



МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М.В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА МАТЕМАТИЧЕСКИХ МЕТОДОВ ПРОГНОЗИРОВАНИЯ

НИКИШИН ЕВГЕНИЙ СЕРГЕЕВИЧ

Методы выделения сообществ в социальных графах

КУРСОВАЯ РАБОТА

Научный руководитель:
д.ф-м.н., профессор
А.Г. Дьяконов

Образец титульного брал [отсюда](#)

Москва, 2016

version 0.04

Содержание

1 Введение (неполное)	1
1.1 Модулярность	1
2 Разбиение на непересекающиеся сообщества	1
2.1 Edge betweenness	1
2.2 Label propagation	2
2.3 FastGreedy	2

1 Введение (неполное)

1.1 Модулярность

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

Testtest [2, 5, 3, 4, 1]

2 Разбиение на непересекающиеся сообщества

2.1 Edge betweenness

Для каждой пары вершин связного графа можно вычислить кратчайший путь, их соединяющий. Будем считать, что каждый такой путь имеет вес, равный $1/N$, где N — число возможных кратчайших путей между выбранной парой вершин. Если такие веса посчитать для всех пар вершин, то каждому ребру можно поставить в соответствие значение Edge betweenness — сумму весов путей, прошедших через это ребро.

Для ясности приведём следующую иллюстрацию:

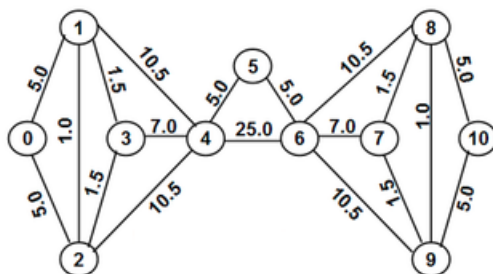


Рис. 1: Граф, для ребёр которого посчитаны значения Edge betweenness

В данном графе хочется выделить два сообщества: "слева" и "справа". Граница же будет проходить через ребро, имеющее максимальный вес, 25. На этой идее и основывается алгоритм: поэтапно удаляем ребра с наибольшим весом, а оставшиеся компоненты связности объявляем сообществами.

Собственно, сам алгоритм:

1. Инициализировать веса
2. Удалить ребро с наибольшим весом
3. Пересчитать веса для ребёр, затронутых удалением
4. Сообществами считаются все компоненты связности
5. Посчитать функционал модулярности (о нём будет сказано ранее)
6. Повторять с шага 2-6, пока есть рёбра

Результатом будет разбиение, на котором модулярность максимальна.

Из необходимости каждый раз пересчитывать веса следует главный недостаток: вычислительная сложность составляет $O(m^2n)$, где m — количество ребёр, n — количество вершин.

2.2 Label propagation

Допустим, что большинство соседей какой-либо вершины принадлежат одному сообществу. Тогда, с высокой вероятностью, ему также будет принадлежать выбранная вершина. На этом предположении и строится алгоритм Label propagation: каждая вершина в графе определяется в то сообщество, которому принадлежит большинство его соседей. Если же таких сообществ несколько, то выбирается случайно одно из них. Пример:

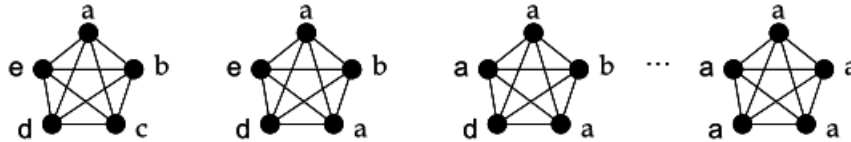


Рис. 2: Демонстрация работы алгоритма для полного графа

В начальный момент времени всем вершинам ставится в соответствие отдельное сообщество. Затем происходят перераспределения сообществ. Из-за случайности важно на каждой итерации изменять порядок обхода вершин. Алгоритм заканчивает работу, когда нечего изменять: все вершины относятся к тем сообществам, что и большинство их соседей. Авторы также советуют запускать несколько раз алгоритм и выбирать наилучшее из результирующих разбиений, либо пересекать их. Главное достоинство данного алгоритма, в противовес предыдущему, — почти линейная сложность. Однако на зашумленных графах зачастую происходит объединение всех вершин в одно сообщество.

2.3 FastGreedy

Алгоритм заключается в жадной оптимизации модулярности. Как и в прошлом методе, в каждой вершине графа инициализируется отдельное сообщество, а затем объединяются пары сообществ, приводящие к максимальному увеличению модулярности. При этом объединяются только инцидентные пары вершин, так как, в противном

случае, модулярность не может увеличиться [во введении необходимо будет объяснить смысл модулярности, чтобы этот факт не вызывал вопросов].

На каждой итерации процесса получается некое разбиение вершин. Последовательность таких разбиений, имеющая вид дерева, в листьях которого находятся сообщества с одной вершиной, а в корне — большое сообщество, содержащее все вершины, называется дендрограммой. Результатом работы алгоритма является ярус дендрограммы (т.е. разбиение), имеющий максимальную модулярность.

Метод является вычислительно нетрудоёмким ($O(m \log n)$), легко применим к большим графам и, несмотря на жадность, зачастую неплохо справляется с задачей.

2do: разобраться с русской кодировкой и правильным оформлением списка литературы

Список литературы

- [1] Clauset, A. Finding community structure in very large networks / Aaron Clauset, M. E. J. Newman, Cristopher Moore // Physical Review E. — 2004. — <http://arxiv.org/abs/cond-mat/0408187>.
- [2] Girvan, M. Community structure in social and biological networks / Michelle Girvan, M. E. J. Newman // Proceedings of the National Academy of Sciences. — 2001. — <http://arxiv.org/abs/cond-mat/0112110>.
- [3] igraph library. — 2016. — <http://igraph.org/python/>.
- [4] Raghavan, U. N. Near linear time algorithm to detect community structures in large-scale networks / Usha Nandini Raghavan, Reka Albert, Soundar Kumara // Physical Review E. — 2007. — <http://arxiv.org/abs/0709.2938>.
- [5] Slavnov, K. A. Social graph analysis. — 2015. — http://www.machinelearning.ru/wiki/images/6/60/2015_417_SlavnovKA.pdf.