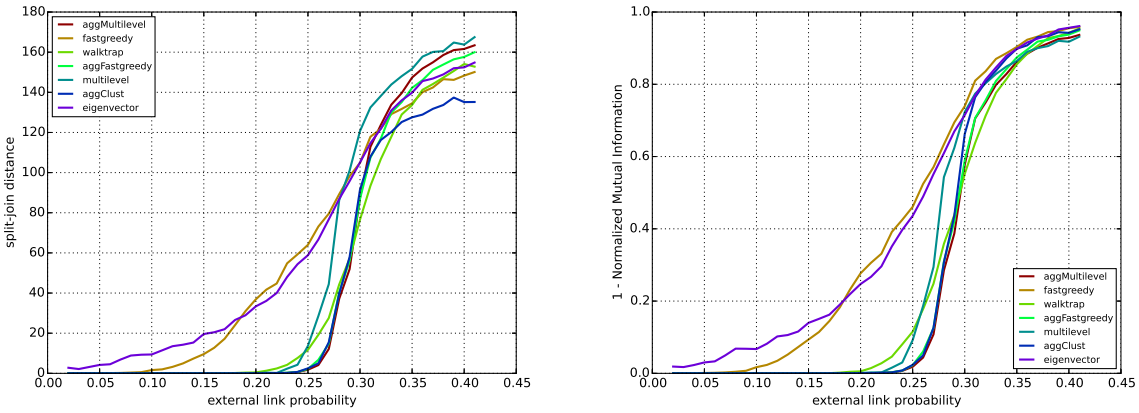

Рис. 12. Зависимость функционалов 1−nmi (слева) и sjd (справа) от вероятности соединенности вершин из разных сообществ для базовых методов и их объединений с помощью кластеризации (**aggClust**) и методов **multilevel** (**aggMultilevel**) и **fastgreedy** (**aggFastgreedy**). Усреднение по 100 запускам на тесте Гирвана-Ньюмана. $p_{in} = 0.5$

§9.2 Тест Lancichinetti

Запустим тест Lancichinetti по такой же схеме с параметрами:

- количество вершин — 1000,
- средняя степень вершины — 40,
- максимальная степень — 100,
- степень в экспоненциальном распределении для степеней вершин — 2,
- степень в экспоненциальном распределении для размеров сообществ — 2,
- минимальный размер сообщества — 30,
- максимальный размер сообщества — 100.

На рис. 13 продемонстрирован график зависимости значения функционала от вероятности соединения вершин из разных сообществ.

Видно, что принципиальные выводы, которые можно было сделать по тесту Гирвана-Ньюмана не изменились. Однако, из-за того, что тест обладает более сложной структурой изменилось поведение методов с увеличением вероятности соединения вершин из разных сообществ. Особенно заметны отличия для метода **fastgreedy**, но результат, который получен с помощью метода **aggFastgreedy** является лучшим вместе с **aggMultilevel**. Метод, который использует алгоритм кластеризации **aggAggClust** немного уступает этим двум методам, что понятно, так как кластеризация никак не связана и не оптимизирует целевой функционал в отличие от двух лидирующих методов.

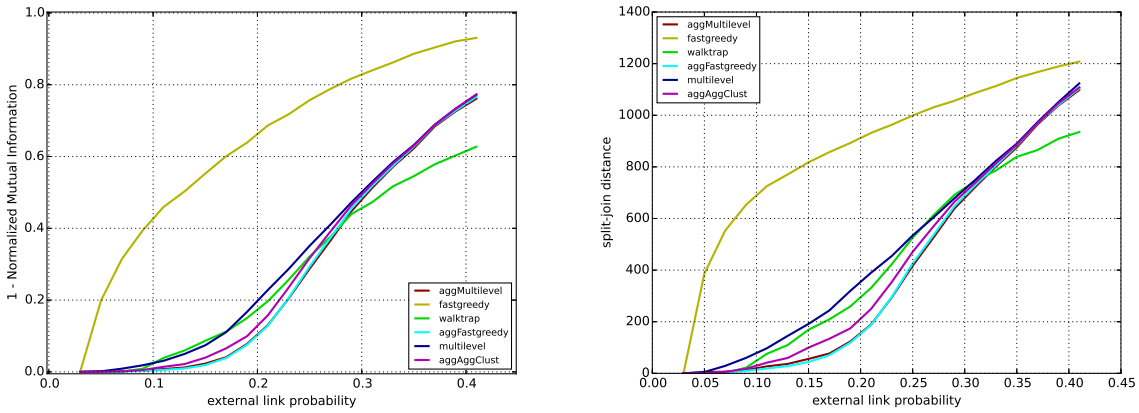

Рис. 13. Зависимость функционалов 1−nmi (слева) и sjd (справа) от вероятности соединенности вершин из разных сообществ для базовых методов и их объединений с помощью кластеризации (**aggAggClust**) и методов **multilevel** (**aggMultilevel**) и **fastgreedy** (**aggFastgreedy**). Усреднение по 100 запускам на тесте Lancichinetti с описанными в тексте параметрами. $p_{in} = 0.5$

§9.3 Тест на реальных данных

Запустим все методы на реальных данных и посмотрим на распределение значений модулярности для каждого метода. К сожалению, уже не возможно производить зашумление данных, поскольку неизвестна истинная разметка сообществ.

Результаты работы методов представлены на рис. 14. Ось икс не несет никакой информативной нагрузки и введена для того, чтобы лучше отобразить результаты. Не представленные методы либо слишком долго обсчитывались, либо не сходились. Отметим, что метод, который плохо себя зарекомендовал для модельных данных (**infomap**), справился не хуже остальных методов на реальном наборе данных. Также видно, что с точки зрения модулярности, сами данные являются достаточно разнородными.

На рис. 15 показан аналогичный график для методов агрегирования результатов. На вертикальной оси теперь изображена разность между максимальным значением модулярности среди простых методов и полученным объединением их результатов. Для примера приведен еще один метод агрегации результатов с помощью кластеризации методом k-means. Видно, что часто метод ухудшает лучший результат.

На рис. 16 даны гистограммы для этих методов. Видно, что часто разница в полученных решениях нулевая, это значит, что метод повторил лучший из имеющихся результатов.

Отметим, что на практике, из-за того что модулярность считается без учета какой-либо информации о разметке, можно запустить любой из четырех методов, и если он выдал результат хуже, чем уже имеется у базовых методов, то в качестве конечной разметки выбрать разметку с максимальной модулярностью.

В среднем методы дают следующие значения в приросте качества:

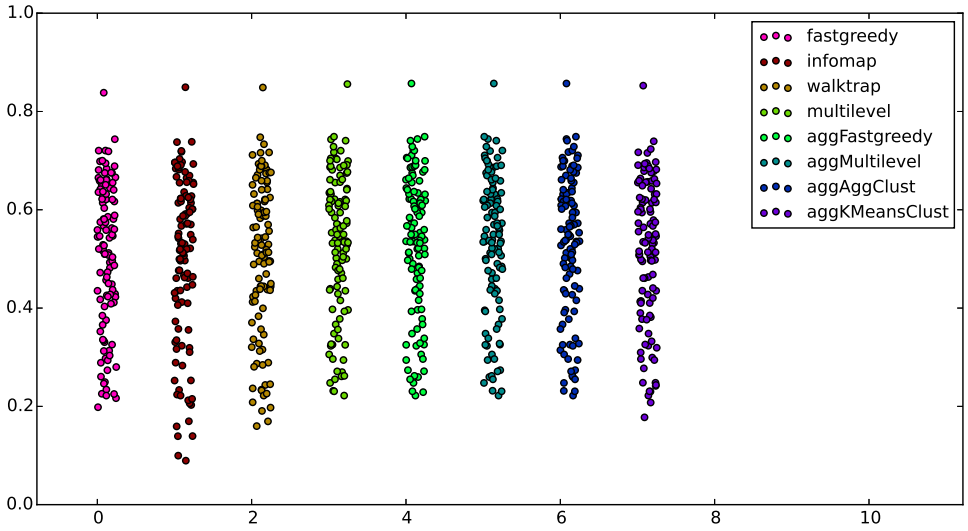

Рис. 14. Значение модулярности для разных базовых методов выделения сообществ для каждого эго-графа. Ось икс введена искуственным образом для того, чтобы разнести результаты разных методов на разных данных.

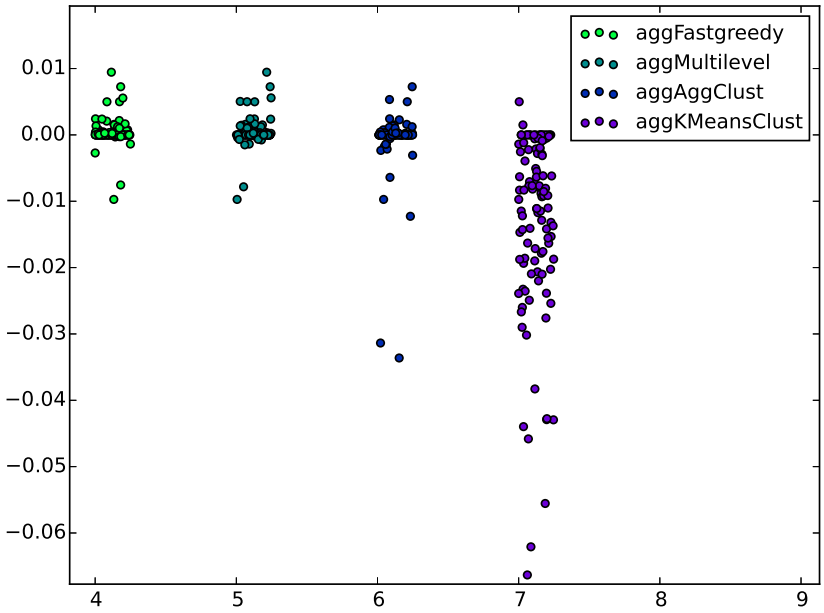

Рис. 15. Значение разности между максимальным значением модулярности среди простых методов и полученным объединением их результатов для каждого эго-графа. Ось икс введена искуственным образом для того, чтобы разнести результаты разных методов на разных данных.

Метод	средний прирост	среднее для максимума из 0 и прироста
aggFastgreedy	0.000336	0.00053
aggMultilevel	**0.000364**	**0.00057**
aggAggClust	-0.000629	0.00037
aggKMeansClust	-0.014295	0.00006

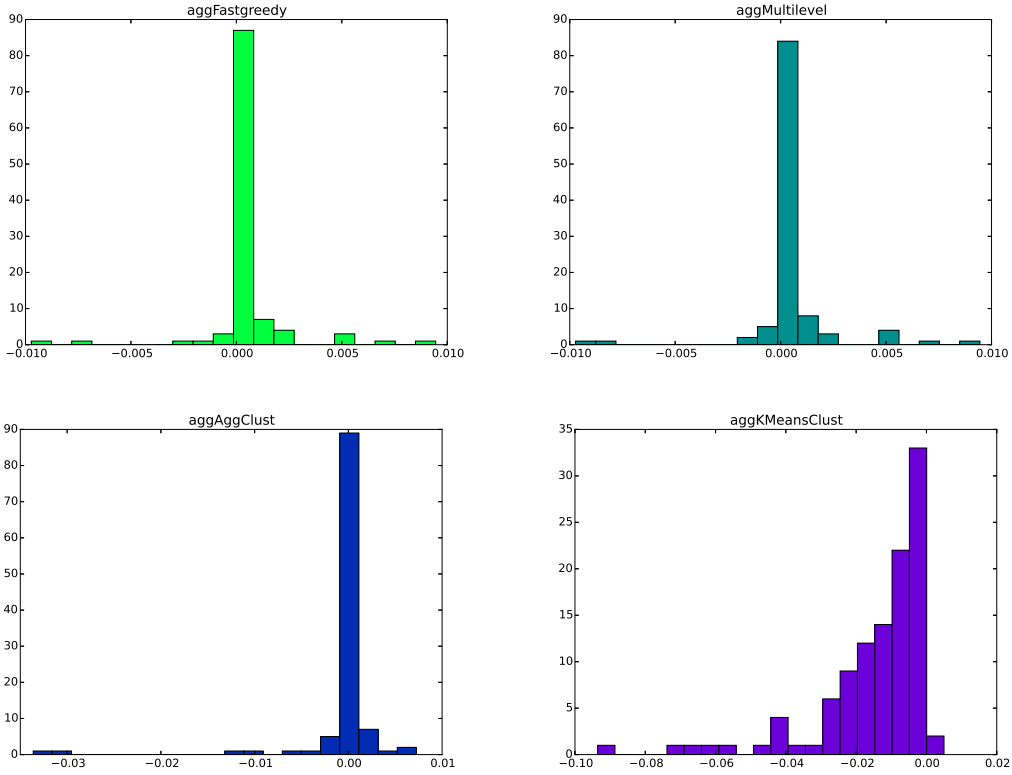

Рис. 16. Гистограммы для 4 методов агрегации для значения разности между максимальным значением модулярности среди простых методов и полученным объединением их результатов для каждого эго-графа.

В третьем столбце указан средний прирост для такого выбора разбиения: если метод выдал результат хуже, чем уже был получен, то он заменяется на лучший. То есть отрицательный прирост становится нулевым.

По значениям в таблице можно сделать вывод, что методы кластеризации в среднем не улучшают значение функционала.

Самым эффективным методов оказался метод **aggMultilevel**.

Отметим, что в ходе экспериментов было выявлено, что большее количество базовых методов ухудшает выдаваемый результат методами агрегирования базовыми алгоритмами, и улучшает для кластеризационного подхода. Например, если объединять результаты трех методов **fastgreedy**, **multilevel** и **walktrap** получатся следующая таблица для прироста:

Метод	средний прирост	среднее для максимума из 0 и прироста
aggFastgreedy	0.000214	**0.00036**
aggMultilevel	**0.000250**	0.00034
aggAggClust	0.000153	0.00027
aggKMeansClust	-0.012697	0.00007

Обратим внимание, что метод **aggAggClust** в среднем стал выдавать положительный прирост.

§9.4 Тест на реальных данных 2

Проведем абсолютно аналогичное испытание методов на втором наборе данных рис 17, 18, 19. Видно, что общее поведение методов осталось неизменным. Лучшими методами можно считать **aggFastgreedy** и **aggMultilevel**.

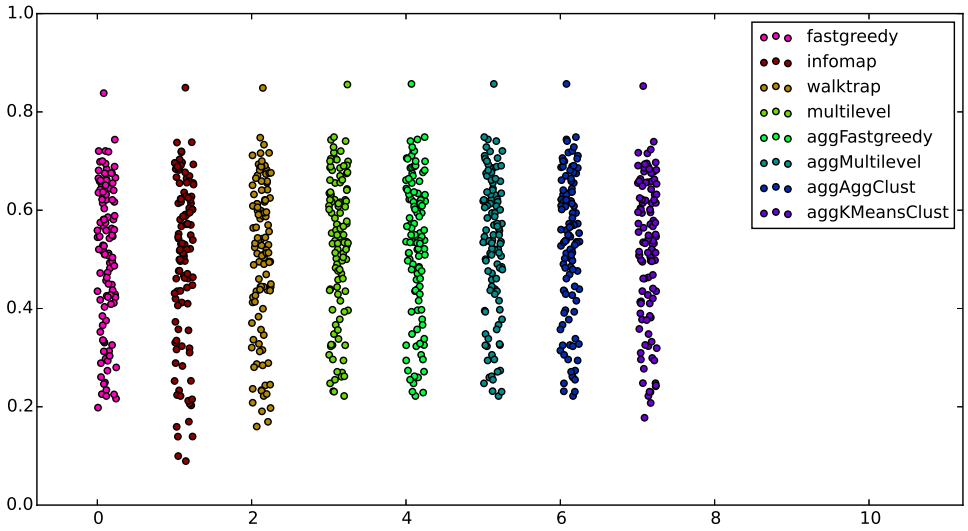

Рис. 17. Значение модулярности для разных базовых методов выделения сообществ для каждого эго-графа. Ось икс введена искуственным образом для того, чтобы разнести результаты разных методов на разных данных. Данные из социальной сети “В Контакте”.

Метод	средний прирост	среднее для максимума из 0 и прироста
aggFastgreedy	0.000699	0.000809
aggMultilevel	0.000719	0.000801
aggAggClust	-0.000733	0.000734
aggKMeansClust	-0.019532	0.00008

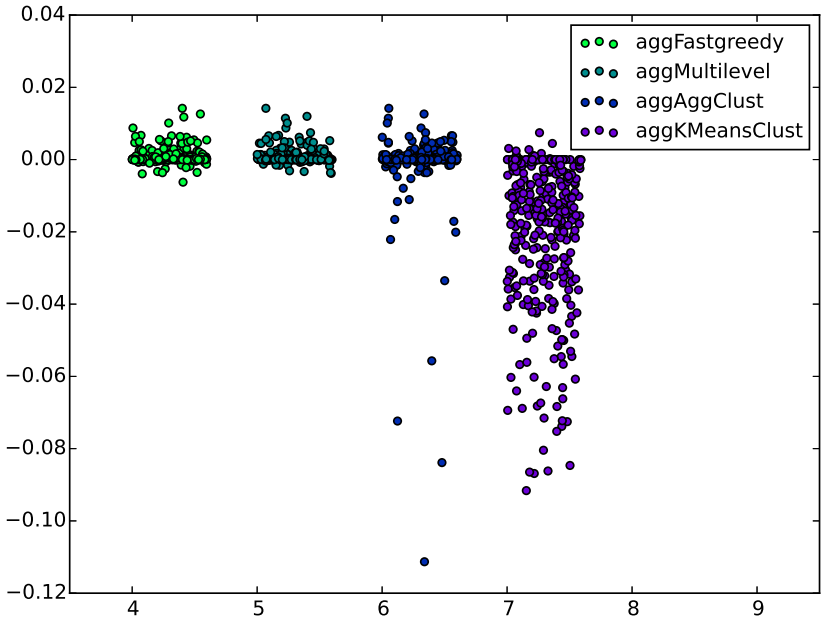

Рис. 18. Значение разности между максимальным значением модулярности среди простых методов и полученным объединением их результатов для каждого эго-графа. Ось икс введена искуственным образом для того, чтобы разнести результаты разных методов на разных данных. Данные из социальной сети “В Контакте”.

10 Выводы и результаты

В ходе работы были изучены и описаны основные базовые методы для выделения сообществ из социального графа. Было представлено два подхода к объединению результатов работы базовых методов для улучшения получаемой разметки. По результатам анализа подходов были предложены конкретные реализации методов. Методы были проверены на разных данных, как модельных, так и реальных действительно продемонстрировали улучшение получаемого качества.

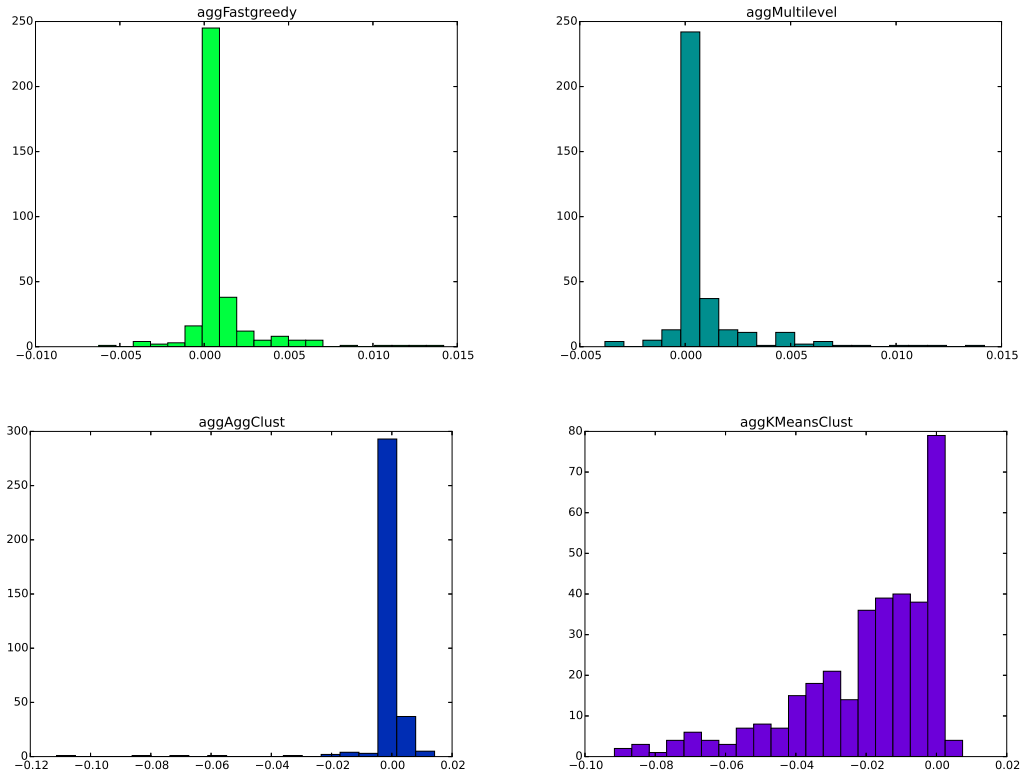

Рис. 19. Гистограммы для 4 методов агрегации для значения разности между максимальным значением модулярности среди простых методов и полученным объединением их результатов для каждого эго-графа. Данные из социальной сети “В Контакте”.

11 Список литературы

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

[2] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. http://www.arxiv.org/abs/0906.0612.

[3] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011. http://arxiv.org/abs/1111.4503.

[4] Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM, 2007.

[5] Stijn Dongen. Performance criteria for graph clustering and markov cluster experiments. 2000. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.9783&rep=rep1&type=pdf.

[6] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[7] 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(10):P10008, 2008. http://www.arxiv.org/abs/0803.0476.

[8] Erwan Le Martelot and Chris Hankin. Fast multi-scale detection of relevant communities in large-scale networks. *The Computer Journal*, page bxt002, 2013. http://comjnl.oxfordjournals.org/content/56/9/1136.full.pdf?keytype=ref&ijkey=Eqs2wpLhn8ZDmvA.

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

[10] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004. http://www.arxiv.org/abs/cond-mat/0408187.

[11] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007. http://www.arxiv.org/abs/0709.2938.

[12] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005. http://arxiv.org/abs/physics/0512106v1.

[13] Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. The map equation. *The European Physical Journal-Special Topics*, 178(1):13–23, 2009.

[14] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006. http://arxiv.org/abs/physics/0605087.

[15] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963. http://iv.slis.indiana.edu/sw/data/ward.pdf.

[16] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008. http://www.arxiv.org/abs/0805.4770.

[17] Learning social circles in networks competition, 2014. https://www.kaggle.com/c/learning-social-circles 2015-04-26.