Multiple Local Community Detection

Alexandre Hollocou, Thomas Bonald, Marc Lelarge

Multiple Local Community Detection

Alexandre Hollocou
INRIA, Paris, France
alexandre.hollocou@inria.fr

Thomas Bonald
Telecom Paristech, Paris, France
thomas.bonald@telecom-paristech.fr

Marc Lelarge
INRIA-ENS, Paris, France
marc.lelarge@ens.fr

ABSTRACT

Community detection is a classical problem in the field of graph mining. We are interested in *local* community detection where the objective is the recover the communities containing some given set of nodes, called the *seed set*. While existing approaches typically recover only one community around the seed set, most nodes belong to multiple communities in practice. In this paper, we introduce a new algorithm for detecting multiple local communities, possibly overlapping, by expanding the initial seed set. The new nodes are selected by some local clustering of the graph embedded in a vector space of low dimension. We validate our approach on real graphs, and show that it provides more information than existing algorithms to recover the complex graph structure that appears locally.

Keywords

Graphs; Clustering; Community detection; Random walks

1. INTRODUCTION

Community detection is a fundamental problem in the field of graph mining, with applications to the analysis of social, information or biological networks [12, 23]. The objective is to find dense clusters of nodes, the underlying *communities* of the graph. While most existing algorithms work on the entire graph [25, 4, 27, 17], it is often irrelevant in practice to cluster all nodes. A more practically interesting problem is to detect the communities containing some given set of nodes, the so-called *seed set*. This problem, known as *local* community detection, is particularly relevant in large datasets where the exploration of the whole graph is computationally expensive, if not impossible.

The problem of local community detection is generally stated as follows: the objective to recover some unknown community C in a graph given some subset $S \subset C$ of these nodes. The implicit assumption is that the communities form a partition of the graph. However, nodes typically belong to multiple, overlapping communities in practice [26, 32]. The problem is then to recover all communities containing the seed set S.

In the present paper, we propose a novel approach to this problem by introducing the `MULTICOM` algorithm, which is able to detect multiple communities nearby a given seed set S. This algorithm uses local scoring metrics to define an embedding of the graph around the seed set. Based on this embedding, it picks new seeds in the neighborhood of the original seed set, and uses these new seeds to recover multiple communities.

The rest of the paper is organized as follows. In the next section, we introduce some mathematical notations that we use throughout the paper. Existing approaches for local community detection are presented in Section 3. Our approach to *multiple* local community detection is presented in Section 4. The algorithm itself is presented in Section 5 and tested on real graphs in Section 6. Section 7 concludes the paper.

IFIP WG 7.3 Performance 2017. Nov. 14-16, 2017, New York, NY USA

2. NOTATIONS

Let $G = (V, E)$ be an undirected, unweighted graph. We use n to denote the number of nodes and m the number of edges in G. Without loss of generality we consider that $V = \{1, ..., n\}$. We denote by A the adjacency matrix of the graph. We use d_u to denote the degree of node $u \in V$ and D the diagonal matrix $\text{diag}(d_1, ..., d_n)$. If U is a set of nodes, its volume is defined by $\text{Vol}(U) = \sum_{u \in U} d_u$.

3. RELATED WORK

Local community detection has recently drawn the attention of many researchers, motivated both by the ever increasing size of the datasets and the complex local structure of real graphs [19, 14].

3.1 Scoring and sweeping

The classical approach to local community detection is based on two steps [2]: first, nodes are scored according to their proximity to the seed set; then, nodes are considered in decreasing order of their score and added to the community, with a stopping rule based on some goodness metric.

Formally, the algorithm is based on some scoring function f such that, for any seed set $S \subset V$, f_S is a vector in $\mathbb{R}_+^V$ whose component $f_S(v)$ characterizes the attachment of node v to S. We expect that the higher the score $f_S(v)$, the higher the probability that v is in the same community as the nodes of S. Examples of scoring functions are the Personalized PageRank, the heat kernel diffusion and the local spectral score, described below.

Given some seed set S, the score $f_S(v)$ of each node $v \in V$ is computed. The nodes are then numbered in decreasing order of their score, so that $f_S(v_1) \geq f_S(v_2) \geq \ldots \geq f_S(v_J)$, where J is the number of nodes with non-zero scores. This numbering defines a sequence of nested sets $S_1, ..., S_J$, with $S_j = \{v_i,\ i \leq j\}$.

The second step consists in finding the set S_j that defines the best community. To measure the quality of a community C, a classical metric is the conductance, defined by

$$\Phi(C) = \frac{\sum_{u\in C}\sum_{v\notin C} A_{uv}}{\min(\mathrm{Vol}(C), \mathrm{Vol}(V \setminus C))}.$$

A community of good quality has low conductance. The conductance of each set S_{j+1} can be computed from the conductance of S_j in time proportional to $d_{v_{j+1}}$. This step is called the *sweep* process [2]. The outcome of the algorithm is the set S_j having the lowest conductance. This set is called the *sweep cut*. Other goodness metrics like modularity or density can also be used for the sweep cut [33].

3.2 Personalized PageRank

The Personalized PageRank is certainly the most common score used for local community detection [16]. It is based on a random walk with restart. Given a parameter $\alpha \in (0, 1)$, we consider the random walk $X_0, X_1, X_2, ...$ that starts uniformly at random from seed set S and that, at each step, moves from node u to node v with probability $\alpha\frac{A_{uv}}{d_u}$ and restarts with probability $1-\alpha$ from a node of S chosen uniformly at random. For all $t \geq 0$ and all $v \in V$, we have

$$\Pr(X_{t+1} = v) = (1-\alpha)\frac{\mathbf{1}_S(v)}{|S|} + \alpha\sum_{u=1}^{n}\frac{A_{uv}}{d_u}\Pr(X_t = u).$$

There is a unique stationary distribution p for the Markov chain $(X_t)_{t\geq 0}$ and it is the limiting distribution of X_t when $t \to \infty$. This distribution satisfies

$$p(v) = (1-\alpha)\frac{\mathbf{1}_S(v)}{|S|} + \alpha\sum_{u=1}^{n}\frac{A_{uv}}{d_u}p(u). \qquad (1)$$

The vector p is known as the Personalized PageRank (PPR) associated with the seed set S.

Solving the linear system (1) exactly is computationally expensive. Efficient methods for approximating the PPR have been proposed [1]. Recent experimental results show that, in practice, a few iterations of the fixed-point equation (1) are sufficient to get a very good ordering of the nodes [16].

3.3 Heat kernel diffusion

The linear system (1) can be written in vector form

$$p = (1-\alpha)\chi_S + \alpha P p,$$

where $P = AD^{-1}$ and $\chi_S = 1_S/|S|$, leading to the following expression for the PPR:

$$p = (1-\alpha)(I-\alpha P)^{-1}\chi_S = (1-\alpha)\sum_{k=0}^{\infty}\alpha^k P^k \chi_S.$$

Another form of diffusion has been introduced by Chung in [6, 7]. It is called the heat kernel diffusion and is defined by

$$h = e^{-t}\left(\sum_{k=0}^{\infty}\frac{t^k}{k!}P^k\right)\chi_S = \exp\{-t(I-P)\}\chi_S,$$

where t is a parameter capturing the spread of the diffusion. A method of approximation of h is proposed in [15] and used in the sweep method to detect local communities.

3.4 Local spectral analysis

Another class of algorithms applies spectral techniques to detect local communities [21, 20]. In [20] for instance, the `LEMON` algorithm is based on the extraction of a sparse vector y in the span of the so-called local spectral subspace of the graph around the seed set S. This vector y is then used as the scoring function. Unlike the previous algorithms, the `LEMON` algorithm is *iterative*: the nodes of highest scores are used to expand the seed set S and to find a new vector y, and so on. The iteration stops when the conductance starts increasing.

3.5 Other approaches

A number of other approaches have been proposed for local community detection. These include greedy algorithms [8, 5, 22], flow-based algorithms [24], degree-based techniques [28], motif detection [13, 34] and subgraph extraction [30]. None recover multiple communities.

4. MULTIPLE COMMUNITY DETECTION

4.1 Scoring gap

Let $v_1, \ldots, v_J$ be the nodes numbered in decreasing order of their score, as described in §3.1, and $S_1, \ldots, S_J$ be the corresponding nested sets. For a strong community, the scoring function should give high values to nodes belonging to the community and small values for nodes outside the community. As a result, we expect the existence of some $a > 0$ such that

$$\begin{aligned} &\max\left(f_S(v_2) - f_S(v_1), \ldots, f_S(v_j) - f_S(v_{j-1})\right) \leq a, \\ &\text{while } f_S(v_{j+1}) - f_S(v_j) > a, \end{aligned} \qquad (2)$$

where j is the index of the target set S_j. The parameter a is called the *scoring gap*. Conditions for the existence of such a gap have been derived in [29, 2]. Experimental studies showing the presence of a scoring gap in real graphs can be found in [2, 9].

To illustrate this, we use the DBLP dataset presented in [33] and available on the Stanford Social Network Analysis Project (SNAP) website. DBLP is a database that collects the main publications in computer science. The graph we consider is a collaboration network: nodes correspond to authors and there is an edge between two authors if they have co-authored a paper. This dataset comes with ground-truth communities that correspond to journals and conferences (i.e., all authors having published in the same journal or conference form a community).

In our experiment, we pick a ground-truth community C in the graph and a seed node $s \in C$. We compute the attachment scores f_s to s with three different scoring functions: Personal PageRank p, Heat Kernel score h and LEMON score y, introduced in Section 3. We compare the corresponding sets $S_1, S_2, \ldots$ to the ground-truth community C with the F1-Score. The F1-Score $\mathrm{F1}(\hat{C}, C)$ between two sets C and $\hat{C}$ is defined as the harmonic mean of the precision and the recall of $\hat{C}$ with respect to C:

$$\mathrm{F1}(\hat{C}, C) = H(\mathrm{precision}(\hat{C}, C), \mathrm{recall}(\hat{C}, C))$$

where $H(a, b) = \frac{2ab}{a+b}$ and

$$\mathrm{precision}(\hat{C}, C) = \frac{|\hat{C}\cap C|}{|\hat{C}|}, \quad \mathrm{recall}(\hat{C}, C) = \frac{|\hat{C}\cap C|}{|C|}.$$

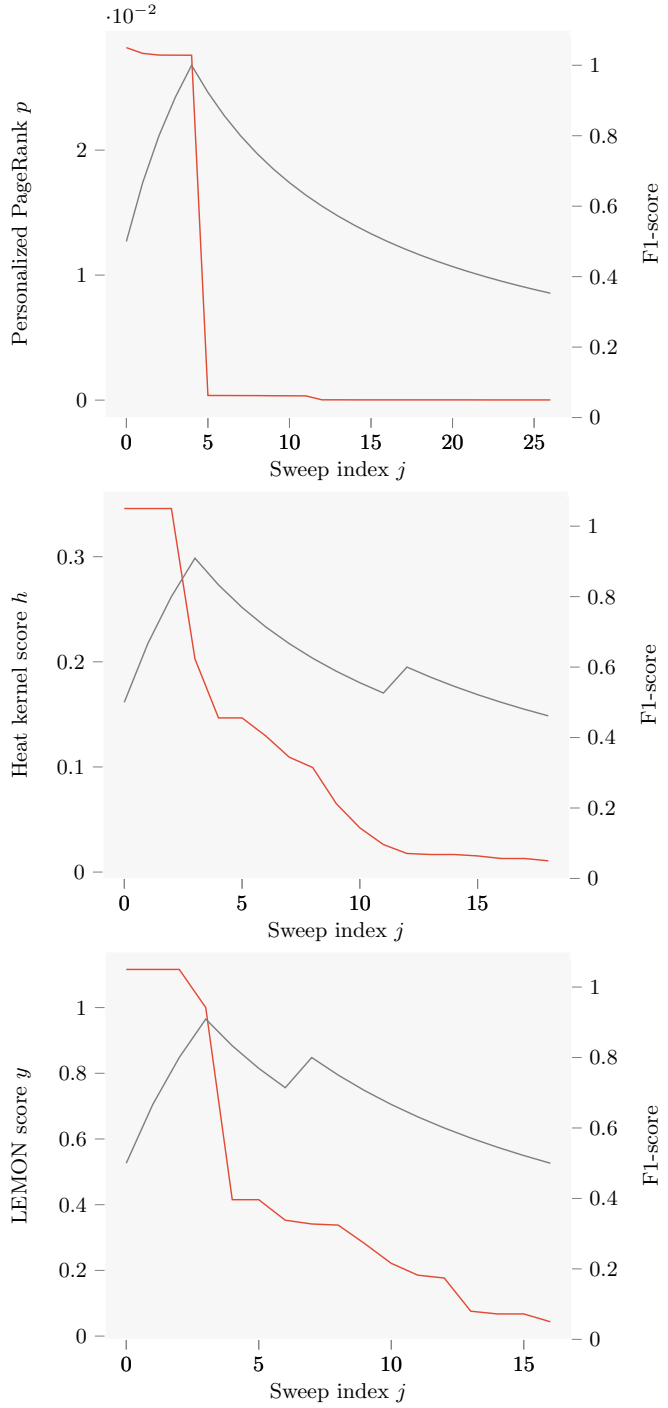


Figure 1: Attachment scores and F1 scores for the sweep sets in a ground-truth community of DBLP. We consider three scoring functions to compute the attachment scores: Personalized PageRank, Heat Kernel and LEMON. The attachment score curves are drawn in orange and the F1 Score curves in grey.

In Figure 1, we jointly plot for a given target community C and seed node $s \in C$ the values of the score $f_s(v_j)$ and the F1 score $\mathrm{F1}(S_j, C)$, for each scoring function. We observe in each case the existence of a clear scoring gap. Moreover we observe that the drop in the scoring function corresponds each time to a peak in the F1 Score, which means that the associated set is actually the best candidate for a community among all the nested sets. In order to find multiple communities, we elaborate on this idea as described in the next section.

4.2 Local embedding

The main idea of our algorithm is to use the scoring function to get a local embedding of the graph around the seed set S. Specifically, each node v is embedded into the vector $(f_s(v))_{s\in S}$. Note that for a node far from the seed set S, this vector will be the all zero vector and it can be safely removed since we are only interested in local communities. We then cluster nodes with respect to their mutual distances in the embedding space.

To motivate the proposed embedding, let us consider the case of a *perfect* scoring function f such that $f_s(v) = c_s$ if v is in the same community as s and 0 otherwise, where c_s is some positive constant that depends on s. Let us assume that we have disjoint communities $C_1, \ldots, C_K$ and seed nodes $s_1, \ldots, s_K$ in each of these communities ($s_k \in C_k$). Then, we see that the embedding $(f_{s_k}(v))_{1\le k\le K}$ associated with these seed nodes and this perfect scoring function takes exactly K distinct non-zero values and that each of these values corresponds to a community C_k. Thus, by using a clustering algorithm on the vectors in the embedding space, we can exactly recover the communities $(C_k)_{k=1,\ldots,K}$. The scoring gap property presented above for Personalized PageRank, LEMON and Heat Kernel guarantees that these three functions are close to the *perfect* scoring function defined above. Therefore, applying clustering on the embedding $(f_s(v))_{s\in S}$ should lead to results similar to these perfect setup.

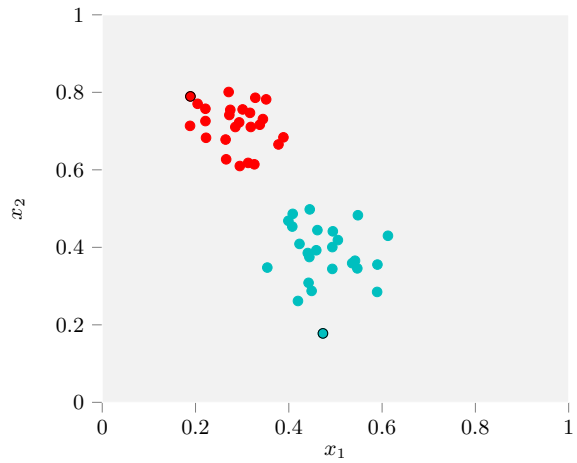
To illustrate the behavior of such embeddings, we show in Figure 2 the result of a local embedding using the Personal PageRank scoring function on a random graph generated from a mixture of two gaussian vectors in $\mathbb{R}^2$ by putting an edge between two points if they are within distance r from each other. We choose a seed node in each gaussian. We observe that the local embedding from these seed nodes clearly separates the nodes from the different groups and that a clustering algorithm can be used in order to recover each gaussian.

4.3 Finding new seeds

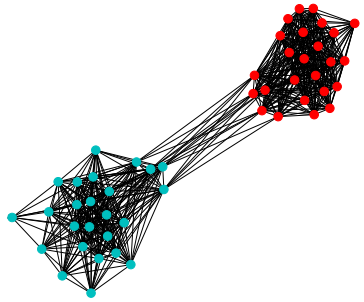
We use the local embedding introduced in the previous section to iteratively find new seeds $\mathcal{S}$ around the original seed set S. The main idea of the algorithm is simple: we start with $\mathcal{S} = S$ and we grow this new seed set by repeating the following three steps:

1. Perform the local embedding using $\mathcal{S}$: $(f_s(v))_{s\in\mathcal{S}}$.
2. Cluster the nodes in the embedding space.
3. Pick a new seed node in each *unexplored* cluster and add it to $\mathcal{S}$.

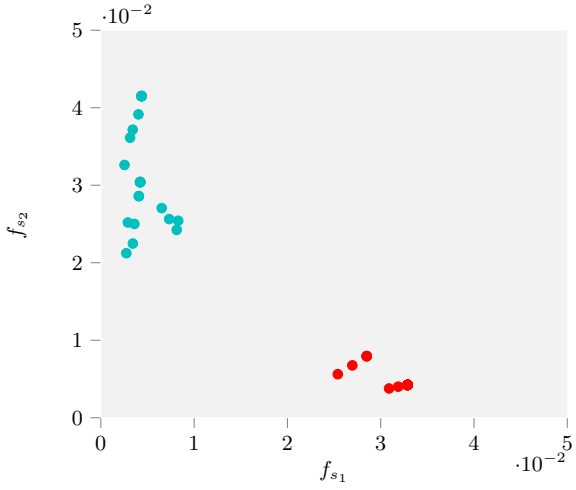
The new seed nodes of $\mathcal{S}$ are then used to detect multiple communities in the neighborhood of the initial seed set S. We define more formally the algorithm in the next section, and, in particular, we clarify the notion of *unexplored* cluster.



(a) Mixture of two gaussian vectors in $\mathbb{R}^2$ (one seed is picked in each cluster)



(b) Corresponding graph for $r = 0.2$



(c) Graph embedding using PPR

Figure 2: Example of local graph embedding for a random graph. The graph is generated from a mixture of two gaussians in $\mathbb{R}^2$ by putting an edge between two points if they are within a given distance r from each other. The local embedding is performed using the Personalized PageRank scoring function and two seed nodes s_1 and s_2 picked in each gaussian.

5. ALGORITHM

We now present our algorithm, named `MULTICOM`, that recovers multiple local communities.

5.1 Inputs

The inputs of `MULTICOM` are a graph $G = (V, E)$ and a seed set $S \subset V$. Our algorithm also takes a scoring function f, which is typically one of the functions presented in Section 3, and a local community detection function `local` that, given a seed node s and a score f_s, returns a local community around S. For instance, this `local` function may return the sweep set with the lowest conductance by applying the sweep process described in §3.1. Finally the algorithm takes an integer parameter I that controls the number of detected communities and a *re-seeding threshold* $\beta \in [0, 1]$ that controls the number of new seeds at each iteration of the algorithm.

5.2 Description

In order to find multiple communities in the neighborhood of the initial seed set S, the algorithm uses a larger seed set $\mathcal{S}$ that contains S at the end of the algorithm. The new seed nodes found during a step of the algorithm are stored in $\mathcal{S}_{\text{new}}$. Initially, $\mathcal{S}_{\text{new}}$ is initialized with S.

Step 1: Local community detection for new seeds

The algorithm starts by computing the score vectors f_s for each new seed node $s \in \mathcal{S}_{\text{new}}$. Then, for each $s \in \mathcal{S}_{\text{new}}$ we use the `local` algorithm to extract a local community C_s around s. Thus, at the end of this step of the algorithm, we obtain one community C_s per seed node s. In particular, if the set $\mathcal{S}_{\text{new}}$ is a singleton, we only recover one community at this stage.

Step 2: Embedding and clustering

We use the scores f_s for $s \in \mathcal{S}$ to define an embedding which maps each node $v \in V$ to a vector $x_v = (f_s(v))_{s \in \mathcal{S}}$ in $\mathbb{R}_+^{|\mathcal{S}|}$. We then apply the popular clustering algorithm DBSCAN [11] to the non-zero vectors x_v and obtain K clusters of nodes, $D_1, ..., D_K$.

Step 3: Picking new seeds

The second step of the algorithm applies the ideas of Section 4 to find new seeds. We expect to obtain two types of clusters in Step 2:

- clusters with significant overlap with the communities $C_s, s \in \mathcal{S}$, already detected;
- clusters with low overlap with these communities.

We want to select seed nodes in the later clusters in order to detect new communities. To identify these clusters, we compute for each cluster D_k the ratio

$$\frac{|D_k \cap \cup_{s \in \mathcal{S}} C_s|}{|D_k|}.$$

Clusters with a ratio lower that the threshold β are considered as new directions that are worth exploring: a new seed node is picked in each of them. In order to have a central node in each cluster, we simply choose the node with the highest degree.

Loop

The seeds selected at the end of step 3 form the new seed set $\mathcal{S}_{\text{new}}$ to which we apply the same three steps. We stop the algorithm when the number of communities is greater than I or if there is no new seed.

Finally we output all the communities C_s found from the seed nodes $s \in \mathcal{S}$. The pseudo-code of our algorithm is given below.

Algorithm 1 `MULTICOM`

Require: Graph $G = (V, E)$, seed set $S \subset V$, score function f, function `local`, parameters I, β.

$\mathcal{S} \leftarrow \emptyset$ (all seed nodes)
$\mathcal{S}_{\text{new}} \leftarrow S$ (new seed nodes)
$\mathcal{C} \leftarrow []$ (list of communities)
while $\mathcal{S}_{\text{new}} \neq \emptyset$ and $|\mathcal{C}| \leq I$ **do**
  # Detecting communities from new seeds
  for $s \in \mathcal{S}_{\text{new}}$ **do**
    $f_s \leftarrow$ compute score function
    $C_s \leftarrow \texttt{local}(f_s)$ (local community detection)
    $\mathcal{C}.\texttt{push}(C_s)$
  end for
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{\text{new}}$ (add new seeds)
  # Embedding and clustering
  $\forall v \in V$, $x_v \leftarrow (f_s(v))_{s \in \mathcal{S}}$
  $D_1, ..., D_K \leftarrow$ clustering of $(x_v)_{v \in V}$ in $\mathbb{R}_+^{|S|} \setminus \{0\}$
  # Picking new seeds
  $\mathcal{S}_{\text{new}} \leftarrow \emptyset$
  $\mathcal{E} \leftarrow \cup_{C \in \mathcal{C}} C$ (explored nodes)
  for $k = 1, ..., K$ **do**
    if $|D_k \cap \mathcal{E}| < \beta |D_k|$ **then**
      $s_{\text{new}} \leftarrow$ node with highest degree in $D_k \setminus \mathcal{E}$
      $\mathcal{S}_{\text{new}} \leftarrow \mathcal{S}_{\text{new}} \cup \{s_{\text{new}}\}$
    end if
  end for
end while
return $\mathcal{C}$

5.3 Post-processing

Note that the communities returned by `MULTICOM` might intersect. This is consistent with the fact that nodes often belong to multiple communities in practice [26, 32]. However, we might want to limit the number of nodes that the communities have in common, and consider that if two communities C_i, C_j share too many nodes, then they form only one community $C_i \cup C_j$. To do so, we apply a post-processing step, called `MERGE`, at the end of `MULTICOM` that merges two communities C_i and C_j if their F1 score $\text{F1}(C_i, C_j)$ is greater than a given threshold $\gamma \in [0, 1]$. In the following experiments we use `MERGE` with parameter $\gamma = \frac{1}{2}$.

6. EXPERIMENTS

We now analyse the performance of our algorithm, both qualitatively and quantitatively, on real graphs.

6.1 Case study: Wikipedia

First, we illustrate the interest of our algorithm on an extract of Wikipedia presented in [31]. The dataset is built from a selection of articles from the English version of Wikipedia that matches the UK National Curriculum[1]. The nodes of the graph corresponds to Wikipedia articles, and we put an edge between two articles a and a' if there is an hyperlink to article a' in article a. We apply `MULTICOM` with a Personalized PageRank scoring function and using a cut based on conductance on the seed set $S = \{\text{Albert Einstein}\}$. With the parameter $I = 5$, the algorithm returns 6 communities, for a total of 153 nodes. For each of these communities, we list the top-5 nodes according to their PageRank:

- $C_1 = \{$Albert Einstein, Special relativity, Euclid, Wave, String theory$\}$
- $C_2 = \{$Light, Electric field, Contact lens, Maxwell's equations, Semiconductor device$\}$
- $C_3 = \{$Gottfried Leibniz, Algebra, Pi, Game theory, Thermo- dynamics$\}$
- $C_4 = \{$Star, Red dwarf, Open cluster, Orion Nebula, Gliese 876$\}$
- $C_5 = \{$Quantum mechanics, Photon, Electromagnetic radiation, Electric charge, Linus Pauling$\}$
- $C_6 = \{$Atom, Renormalization, Mechanical work, Quark, Ununpentium$\}$

The top node of each community $C_2, \ldots, C_6$ turns out to be a seed found by `MULTICOM`. We see that each community corresponds to a different facet of Einstein's work. Note that the state-of-the-art algorithms like Personalized PageRank only recover community C_1, as it corresponds to the first step of the `MULTICOM` algorithm. We see that we gain precious information by considering additional communities around the Albert Einstein article.

6.2 Real-world data

Datasets

For a quantitative evaluation of our algorithm on real-world graphs, we use the datasets available on the SNAP website [33]. All these datasets include ground-truth community memberships. We consider graphs of different types: the social network YouTube [3], the product co-purchasing graph built from Amazon data [18], and the DBLP dataset described in Section 5.

Algorithms

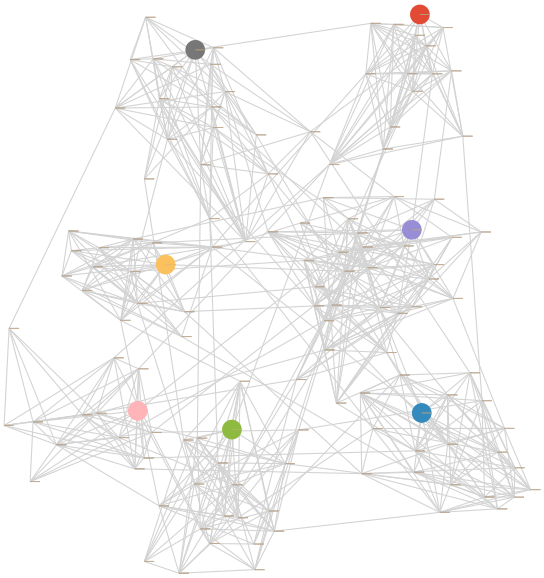
We compare `MUTLTICOM` to the three state-of-the-art algorithms presented in Section 3: Personalized PageRank (`PPR`), Heat Kernel (`HK`) and LEMON (`LEMON`). For `MULTICOM` we use a function `cut` that finds the local minimum for the conductance $\Phi(S_j)$, and we take Personalized PageRank for the scoring function f. We have implemented `MULTICOM` in Python and made it available on GitHub[2].

Performance evaluation

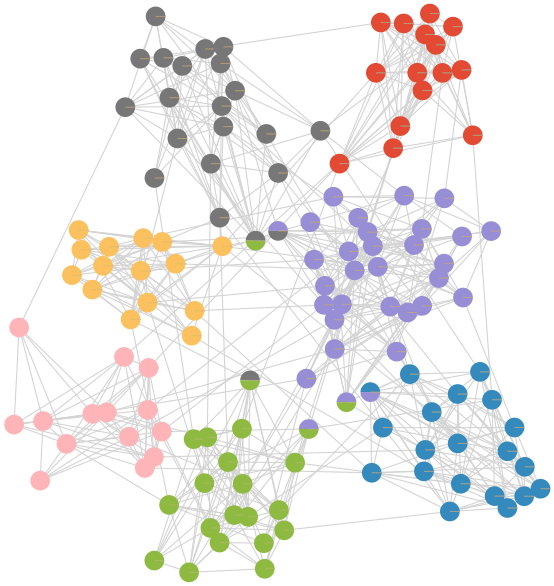
In order to evaluate the performance of the algorithms, we measure for each returned community $\hat{C}$ its conductance $\Phi(\hat{C})$ and the maximum F1-score between $\hat{C}$ and any ground-truth community.

The state-of-the-art algorithms return only one community so the computation of these scores is straightforward.

[1] `http://schools-wikipedia.org`
[2] `https://github.com/ahollocou/multicom`



(a) Detected seed nodes
(initial seed in red)



(b) Detected communities

Figure 3: Seeds and communities detected by `MULTICOM` in a SBM. The algorithm has been applied to a SBM random graph with an initial seed displayed **in red** in Figure (a). Figure (a) shows the new seeds detected by `MULTICOM`. Figure (b) shows the communities detected from these seeds. Note that some nodes belong to several communities (bi-colored nodes).

`MUTLTICOM` generally outputs several communities, so we consider the average values of conductance and F1-score for these communities.

We also compute the total number of nodes $|\mathcal{E}|$ *labeled* or *explored* by each algorithm, i.e. the total number of nodes that belong to a community at the end of each algorithm.

Benchmark

For each dataset, we pick 100 seed nodes uniformly at random and run the algorithms on each of these seed nodes. We use the parameter $I = 5$ and $\beta = 0.8$ for `MULTICOM`. The results are shown in Table 1.

DBLP

Algo.	$\lvert\mathcal{E}\rvert$	Φ	F1
`MULTICOM`	103	0.45 ± 0.14	0.23 ± 0.15
`PPR`	21	0.41 ± 0.17	0.23 ± 0.22
`HK`	15	0.32 ± 0.14	0.23 ± 0.26
`LEMON`	18	0.43 ± 0.20	0.26 ± 0.24

Amazon

Algo.	$\lvert\mathcal{E}\rvert$	Φ	F1
`MULTICOM`	80	0.35 ± 0.14	0.44 ± 0.19
`PPR`	26	0.31 ± 0.16	0.46 ± 0.24
`HK`	24	0.27 ± 0.16	0.49 ± 0.26
`LEMON`	21	0.40 ± 0.27	0.48 ± 0.25

YouTube

Algo.	$\lvert\mathcal{E}\rvert$	Φ	F1
`MULTICOM`	284	0.69 ± 0.11	0.05 ± 0.03
`PPR`	95	0.56 ± 0.25	0.05 ± 0.11
`HK`	100	0.55 ± 0.33	0.04 ± 0.11
`LEMON`	5	0.96 ± 0.12	0.12 ± 0.20

Table 1: Benchmark results on SNAP datasets

Observe that the total number of nodes found by `MULTICOM`, corresponding to multiple communities, is much higher than those found by the other algorithms. In other words, the volume of information that `MULTICOM` collects in the neighborhood of the seed set is much more important. Moreover, the quality of this information is not too much downgraded by the multi-directional approach. Indeed, the average F1-score measured on the multiple communities returned by `MULTICOM` is essentially the same as the F1-scores of the other algorithms. The much higher F1-score of `LEMON` for the YouTube dataset is due to the much smaller communities detected by this algorithm.

6.3 Synthetic data

We evaluate our algorithm on synthetic graphs generated with the Stochastic Block Model (SBM) [10]. We illustrate the results of `MULTICOM` on such a synthetic graph in Figure 3. In the sub-figure (a), we display the new seeds found by the algorithm when initialized with an initial seed colored in red in the figure. In the sub-figure (b), we display the corresponding communities returned by the algorithm. Note that some communities are overlapping i.e. some nodes belong to more than one community.

In order to numerically evaluate the results of `MULTICOM` on the SBM model, we generate 100 graphs with 100 nodes and 5 communities of equal size. We pick a random seed node in each of these graphs and run `MULTICOM` starting from this seed with the same parameters as in §6.2. We use the same performance metrics (average conductance, average F1-score and number of *explored* nodes) to evaluate the

performance of MULTICOM and we compare it to PPR. The results are shown in Table 2. We observe that MULTICOM recovers almost perfectly all 5 communities in each generated graph, whereas PPR recover only one of these communities.

Algo.	$\lvert\mathcal{E}\rvert$	Φ	F1
MULTICOM	98	0.11	0.98
PPR	20	0.11	0.97

Table 2: Benchmark results on a SBM model. Graph generated have 100 nodes and 5 equal-sized communities.

7. CONCLUSION

In this paper, we have presented an algorithm for multiple local community detection. The approach relies on the local embedding of the graph near the seed set, using a scoring function such as Personal PageRank. We have seen that the target communities are typically well separated in the embedding space. Building on this observation, we have proposed a clustering method to identify new seeds in the neighborhood of the initial seed set, so as to recover different communities in various directions of the graph.

For future work, we would like to compare the local embedding obtained with MULTICOM with the existing local spectral embedding, and use it for other tasks such as network visualization and link prediction.

8. REFERENCES

[1] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[2] R. Andersen and K. J. Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, pages 223–232. ACM, 2006.

[3] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM, 2006.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[5] C.-S. Chang, C.-J. Chang, W.-T. Hsieh, D.-S. Lee, L.-H. Liou, and W. Liao. Relative centrality and local community detection. *Network Science*, 3(4):445–479, 2015.

[6] F. Chung. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences*, 104(50):19735–19740, 2007.

[7] F. Chung. A local graph partitioning algorithm using heat kernel pagerank. *Internet Mathematics*, 6(3):315–330, 2009.

[8] A. Clauset. Finding local community structure in networks. *Physical review E*, 72(2):026132, 2005.

[9] M. Danisch, J.-L. Guillaume, and B. Le Grand. Learning a proximity measure to complete a community. In *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, pages 90–96. IEEE, 2014.

[10] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[12] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[13] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1311–1322. ACM, 2014.

[14] L. G. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91(1):012821, 2015.

[15] K. Kloster and D. F. Gleich. Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1386–1395. ACM, 2014.

[16] I. M. Kloumann and J. M. Kleinberg. Community membership identification from small seed sets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1366–1375. ACM, 2014.

[17] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[18] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

[19] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.

[20] Y. Li, K. He, D. Bindel, and J. E. Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In *Proceedings of the 24th international conference on world wide web*, pages 658–668. International World Wide Web Conferences Steering Committee, 2015.

[21] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally. *Journal of Machine Learning Research*, 13(Aug):2339–2365, 2012.

[22] A. Mehler and S. Skiena. Expanding network communities from representative examples. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(2):7, 2009.

[23] M. E. Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[24] L. Orecchia and Z. A. Zhu. Flow-based algorithms for local graph clustering. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1267–1286. Society for Industrial and Applied Mathematics, 2014.

[25] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*, pages 284–293. Springer, 2005.

[26] F. Reid, A. McDaid, and N. Hurley. Partitioning breaks communities. In *Mining Social Networks and Security Informatics*, pages 79–105. Springer, 2013.

[27] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[28] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 939–948. ACM, 2010.

[29] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

[30] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 404–413. ACM, 2006.

[31] R. West, J. Pineau, and D. Precup. Wikispeedia: An online game for inferring semantic distances between concepts. In *IJCAI*, pages 1598–1603, 2009.

[32] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):43, 2013.

[33] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[34] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. 2017.