Доклад о современных методах выделения сообществ в социальных графах

Спецсеминар "Алгебра над алгоритмами и эвристиеский поиск закономерностей"

Евгений Никишин

ВМК МГУ

25.11.2015

1. Ликбез
2. A General Optimization Technique for High Quality Community Detection in Complex Networks
3. High Quality, Scalable and Parallel Community Detection for Large Real Graphs
4. Efficient Community Detection in Large Networks using Content and Links

- Понятие сообщества неформально: связи внутри сообщества плотнее межсообщественных
- Критерий качества: модулярность

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

где A - матрица смежности, m - число ребер, d - степень, C - сообщество

1. Ликбез
2. A General Optimization Technique for High Quality Community Detection in Complex Networks
3. High Quality, Scalable and Parallel Community Detection for Large Real Graphs
4. Efficient Community Detection in Large Networks using Content and Links

A General Optimization Technique for High Quality Community Detection in Complex Networks

Многие стратегии выделения сообществ используют:

- Соединение двух сообществ в одно
- Разбиение сообщества на два
- Перемещение вершин из одного сообщества в другое

Будем использовать все три

A General Optimization Technique for High Quality Community Detection in Complex Networks

```
Procedure PerformKernighanLinShifts(origin, dest)
    Calculate gains from moving each node to opposite community;
    for i ← 1 to size of origin community do
        Perform temporary movement that produces maximal gain;
        Remember current gain and moved node;
        Recalculate all gains;
    Retrieve the movements leading to a maximal gain among intermediately calculated
    and perform them;
```

Пояснения:

- Создаем список из всех вершин origin
- После перемещения вершины она удаляется из списка

A General Optimization Technique for High Quality Community Detection in Complex Networks

```
Procedure ReCalculateGain(origin, dest)
    if dest is new community and we already have max_communities then
        return;
    Define and initialize number_of_tries;
    for tryI ← 1 to number_of_tries do
        foreach vertex v from origin community do
            move v to dest or leave in origin with equal probability;
        Calculate new gain, assign zero to previous gain;
        while new gain > previous gain do
            PerformKernighanLinShifts(origin, dest);
        if achieved gain is greater then current maximum then
            Remember current partition and gain;
```

Пояснения:

- number_of_tries нужен для компенсации случайности при выборе вершин
- PerformKernighanLinShifts считает new gain

A General Optimization Technique for High Quality Community Detection in Complex Networks

```
Initialize variables for storing partitions and their gains;
for each pair (origin, dest) of communities do // dest may be empty community
    // Calculate best gain from moving nodes from origin to dest
    ReCalculateGain(origin, dest);
while BestGain() > THRESHOLD do
    PerformMove(best_origin, best_dest, best_partition);
    // Update gains for changed communities
    for each community i do
        ReCalculateGain(best_origin, i); ReCalculateGain(i, best_origin);
        ReCalculateGain(best_dest, i); ReCalculateGain(i, best_dest);
```

```
Procedure PerformMove(origin, dest, partition)
    Move nodes from origin to dest according to partition;
Procedure BestGain()
    Select from remembered partitions one with the best gain;
    Return this gain and corresponding best_origin, best_dest and best_partition;
```

A General Optimization Technique for High Quality Community Detection in Complex Networks

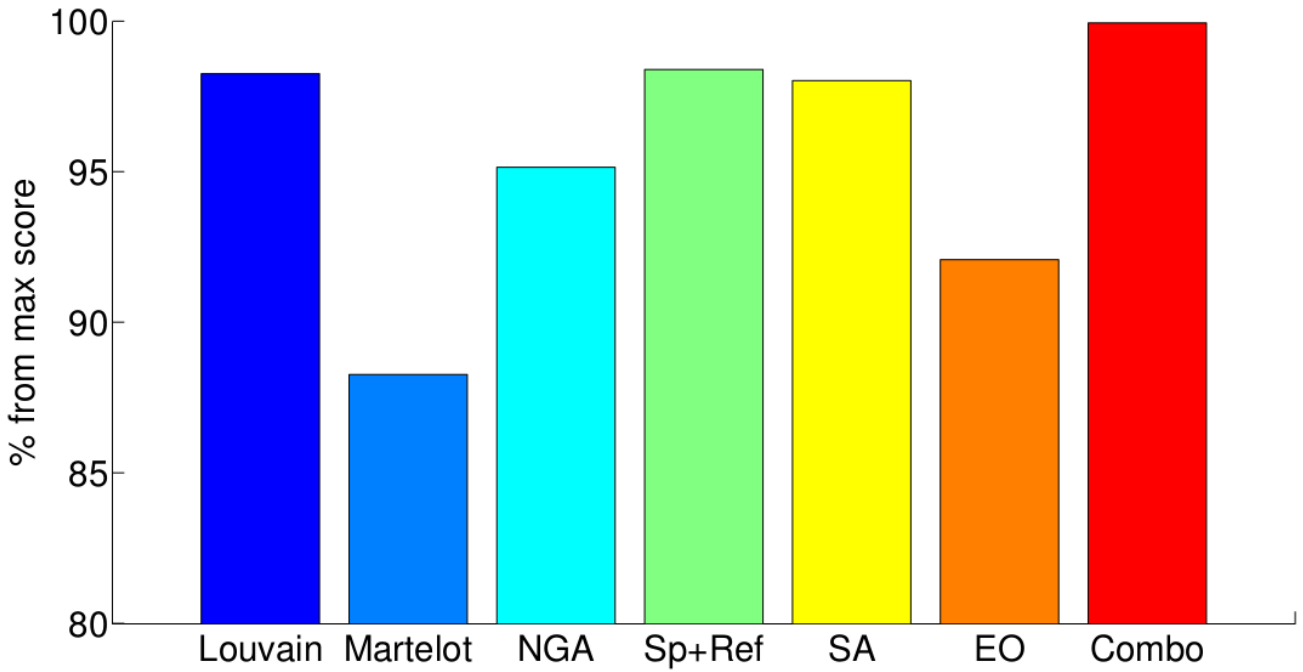


FIG. 3. (Color online) Performance of algorithms as average percent of their resulting modularity score to the maximum, achieved by the best algorithm.

Тестировалось как по реальным, так и по искуственным данным

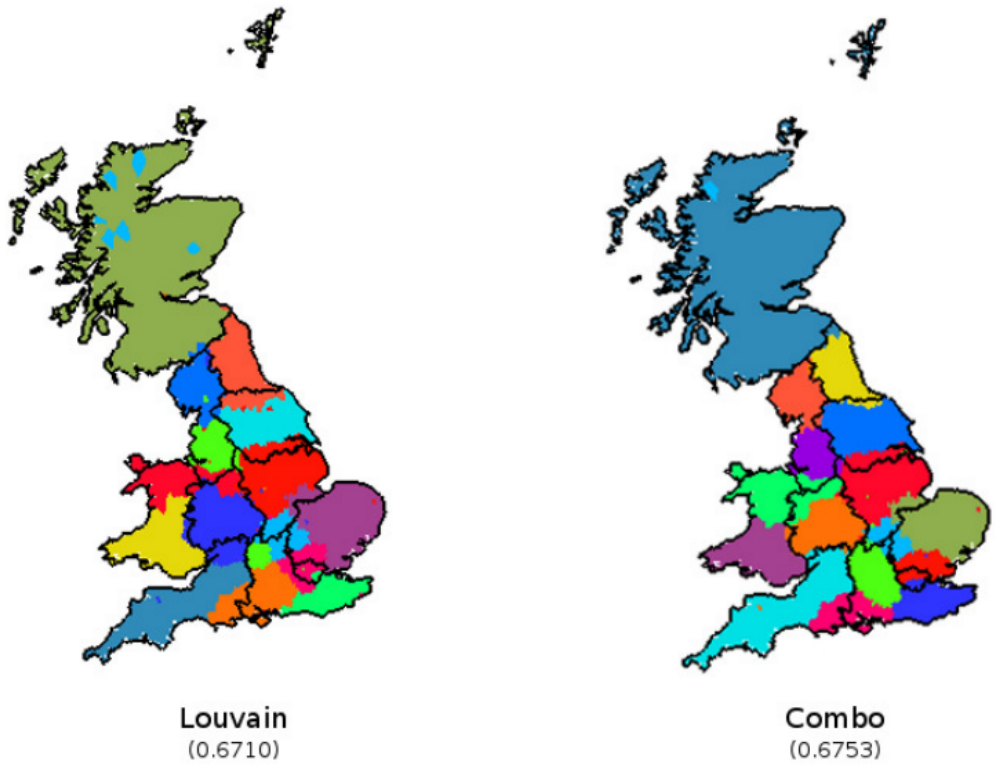


FIG. 5. (Colored online) Partitioning of a network based on the number of all calls entertained between each pair of locations. A minimal variation in modularity (less than 1 percent) can turn into a sizable difference in partitioning. Here the Combo results show cleaner geographical separation of communities and are substantially more similar to official administrative divisions with modularity equal to 0.6753 and NMI = 0.804, compared to modularity = 0.6710 and NMI = 0.703 for Louvain.

1. Ликбез
2. A General Optimization Technique for High Quality Community Detection in Complex Networks
3. High Quality, Scalable and Parallel Community Detection for Large Real Graphs
4. Efficient Community Detection in Large Networks using Content and Links

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

Специальная метрика WCC:

- Базируется на идее о том, что вероятность образования треугольника из вершин в сообществе выше, чем между сообществами
- Дан граф G(V, E)
- t(x, C) — количество треугольников, которые образованы вершинами C, содержащие x
- vt(x, C) — количество вершин в C, которые содержатся в треугольнике с x и любой другой вершиной в графе
- Непосредственно метрика:

$$WCC(x, C) = \frac{t(x, C)}{t(x, V)} \cdot \frac{vt(x, V)}{|C \setminus \{x\}| + vt(x, V \setminus C)}$$

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

Инициализация:

- Препроцессинг: уберем все ребра, которые не входят в треугольники
- Построим начальное разбиение очень простым алгоритмом (сортируем вершины по коэффициенту кластеризации, затем для списка пробегаем все непосещенные вершины, добавляя в сообщество всех соседей просматриваемой вершины)

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

```
   Data: Given a graph G(V,E) and a partition P
   Result: A refined partition P'
 1 newP ← P;
 2 newWCC ← computeWCC(P);
 3 repeat
 4 |   WCC' ← newWCC;
 5 |   P' ← newP;
 6 |   M ← ∅;
 7 |   foreach v in V do
 8 |   |   M.add( bestMovement(v,P') );
 9 |   end
10 |   newP ← applyMovements(M,P');
11 |   newWCC ← computeWCC(newP);
12 until (newWCC − WCC')/WCC' ≥ t;
13 return P';
```

Algorithm 2: Phase 2, refinement.

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

Пояснения:

- Авторы советуют выбирать порог 1%
- bestmovement не изменяет разбиение и может быть таким:
  1. Оставить как есть
  2. Изъять из сообщества и переместить в сообщество только из этой вершины
  3. Переместить из одного сообщества в другое

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

```
Data: Given a graph G(V,E) a partition P
      and a vertex v
Result: Computes the best movement of v.
1  m ← [NO_ACTION];
2  sourceC ← GetCommunity(v,P);
3  wcc_r ← WCC_R(v,sourceC);
4  wcc_t ← 0.0;
5  bestC ← ∅;
6  Candidates ← candidateCommunities(v,P);
7  for c in Candidates do
8  |   aux ← WCC_T(v,sourceC,c) ;
9  |   if aux > wcc_t then
10 |   |   wcc_t ← aux;
11 |   |   bestC ← c;
12 |   end
13 end
14 if wcc_r > wcc_t and wcc_r > 0.0 then
15 |   m ← [REMOVE];
16 else if wcc_t > 0.0 then
17 |   m ← [TRANSFER , bestC];
18 end
19 return m;
```

Algorithm 3: `bestMovement`.

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

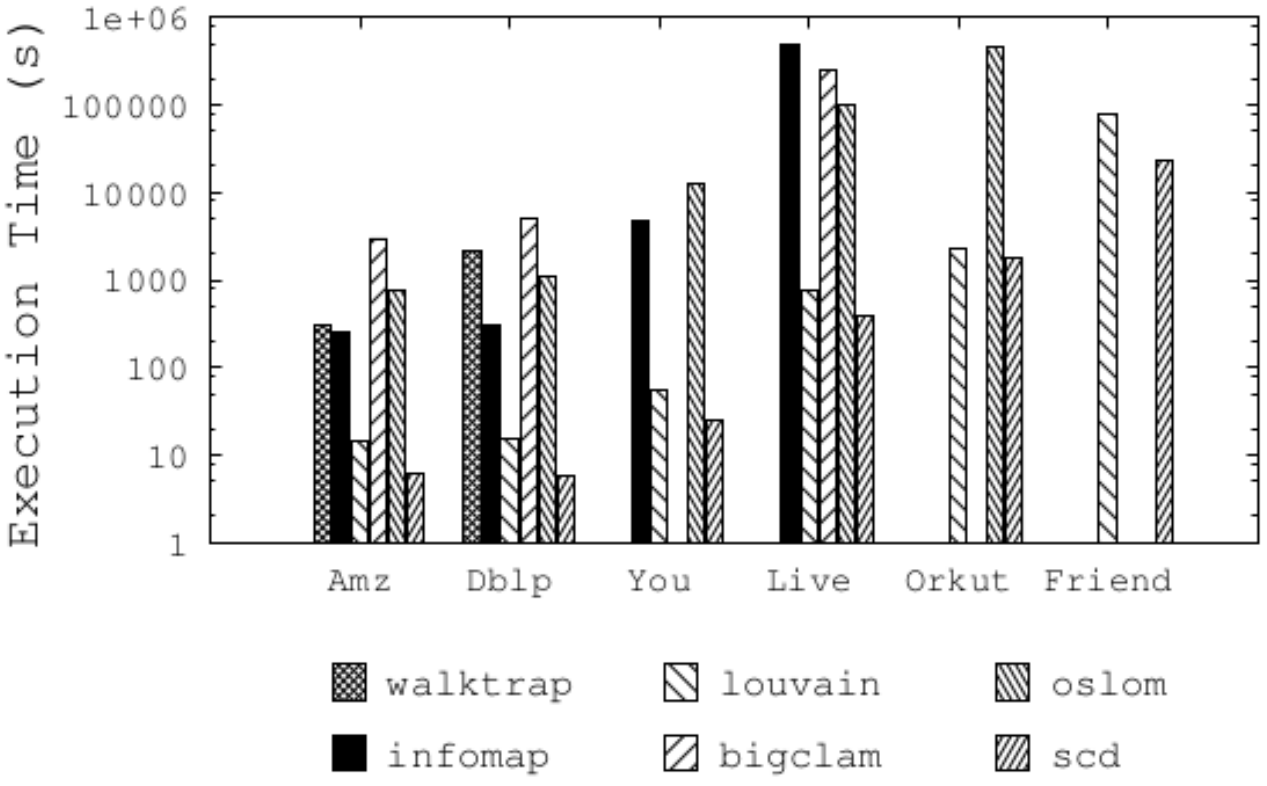


Figure 4: Execution times in seconds.

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

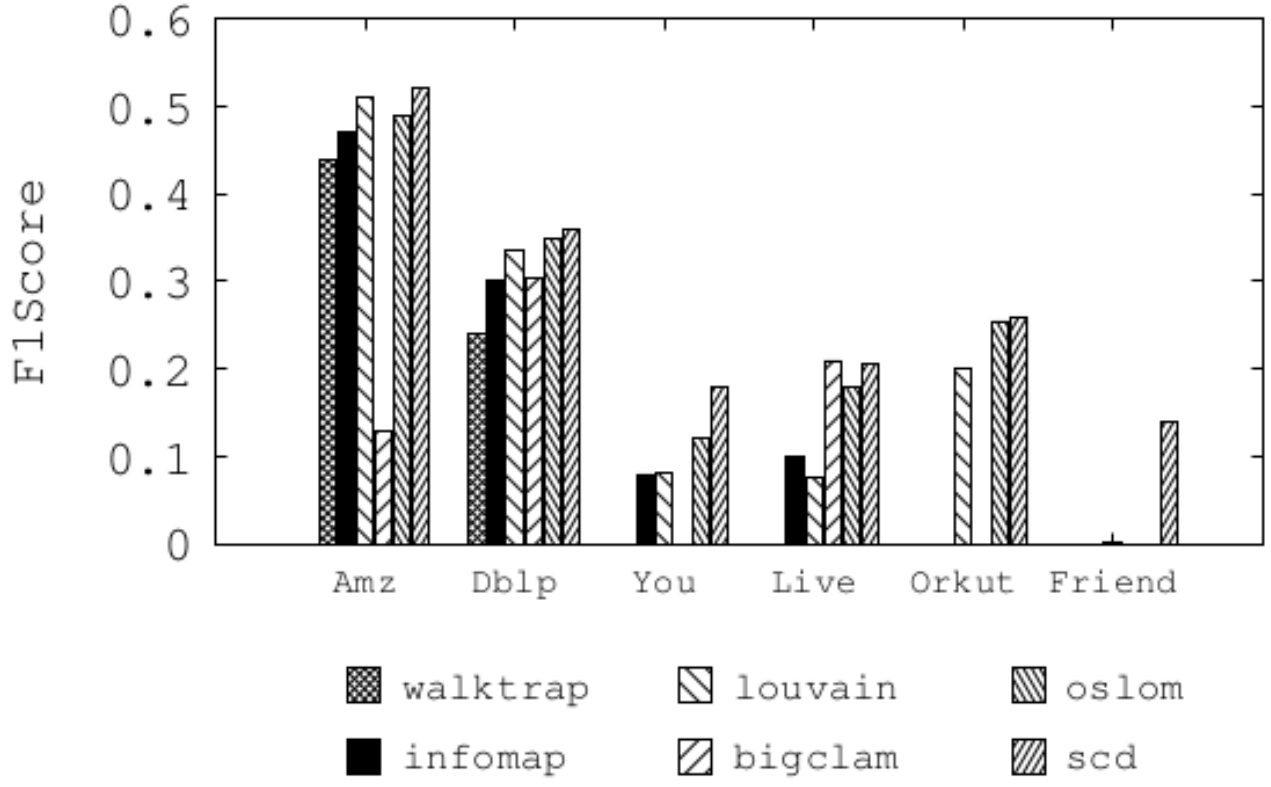


Figure 5: $\bar{F}_1$ score using ground truth.

High Quality, Scalable and Parallel Community Detection for Large Real Graphs

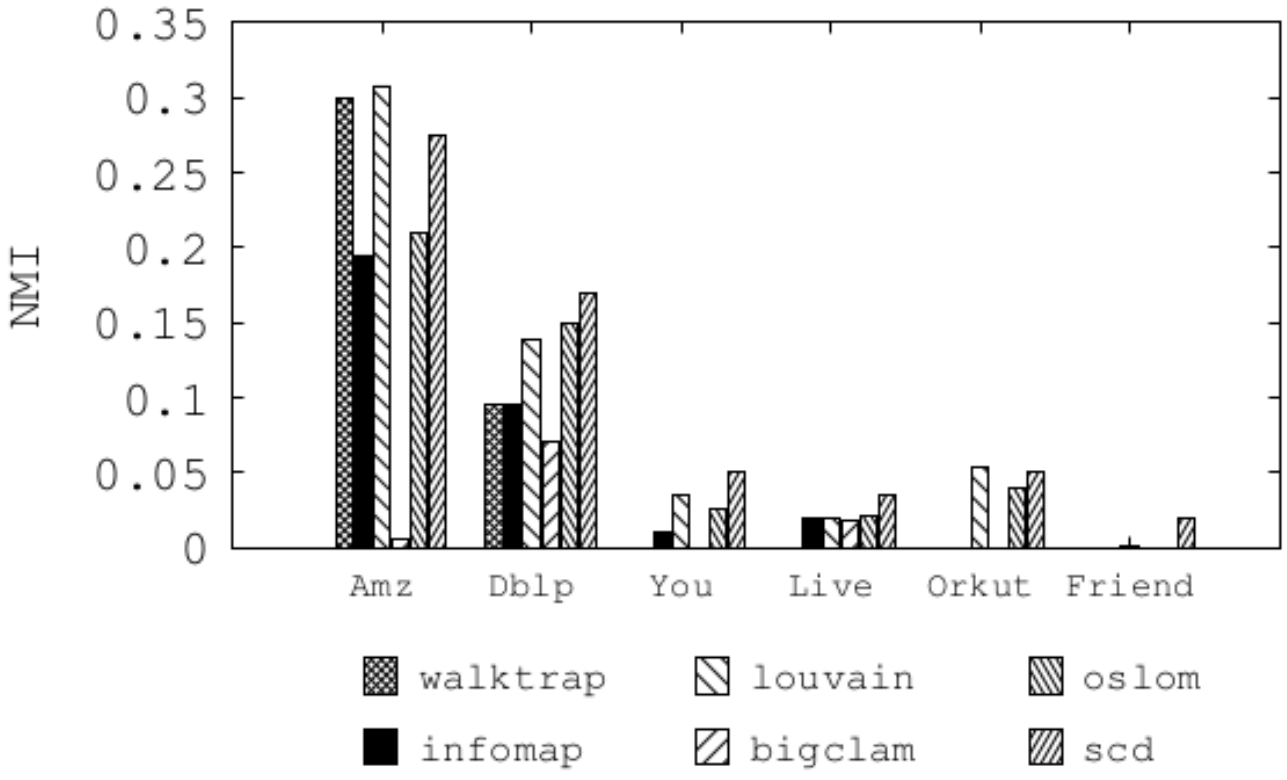
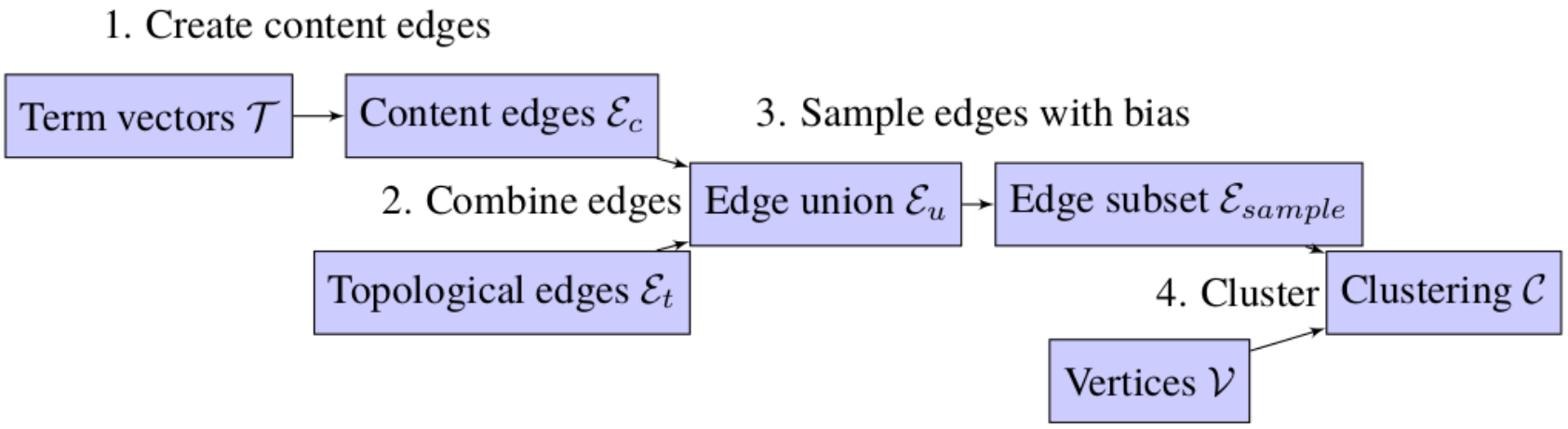


Figure 6: NMI using ground truth.

1. Ликбез
2. A General Optimization Technique for High Quality Community Detection in Complex Networks
3. High Quality, Scalable and Parallel Community Detection for Large Real Graphs
4. Efficient Community Detection in Large Networks using Content and Links

Efficient Community Detection in Large Networks using Content and Links

Пусть дан граф $\mathcal{G} = (\mathcal{V}, \mathcal{E}_t, \mathcal{T})$
$\mathcal{T} = \mathbf{t}_1, \ldots, \mathbf{t}_n$, элементами $\mathbf{t}_i$ являются единицы контента такие, как слова, тэги, n-граммы



Efficient Community Detection in Large Networks using Content and Links

Входные параметры алгоритма:

1. Граф
2. k — количество ближайших контекстных соседей к вершине
3. similarity(**x**, **y**)
4. normalize(**x**)
5. α — параметр, уточняющий веса при комбинации контекстной и топологической информации
6. l — желаемое количество кластеров
7. clusteralgo($\mathcal{G}, l$)

Efficient Community Detection in Large Networks using Content and Links

$\mathcal{E}_c \leftarrow \emptyset$
for $i = 1$ to $|\mathcal{V}|$ **do**
  foreach $v_j \in TopK(v_i, k, \mathcal{T})$ **do**
    $\mathcal{E}_c \leftarrow \mathcal{E}_c \cup (v_i, v_j)$
  end for
end for

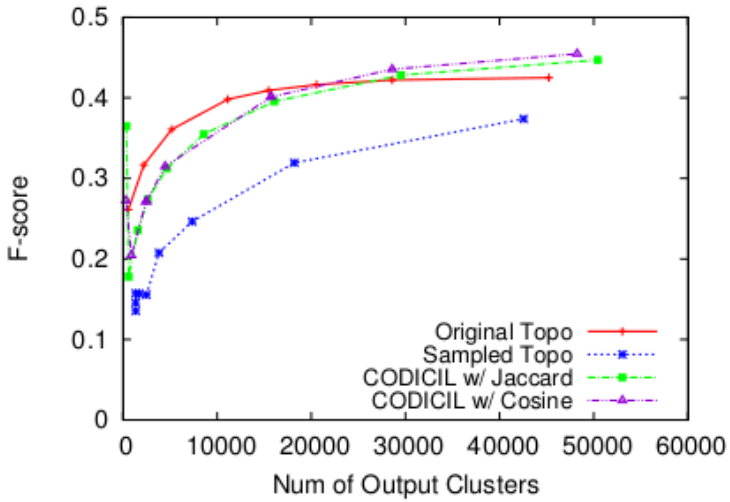
Пояснения:

- Топ K схожих считается по косинусной метрике векторов, полученных с помощью TF-IDF

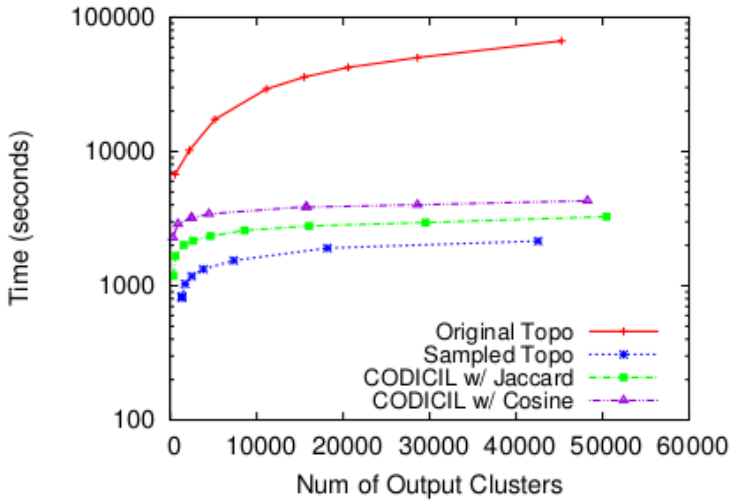
Efficient Community Detection in Large Networks using Content and Links

$\mathcal{E}_u \leftarrow \mathcal{E}_t \cup \mathcal{E}_c$
$\mathcal{E}_{sample} \leftarrow \emptyset$
for $i = 1$ to $|\mathcal{V}|$ **do**
  \\Γ_i contains v_i's neighbors in the edge union
  $\Gamma_i \leftarrow ngbr(v_i, \mathcal{E}_u)$
  for $j = 1$ to $|\Gamma_i|$ **do** $\boldsymbol{sim^t}_{ij} \leftarrow similarity(ngbr(v_i, \mathcal{E}_t), ngbr(\gamma_j, \mathcal{E}_t))$
  $\boldsymbol{simnorm^t}_i \leftarrow normalize(\boldsymbol{sim^t}_i)$
  for $j = 1$ to $|\Gamma_i|$ **do** $\boldsymbol{sim^c}_{ij} \leftarrow similarity(\boldsymbol{t}_i, \boldsymbol{t}_{\gamma_j})$
  $\boldsymbol{simnorm^c}_i \leftarrow normalize(\boldsymbol{sim^c}_i)$
  for $j = 1$ to $|\Gamma_i|$ **do** $\boldsymbol{sim}_{ij} \leftarrow \alpha \cdot \boldsymbol{simnorm^t}_{ij} + (1 - \alpha) \cdot \boldsymbol{simnorm^c}_{ij}$
  \\Sort similarity values in descending order. Store the corresponding node IDs in idx_i
  $[val_i, idx_i] \leftarrow descsort(\boldsymbol{sim}_i)$
  for $j = 1$ to $\left\lceil \sqrt{|\Gamma_i|} \right\rceil$ **do**
    $\mathcal{E}_{sample} \leftarrow \mathcal{E}_{sample} \cup (v_i, v_{idx_{ij}})$
  end for
end for
$\mathcal{G}_{sample} \leftarrow (\mathcal{V}, \mathcal{E}_{sample})$
$\mathcal{C} \leftarrow clusteralgo(\mathcal{G}_{sample}, l)$ \\Partition into l clusters

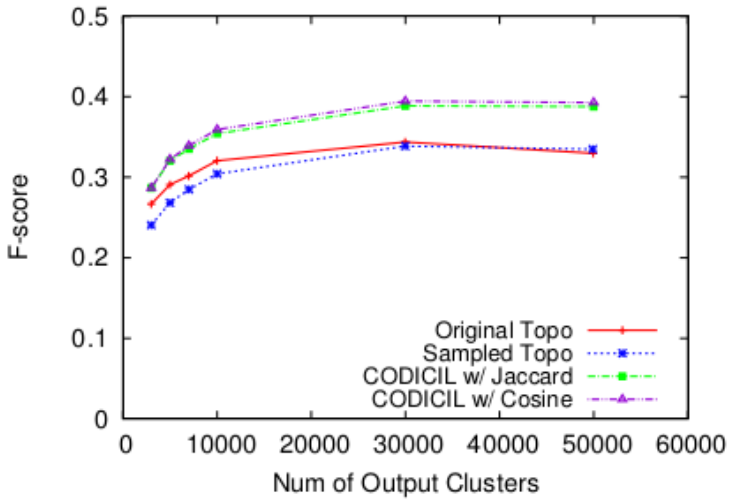
Efficient Community Detection in Large Networks using Content and Links



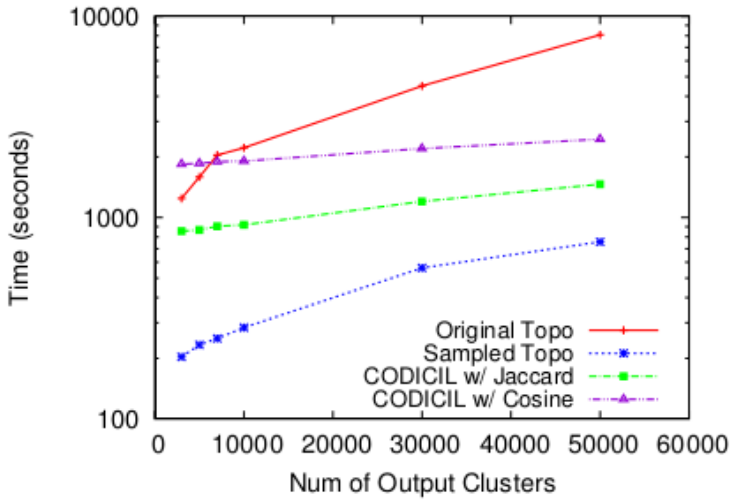
(a) F-score of MLR-MCL on Wikipedia



(b) Running time of MLR-MCL on Wikipedia

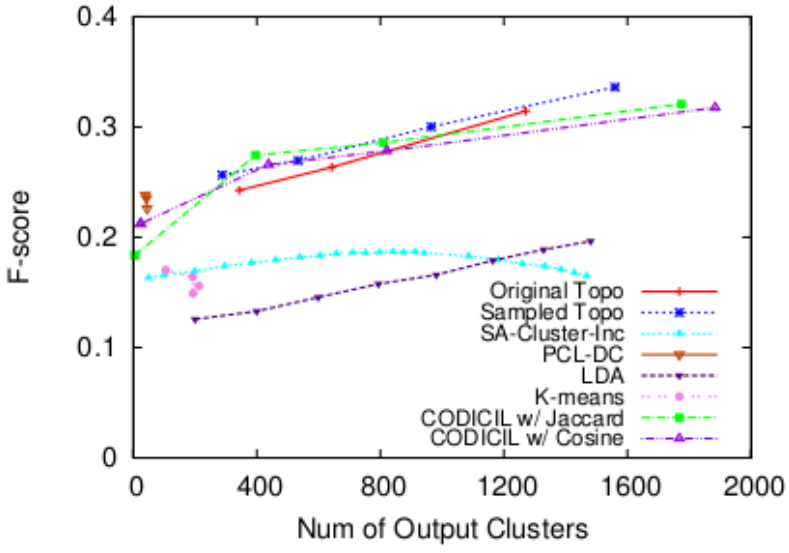


(c) F-score of Metis on Wikipedia

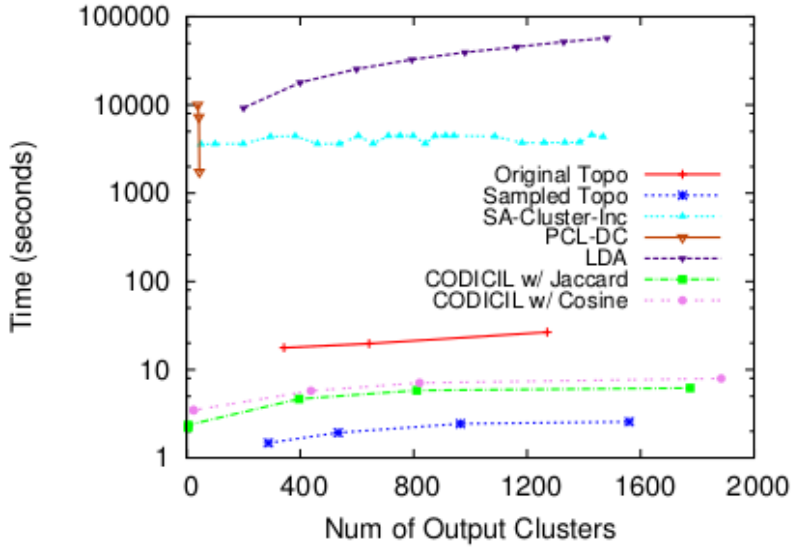


(d) Running time of Metis on Wikipedia

Efficient Community Detection in Large Networks using Content and Links



(a) F-score of MLR-MCL on Flickr



(b) Running time of MLR-MCL on Flickr

Figure 5: Experiment Results on Flickr

Вопросы?