САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Информатики

Проданов Тимофей Петрович

Адаптивный рандомизированный алгоритм выделения сообществ в графах

Бакалаврская работа

Допущена к защите.
Зав. кафедрой:

Научный руководитель:
д. ф.-м. н., профессор О.Н. Граничин

Рецензент:
В.А. Ерофеева

Санкт-Петербург
2015

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Department of Computer Science

Timofey Prodanov

Adaptive randomised algorithm for community detection in graphs

Bachelor's Thesis

Admitted for defence.
Head of the chair:

Scientific supervisor:
Professor Oleg Granichin

Reviewer:
Victoria Erofeeva

Saint-Petersburg
2015

Оглавление

Введение

1. Предварительные сведения

1.1. Выделение сообществ в графах

Исторически, изучение сетей происходило в рамках теории графов, которая начала своё существование с решения Леонардом Эйлером задачи о кёнигсбергских мостах. В 1920-х взял своё начало анализ социальных сетей и лишь последние двадцать лет развивается изучение *сложных сетей*, то есть сетей с неправильной, сложной структурой, в некоторых случаях рассматривают динамически меняющейся во времени сложные сети. От изучения маленьких сетей внимание переходит к сетям из тысяч или миллионов узлов.

В процессе изучения сложных систем, построенных по реальным системам, оказалось, что распределение степеней $P(s)$, определённое как доля узлов со степенью s среди всех узлов графа, сильно отличается от распределения Пуассона, которое ожидается для случайных графов. Также сети, построенные по реальным системам характеризуются короткими путями между любыми двумя узлами и большим количеством маленьких циклов [1]. Это показывает, что модели, предложенные теорией графов, часто будут оказываться далеко от реальных потребностей.

Современное изучение сложных сетей привнесло значительный вклад в понимание реальных систем. Сложные сети с успехом были применены в таких разных областях, как изучение структуры и топологии интернета [2, 3], эпидемиологии [4], биоинформатике [5], поиске преступников [6], социологии [7] и многих других.

Свойством, присутствующим почти у любой сети, является структура сообществ, разделение узлов сети на разные группы узлов так, чтобы внутри каждой группы соединений между узлами много, а соединений между узлами разных групп мало. Способность находить и анализировать подобные группы предоставляет большие возможности в изучении реальных систем, представленных с помощью сложных сетей. Плотно связанные группы узлов в социальных сетях представляют людей, принадлежащих социальным сообществам, плотно сплочённые группы узлов в интернете соответствуют страницам, посвящённым распространённым темам, а сообщества в генетических сетях связаны с функциональными модулями [1]. Таким образом, выделение сообществ в сети является мощным инструментом для понимания функциональности сети.

1.2. Определения и обозначения

Формально, сложная система может быть представлена с помощью графа. В этой работе будут рассматриваться только невзвешенные неориентированные графы. Неориентированный невзвешенный граф $G = (\mathscr{N}, \mathscr{L})$ состоит из двух множеств — множества $\mathscr{N} \neq \emptyset$, элементы которого называются *узлами* или *вершинами* графа, и мно-

жества $\mathscr{L}$ неупорядоченных пар из множества $\mathscr{N}$, элементы которого называются *рёбрами* или *связями*. Мощности множеств $\mathscr{N}$ и $\mathscr{L}$ равны N и L соответственно.

Подграфом называется граф $G' = (\mathscr{N}', \mathscr{L}')$, где $\mathscr{N}' \subset \mathscr{N}$ и $\mathscr{L}' \subset \mathscr{L}$.

Узел обычно обозначают по его порядковому месту i в множестве $\mathscr{N}$, а ребро, соединяющее пару узлов i и j обозначается l_{ij}. Узлы, между которыми есть ребро называются *смежными*. Степенью узла назовём величину s_i, равную количеству рёбер, выходящих узла i.

Прогулка из узла i в узел j — это последовательность узлов, начинающаяся с узла i и заканчивающаяся узлом j. Путь — это прогулка, в которой каждый узел встречается единожды. Геодезический путь — это кратчайший путь, а количество узлов в нём на один больше геодезического расстояния.

До того, как мы определили понятие *сообщество*, определим *разбиение* на сообщества. Пусть $G = (\mathscr{N}, \mathscr{L})$ — граф, тогда разбиением на сообщества будет называться разбиение множества его вершин $P = \{C_1, \ldots, C_K\}$, то есть $\bigcup_{i=1}^{K} C_i = \mathscr{N}$ и $C_i \cap C_j = \emptyset \ \forall i \neq j \in 1..K$.

Сообщество — это такой подграф, чьи узлы плотно связаны, однако структурная сплочённость узлов можно определить по разному. Одно из определений вводит понятие *клик*. Клик — это максимальный такой подграф, состоящий из трёх и более вершин, каждая из которых связана с каждой другой вершиной из клика. n-клик — это максимальный подграф, в котором самое большое геодезическое расстояние между любыми двумя вершинами не превосходит n. Другое определение гласит, что подграф G' является сообществом, если сумма всех степеней внутри G' больше суммы всех степеней, направленных в остальную часть графа [8]. Сообщества называются смежными, если существует ребро, направленное из вершины первого сообщества в вершину второго.

1.3. Модулярность

Однако подобными определениями сообществ пользоваться неудобно и их проверка достаточно долгая. В 2004 году была представлена *модулярность* — целевая функция, оценивающая неслучайность разбиения графа на сообщества [9]. Допустим, у нас K сообществ, определим тогда симметричную матрицу $\mathbf{e}$ размером $K \times K$. Пусть e_{ij} — отношение количества рёбер, которые идут из сообщества i в сообщество j, к полному количеству рёбер в графе (рёбра l_{mn} и l_{nm} считаются различными, m, n — узлы). След такой матрицы $\mathrm{Tr}\mathbf{e} = \sum_{i \in 1..K} e_{ii}$ показывает отношение рёбер в сети, которые соединяют узлы одного и того же сообщества, и хорошее разбиение на сообщества должно иметь высокое значение следа. Однако если поместить все вершины в одно сообщество — след примет максимальное возможное значение, притом, что такое разбиение не будет сообщать ничего полезного о графе.

Поэтому далее определяется вектор $\mathbf{a}$ длины K, элементы которой $a_i = \sum_{j \in 1..K} e_{ij}$, которая обозначает долю количества рёбер, идущих к узлам, принадлежащим сообществу i, к полному количеству рёбер в графе. Если в графе рёбра проходят между вершинами независимо от сообществ — e_{ij} будет в среднем равно $a_i a_j$, поэтому модулярность можно определить следующим образом:

$$Q(G, P) = \sum_{i \in 1..K} \left(e_{ii} - a_i^2\right) = \mathrm{Tr}\mathbf{e} - \|\mathbf{e}^2\|, \tag{1}$$

где $\|\mathbf{x}\|$ является суммой элементов матрицы $\mathbf{x}$. Если количество рёбер внутри сообществ не будет отличаться от случайного взятого количества — модулярность будет примерно равна 0. Максимальным возможным значением функции будет 1, но на практике модулярности графов лежат между 0.3 и 0.7.

Было предложено несколько вариаций модулярности [10, 11]. Так, эквивалентным приведённому выше определению будет

$$Q(G, P) = \frac{1}{2L} \sum_{x,y \in 1..N} \left(w_{xy} - \frac{s_x s_y}{2L}\right) \ \delta(c_P(x), c_P(y)), \tag{2}$$

где L — мощность $\mathscr{L}$, w_{xy} — вес ребра между вершинами x и y, s_x и s_y — степени вершин x и y соответственно, δ — символ Кронекера, а отображение $c_P(\cdot)$ указывает, в каком сообществе разбиения лежит узел графа.

Теперь можно поставить задачу выделения сообществ следующим образом: требуется найти такое разбиение графа, что модулярность примет максимальное значение. Можно заметить, что такая постановка не использует какого-либо определения сообществ, и получившиеся разбиение не проверяется на дополнительные свойства, кроме подсчёта модулярности. Однако такая задача всё ещё будет NP-сложной [12].

Преимущество модулярности состоит в том, что для того, чтобы посчитать, какой выигрыш мы извлечем из объединения двух сообществ, необходимо произвести только одну операцию. В рамках определения (1) такой выигрыш будет равен $\Delta Q = 2(e_{ij} - a_i a_j)$, где i и j — потенциально объединяемые сообщества.

Для того, чтобы объединить два сообщества необходимо сделать $O(\min\{n_i, n_j\})$ операций, где n_i и n_j обозначают количество смежных к i и j сообществ. Не умоляя общности, $n_j \leq n_i$, тогда необходимо обновить столбец i-ый столбец и i-ую строку матрицы $\mathbf{e}$, а так же i-ый элемент вектора $\mathbf{a}$: $e_{ik} = e_{ki} = e_{ki} + e_{kj}$, где k — смежное к j сообщество, и $a_i = a_i + a_j$. При этом сообщество j следует удалить из дальнейшего рассмотрения.

Имея матрицу $\mathbf{e}$ и вектор $\mathbf{a}$ не очень важно, как устроен граф и сообщества, что позволяет искать сообщества, основываясь на некотором начальном разбиении, для которого построены $\mathbf{e}$ и $\mathbf{a}$.

1.4. Рандомизированный жадный алгоритм

Ньюман в 2004 году предложил алгоритм, максимизирующий модулярность [13]. Алгоритм начинается с разбиения графа на N сообществ из одной вершины, а затем на каждой итерации просматривает все пары сообществ и соединяет ту пару, которая даст наибольший выигрыш модулярности. Такой алгоритм достаточно долго работает и страдает от несбалансированного объединения сообществ — сообщества растут с разной скоростью, большие кластеры соединяются со своими небольшими соседями независимо от того, выгодно это глобально или нет [14].

Поэтому был предложен рандомизированный жадный алгоритм (RG) [15], который на каждой итерации рассматривал k случайных сообществ и смежных к ним сообществ, а затем так же соединял пару, дающую наибольший выигрыш. Трудоёмкость такого алгоритма примерно равна $O(kL \ln N)$. И первый алгоритм, и его рандомизированная вариация соединяют сообщества, записывая только номера соединений, до тех пор, пока не останется только одно сообщество, а затем создают разбиение из списка соединений до того момента, когда достигалась максимальная модулярность (так как в результате лучшего соединения модулярность может уменьшиться).

Можно отметить, что таким алгоритмом можно кластеризовать не только граф, но и граф с некоторым начальным разбиением, в котором можно сообщества разбивать дальше, но нельзя их соединять. При этом только немного поменяется начальный этап инициализации матрицы $\mathbf{e}$ и вектора $\mathbf{a}$ (смотри Алгоритм 1).

1.5. Ансамблевая стратегия

Овельгённе и Гейер-Шульц в 2012 году выиграли 10th DIMACS Implementation Challenge с ансамблевой стратегией выделения сообществ (ES). Ансамблевая стратегия заключается в том, что сначала s начальных алгоритмов разбивают граф на сообщества, и считается, что те вершины, в которых начальные алгоритмы сошлись во мнении определены по сообществам правильно, а те, которые остались, распределяет по сообществам финальный алгоритм [16].

Формализовать это можно следующим образом:

1. Создать множество S из s разбиений G с помощью начальных алгоритмов
2. Создать разбиение $\hat{P}$, равное максимальному перекрытию разбиений из множества S
3. Финальным алгоритмом создать разбиение $\widetilde{P}$ графа G на основе разбиения $\hat{P}$

Необходимо определить понятие *максимальное перекрытие*. Пусть у нас есть множество $S = \{P_1, \ldots, P_s\}$, $c_P(v)$ указывает, в каком сообществе находится узел v с

Входные данные: Невзвешенный неориентированный граф $G = (\mathscr{N}, \mathscr{L})$, параметр k

Выходные данные: Разбиение на сообщества P

for $i \in 1..N$ **do**
  for $j \in 1..N$ **do**
    if i и j смежные **then**
      $e[i, j] = 1/(2 * L)$;
    else
      $e[i, j] = 0$;
    end
  end
  $a[i] = \sum_j e[i, j]$;
end
$global\Delta Q \leftarrow 0$;
$max\_global\Delta Q \leftarrow -\infty$;

for $i \in 1..N$ **do**
  $max\Delta Q \leftarrow -\infty$;
  for $j \in 1..k$ **do**
    $c1 \leftarrow$ случайное сообщество;
    forall сообщества $c2$, смежные с $c1$ **do**
      $\Delta Q \leftarrow 2 * (e[i, j] - a[i] * a[j])$;
      if $\Delta Q > max\Delta Q$ **then**
        $max\Delta Q \leftarrow \Delta Q$;
        $next\_join \leftarrow (c1, c2)$;
      end
    end
  end
  $joins\_list.push(next\_join)$;
  $global\Delta Q \leftarrow global\Delta Q + max\Delta Q$;
  if $global\Delta Q > max\_global\Delta Q$ **then**
    $max\_global\Delta Q \leftarrow global\Delta Q$;
    $best\_step \leftarrow i$;
  end
  $(c1, c2) \leftarrow next\_join$;
  if количество соседей($c2$) $>$ количество соседей($c1$) **then**
    поменять местами $c1$ и $c2$;
  end
  forall соседи $c3$ сообщества $c2$, где $c3 \neq c1, c2$ **do**
    $e[c3, c1] \leftarrow e[c3, c1] + e[c3, c2]$;
    $e[c1, c3] \leftarrow e[c3, c1]$;
  end
  $e[c1, c1] \leftarrow e[c1, c1] + e[c2, c2] + e[c1, c2] + e[c2, c1]$;
  $a[c1] \leftarrow a[c1] + a[c2]$;
end

$P \leftarrow$ создать разбиение из $joins\_list[1..best\_step]$;

Алгоритм 1: Рандомизированный жадный алгоритм

разбиении P. Тогда у максимального перекрытия $\hat{P}$ множества S будут следующие свойства:

$$v, w \in \mathscr{N}, \forall i \in 1..s \ : \ c_{P_i}(v) = c_{P_i}(w) \Rightarrow c_{\hat{P}}(v) = c_{\hat{P}}(w)$$

$$v, w \in \mathscr{N}, \exists i \in 1..s \ : \ c_{P_i}(v) \neq c_{P_i}(w) \Rightarrow c_{\hat{P}}(v) \neq c_{\hat{P}}(w)$$

Ансамблевую стратегию можно итерировать, заставляя начальные алгоритмы разбивать максимальное перекрытие и получившееся максимальное перекрытие до тех пор, пока это будет увеличивать модулярность. В таком случае схема будет выглядеть следующим образом:

1. Инициализировать $\hat{P}$ разбиением из сообществ из одного узла
2. Создать множество S из s разбиений графа G на основе разбиения $\hat{P}$ с помощью начальных алгоритмов
3. Записать в $\hat{P}$ максимальное перекрытие множества S
4. Если P_{best} не существует или оно хуже, чем $\hat{P}$, то присвоить $P_{best} \leftarrow \hat{P}$ и вернуться на второй шаг
5. Финальным алгоритмом создать разбиение $\widetilde{P}$ графа G на основе разбиения P_{best}

В качестве начальных и финального алгоритма можно брать рандомизированный жадный алгоритм.

1.6. Одновременно возмущаемая стохастическая аппроксимация

Стохастические аппроксимация была введена Роббинсом и Монро в 1951 году [17] и затем была использована для решения оптимизационный задач Кифером и Вольфовицем (KW) [18]. В [19] алгоритм стохастической аппроксимации был расширен до многомерного случая. В m-мерном пространстве обычная KW-процедура, основанная на конечно-разностной аппроксимации градиента, использовала $2m$ измерений на каждой итерации (по два измерения на каждую координату градиента). Спалл предложил алгоритм *одновременно возмущаемой стохастической аппроксимации* (SPSA) [20], который на каждой итерации использует всего два измерения. Он показал, что SPSA алгоритм имеет такую же скорость сходимости, несмотря на то, что в многомерном случае (даже при $m \to \infty$), несмотря на то, что в нём используется заметно меньше измерений [21].

Стохастическая аппроксимация первоначально использовалась как инструмент для статистических вычислений и в дальнейшем разрбатывалась в рамках отдельной ветки теории управления. На сегодняшний день стохастическая аппроксимация

имеет большое разнообразие применений в таких областях, как адаптивная обработка сигналов, адаптивное выделение ресурсов, адаптивное управление.

Алгоритмы стохастической аппроксимации показали свою эффективность в решении задач минимизации стационарных функционалов. В [22] для функционалов, меняющихся со временем были применены метод Ньютона и градиентный метод, но они применимы только в случае дважды дифференцируемых функционалов и в случае известных ограничений на Гессиан функционала. Так же оба метода требуют возможности вычисления градиента в произвольной точке.

Общую схему одновременно возмущаемой стохастической аппроксимации можно представить следующим образом:

1. Выбор начальной центральной точки $\theta_0 \in \mathbb{R}^m$, счётчик $n = 0$, выбор параметров алгоритма $d \in \mathbb{R} \setminus \{0\}$, $\{\alpha_n\} \subset \mathbb{R}^m$
2. Увеличение счётчика $n \to n + 1$
3. Выбор вектора возмущения $\Delta_n \in \mathbb{R}^m$, чьи координаты независимо генерируются и в среднем дают ноль. Часто для генерации компонент вектора используют распределение Бернулли, дающее ± 1 с вероятностью $\frac{1}{2}$ для каждого значения
4. Определение новых аргументов функции $\theta_n^- = \hat{\theta}_{n-1} - d\Delta_n$ и $\theta_n^+ = \hat{\theta}_{n-1} + d\Delta_n$
5. Вычисление значений функционала $y_n^- = f(\theta_n^-), y_n^+ = f(\theta_n^+)$
6. Вычисление следующей центральной точки

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \alpha_n \frac{y_n^+ - y_n^-}{|\theta_n^+ - \theta_n^-|} \tag{3}$$

7. Далее происходит либо остановка алгоритма, либо переход на второй пункт

В [23] был представлен метод стохастической аппроксимации с константным размером шага, в таком случае вместо последовательности $\{\alpha_n\}$ используется единственный параметр $\alpha \in \mathbb{R}^m$, и следующая центральная точка вычисляется по следующей формуле: $\hat{\theta}_n = \hat{\theta}_{n-1} - \alpha \frac{y_n^+ - y_n^-}{|\theta_n^+ - \theta_n^-|}$

1.7. Постановка задачи

Рандомизированный жадный алгоритм имеет один параметр k, в то время как ансамблевая имеет стратегия один параметр s и, в случае использования рандомизированного жадного алгоритма в качестве начального и финального алгоритма, будет иметь два дополнительных параметра k_1 и k_2, то есть в целом три параметра. Если на

параметры рандомизированного жадного алгоритма поставить ограничения, к примеру, $k_1 = k_2$, то то можно считать, что ансамблевая стратегия имеет два параметра, s и k.

Часто алгоритмы на каждых входных данных имеют разные оптимальные параметры, то есть параметры, решающие задачу наилучшим образом. И алгоритм SPSA показал хорошие результаты в адаптировании параметров подобных алгоритмов, в ходе работы рассматриваемого алгоритма его параметры варьируются, довольно быстро достигая оптимальной точки.

В данной работе будет рассматриваться применение алгоритма SPSA к двум алгоритмам выделения сообществ в графах, а так же сравнение модулярностей этих алгоритмов и их адаптивных модификаций.

1.8. Тестовые графы

В качестве тестовых графов были взяты графы, используемые для оценки алгоритмов на 10th DIMACS Implementation Challenge. Далее представлены используемые графы с кратким описанием, в порядке увеличения веса файла в формате METIS Graph.

- **karate**, $N = 34,\ L = 78$: социальная сеть между 34 членами карате клуба с 1970 по 1972 год [24]
- **dolphins**, $N = 62,\ L = 159$: социальная сеть частых общений между 62 дельфинами [25]
- **chesapeake**, $N = 39,\ L = 170$: сеть мезогалинных вод Чесапикского залива [26]
- **adjnoun**, $N = 112,\ L = 425$: сеть смежности частых прилагательных и существительных в романе «Давид Копперфильд» Чарльза Диккенса [27]
- **polbooks**, $N = 105,\ L = 441$: сеть книг о политике США, изданных во время президентских выборов 2004 года. Рёбра между книгами означают частые покупки одними и теми же покупателями. Сеть скомпилирована Валдисом Кребсом, однако не была опубликована
- **football**, $N = 115,\ L = 613$: сеть игр в американский футбол между колледжами из дивизиона IA во время регулярного сезона осенью 2000 года [28]
- **celegans_metabolic**, $N = 453,\ L = 2025$: метаболическая сеть Caenorhabditis elegans [29]
- **jazz**, $N = 198,\ L = 2742$: сеть джазовых музыкантов [30]

- **netscience**, $N = 1589,\ L = 2742$: сеть соавторства среди учёных, работающих над сложными сетями [27]

- **email**, $N = 1133,\ L = 5451$: сеть связей по электронной почте между членами Университета Ровира и Вирхилий [31]

- **polblogs**, $N = 1490,\ L = 16715$: сеть ссылок между веб блогами о политике США в 2005 году [32]

- **PGPgiantcompo**, $N = 10680,\ L = 24316$: список узлов гигантской компоненты сети пользователей алгоритма Pretty-Good-Privacy для защищенного обмена информацией [33]

- **as-22july06**, $N = 22963,\ L = 48436$: структура интернета на уровне автономных систем на 22 июля 2006 года. Сеть создана Марком Ньюманом и не опубликована

- **cond-mat-2003**, $N = 31163,\ L = 120029$: сеть соавторства среди учёных, публиковавших препринты в определённые архивы между 1995 и 2003 годами [34]

- **caidaRouterLevel**, $N = 192244,\ L = 609066$: граф структуры интернета на уровне роутера, собранный ассоциацией CAIDA в апреле и мае 2003 года

- **cnr-2000**, $N = 325557,\ L = 2738969$: небольшая часть обхода итальянского домена .cnr [35, 36, 37]

- **in-2004**, $N = 1382908,\ L = 13591473$: небольшая часть обхода индийского домена .in [35, 36, 37]

- **eu-2005**, $N = 862664,\ L = 16138468$: небольшая часть обхода домена Европейского союза .eu [35, 36, 37]

Кроме того, некоторые тесты используют автоматически сгенерированные графы с заранее известным количеством сообществ. Такой граф имеет четыре параметра — количество узлов N, количество сообществ K (все сообщества одинаковых размеров), вероятность появления ребра между узлами одного сообщества p_1 и вероятность появления ребра между вершинами разных сообществ p_2. Преимущество автоматически сгенерированных графов заключается в проверке работы алгоритма на разных по размеру графах с разными по плотности сообществами. Так же из построения известны реальные сообщества. Однако графы, построенные по реальным системам могут сильно отличаться от подобных графов.

Так, далее в работе используется автоматически сгенерированный граф под названием *auto40*, со следующими параметрами: $N = 40,000,\ K = 40,\ p_1 = 0.1,\ p_2 = 5 \cdot 10^{-4}$.

2. Адаптивный рандомизированный жадный алгоритм

2.1. Применимость алгоритма SPSA

Для того, чтобы алгоритм SPSA был применим — необходимо иметь выпуклую функцию качества, которую необходимо минимизировать. В большинстве модулярность результатов работы рандомизированного жадного алгоритма на разных графах с разными значениями параметра k будет выглядеть следующим образом:

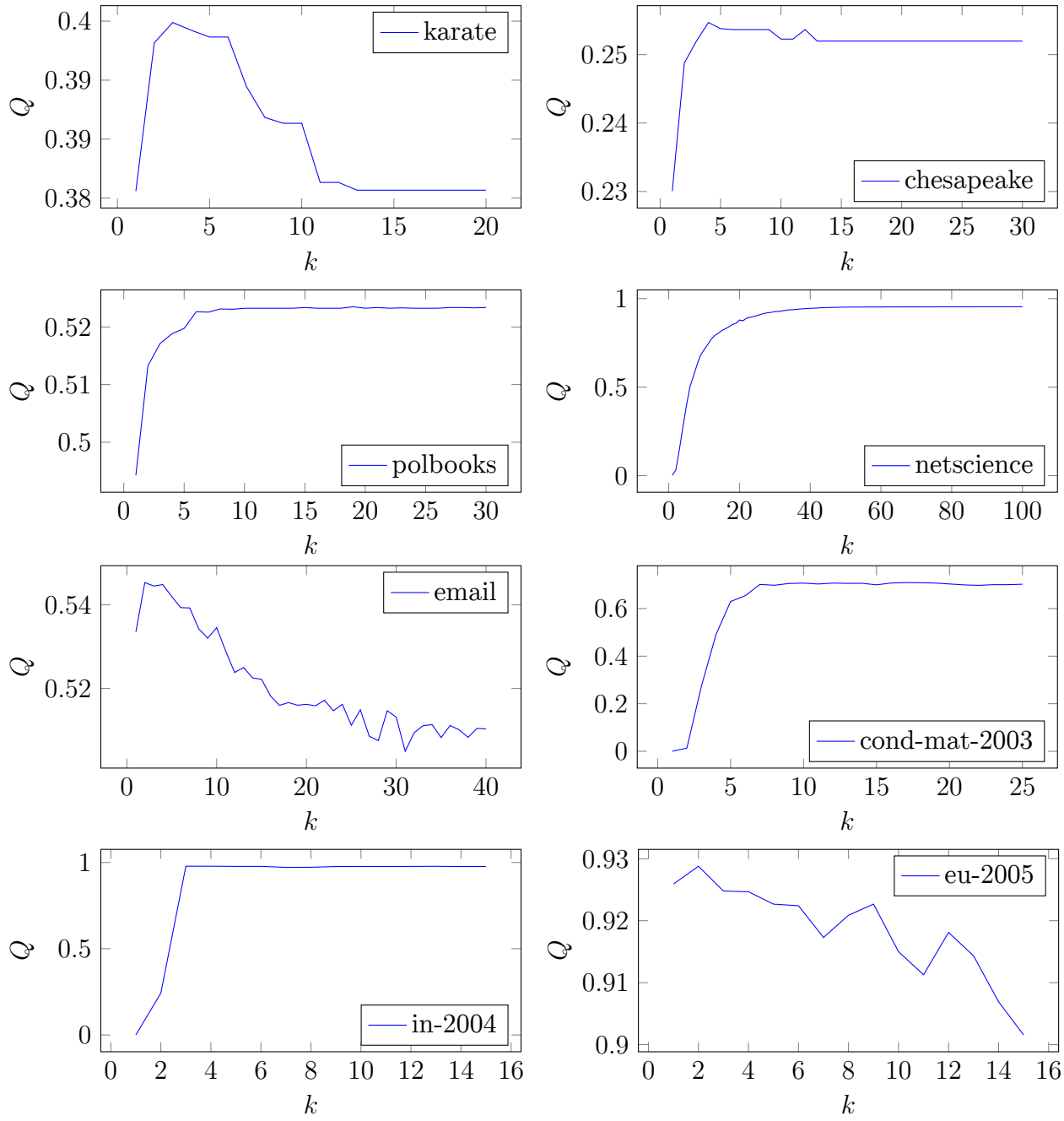

Рис. 1: Зависимость модулярности от k для рандомизированного случайного алгоритма на разных графах

Результаты показывают разделение поведения алгоритма на разных графах на два возможных случая: в первом алгоритм принимает наилучший результат при некотором небольшом, но разном k_{best} (например работа алгоритма на графе *karate*). Во втором результаты алгоритма постепенно возрастают, приближаясь к некоторой асимптоте (хорошим примером будет работа алгоритма на графе *netscience*. К первому случаю так же относится такое поведение, в котором алгоритм быстро (с ростом k) достигает своего лучшего значения, и затем его результаты очень несильно падают, и в дальнейшем держатся того же значения (такое выполняется, например, на графе *in-2004*).

Похожее поведение алгоритм показывает на автоматически сгенерированных графах, но в таких графах можно получить более отличающиеся значения k_{best} и предположить, что будет происходить с модулярностью при дальнейшем росте k.

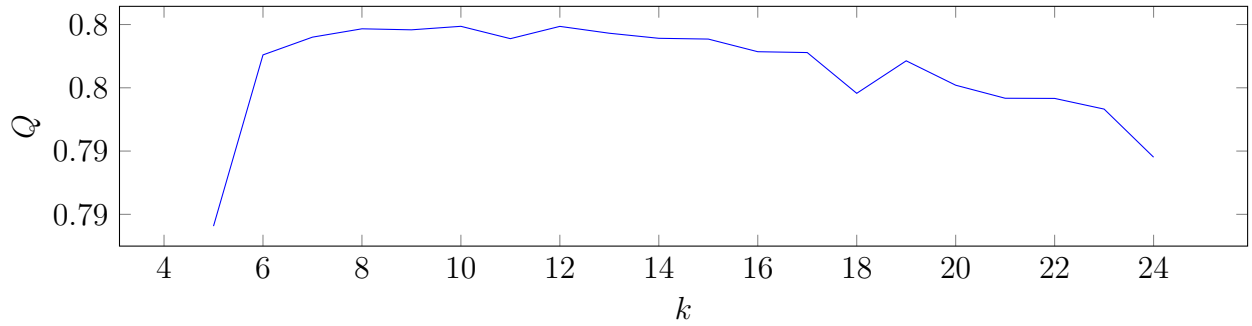

Рис. 2: Зависимость модулярности от k для рандомизированного случайного алгоритма на автоматически сгенерированном графе *auto40* с параметрами $N = 40.000$, $K = 40$, $p_1 = 0.1$, $p_2 = 5 \cdot 10^{-4}$

2.2. Функция качества

Таким образом, имеет смысл использовать в функции качества не только модулярность, но и время. Подсчёт времени сам по себе занимает время и в разных случаях может давать сильно отличающиеся результаты. Теоретическая трудоёмкость алгоритма линейно зависит от параметра k, на реальных графах зависимость тоже близка к линейной:

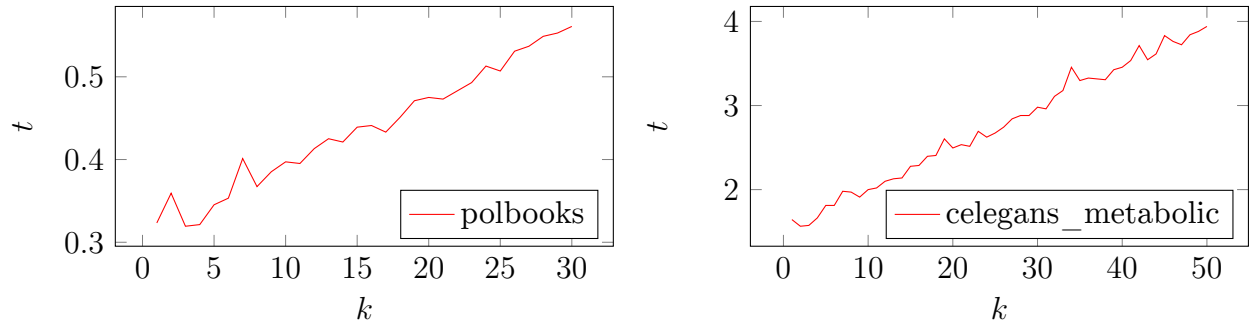

Рис. 3: Зависимость времени t в миллисекундах от k для рандомизированного случайного алгоритма на графах *polbooks* и *celegans_ metabolic*

Таким образом, вместо времени можно использовать значение k. Максимальное значение модулярности, которое может быть достигнуто на графе очень сильно отличается, поэтому нет смысла использовать абсолютное значение модулярности, но имеет смысл использовать относительные значения. При вычислении центрального значения (3) в алгоритме одновременно возмущаемой стохастической аппроксимации используется только разность функций качества. Таким образом, если использовать в функции качества логарифмы от модулярности — разность функций укажет, во сколько раз модулярность изменилась.

Так же функция качества должна принимать минимум, а не максимум, поэтому первой версией подобной функции может быть $f(Q) = -\alpha \ln Q,\ \alpha > 0$. Для того, чтобы принимать во внимание время работы, разумно добавить логарифм от k:

$$f(Q, k) = -\alpha(\ln Q - \beta \ln k),\ \alpha > 0, \beta \geq 0 \tag{4}$$

Коэффициент β в таком случае можно рассматривать в следующем виде $\beta = \frac{\ln \gamma}{\ln 2}$, где коэффициент β указывает, во сколько раз необходимо увеличиться модулярности для того, чтобы покрыть увеличение времени (то есть k) вдвое. Если время не имеет значение коэффициент γ принимает значение 1, а следовательно коэффициент β принимает значение 0. Коэффициент α же играет роль размера шага при выборе следующей центральной точки.

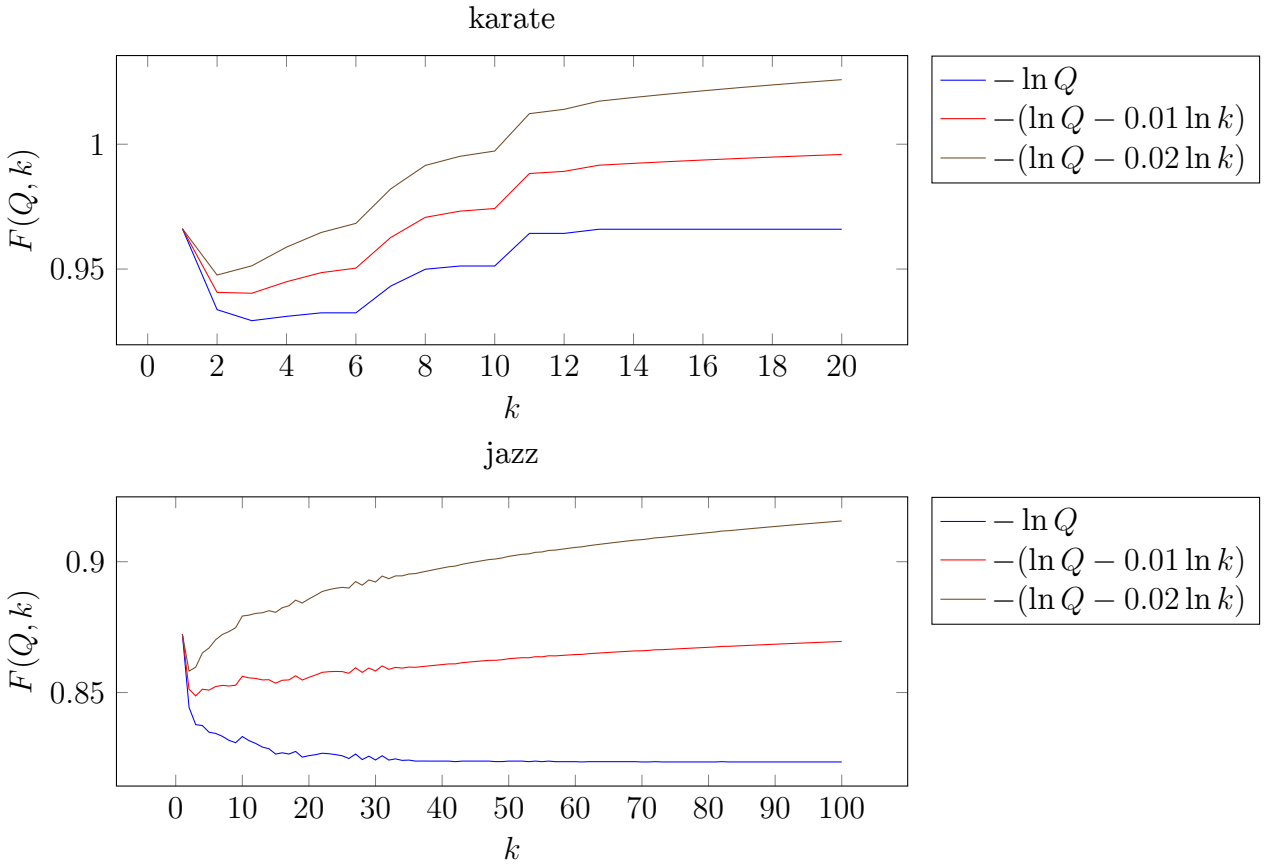

Рис. 4: Функции качества с разными коэффициентами β для графов *karate* и *jazz*

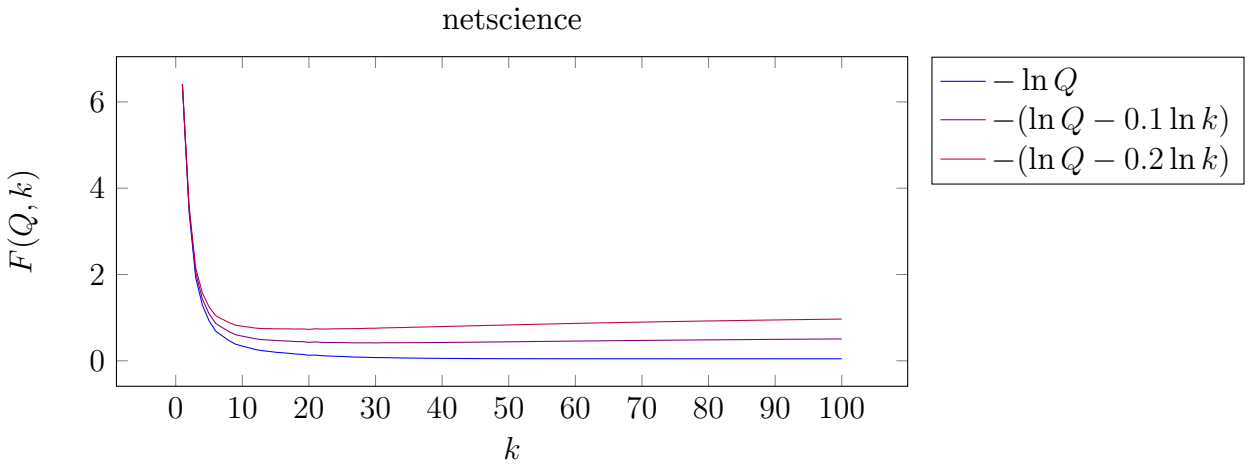

Рис. 5: Продолжение Рис. 4. Функции качества с разными коэффициентами β для графа *netscience*

Как видно из рисунка 5, иногда для того, чтобы функция качества имела минимум на небольших k, надо задавать довольно большое значение коэффициента β. Однако это логично, на данном графе если нас устраивает время работы — выгоднее всё время увеличивать значение k.

2.3. Адаптивный алгоритм

При использовании алгоритма SPSA для рандомизированного жадного алгоритма предлагается разбить действие алгоритма на шаги длиной в σ итераций. В течении каждого шага используется одно значение k. После каждого шага можно считать прирост модулярности, но вместо этого имеет смысл считать медиану прироста модулярности за σ итераций — так как алгоритм рандомизированный, время от времени будут появляться очень хорошие соединения сообществ, которые будут портить функцию качества, такой большой прирост может появиться даже при очень плохом k. В таком случае схема алгоритма будет выглядеть следующим образом:

1. Выбор начальной центральной точки $\hat{k}_0 \in \mathbb{N}$, счётчик $n = 0$, выбор размера возбуждения $d \in \mathbb{N}$, коэффициентов функции качества $\alpha, \beta \in \mathbb{R},\ \alpha > 0,\ \beta \geq 0$, и $\sigma \in \mathbb{N}$ — количество итераций в одном шаге

2. Увеличение счётчика $n \to n + 1$

3. Определение новых аргументов функции $k_n^- = \max\{\hat{k}_{n-1} - d, 1\}$ и $k_n^+ = \hat{k}_{n-1} + d$

4. Выполнение σ итераций с параметром k_n^-, вычисление медианы прироста модулярности μ_n^-

5. Выполнение σ итераций с параметром k_n^+, вычисление медианы прироста модулярности μ_n^+

6. Вычисление функций качества $y_n^- = -\alpha(\ln \mu_n^- - \beta \ln k_n^-)$, $y_n^+ = -\alpha(\ln \mu_n^+ - \beta \ln k_n^+)$

7. Вычисление следующей центральной точки

$$\hat{k}_n = \max\left\{1, \left[\hat{k}_{n-1} - \frac{y_n^+ - y_n^-}{k_n^+ - k_n^-}\right]\right\} \tag{5}$$

8. Далее происходит переход на второй пункт

Алгоритм заканчивает работу в тот момент, когда для рассмотрения осталось ровно одно сообщество.

2.4. Чувствительность к перепадам функции качества

Коэффициент α отвечает за то, насколько чувствителен алгоритм будет к перепадам функций качества — чем больше α, тем сильнее будет отличаться новая центральная точка от предыдущей в одной и той же ситуации. На трёх графах были измерены зависимости модулярности от параметра α при $d = 2,\ \sigma = 500,\ \hat{k}_0 = 10,\ \beta = 0$:

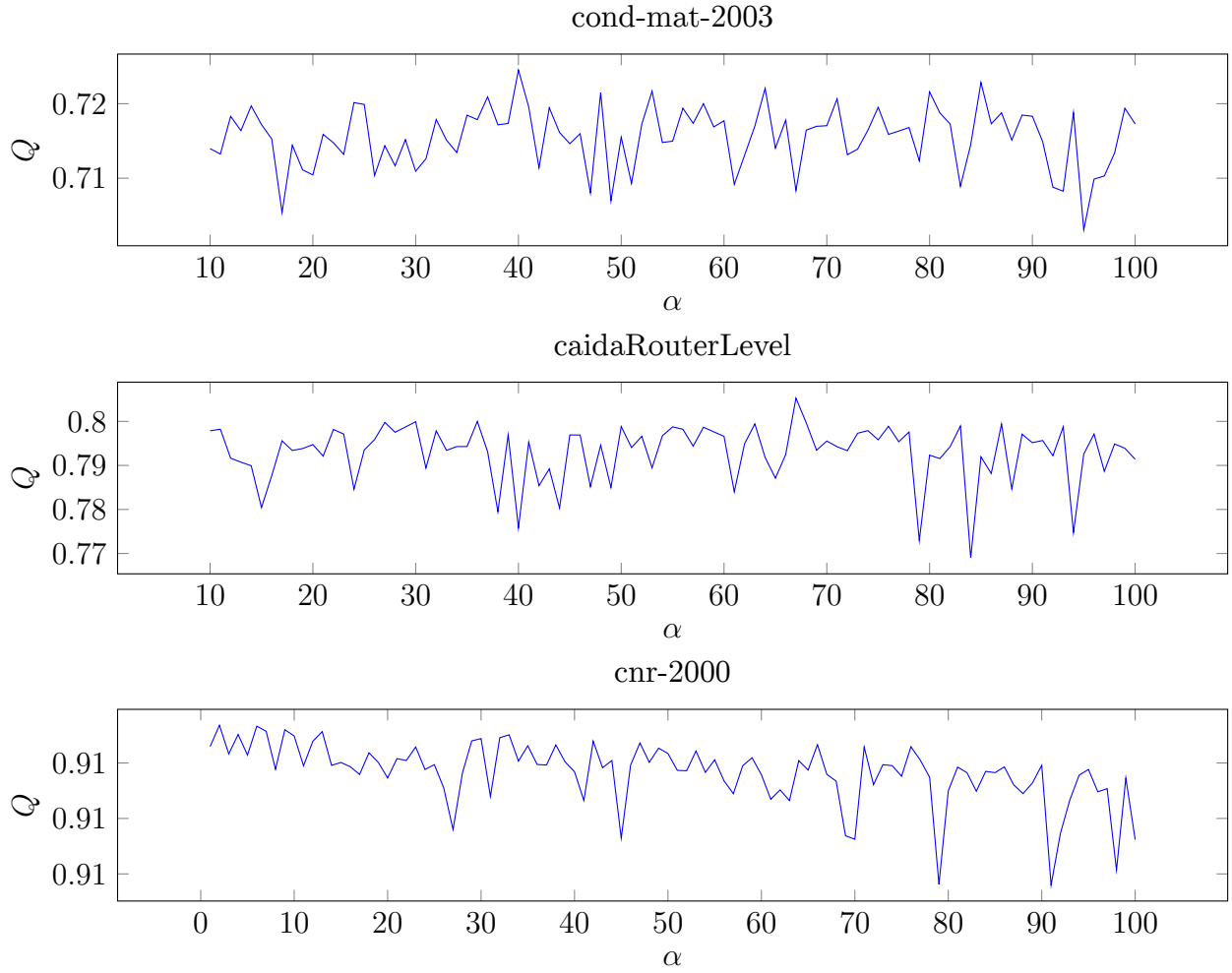

Рис. 6: Зависимость модулярности от параметра α в работе адаптивного рандомизированного жадного алгоритма на графах *cond-mat-2003*, *caidaRouterLevel*, *cnr-2000*

Из рисунка 6 видно, что результат не очень стабильный, хотя значение модулярности и колеблется на небольшом промежутке. Так же на третьем графе заметно, что увеличение параметра α влечёт за собой уменьшение модулярности в среднем.

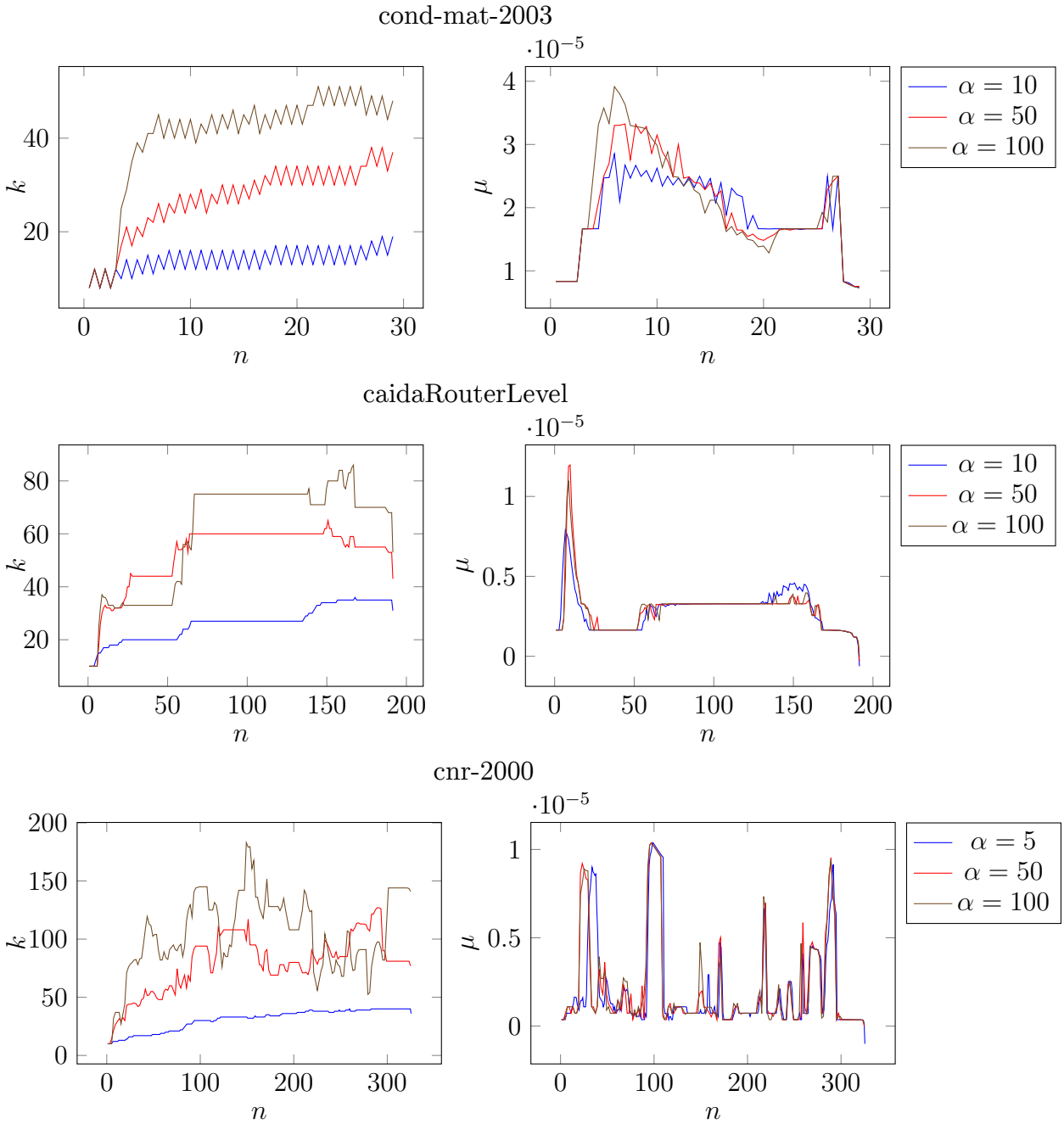

Рис. 7: Изменения параметра k и медианы приросты модулярности со временем для разных параметров α на трёх графах. Графики для второго и третьего графа смазаны для повышения читаемости

Из рисунков видно, что в зависимости от α параметр k, с которым прогоняются шаги по σ шагов, и который поочередно принимает значения k_n^- и k_n^+, меняется с разной интенсивностью, но результаты при этом получаются приблизительно одинаковые.

2.5. Размер возмущения

Коэффициент d отвечает за то, насколько сильно будет возмущаться центральная точка для получения следующих измерений. То есть, насколько k_n^+ и k_n^- будут отличаться от $\hat{k}_{n-1}$. Зависимость модулярности от размера возмущения при $f(\mu, k) = -10 \ln \mu,\ \sigma = 500,\ \hat{k}_0 = 10$ будет выглядеть следующим образом:

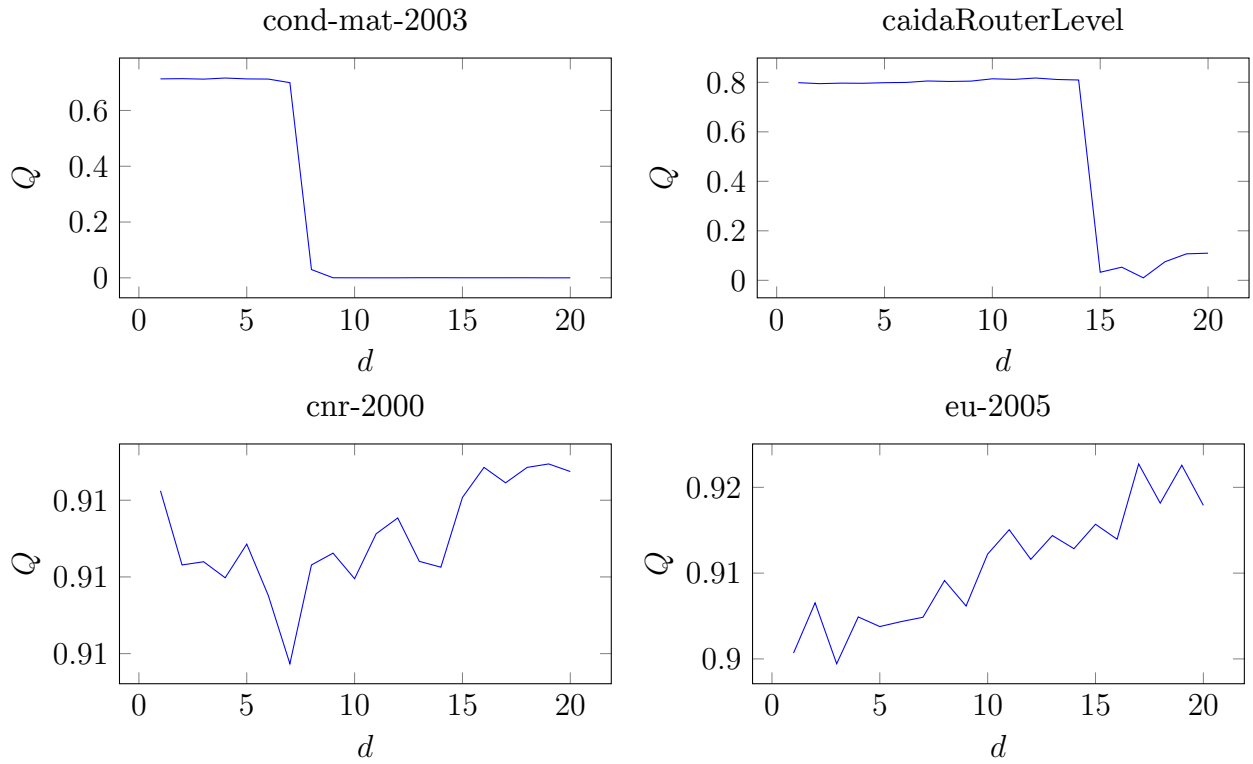

Рис. 8: Зависимость модулярности от размера возмущения на четырёх графах

На графах *cnr-2000* и *eu-2005* значения модулярности не очень сильно менялись в зависимости от параметра d, хотя некоторые значения d и давали более большие значения. Однако на графах *cond-mat-2003* и *caidaRouterLevel* после некоторого порогового значения возмущения модулярность показывала, что получившееся разбиение не лучше случайного.

2.6. Количество итераций в одном шаге

Параметр σ указывает, как часто меняется k в рандомизированного жадном алгоритме. Так как в функции качества используется медиана прироста модулярности, а не прирост модулярности за все σ шагов — при изменении σ нет необходимости менять функцию качества, модулярность прироста будет оставаться приблизительно такой же по величине, в то время как прирост модулярности линейно зависит от σ. Зависимость модулярности от количества итераций в одном шаге при $d = 5,\ f(\mu, k) = -10 \ln \mu,\ \hat{k}_0 = 10$ принимает такой вид:

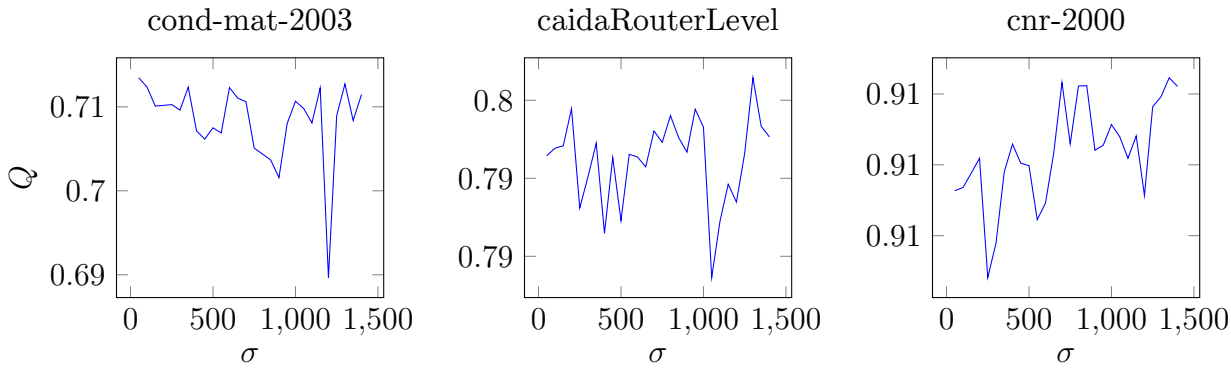

Рис. 9: Зависимость модулярности от значения σ на трёх графах. σ принимает значения от 50 до 1400

На всех трёх графах модулярность несильно, но непредсказуемо меняется при изменении σ. Во всех случаях есть хорошие и плохие значения σ, но в целом значения параметра достаточно равнозначны.

2.7. Начальная центральная точка

Начальная центральная точка указывает, с какой точки будут находиться k_1^- и k_1^+, с которыми будут проходиться первые и вторые σ итераций, соответственно. Зависимость модулярности от начальной центральной точки $\hat{k}_0$ при $d = 5,\ f(\mu, k) = -10 \ln \mu,\ \sigma = 1000$ будет выглядеть следующим образом:

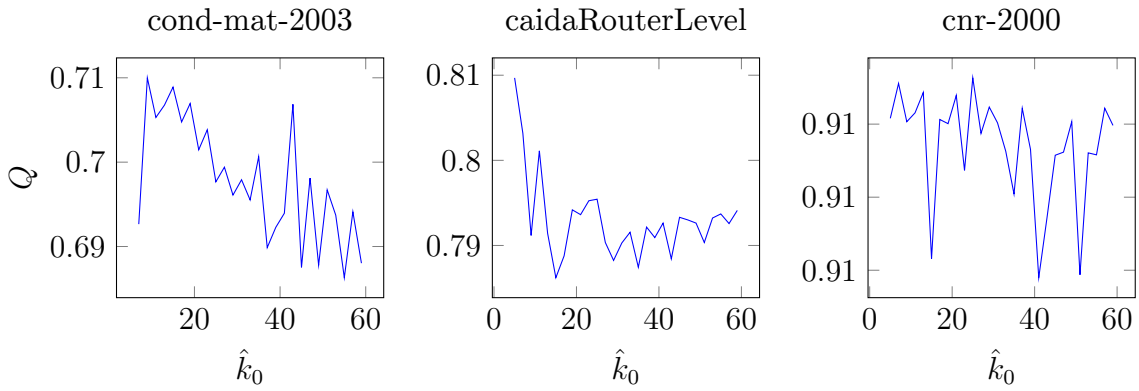

Рис. 10: Зависимость модулярности от значения $\hat{k}_0$ на трёх графах

Как можно заметить, на разных графах модулярность ведёт себя по разному в зависимости от $\hat{k}_0$. График для графа *cond-mat-2003* на рисунке 10 начинается от первого вхождения $\hat{k}_0 = 7$ (остальные от $\hat{k}_0 = 5$), так как при меньших значениях модулярность очень сильно падает. Это связано с тем, что $d = 5$ и следовательно $k_1^- = 1$, в то время как рандомизированный алгоритм с параметром $k = 1$ часто работает сильно хуже остальных. Однако выгоднее использовать небольшое значение $\hat{k}_0$, хоть и большее $d + 1$.

2.8. Коэффициент β функции качества

Коэффициент β функции качества (4) указывает на то, насколько много внимания алгоритм обращает на величину точек k_n^- и k_n^+ при выборе следующей центральной точки. Предполагалось, что чем больше β, тем быстрее будет работать алгоритм, однако возможно будет находить более плохие разбиения. Такие результаты получились при параметрах $d = 5,\ \sigma = 1000,\ \hat{k}_0 = 8$, функция качества $f(\mu, k) = -10(\ln \mu - \beta \ln k)$:

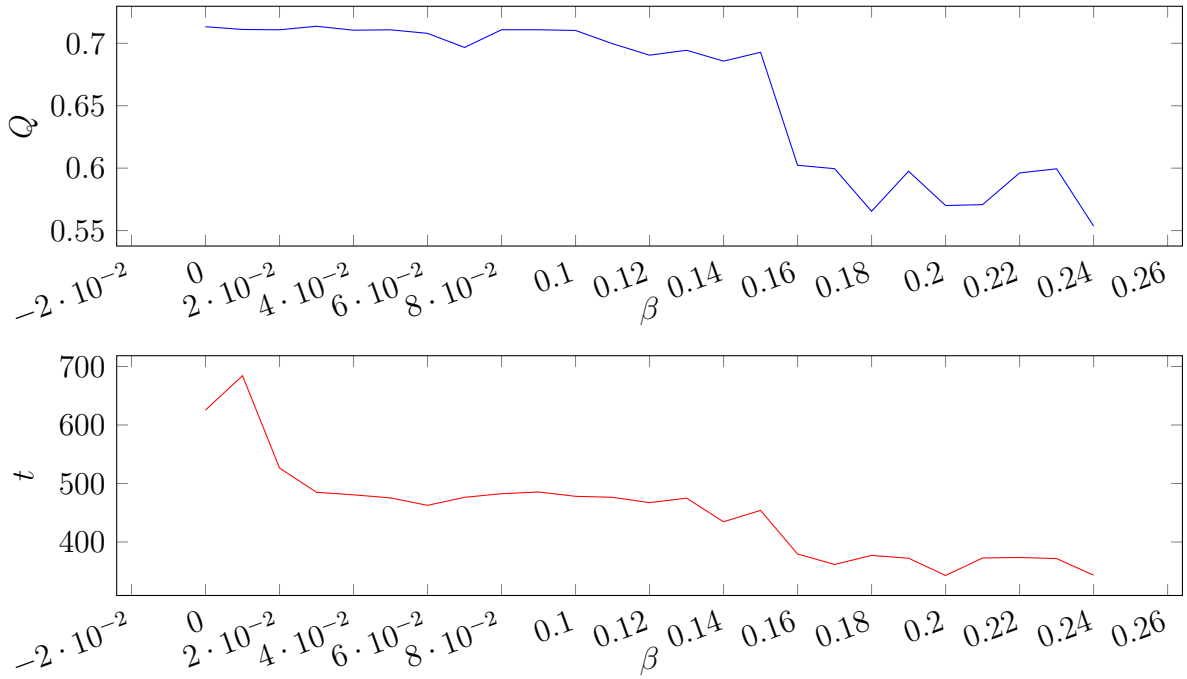

Рис. 11: Зависимость модулярности и времени от параметра функции качества β на графе *cond-mat-2003*

2.9. Результаты

На таблицах 1 и 2 показано сравнение результатов работы рандомизированного жадного алгоритма и адаптивного рандомизированного жадного алгоритма. В качестве графов для рассмотрения выбраны 7 графов.

Рандомизированный жадный алгоритм в таблицах обозначается как $RG(\cdot)$ с параметром k в качестве аргумента. Адаптивный рандомизированный жадный алгоритм обозначается как $ARG(\cdot)$ с параметром функции качества β в качестве аргумента, остальные параметры при этом равны $d = 5,\ f(\mu, k) = -10(\ln \mu - \beta \ln k),\ \sigma = 1000,\ \hat{k}_0 = 8$. Такой набор параметров не гарантирует наилучший результат на каждом графе или даже наилучший в среднем по всем графам результат для адаптивного алгоритма, однако показывает неплохие результаты.

Оба алгоритма — рандомизированные, поэтому на каждом отдельном запуске результат может оказаться очень хорошим или сравнительно плохим, однако даст мало сведений о качестве алгоритма. Поэтому для каждого графа производилось опреде-

лённое количество запусков, и в качестве модулярности бралась медиана значений, а в качестве времени — среднее, для повышения точности измерения времени.

Так, для графа *as-22july06* производился 101 запуск, для *cond-mat-2003* 51 запуск, для *caidaRouterLevel* и *cnr-2000* 11 запусков, для *in-2004* пять запусков, и для *eu-2005* три запуска. Для кратости в таблице эти графы записываются как *22july* вместо *as-22july06*, *cond* вместо *cond-mat-2003*, *caida* вместо *caidaRouterLevel* и *cnr* вместо *cnr-2000*.

Кроме того, в таблицу вошли результаты работы над автоматически сгенерированным графом *auto40*, который уже появлялся в работе, к примеру в рисунке 2. Для этого графа для повышения точности производилось 11 запусков.

Таблица 1: Модулярности разбиений, полученных в результатах работы рандомизированного жадного алгоритма и адаптивного рандомизированного жадного алгоритма с разными параметрами

	22july	cond	auto40	caida	cnr	eu-2005	in-2004
RG(1)	0.65281	0.00012	0.78944	0.01938	0.90237	0.92765	0.00026
RG(3)	0.64658	0.19727	0.79988	0.81101	0.91192	0.92559	0.97836
RG(10)	0.64024	0.70738	0.80417	0.79883	0.91144	0.91780	0.97185
RG(50)	0.63479	0.69403	0.80273	0.79300	0.90997	0.90416	0.97596
ARG(0)	0.64262	0.71129	0.80136	0.79970	0.91028	0.91062	0.97614
ARG(0.01)	0.64041	0.71232	0.80145	0.80114	0.91041	0.91047	0.97615
ARG(0.05)	0.64264	0.71193	0.80174	0.80216	0.91039	0.91048	0.97616
ARG(0.1)	0.64134	0.69749	0.80152	0.80059	0.91108	0.91199	0.97618
ARG(0.2)	0.64192	0.56631	0.80102	0.80176	0.91075	0.91242	0.97588

Таблица 2: Время работы (в миллисекундах) рандомизированного жадного алгоритма и адаптивного рандомизированного жадного алгоритма с разными параметрами

	22july	cond	auto40	caida	cnr	eu-2005	in-2004
RG(1)	177	58	4,652	852	26,083	202,188	9,208
RG(3)	189	184	4,591	9,114	26,056	200,686	487,953
RG(10)	231	463	6,017	10,244	27,137	207,689	553,196
RG(50)	464	931	12,558	15,217	33,592	246,170	607,408
ARG(0)	238	477	6,526	11,573	30,033	233,761	622,813
ARG(0.01)	241	477	6,807	11,607	30,465	226,869	625,124
ARG(0.05)	238	474	6,479	11,514	29,054	225,748	617,345
ARG(0.1)	233	476	6,428	11,509	29,971	226,427	640,454
ARG(0.2)	222	351	6,105	11,220	29,784	266,038	616,187

В большинстве случаев один или несколько параметров k дают рандомизированному жадному алгоритму лучшие результаты, чем все результаты адаптивного рандо-

мизированного алгоритма, однако адаптивный вариант даёт более стабильные результаты при не очень большом увеличении по времени. Можно заметить, что небольшие значения параметра β дают лучшие результаты, чем нулевое значение, а при более больших значениях параметра время работы алгоритма уменьшается, но незначительно. В таблицу не попали результаты алгоритмов со значениями β больше 0.2, где время работы действительно сильно снижалось, однако и значения модулярности получались слишком маленькими.

Адаптивный рандомизированный жадный алгоритм можно сравнить с рандомизированным жандым алгоритмом при $k = 10$, так как он тоже даёт стабильные результаты, в отличии от $k = 1,\ k = 3$. Однако заметно, что адаптивный алгоритм в большинстве случаев даёт бóльшую модулярность.

3. Ансамблевая стратегия

3.1. Адаптивный рандомизированный жадный алгоритм в ансамблевой стратегии

Ансамблевая стратегия была более подробно описана в подразделе 1.5, однако общая схема работы заключается в том, что сначала s начальных алгоритмов выделяют сообщества, а те узлы, в которых они разошлись по мнению, затем разбивает на сообщества финальный алгоритм. Трудоёмкость таким образом складывается из s трудоёмкостей начальных алгоритмов и трудоёмкости финального алгоритма. Таким образом имеет смысл в качестве начальных алгоритмов брать алгоритмы, работающие быстро. Однако в случае, если начальные алгоритмы очень плохие, то есть дают разбиения не лучше случайного — модулярность ансамблевой стратегии будет приблизительно равна модулярности финального алгоритма.

Далее разбиение, полученное начальными алгоритмами называется *начальным* разбиением, полученное финальным алгоритмом — *финальным* разбиением. Также разбиение, равное максимальному перекрытию начальных разбиений далее называется *промежуточным*.

Если промежуточное разбиение состоит из достаточно большого количества сообществ (как минимум в несколько раз больше 2σ) — на нём разумно использовать в качестве финального алгоритма адаптивную версию рандомизированного жадного алгоритма, чтобы получать стабильные результаты. Однако на небольших графах адаптивный алгоритм не имеет смысла.

Так же, если граф, подающийся на вход, достаточно большой — в качестве начального алгоритма на нём имеет смысл использовать адаптивный алгоритм.

В таблице 3 сравниваются результаты работы ансамблевой стратегии с разными начальными и финальными алгоритмами. В качестве начального алгоритма используюся два рандомизированных жадных алгоритма с параметрами $k = 3$ (обозначается RG_3) и $k = 10$ (обозначается RG_{10}), и адаптивный рандомизированный жадный алгоритм с параметрами $d = 5,\ f(\mu, k) = -10(\ln \mu - 0.02 \ln k),\ \sigma = 1000,\ \hat{k}_0 = 8$ (обозначается ARG). А в качестве финального алгоритма используются только рандомизированные жадные алгоримы с k равными 3 и 10, так как на рассматриваемых графах промежуточное разбиение состоит из недостаточно большого количества сообществ, чтобы использовать адаптивный рандомизированный алгоритм.

Ансамблевая стратегия используется с параметром $s = 10$, а обозначается как $ES(init, final)$, где в первом аргументе указывается начальный алгоритм, а во втором — финальный. Алгоритмы на графе *cond-mat-2003* запускались 21 раз, на *caidaRouterLevel* и *auto40* — три раза.

Таблица 3: Сравнение модулярности и времени ансамблевой стратегии с $RG(3)$, $RG(10)$ и ARG в качестве начальных алгоритмов и $RG(3)$ и $RG(10)$ в качестве финальных алгоритмов

	cond-mat-2003		auto40		caidaRouterLevel	
	Q	t	Q	t	Q	t
$ES(RG_3, RG_3)$	0.16840	2.0	0.806277	47.0	0.84078	94.0
$ES(RG_{10}, RG_3)$	0.44934	4.7	0.806325	57.7	0.84031	104.2
$ES(ARG, RG_3)$	0.42708	4.8	0.806276	65.7	0.83671	115.1
$ES(RG_3, RG_{10})$	0.71155	2.3	0.806445	46.6	0.85372	93.7
$ES(RG_{10}, RG_{10})$	0.74794	4.8	0.806445	57.2	0.84448	104.3
$ES(ARG, RG_{10})$	0.74872	4.9	0.806472	65.8	0.85279	118.5

Граф *cond-mat-2003* плохо разбивается случайными жадными алгоритмами с маленьким k, если обратить внимание на таблицу 3 — в случаях, когда RG_3 используется в качестве начального алгоритма — модулярность ансамблевой стратегии сравнима с модулярностью финального алгоритма на этом графе (можно увидеть в таблице 1). В случаях, когда RG_3 используется в качестве финального алгоритма — модулярность получается сравнительно плохой.

Заранее неизвестно, при каких k алгоритм будет хорошо работать, а при каких плохо, кроме того неизвестно по результату, хорошее ли это разбиение для конкретного графа или нет. Кроме того, ансамблевая стратегия работает заметно дольше, чем рандомизированный жадный алгоритм, поэтому запускать ансамблевую стратегию несколько раз для определения хороших параметров не выгодно. Таким образом, выгодно использовать адаптивный рандомизированный жадный алгоритм и в качестве начального алгоритма, и в качестве финального, если промежуточное разбиение подходит по размерам.

3.2. Адаптивное построение промежуточного разбиения

Так как адаптивный рандомизированный жадный алгоритм не может работать на графах с маленьким количеством узлов или на разбиениях с маленьким количеством сообществ — была предложена другая схема адаптивной ансамблевой стратегии:

1. На вход подаётся граф G. Выбор параметров алгоритма: $d \in \mathbb{N}$ — размер возмущения, $f(Q, k) : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$ — функция качества, $l_1 \in \mathbb{N}$ — количество шагов, $l_2 \leq l_1 \in \mathbb{N}$ — количество лучших начальных разбиений, учавствующих в создании промежуточного разбиения. И наконец $\hat{k}_0 \in \mathbb{N}$ — начальная центральная точка. Установка счётчика $n = 0$. Установка список начальных разбиений $S = \emptyset$

2. Увеличение счётчика $n \to n + 1$

3. Вычисление следующий параметров рандомизированного жадного алгоритма $k_n^- = \max\{1, \hat{k}_{n-1} - d\}$ и $k_n^+ = \hat{k}_{n-1} + d$

4. Выделение сообществ в графе G рандомизированным жадным алгоритмом с параметром $k = k_n^-$, пара из результирующего разбиения P_n^- и его модулярности Q_n^- записывается в список S

5. Выделение сообществ в графе G рандомизированным жадным алгоритмом с параметром $k = k_n^+$, пара из результирующего разбиения P_n^+ и его модулярности Q_n^+ записывается в список S

6. Вычисляются функции качества $y_n^- = f(Q_n^-, k_n^-),\ y_n^+ = f(Q_n^+, k_n^+)$

7. Вычисляется следующая центральная точка аналогично (5)

$$\hat{k}_n = \max\left\{1, \left[\hat{k}_{n-1} - \frac{y_n^+ - y_n^-}{k_n^+ - k_n^-}\right]\right\} \tag{6}$$

8. Если $n \neq l_1$ — переход на второй пункт, иначе — на следующий

9. По l_2 лучшим разбиениям (с наибольшей модулярностью) в S создаётся промежуточное множество, которое затем передаётся в финальный алгоритм

При этом граф G может быть иметь сколь угодно маленькое количество узлов.

3.3. Количество начальных разбиений

Список литературы

[1] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[2] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 251–262. ACM, 1999.

[3] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[4] Cristopher Moore and Mark EJ Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61(5):5678, 2000.

[5] Jing Zhao, Hong Yu, Jianhua Luo, ZW Cao, and Yixue Li. Complex networks theory for analyzing metabolic networks. *Chinese Science Bulletin*, 51(13):1529–1537, 2006.

[6] Wang Hong, Wang Zhao-wen, Li Jian-bo, and Qiu-hong Wei. Criminal behavior analysis based on complex networks theory. In *IT in Medicine & Education, 2009. ITIME'09. IEEE International Symposium on*, volume 1, pages 951–955. IEEE, 2009.

[7] John Scott. *Social network analysis*. Sage, 2012.

[8] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[9] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, Feb 2004.

[10] Stefanie Muff, Francesco Rao, and Amedeo Caflisch. Local modularity measure for network clusterizations. *arXiv preprint cond-mat/0503252*, 2005.

[11] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[12] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2):172–188, 2008.

[13] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, Jun 2004.

[14] Michael Ovelgönne and Andreas Geyer-Schulz. A comparison of agglomerative hierarchical algorithms for modularity clustering. In *Challenges at the Interface of Data Analysis, Computer Science, and Optimization*, pages 225–232. Springer, 2012.

[15] Michael Ovelgönne and Andreas Geyer-Schulz. Cluster cores and modularity maximization. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 1204–1213. IEEE, 2010.

[16] Michael Ovelgönne and Andreas Geyer-Schulz. An ensemble learning strategy for graph clustering. *Graph Partitioning and Graph Clustering*, 588:187, 2012.

[17] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[18] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[19] Julius R Blum. Multidimensional stochastic approximation methods. *The Annals of Mathematical Statistics*, pages 737–744, 1954.

[20] James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[21] James C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[22] Boris T. Polyak. *Introduction to optimization*. Optimization Software New York, 1987.

[23] Oleg Granichin and Natalia Amelina. Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances. *IEEE Transactions on Automatic Control*, 60(5), 2015.

[24] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

[25] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[26] Daniel Baird and Robert E Ulanowicz. The seasonal dynamics of the Chesapeake Bay ecosystem. *Ecological Monographs*, pages 329–364, 1989.

[27] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[28] 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.

[29] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2):027104, 2005.

[30] Pablo M. Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[31] Roger Guimerà Manrique, L Danon, Albert Díaz Guilera, Francesc Giralt, and Àlex Arenas. Self-similar community structure in a network of human interactions. *Physical Review E, 2003, vol. 68, núm. 6, p. 065103-1-065103-4*, 2003.

[32] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[33] Marián Boguñá, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70:056122, Nov 2004.

[34] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[35] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

[36] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*. ACM Press, 2011.

[37] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.