

Contents

1	Introduction	2
2	Information Retrieval	3
2.1	Crawling	3
2.2	Parsing	3
3	The PageRank Algorithm	3
3.1	How to compute PageRank values	4
4	The Graph Partitioning Algorithm	5
4.1	How to apply spectral graph partitioning	6
4.2	K-way Partitioning	7
5	Numerical results / Data analysis / ...	8
5.1	PageRank – forse (inclusi in R.analysis)	8
5.2	Graph Partitioning – forse (inclusi in R.analysis)	8
6	Bibliography	9

1 Introduction

A social network consists of a set of objects connected to each other by social relations. The best way to model social networks is using graphs (see Figure 1): the objects (entities) are represented as nodes and the connections as edges between two different nodes. The most common example we can take is the World Wide Web (WWW) where we have web pages as nodes connected by hyperlinks, the edges.

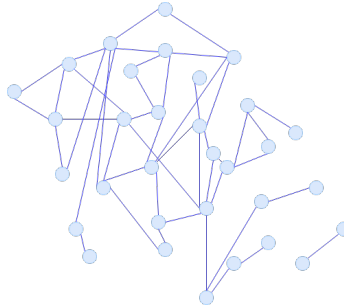


Figure 1. Example of social network graph

The goal of this project is to create a social network of computational science authors belonging to Swiss institutions and then analyze the relative graph.

I want to find the relations between authors of the same or different institutions and belonging to the same or to different research scenarios: how they interact with each other and if they are strictly related to the institution or cooperate throughout Switzerland. The result will provide also a ranking of the authors and the institutions to see which are the most influential in the computational science research.

The first step is retrieving all the necessary information for the construction of the social network, i.e., names of the authors and relationships between each other. I crawled the 2015, 2016 and 2017 PASC Conferences, interdisciplinary conferences which bring together researchers across the areas of computational science, high-performance computing, and various domain sciences, and different SIAM conferences of several years, selecting the topics relevant to us (e.g. Optimization, Parallel Computing,...). Split previous long sentence into shorter ones.

The second step is the information analysis to find out relations between institutions and between members belonging to the same institution. With PageRank algorithm we obtain a “ranking” of all conferences’ participants. PageRank is an algorithm used by Google Search to rank websites in their search engine results, but it can be applied to any social network. In this project I use the algorithm to measure the importance of institutions’ members considering the number and quality of their collaborations. I use graph partitioning techniques to invert the process and obtain the institutions from members collaborations: it is reasonable to assume that members of the same institution collaborate more between each other than with other institutions’ representatives.

I also analyze the institutions’ connectivity matrices and their structure: looking at the cliques (i.e. a sub matrix where every two distinct members collaborate with each other; this means that all entries of the sub matrix are ones)–REFORMULATE: the submatrix is not “where” members are, but the dense submatrix “represents” a group of authors densely connected. Also, I would not put this in parenthesis, but I’d describe cliques clearly in a paragraph. present in the matrices we can for example detect the different research areas of the institutions and the connection between them.

The results will provide an interesting picture of the different research scenarios in Switzerland and how they interact with each other.

2 Information Retrieval

...

2.1 Crawling

...

2.2 Parsing

...

3 The PageRank Algorithm

PageRank is an algorithm used by Google Search to rank websites in their search engine results; it was developed by Larry Page and Sergey Brin (the Google's founders) in 1996 as part of their research project at Stanford University.

PageRank is used, together with other algorithms, to measure the importance of web pages and sort them by popularity in the result; it can be used in any graph to compute the importance of each node with its PageRank value. Larry Page and Sergey Brin algorithm works by counting the number and quality of links to a page to compute a value which represents its importance; more popular a page is, more likely it receives links from other websites and more likely from important websites.

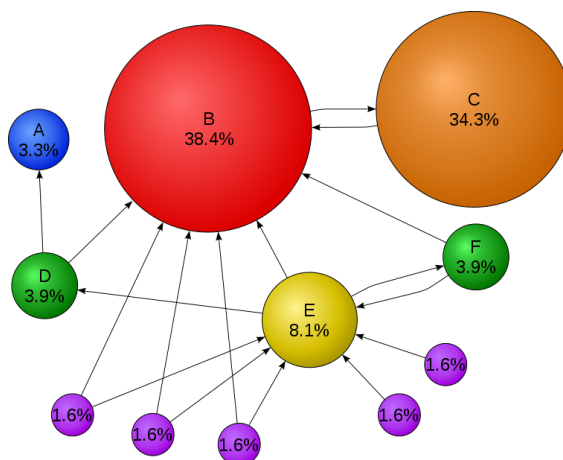


Figure 2. PageRank example **Either change this figure or introduce reference to online source (I prefer changing the figure).**

As we can see in Figure 2, node C has higher PageRank than E even if it has lower number of inner links; this is because the only inner link C has is from the most important node of the graph B so it is considered to be more important than all the inner links of node E from the purple lower nodes.

Imagine surfing websites, going from one page to another: – I don't like the “imagine” part ... to avoid problems with pages without outer links we have two different methods to choose the next page. The surfer can:

- randomly choose an outgoing link from the page,

- choose a random page from the network “network” instead of “Web”, since we are not surfing the whole WWW.

We assume that with probability p (typically $p = 0.85$) our surfer follows the first option and with probability $1 - p$ the second. **Invert subject-verb: We assume that the surfer (get rid of “our”) follows the first option with probability ...**

The probability that an infinite random surfer visits a specific website is called its PageRank; a page will have high rank if other pages with high rank link to it.

3.1 How to compute PageRank values

To apply PageRank, from a graph of n nodes we construct a n -by- n connectivity matrix G such that

$$g_{ij} = 1 \quad \text{if and only if nodes } i \text{ and } j \text{ are connected.}$$

The number of non-zeros in G is the total number of connections in the graph.

The row r_i represents the number of inner-links of page i , while column c_i the number of outer-links. So we can define:

- In-degree of page i : $r_i = \sum_{j=1}^n g_{ij}$
- Out-degree of page i : $c_i = \sum_{j=1}^n g_{ji}$

Try to use an align command or something similar to align the previous list. Let p be the probability that the surfer follows a link and $1 - p$ the probability that it chooses a random page, then $\delta = \frac{1-p}{n}$ is the probability that a particular random page is chosen.

We construct a transition probability matrix A where all elements are positive, smaller than one and sums of columns are equal to one; column j represents the probabilities to go from page j to all other pages. So we set

$$a_{ij} = \begin{cases} \frac{pg_{ij}}{c_j} + \delta & \text{if } c_j \neq 0 \\ \frac{1}{n} & \text{if } c_j = 0 \end{cases}$$

In other words, if page j has no outer links, it means that column j of A present a uniform probability $\frac{1}{n}$ in all its elements. We can now compute PageRank values by solving the **homogeneous linear system – this is an eigenvalue problem**. I guess we could add a paragraph or two about eigenvalue problems inspired by Chapter 8 of the Num. Comp. book.

$$x = Ax \tag{1}$$

The transition matrix A can be written as

$$A = pGD + ez^T,$$

where D is a diagonal matrix with the reciprocals of the out-degrees, i.e.,

$$d_{jj} = \begin{cases} \frac{1}{c_j} & \text{if } c_j \neq 0 \\ 0 & \text{if } c_j = 0 \end{cases},$$

z is the vector formed by

$$z_j = \begin{cases} \delta & \text{if } c_j \neq 0 \\ \frac{1}{n} & \text{if } c_j = 0 \end{cases},$$

and e is the vector of length n with all ones.

We can write the linear system

$$(I - A)x = 0$$

as

$$(I - pGD)x = \gamma e$$

where $\gamma = z^\top x$. To conclude, **since the value of the scalar γ depends on the unknown x , we impose the condition $\gamma = 1$** , then solve

$$(I - pGD)x = e \tag{2}$$

and then rescale the solution so that $\sum_{i=1}^n x_i = 1$.

cosa inserire in procedure

Algorithm 1 (PageRank)

```

1: procedure PAGERANK
2:    $U \leftarrow$  list of authors
3:    $G \leftarrow$  adjacency matrix
4:    $p \leftarrow$  dumping factor
5:    $G \leftarrow G - \text{diag}(G)$  (remove self collaborations)
6:    $c \leftarrow \text{sum}(G,1)$   $r \leftarrow \text{sum}(G,2)$  (Compute out-degree and in-degree of each node)
7:    $D \leftarrow$  Diagonal matrix with reciprocals of out-degrees
8:    $(I - pGD)x = e$ 
9:    $x \leftarrow x / \text{sum}(x)$  (normalize the result  $x$ )
10:   $x \leftarrow$  sort  $x$  in descending order
11: return  $x$ 

```

4 The Graph Partitioning Algorithm

Graph partitioning consists in dividing a graph $G = (V, E)$ ¹ cutting the smaller number of edges and obtaining sub-graphs with specific properties. For example k -way partitioning divides the graph G into k smaller components.

A partitioning algorithm is said to be efficient if it divides a given graph into smaller components with about the same size, cutting as few edges as possible. A valid and fast method is the coordinate bisection, which simply chooses a partitioning plane perpendicular to one of the coordinate axes but spectral algorithm is more efficient: it finds a partition from the adjacency matrix of the graph. – Un po' sbrigativo, expand, lower the “speed” of the sentences and try to explain more details.

As we can see in Figure 3 sometimes coordinate and spectral partitioning behave the same but usually the second one is more effective.

¹ A graph G with vertexes V and edges E .

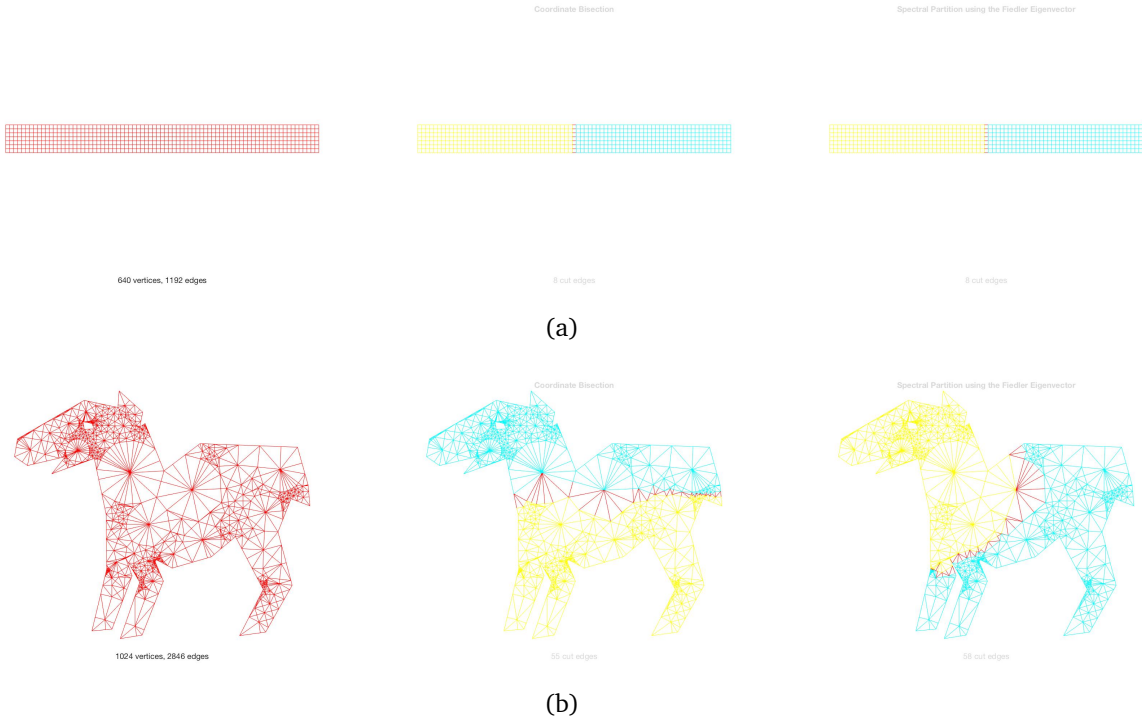


Figure 3. In both (a) and (b) we have the graph at the left, the coordinate bisection in the middle and the spectral partition on the right

4.1 How to apply spectral graph partitioning

To apply graph partitioning we need the adjacency matrix A of the graph, which is symmetric since the graph is considered to be undirected, and the Laplacian matrix L constructed such that:

$$L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

where d_i is the vertex degree of node $i \in V$. The relation between A and L is

$$L = D - A \quad (3)$$

where D is the diagonal matrix having the d_i 's on the diagonal. Since L is symmetric and positive semidefinite by construction, all its eigenvalues are real and nonnegative. The first eigenvalue is always zero and the second is not equal to zero if the graph is connected². The eigenvalues of L are ordered as follows:

$$0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_n. \quad (4)$$

The eigenvector corresponding to the first eigenvalue λ_1 is the vector of all ones and it does not provide information about the graph structure; the second lowest eigenvector, instead, called “Fiedler eigenvector”, is used by spectral partitioning to return the smaller sub-graphs. Dividing the nodes of graph G according to the median s of the Fiedler vector $v = (v_1, v_2, \dots, v_n)$ gives two partitions V_1 and V_2 such that

$$\begin{cases} v_i \in V_1 & \text{if } v_i \leq s \\ v_i \in V_2 & \text{if } v_i > s \end{cases}.$$

² A graph is connected if there is a path from any node to any other node in the graph.

Such a partition is called the Fiedler cut and leads to two parts of G with nearly equal number of vertices with a small number of cutting edges. **Is it the best edge-cut? We should summarize briefly the actual formulation of the problem which is NP-hard and show how spectral partitioning approximates the solution (see Prof. Schenk's lecture notes).**

Algorithm 2 (Graph Partitioning)

```

1: procedure GRAPH PARTITIONING
2:    $G \leftarrow (V, E)$ 
3:   compute Fiedler vector of  $L = D - A$ 
4:   Sort the vector values
5:   Put first half of the nodes in  $V_1$  and second half in  $V_2$ 
6:    $E_1 \leftarrow$  edges with both nodes in  $V_1$ 
7:    $E_2 \leftarrow$  edges with both nodes in  $V_2$ 
8: return  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$ 

```

4.2 K-way Partitioning

K-way partitioning of a graph G of n nodes, consists in a division of its nodes in k disjoint subset, all with size nearly $\frac{n}{k}$ (see Figure 4).

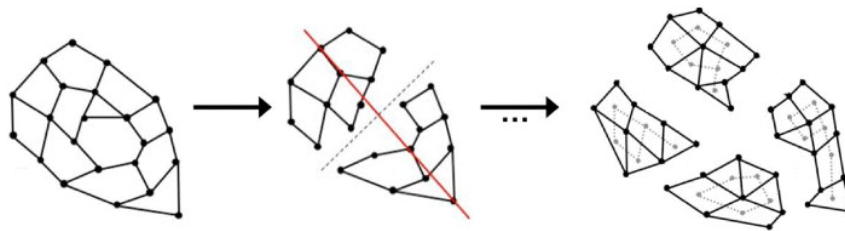


Figure 4. Example of k-way graph partitioning

We consider only the simple case where we want to divide the graph in k partitions and k is a power of two. The idea is just to apply the spectral partitioning $\frac{k}{2}$ times recursively to each of the partitions obtained. The method is presented in Algorithm 3.

Algorithm 3 (k-way Partitioning)

```

1: procedure K-WAY PARTITIONING
2:    $G \leftarrow (V, E)$ 
3:    $k \leftarrow$  number of desired partitions
4:   Apply spectral partitioning on  $G$  and find  $G_1$  and  $G_2$ 
5: loop:
6:   if  $\frac{k}{2} > 1$  then
7:     Recursive partition on  $G_1$  with  $\frac{k}{2}$ 
8:     Recursive partition on  $G_2$  with  $\frac{k}{2}$ 
9:      $k \leftarrow \frac{k}{2}$ 
10:  goto loop
11: return  $G_1 = (V_1, E_1) \dots G_K = (V_K, E_K)$ 

```

5 Numerical results / Data analysis / ...

5.1 PageRank – forse (inclusi in R.analysis)

5.2 Graph Partitioning – forse (inclusi in R.analysis)

6 Bibliography

- <https://en.wikipedia.org/wiki/PageRank>
- Assignment1 1 Course Numerical Computing 2107/2018
- Assignment2 1 Course Numerical Computing 2107/2018
- "https://am.vsb.cz/export/sites/am/cs/theses/kabelikova_ing.pdf"
- https://en.wikipedia.org/wiki/Graph_partition