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 0.03

Содержание

1 Введение (неполное)

1.1 Модулярность

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

Test [1, 4, 2, 3]

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:

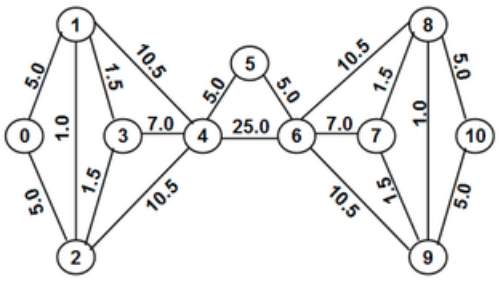

Рис. 1: Граф, для ребёр которого посчитаны значения Edge betweenness

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 шаги 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:

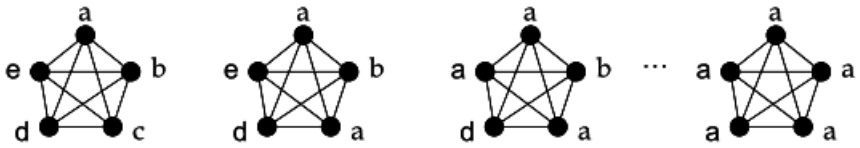

Рис. 2: Демонстрация работы алгоритма для полного графа

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.

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

Список литературы

[1] 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.

[2] igraph library. — 2016. — http://igraph.org/python/.

[3] 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.

[4] Slavnov, K. A. Social graph analysis. — 2015. — http://www.machinelearning.ru/wiki/images/6/60/2015_417_SlavnovKA.pdf.