Выделение сообществ в гиперграфах.

Насыров Руслан Рашидович
студент 2 курса,
физтех-школа прикладной математики и информатики,
Московский физико-технический институт(ГУ)
РФ, г. Долгопрудный
E-mail: nasyrov.rr@phystech.edu
Васильева Екатерина Евгеньевна,
Кандидат физ.-мат. наук
Мусатов Даниил Владимирович,
Кандидат физ.-мат. наук, кафедра дискретной математики МФТИ
РФ, Долгопрудный

17 декабря 2021 г.

1 Аннотация

1. Объект исследования: графы и гиперграфы.
2. Цель: написать программу, позволяющую находить сообщества в графах и гиперграфах. Проанализировать полученные кластеризации.
3. Задачи:
   (a) дать определение ключевым понятиям.
   (b) рассмотреть известные методы выделения сообществ.
   (c) рассмотреть обобщение функции модулярности на случай гиперграфов, предложенное в статье [8]
   (d) рассмотреть обобщение Лувенского алгоритма на случай гиперграфов, предложенное в статье [8]
   (e) написать алгоритм, который выделяет сообщества как в графах, так и в гиперграфах.
   (f) проанализировать клсатеризации синтетических графов.
   (g) сделать вывод.

2 Ключевые понятия

Определение 1 *Гиперграф - обобщенный вид графа, в котором каждым ребром могут соединятся не только две вершины, но и любые подмножества вершин. С математической точки зрения, гиперграф H - это пара H=(V, I) , где V - множество вершин, а I семейство подмножеств V, называемых гиперребрами. [10]*

Определение 2 *Сеть со структурой сообщества - такая сеть, узлы которой могут быть легко сгруппированны в кластеры таким образом, чтобы внутри кластеров лежало много ребер, а между ними - мало. [1]*

Определение 3 *(Вероятностное) Сеть со структурой сообщества - такая сеть, узлы которой могут быть легко сгруппированны в кластеры таким образом, чтобы вероятность ребра внутри кластерв была намного больше, чем между. [1]*

Определение 4 *Модульность - это функция, которая измеряет качество конкретного разделения сети на сообщества. Именно от нее зависит качество кластеризации. [5]*

Определение 5 *Кластеризация сети - процесс нахождения структуры сообщества сети. То есть процесс разбиения вершины на кластеры.*

3 Введение

При изучении сложных сетей, таких как социальные, биологические, компьютерные и т.д. одна из важных структурных характеристик - это их сообщества.

Часто нахождение сообществ в графе рассматривают как разбиение графа на части, то есть когда каждая вершина попадает не более чем в одно сообщество. Большинство алгоритмов выделения сообществ работают именно в такой модели.

Но существуют другие определения сообществ, которые допускают, что различные сообщества могут иметь пересечения. Обычно в таких моделях для каждой вершины вводится набор коэффициентов, который отвечает за принадлежность данной вершине различным сообществам. Это лучше соответсовует действительности, так как очевидно, что, в реальных графах каждая вершина может принадлежать более, чем одному сообществу.

Здесь для упрощения будет рассмотрен только случай не перекрывающихся кластеров.

3.1 Актуальность

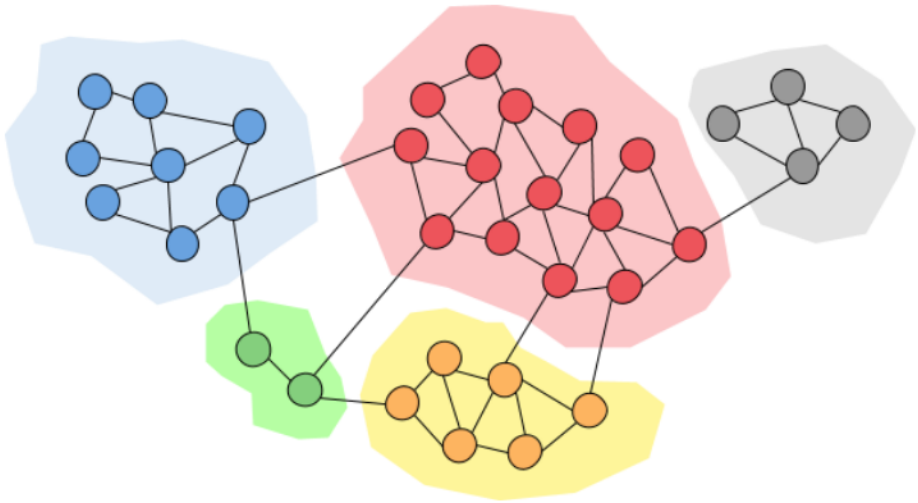

Рис. 1: Сообщества.[16]

Структуры сообществ встречаются почти во всех реальных сетях. Например, в социальных сетях это могут быть люди, живущие в одном районе, или дети, ходящие в одну школу. Поиск структуры сообществ таких сетей существенно упрощает их анализ, так как позволяет перейти от рассмотрения точечных взаимодействий между парами вершин к взаимодействиям между сообществами (кластерами). Такой подход применим, так как вершины, состоящие в одном кластере с большой вероятностью обладают одинаковыми наборами характеристик и нет смысла рассматривать несколько похожих вершин по отдельности.

Алгоритмы выделения сообществ применяются для анализа сетей взаимодействия белков. В них сообщества соответствуют белкам с аналогичной функциональностью внутри биологической клетки. А этот анализ уже применяется для создания лекарств.

Аналогично, сети цитирования формируют сообщества по темам исследований, а сети общения - выделяют круг общения человека.

Одна из причин, по которым выделение сообществ важно состоит в том, что сообщества часто имеют совсем другие свойства, чем средние свойства сетей. Поэтому анализ лишь средних характеристик упускает такой момент, как например существование замкнутых групп общения или малообщительных групп.

Еще одно применение - анализ распространения слухов и эпидемий. Последнее стало особенно актуальным в связи с новой коронавирусной инфекцией. Поскольку анализ сетей общения людей позволит разработать грамотные действия по предотвращению эпидемии. Так как отчетливо будет видно, в каких регионах и с какой вероятностью будет определенное количество заболевших, что позволит вводить точечные ограничения, которые урегулируют ситуацию.

Еще одним из важных приложений является прогнозирование недостающих ребер и выявление ложных ребер в сети. Так как в процессе сбора информации могут встречаться ошибки, то

при небольшом их количестве с помощью выделения сообществ из легко выявить и устранить. Например, единственное ребро между никак не связанными между собой кластерами.

4 Алгоритмы выделения сообществ в графе.

Поиск сообществ в произвольной сети может быть вычислительно сложной задачей. Количество сообществ, как правило, неизвестно, и сообщества часто имеют неодинаковый размер. Однако, несмотря на эти трудности, было разработано и применено несколько методов поиска сообщества с разной степенью успеха.

4.1 Метод минимального разреза.[6]

Это - один из старейших методов выделения сообществ. Он заключается в том, сеть разрезается на заранее определенное количество частей, обычно примерно равных по размеру. Эти чести выбираются таким образом, чтобы количество ребер между ними было минимально.

Метод хорошо работает во многих приложениях, но не идеален для поиска структуры сообщества в общих сетях, поскольку он будет находить сообщества независимо от того, подразумеваются ли они в структуре, и он найдет только фиксированное их количество.

4.2 Иерархическая кластеризация. [2]

В этом методе определяется мера сходства, количественно определяющая некоторый тип сходства между парами узлов. Обычно используемые показатели включают косинусное сходство, индекс Джаккарда(размер пересечения деленный на размер объединения), используется для выявления похожести между двумя конечными множествами и расстояние Хэмминга между строками матрицы смежности. Затем в соответствии с этой мерой аналогичные узлы группируются в сообщества.

Существует несколько распространенных схем для выполнения группировки, простейшим из которых являются кластеризация с одной связью. Она основана на группировании кластеров по принципу "снизу вверх".

Алгоритм кластеризции следующий:

1. Изначально все вершины находятся в своих собственных кластерах.
2. На каждом шаге объединяются два кластера, которые содержат ближайшую пару элементов, еще не принадлежащих к одному кластеру.

4.3 Алгоритм Гирвана-Ньюмана. [7]

Совершенно другая идея представлена в **алгоритме Гирвана-Ньюмана:**

1. Изначально все вершины назодятся в одном общем кластере.
2. На каждом шаге удаляются ребра, которые "скорее всего"находятся между сообществами.

Здесь вводится понятие центральностьи ребра как доля кратчайших путей между парами узлов, которые проходят вдоль него. Если сеть содержит сообщества или группы, которые слабо связаны несколькими межгрупповыми ребрами, то все кратчайшие пути между различными сообществами должны проходить по одному из этих немногих ребер. Таким образом, ребра, соединяющие сообщества, будут иметь высокую центральность.

Из удаление отделяет кластеры друг от друга и раскрывает структуру сообщества сети.

Алгоритм Гирвана–Ньюмана популярен, поскольку он реализован в ряде стандартных программных пакетов. Но он также работает медленно, занимая время $O(m^2n)$ в сети из n вершин и m ребер, что делает его непрактичным для сетей с более чем несколькими тысячами узлов.

4.4 Алгоритмы максимизации модулярности. [4]

Одним из наиболее широко используемых методов обнаружения сообществ является максимизация модульности.

Он основан на совершенно другой идее по сравнению с предыдущими алгоритмами. Очевидно, если мы хотим найти лучшее разбиение на сообщества, нам надо уметь их сравнивать. Поэтому вводится понятие модулярности разбиения.

В статье [4] предлагается слудующее легко интерпретируемое определение:
Модулярность - это сумма разностей долей ребер внутри кластера минус ожидаемая доля ребер внутри кластера. И предлагается слудующая формула для ее вычисления:

$$Q = \frac{1}{2m}\sum_{i,j}(A_{i,j} - \frac{k_i k_j}{2m})\delta(C_i, C_j).$$

Где:
A - матрица смежности графа. $A_{i,j}$—это вес ребра между вершинами i, j или 0, если ребра нет.
$m = \frac{1}{2}\sum_{i,j} A_{i,j}$ - общее количество ребер.
$k_i = \sum_j A_{i,j}$ - степень вершины i.
C_i - номера сообществ вершин.
$\delta(C_i, C_j)$ - символ Кронекера.

Чтобы это понять, почему формула такова, введем понятие *случайного графа с заданными степенями вершин*. Этот граф образуется в ходе следующего процесса:

1. Рассматривается какой-то граф с заданными степенями вершин.
2. Все его ребра рубятся пополам таким образом, формируя "обрубки".
3. Каждый "обрубок"случайо соединяется с любым другим, формируя ребро случайного графа.

Таким образом, мы получаем граф с заданными степенями вершин, который, тем не менее, случаен.

Теперь покажем интерпретацию формулы: Введем случайные величины $I_i^{v,w}$ - индикатор того, что i-ый "обрубок"вершины v соединился с одним из k_w "обрубков"вершины w.
Тогда $E[I_i^{v,w}] = \frac{k_w}{2m-1}$.

Следовательно, количество ребер между v и w: $J_{v,w} = \sum_{i=1}^{k_v} I_i^{v,w}$.
Тогда $E[J_{v,w}] = \frac{k_v k_w}{2m-1}$. При больших m 2m-1 заменяют просто на 2m.

Тогда доля ребер внутри сообществ: $\frac{1}{2m}\sum_{i,j} A_{i,j}\delta(C_i, C_j)$ а матожижание доли ребер внутри сообществ: $\frac{1}{2m}\sum_{i,j}\frac{k_i k_j}{2m})\delta(C_i, C_j)$. Что подтверждает интерпретацию.

Метод максимизации модульности обнаруживает сообщества путем просмотра всех возможных кластеризаций и выбора лучшей по модулярности. Но так как даже на среднего размера графах полный перебор очень долог, используются эвристические алгоритмы. Одним из самых эффективных и часто используемых является Лувенский алгоритм.

4.5 Лувенский алгоритм. [3]

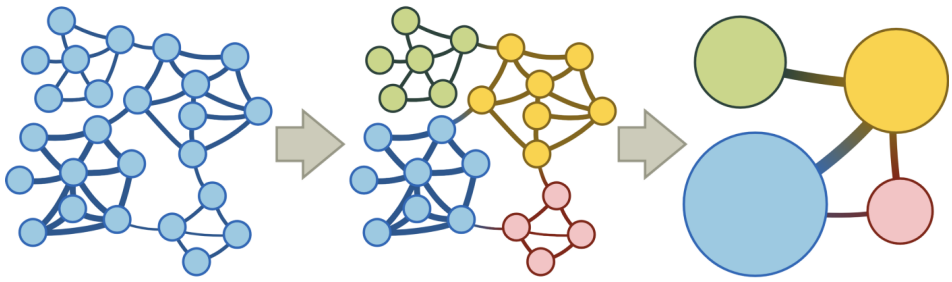

Рис. 2: Привер работы Лувенского алгоритма.[14]

Популярным подходом к максимизации модульности является метод Лувена, который итеративно оптимизирует локальные сообщества до тех пор, пока глобальная модульность больше не сможет быть улучшена с учетом возмущений текущего состояния сообщества. В настоящее время он является лучшим алгоритмом максимизации модульности.

Метод представляет собой метод жадной оптимизации, который, исходя из практических экспериментов, выполняется за время $O(n \log n)$. Значение, которое мы хотим оптимизировать - это модулярность, которая по сути измеряет плотность количества ребер внутри кластеров по сравнению с плотностью в случайном графе.

Обобщение формулы модульности, данной выше на взвешенные ребра - элементарно: просто теперь элементы матрицы A - вещественные неотрицательные числа.

Сам **Лувенский алгоритм** состоит из следующих стадий:

1. Каждая вершина изначально принадлежит своему собственному кластеру.
2. Для каждой вершины v вычисляется изменение модулярности при ее перемещении в соседний кластер (т.е. в тот, который соединен с ней ребром).
3. Если максимум из таких изменений больше 0, то v перемещается с соответствующий кластер.
4. Шаги 2-3 повторяются, пока улучшается значение модулярности.
5. Если увеличение уже невозможно, то создается новая сеть, вершинами которой являются кластеры старой сети, а вес ребра - это сумма весов ребер между кластерами. Важно отметить, что петли допускаются.
6. Далее переход к шагу 2. И так, пока все вершины не объединятся в 1 кластер.

Такой процесс формирования сообществ формирует *дендрограмму.* Важно отметить, что она, наряду с кластеризацией является важной характеристикой структуры сети. Лувенский алгоритм допускает некоторую вольность в порядке рассмотрения вершин. Соответственно, при изменении порядка будут получаться другие дендрограммы. Тогда возникает другая интересная задача связанная с изучением того, насколько произвольные дендрограммы могут получаться.

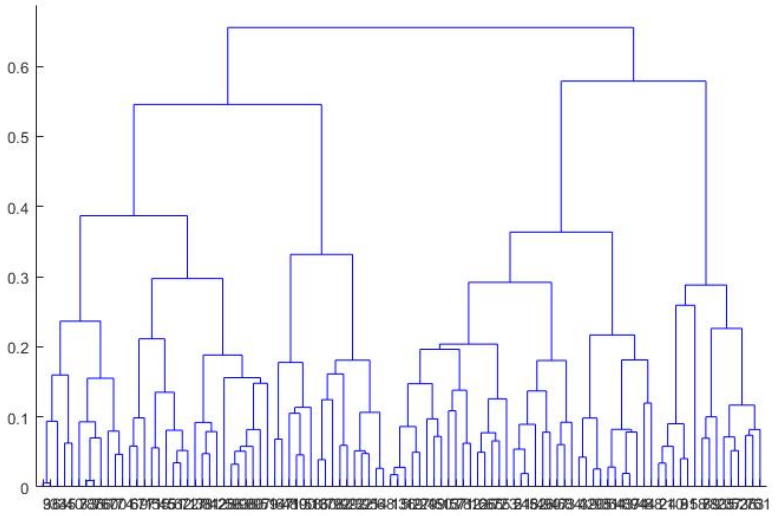

Рис. 3: Дендрограмма.[15]

5 Обобщение Лувенского алгоритма на гиперграфы. [8]

Если мы хотим обобщить Лувенский алгоритм, то надо обобщить и функцию модулярности на случай гиперграфов, а значит нужно научиться их генерировать. В статье [8] предложены следующие обобщения:

5.1 Стохастическая блок-модель гиперграфа с поправкой на степень.

Эта модель генерирует кластеризованные гиперграфы с неоднородными распределениями степеней вершин и размеров гиперребер. Она достаточно гибкая, так как содержит много параметров. Их правильный подбор будет обеспечивать более хорошую кластеризацию графа.

Граф предлагается генерировать следующим образом:

$$a_R = Poisson(\Omega(z_R)\pi(\theta_R)|R|!),$$

где a_R- это случайная величина - вес гиперребра R. Ω - функция, контролирующая вес гиперребра случайного графа в зависимости от текищей кластеризации, $\theta-$ это параметры вершин, каждый параметр контролирует степень конкретной вершины. $\pi(\theta_R) = \prod_{v\in R}\theta_v$. Это имеет следующий смысл: чем больше мы хотим степени вершин, тем больше должна быть степень гиперребра. $|R|!-$ это количество способов упорядочить вершины гиперребра. Интуитивно процесс генерации понимается так: на каждое упорядоченное подмножество вершин кладется гиперребро, вес которого распределен как $Poisson(\Omega(z_R)\pi(\theta_R)$. Тогда сумма |R|! этих случайных величин как раз даст требуемое распределение.

5.1.1 Метод максимального правдопообия.

Теперь, после того как процесс генерации гиперграфа описан, в статье предлагают определить функцию, аналогичную функции модулярносити обычного гиперграфа. Для этого ставится задача максимального правдоподобия:

$$z', \Omega', \theta' = argmax_{z,\Omega,\theta} P(\mathbf{G}|z, \Omega, \theta),$$

где G - гиперграф, который нужно кластеризовать, z - кластеризация, Ω - функция, контролирующая вес гиперребра случайного графа, θ - это параметры вершин, контролирующие их степень.

Так как веса всех ребер - независимые случайные величины, то $P(\mathbf{G}|z, \Omega, \theta) = \prod_{R\in G} P(a_R|z, \Omega, \theta)$. А это можно посчитать легко, так как известно, что веса ребер распределены Пуассоновски. Будем считать логарифм от вероятности (он имеет тот же локальный оптимум). Для упрощения полагаем, что параметры вершин θ - зафиксированы (мы их оптимизировать не будем).

$$L(\mathbf{G}|z, \Omega, \theta) = Q(z, \Omega, \theta) + K(\theta) + C$$

где

$$Q(z, \Omega, \theta) = \sum_{R\in R} a_R \log \Omega(z_R) - |R|!\pi(\theta_R)\Omega(z_R)$$

а K и C не зависят от кластеризации z.

Тогда в статье предлагается взять эту функцию Q за обобщение функции модулярности на гиперграфы.

5.2 Лувенский алгоритм для гиперграфов.

Далее в статье предлагается обобщение хорошо известного Лувенского алгоритма на гиперграфы. Вот описание **одного шага алгоритма:**

1. Перебираются текущие кластеры и для каждого происходит попытка переместить его в один из смежных с ним кластеров. Находится кластер, который максимизирует ΔQ.
2. Если $\Delta Q > 0$, то перемещение происходит, иначе нет.
3. Шаги 1-2 повторяются, пока происходят перемещения кластеров.
4. В результате получена локально оптимальная кластеризация.

Сам алгоритм:

1. Изначально все вершины находятся каждая в своем кластере.

2. Запускаем шаг алгоритма.

3. Если модулярность не увеличилась, заканчиваем. Иначе - повторяем шаг 2.

Важным отличием данного алгоритма от Лувенского является то, что он не сжимает полученный в результате 1 шага кластеры, чтобы не потерять выжную информацию о связях в одном кластере.

6 Нахождение кластеризации гиперграфа.

План исследования:

1. Анализ статьи [8] и написание программы, описанной в ней.
2. Тестирование программы.
3. Написание программы, визуализирующей кластеризацию и позволяющей проводить дальнейший анализ.
4. Изучение полученных кластеризаций графов.

6.1 Написание алгоритма из статьи.

Код получился достаточно объемным. Я приведу только две основные функции. Полная версия разработанного кода доступна здесь [9].

```
def SymmetricHMLLstep(H: Hypergraph, Q, Ω, z):
    '''
        Ω - affinity function (ex: All-or-Nothnig)
        z - current partition vector

    '''
    z = list(z)
    C = list(np.unique(copy(z)))
    new_z = copy(z)
    improving = True

    while improving:
        improving = False
        # все действия происходят отложенно.
        # Я сначала намереваюсь сдвинуть, и только потом реально двигаю.

        for c in C: # цикл по всем кластерам
            Sc = get_nodes_with_cluster(H, z, c) # вершины такого же кластера
            new_C = list(np.unique(new_z)) # новая кластеризация

            Ac = get_adjancted_clusters(H, new_z, Sc)
            assert c not in Ac

            Δ, new_c = -np.inf, -1

            # пробую переместить Sc в кластер с номером c_1
            for c_1 in Ac:
                delta_Q = ΔQ(H, Q, Ω, new_z, Sc, c_1)
                if delta_Q > Δ:
                    Δ = delta_Q
                    new_c = c_1

            if Δ > 0:
                for vertex in Sc:
                    new_z[vertex] = new_c
                improving = True
    return new_z
```

Код 1: 1 шаг алгоритма Лувена.

```
def SymmetricHMLL(H: Hypergraph, Q, Ω):
    n = H.n_vertex()
    z_new = list(range(1, n + 1))
    z = list(range(1, n + 1))
    while True:
        z = copy(z_new)
        z_new = SymmetricHMLLstep(H, Q, Ω, z)

        if z == z_new:
            break
    return z
```

Код 2: Сам Лувенский алгоритм.

6.2 Изучение полученных кластеризаций на обычных графах.

Посмотрим на кластеризацию и модулярность для следующих графов:

1. 2 клики на 10 вершинах, соединенных 1 ребром:

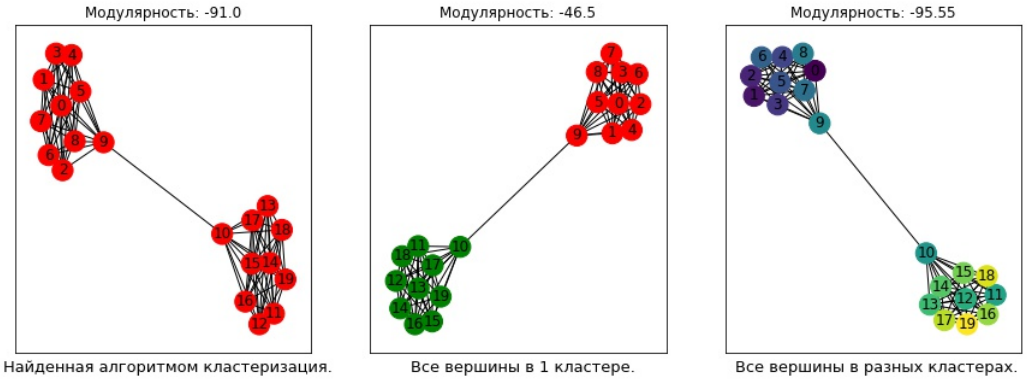

Рис. 4: Различные кластеризации графа 1.

Как видим, алгоритм успешно находит верную кластеризацию, и ее значение модулярности вдвое меньше, чем когда все вершины в 1 кластере или когда все вершины в разных.

2. 4 Клики на 4 вершинах и 10 "зашумляющих ребер"и немного случайных ребер между ними.

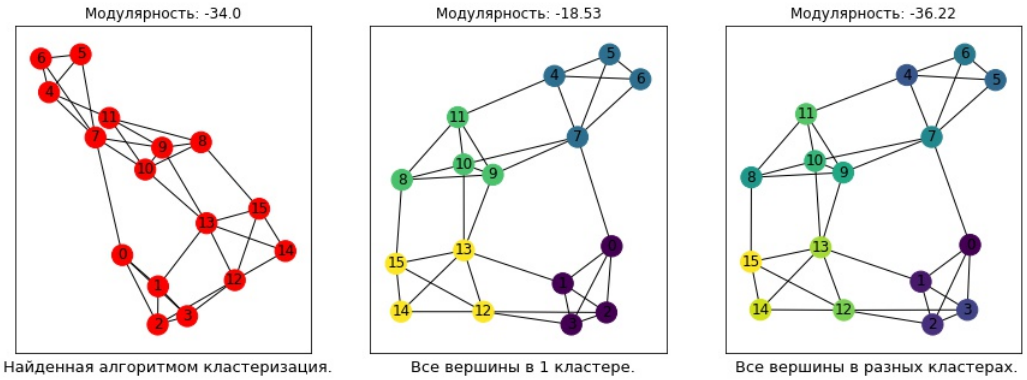

Рис. 5: Различные кластеризации графа 2.

Как видим, алгоритм справился и с этой задачей успешно.

3. 6 клик на 5 вершинах и 40 ребер между ними: А вот здесь уже алгоритм выдает всего 5

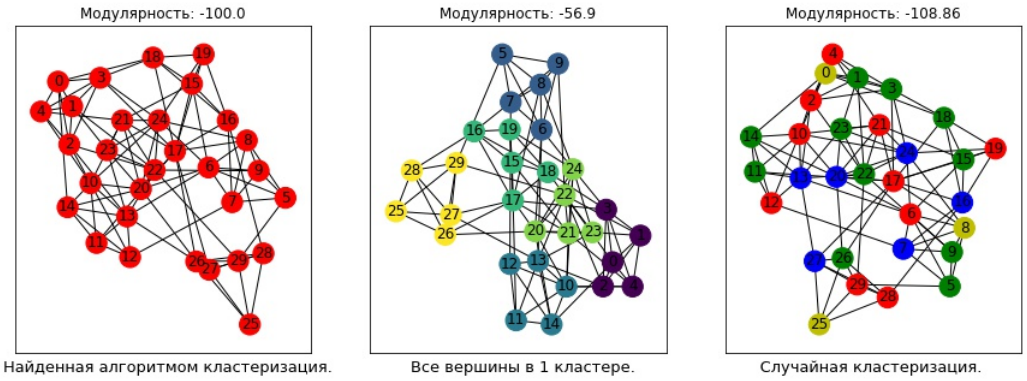

Рис. 6: Различные кластеризации графа 3.

клик. Скорее всего это следствие того, что слишком много "зашумляющих ребер"добавлено. Но все остальные клики выделены правильно.

6.3 Вывод

Алгоритм хорошо работает на обычных графах. Он легко выделяет сообщества даже в зашумленных случаях, а функция модулярности имеет большой скачок при переходе от оптимальной кластеризации к плохой.

7 Итог

В рамках исследования была написана программа позволяющая вычислять кластеризацию произвольного гиперграфа. С её помощью выделены сообщества в синтетических графах на небольшом числе вершин. На основе полученных результатов можно сделать вывод, что алгоритм, предложенный в статье работает хорошо, и его можно применять как для обычных графов, так и для гиперграфов.

Впоследствии в рамках данной темы будут проведены эксперименты по выделению сообществ в гиперграфах и написание более быстрой версии данного алгоритма. Это позволит выделять сообщества намного лучше и применять метод даже на очень больших сетях.

Список литературы

[1] M. Girvan; M. E. J. Newman (2002). "Community structure in social and biological networks". Proc. Natl. Acad. Sci. USA. 99 (12): 7821–7826.

[2] Ahn, Y.-Y.; Bagrow, J.P.; Lehmann, S. (2010). "Link communities reveal multi-scale complexity in networks". Nature. 466 (7307): 761–764.

[3] Blondel, Vincent D; Guillaume, Jean-Loup; Lambiotte, Renaud; Lefebvre, Etienne (9 October 2008). "Fast unfolding of communities in large networks". Journal of Statistical Mechanics: Theory and Experiment. 2008 (10): P10008.

[4] M. E. J. Newman (2004). "Fast algorithm for detecting community structure in networks". Phys. Rev. E. 69 (6): 066133. arXiv:cond-mat/0309508.

[5] Newman, M. E. J. (2006). "Modularity and community structure in networks". Proceedings of the National Academy of Sciences of the United States of America. 103 (23): 8577–8696.

[6] M. E. J. Newman (2004). "Detecting community structure in networks". Eur. Phys. J. B. 38 (2): 321–330. Bibcode:2004EPJB...38..321N. doi:10.1140/epjb/e2004-00124-y.

[7] M. Girvan; M. E. J. Newman (2002). "Community structure in social and biological networks". Proc. Natl. Acad. Sci. USA. 99 (12): 7821–7826

[8] Hypergraph clustering: from blockmodels to modularity Philip S. Chodrow, Nate Veldt, Austin R. Benson

[9] github.com/2001092236/Community-detection

[10] В. А. Емеличев, О. И. Мельников, В. И. Сарванов, Р. И. Тышкевич. Глава XI: Гиперграфы // Лекции по теории графов. — М.: Наука, 1990. — С. 298—315. — 384 с. — ISBN 5-02-013992-0.

[11] matplotlib.org

[12] networkx.org

[13] github.com/mattbierbaum/arxiv-public-datasets

[14] www.machinelearningmastery.ru/generating-twitter-ego-networks-detecting-ego-communities

[15] fin-az.ru

[16] towardsdatascience.com/community-detection-algorithms