

Адаптивный рандомизированный алгоритм выделения сообществ в графах

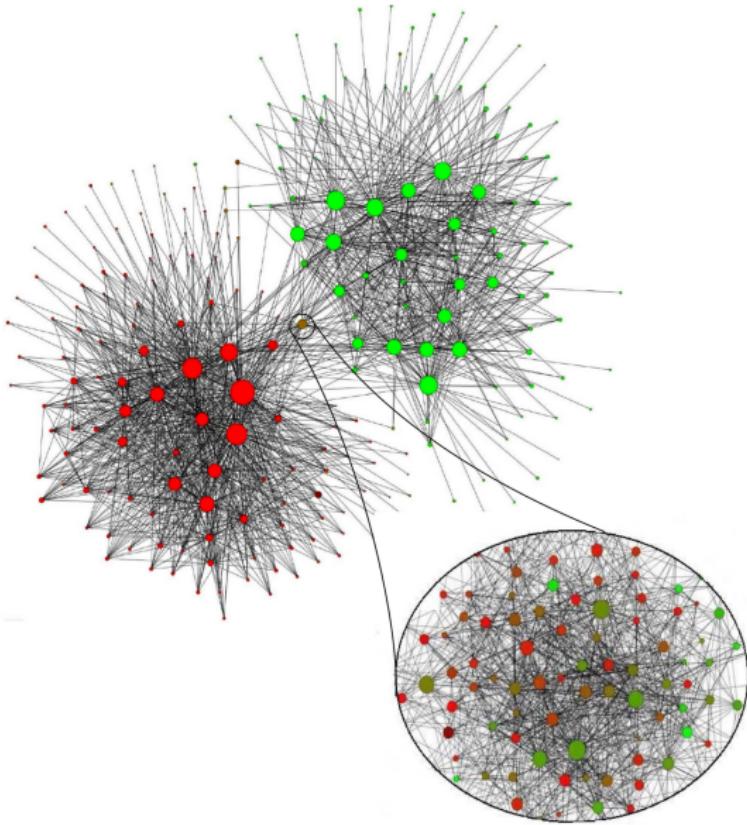
Тимофей Проданов

Санкт-Петербургский государственный университет
Кафедра информатики

Научный руководитель: д.ф.-м.н., проф. Границин О. Н.
Рецензент: Ерофеева В. А.

Санкт-Петербург
2015

Пример разбиения на сообщества



Сообщества — тесно связанные группы узлов в графах
2004 г. Модулярность:

$$Q(G, P) = \sum_{i \in 1..K} (e_{ii} - a_i^2),$$

где

- G — граф
- P — разбиение на сообщества
- K — количество сообществ
- e — нормированная матрица смежности сообществ
- a — вектор нормированных степеней сообществ

Алгоритмы выделения сообществ

2010 г. Рандомизированный жадный алгоритм RG^1 :

- k случайных сообществ и их соседи
- Лучшая пара соседей

2012 г. Схема кластеризации основных групп графа $CGGC^2$:

1. *s начальных алгоритмов*
2. *Финальный алгоритм* распределяет неопределившиеся узлы

Итеративная схема $CGGCI$

RG как начальный и финальный алгоритм

¹Ovelgonne and Geyer-Schulz. *Cluster cores and modularity maximization*. ICDMW'10. IEEE International Conference on Data Mining Workshops. pp. 1204–1213 (2010)

²Ovelgonne and Geyer-Schulz. *A comparison of agglomerative hierarchical algorithms for modularity clustering*. Graph Partitioning and Graph Clustering. 588:187, 2012

Качество работы RG и $CGGC$ зависит от их параметров

Одновременно возмущаемая стохастическая аппроксимация $SPSA$:

- Разбиение алгоритма на $2n$ шагов
- Каждый шаг новый параметр
- Каждые два шага вычисляются следующие два значения параметра

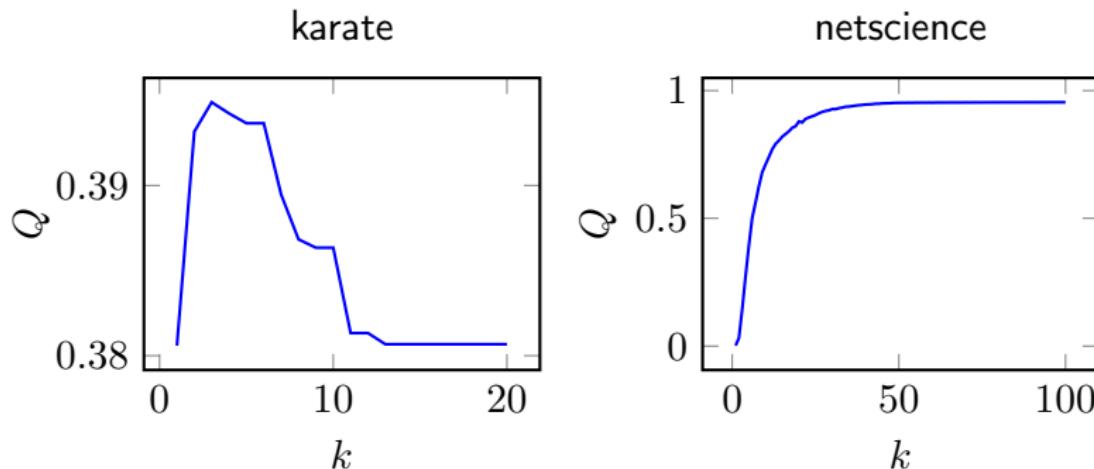
Хорошие результаты на большем количестве входных данных

Применение $SPSA$ к RG и $CGGC$

Применимось SPSA. Функция качества

Выпуклая усреднённая функция качества

Рандомизированный жадный алгоритм RG :



Время растёт линейно от k

Функция качества:

$$f(Q, k) = -\alpha(\ln Q - \beta \ln k), \quad \alpha > 0, \quad \beta \geq 0$$

Адаптивный рандомизированный жадный алгоритм *ARG*:

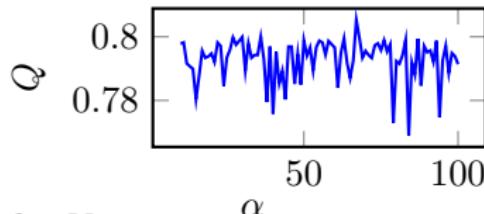
- Шаги по σ итераций
- После окончания шага k меняется
- Функция качества $f(\mu, k) = -\alpha(\ln \mu_n^- - \beta \ln k_n^-)$,
 μ — медиана прироста модулярности
- \hat{k}_{n-1} — текущая оценка, её возмущение:
 $k_n^- = \max\{\hat{k}_{n-1} - d, 1\}$ и $k_n^+ = \hat{k}_{n-1} + d$
- Следующая оценка:

$$\hat{k}_n \leftarrow \max \left\{ 1, \left[\hat{k}_{n-1} - \frac{f_n^+ - f_n^-}{k_n^+ - k_n^-} \right] \right\}$$

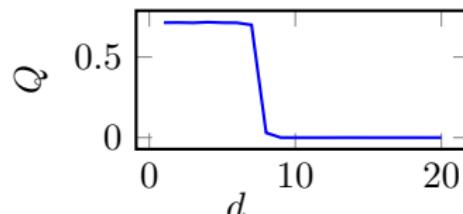
Параметры ARG

ARG имеет пять параметров:

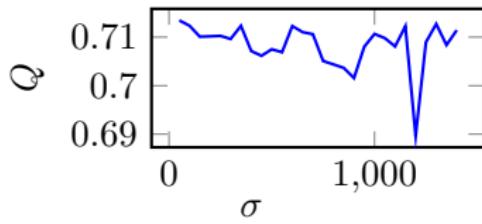
- $\alpha > 0 \in \mathbb{R}$



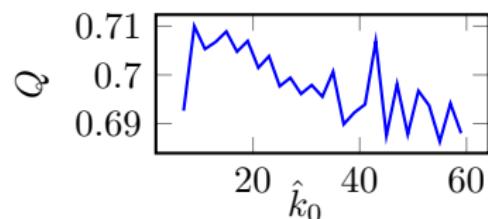
- $d \in \mathbb{N}$



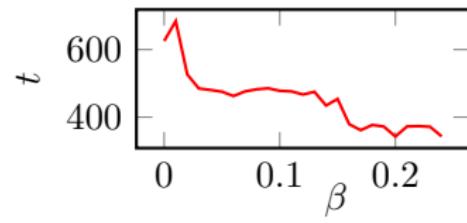
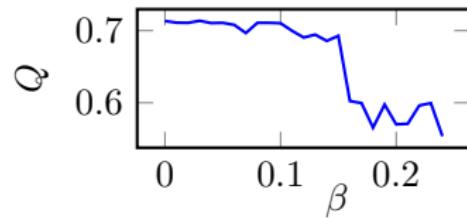
- $\sigma \in \mathbb{N}$



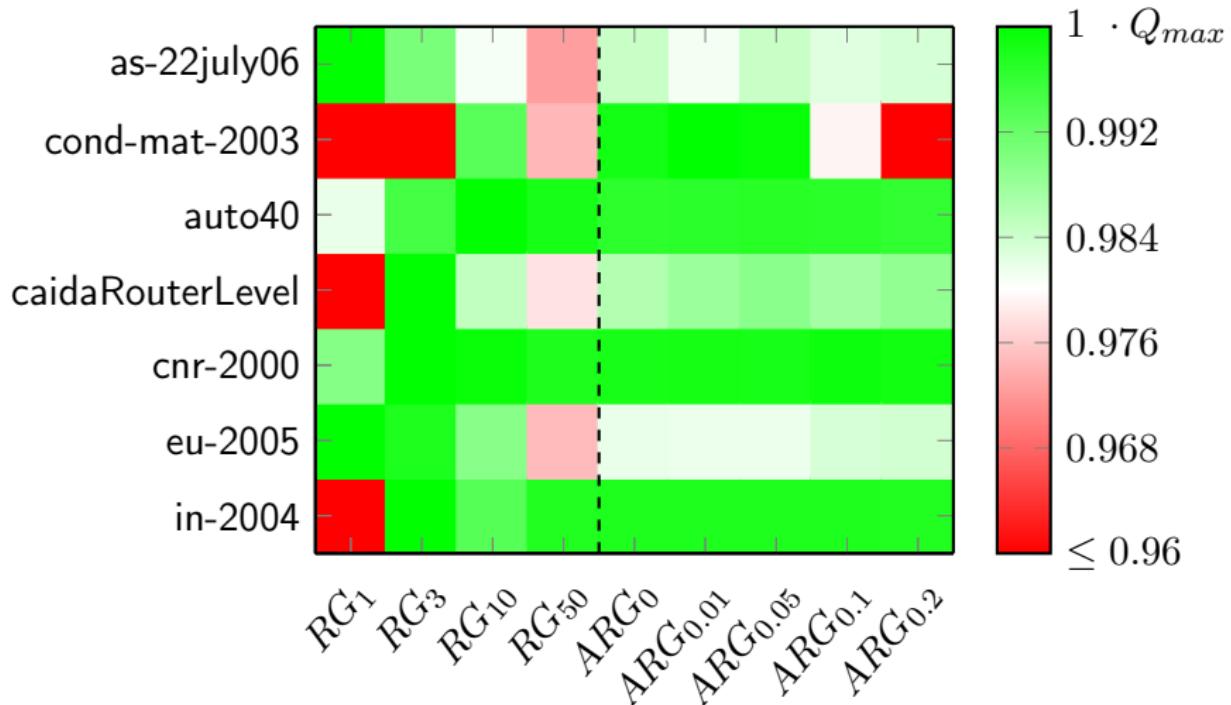
- $\hat{k}_0 \in \mathbb{N}$



- $\beta \geq 0 \in \mathbb{R}$



Сравнение RG и ARG



Если начальные алгоритмы дают плохой результат в схеме $CGGC$, то и окончательный результат будет плохим. По модулярности одного разбиения нельзя сказать, хорошее разбиение или плохое. Запускать RG_k с разными параметрами для проверки качества разбиения не выгодно. Поэтому имеет смысл запускать более стабильный ARG .

Таблица: Модулярность разбиений, полученных в результате работы $CGGC$ с начальным алгоритмом A_{init} и финальным алгоритмом A_{final} на трёх графах

A_{init}		RG_3		RG_{10}		ARG	
A_{final}		RG_3	RG_{10}	RG_3	RG_{10}	RG_3	RG_{10}
G_1	0.16840	0.71155	0.44934	0.74794	0.42708	0.74872	
G_2	0.80628	0.80645	0.80633	0.80645	0.80628		0.80647
G_3	0.84078	0.85372	0.84031	0.84448	0.83671		0.85279

Схема *ACGGC*:

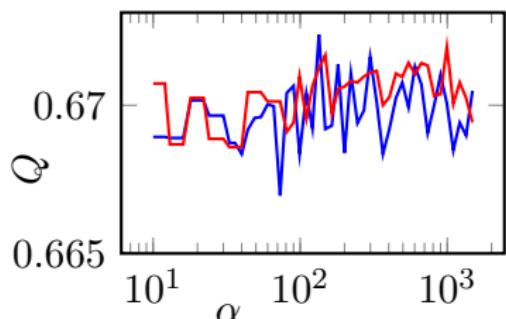
1. $n \leftarrow n + 1$
2. Вычисляются аргументы функционала
 $k_n^- \leftarrow \max\{1, \hat{k}_{n-1} - d\}$ и $k_n^+ \leftarrow \hat{k}_{n-1} + d$
3. $RG_{k_n^-}$ и $RG_{k_n^+}$ создают разбиения P_n^- и P_n^+ , они записываются в множество S , их модулярности Q_n^- и Q_n^+
4. Вычисляются значения функционала
 $y_n^- \leftarrow f(Q_n^-, k_n^-)$, $y_n^+ \leftarrow f(Q_n^+, k_n^+)$
5. Рассчитывается следующая оценка:

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

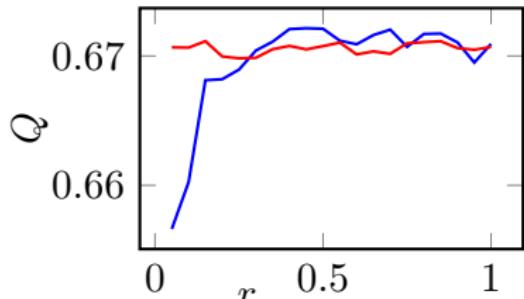
6. При $n \neq l$ — переход на 1 шаг, иначе переход на 7 шаг
7. В зависимости от $r \in (0, 1]$ выбирается несколько хороших разбиений из S , которые формируют максимальное перекрытие \tilde{P}
8. Финальный алгоритм выделяет сообщества на основе \tilde{P}

Параметры CGGC

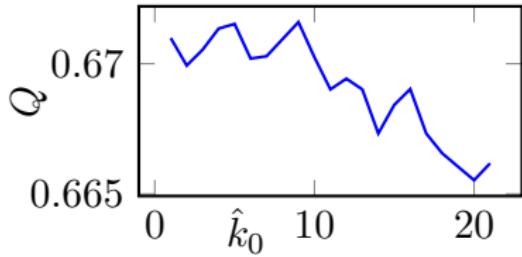
- $d \in \mathbb{N}, \alpha > 0 \in \mathbb{R}$



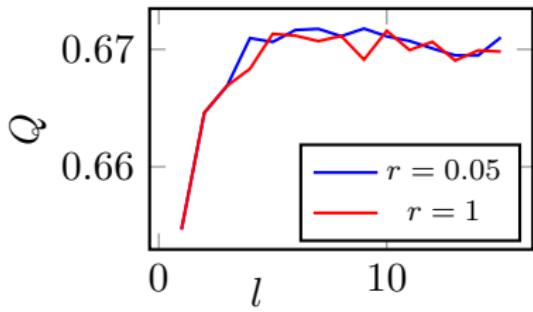
- $r \in (0, 1]$



- $\hat{k}_0 \in \mathbb{N}$

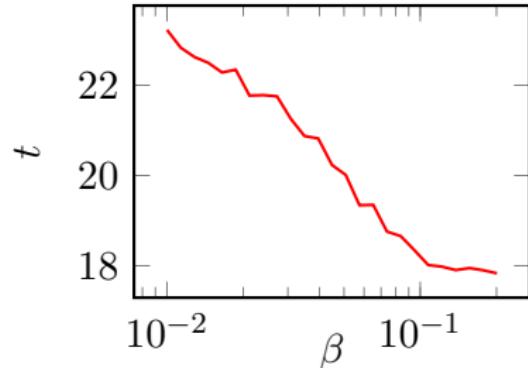
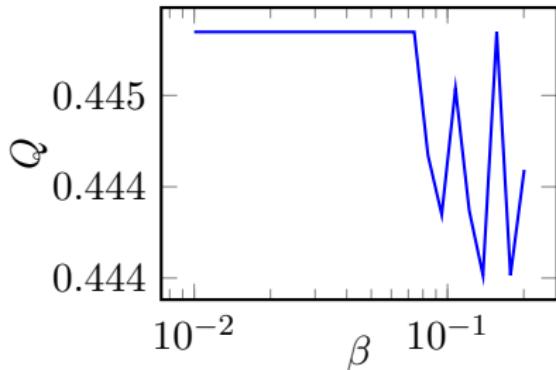


- $l \in \mathbb{N}$



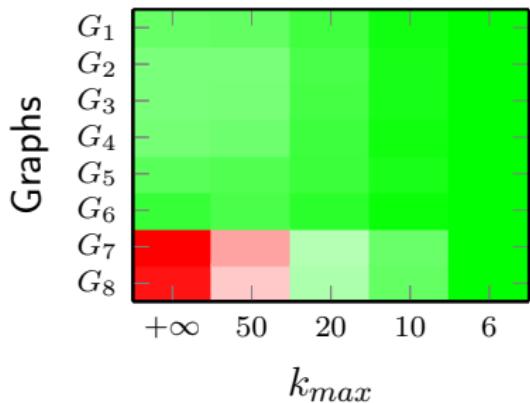
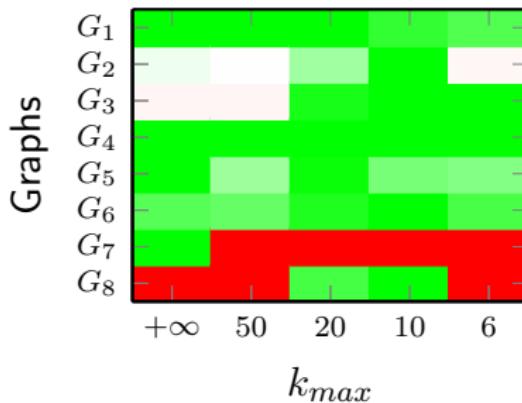
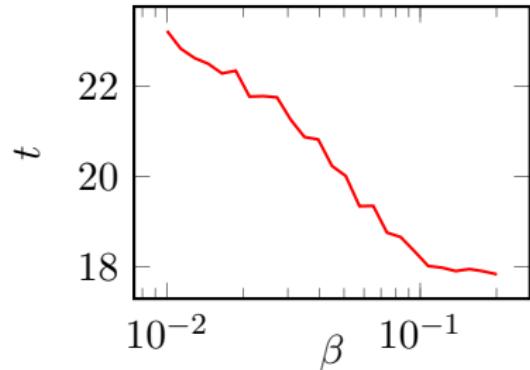
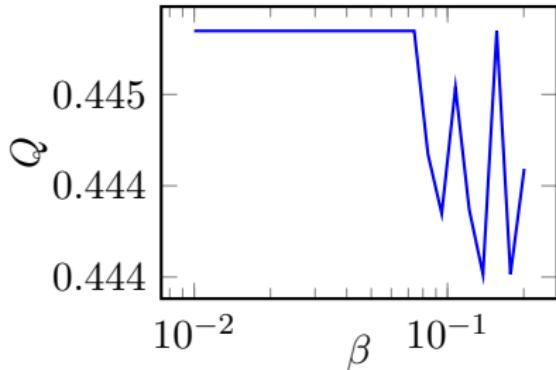
Время работы

Два механизма снижения времени работы:



Время работы

Два механизма снижения времени работы:



Сравнение $CGGC$ и $ACGGC$

Таблица: Модулярность разбиений, полученных $ACGGC$ и $CGGC$ на тестовых графах

	$ACGGC^I$	$ACGGC^{II}$	$CGGC_{10}^{10}$	$CGGC_3^{10}$	$CGGC_{10}^3$
karate	0.417242	0.417406	0.415598	0.396532	0.405243
dolphins	0.524109	0.523338	0.521399	0.523338	0.522428
chesapeake	0.262439	0.262439	0.262439	0.262439	0.262370
adjnoun	0.299704	0.299197	0.295015	0.292703	0.290638
polbooks	0.527237	0.527237	0.527237	0.526938	0.526784
football	0.603324	0.604266	0.604266	0.599537	0.599026
celegans	0.439604	0.438584	0.435819	0.436066	0.432261
jazz	0.444739	0.444848	0.444871	0.444206	0.444206
netscience	0.907229	0.835267	0.724015	0.708812	0.331957
email	0.573333	0.573409	0.571018	0.572667	0.567423
polblogs	0.424107	0.423208	0.422901	0.421361	0.390395
pgpGiantCompo	0.883115	0.883085	0.882237	0.882532	0.880340
as-22july06	0.671249	0.670677	0.666766	0.669847	0.665260
cond-mat-2003	0.744533	0.750367	0.751109	0.708775	0.413719
caidaRouterLevel	0.846312	0.855651	0.851622	0.858955	0.843835
cnr-2000	0.912762	0.912783	0.912500	0.912777	0.912496
eu-2005	0.938292	0.936984	0.935510	0.936515	0.936420
in-2004	0.979844	0.979771	0.979883		

Сравнение $CGGCi$ и $ACGGCi$

Таблица: Модулярность работы четырёх итеративных алгоритмов на небольших тестовых графах

	$ACGGCi^I$	$ACGGCi^{II}$	$CGGCi$	combined
karate	0.417242	0.417406	0.417242	0.417242
dolphins	0.525869	0.525869	0.525869	0.525869
chesapeake	0.262439	0.262439	0.262439	0.262439
adjnoun	0.303731	0.303504	0.303571	0.303970
polbooks	0.527237	0.527237	0.527237	0.527237
football	0.604266	0.604407	0.604429	0.604407
celegans	0.446964	0.446836	0.445442	0.447234
jazz	0.444871	0.444871	0.444871	0.444871
netscience	0.908845	0.888422	0.725781	0.907443
email	0.576778	0.577000	0.576749	0.577110
polblogs	0.424025	0.422920	0.423281	0.423996

В рамках работы

- Проанализированы современные методы выделения сообществ
- Предложен алгоритм ARG , дающий более стабильные результаты, чем RG
- Описано возможное применение ARG
- Представлены схемы $ACGGC$ и $ACGGCi$, показывающие более хорошие и стабильные результаты, чем $CGGC$ и $CGGCI$
- Описан механизм снижения времени работы $ACGGC$

Вопросы?