Università
della
Svizzera
italiana

Faculty
of Informatics

Bachelor Thesis

June 11, 2018

The Swiss Society for Industrial and Applied Mathematics Social Network

Vanessa Braglia

Abstract

In this project I implemented some algorithms to analyze the Swiss society for industrial and applied mathematics social network. To construct the graph representing the relations (the edges) between authors (the nodes) I crawled some conferences' online programs and I have extracted all the necessary information. I used PageRank and graph partitioning to study the relationships between authors: I found the members that collaborate more with each other and the relationship between different institutions. The results provide an interesting picture of the different research scenarios in Switzerland and how they interact with each other internally to an organization but also between different institutions.

Advisor
Prof. Olaf Schenk
Assistant
Fabio Verbosio

Advisor's approval (Prof. Olaf Schenk): Date:

Contents

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: the objects (entities) are represented as nodes and the connections as edges between two different nodes (see Figure 1). 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.

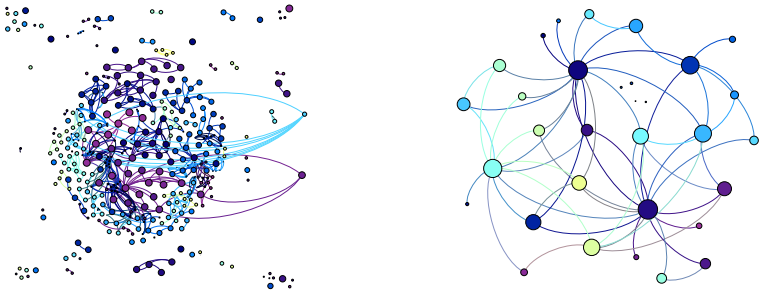


Figure 1. Example of social network graphs

The goal of this project is to create a social network of computational science authors belonging to Swiss institutions and corporations, and then analyze the relative graph, represented by an adjacency matrix. 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 and different SIAM conferences of several years, selecting the topics relevant to us (e.g. Optimization, Parallel Computing,...). The PASC conferences are interdisciplinary and bring together researchers across the areas of computational science, high-performance computing, and various domain sciences.

The second step is the information analysis to find out relations between institutions and between members belonging to the same or different institutes. With PageRank algorithm we obtain a ranking of all the conference participants; it is an algorithm used by Google Search to rank websites in their search engine results, but it can be applied on any social network.

In this project I use the algorithm to measure the importance of the institution members considering the number and quality of their collaborations and to find out the relevant universities for computational science researches.

I use graph partitioning techniques to invert the process, trying to 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. With this assumption the algorithm will cut the relations between members of different universities,

keeping the representatives of each institution in the same group.

I also analyze the institutions connectivity matrices and their structure: looking at the cliques present in the matrices, we can for example detect the different research areas of the institutions and the connection between them. Cliques are dense sub matrices which represent a group of authors that are closely connected with each other.

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

2 Information Retrieval

Information retrieval is the process of extracting, starting from an information need, the relevant information from a collection of resources. In this project, the necessary information are the names of the institutions members and the collaborations between them. First, I try to use Mendeley API but it allows only to take 100 results per query and there is no control on the type of information to be retrieved. Additionally, it does not return all the information needed, i.e., not all authors have a specified institution and only few papers were available.

After this first unproductive attempt to use Mendeley API, I decide to extract the information from some online conferences programs: the idea is that if two authors presented some work together at a conference, then they are connected in my social network.

I crawl the conference programs and I construct a parser to extract the information from the HTML pages and to organize the data.

2.1 Crawling

A web crawler is an Internet bot that browses the World Wide Web, typically for the purpose of web indexing of search engines. In this project the goal is not indexing web pages but save the information contained in the HTML to parse them and construct the social network.

The web pages that I decide to crawl are the online programs of 2015, 2016 and 2017 PASC conferences and of SIAM conferences with arguments concerning computational science i.e., Optimization, Parallel Computing, Computational Science & Engineering, Imaging Science, Uncertainty Quantification, Linear Algebra in Signals, Systems & Control, Data Mining and Analysis of Partial Differential Equations.

I use Apache Nutch [3], a highly extensible and scalable open source web crawler, allowing the user to crawl a lot of web pages with few command lines. It starts with a list of URLs to visit, called the seeds; I create this list including all the links to the computational science conferences web pages previously mentioned.

As the crawler visits the URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit; in my case I crawl only the pages in the starting seed ignoring all the internal links. The crawler copies and saves the information as it goes and stores in a repository each page as a distinct file.

Thanks to Apache Nutch I manage to create a text file with the HTML of all the saved conference pages. After this file is created, the next step is studying the structure of the page and how to extract the information needed, i.e., where we can find the information in the page, if it is included in some particular HTML tag, and how to interpret it. In order to do so, I designed a parser responsible for these tasks.

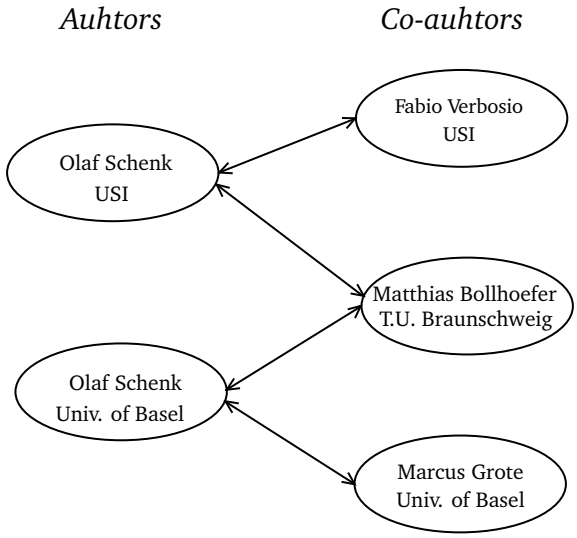


Figure 2. Example of author with two different universities and different coauthors

2.2 Parsing

A parser is a program that receives data as input and breaks it up into smaller elements that can then be used for different purposes. In this project, the parser reads the text file created after the crawling and extracts from it the information about Swiss institutions and their members. The data obtained is used to create the social network ready to be analyzed.

In this project I personally write a parser, using Python programming language, to extract the information from the conferences HTML pages. The parser goes through the structure of the page and extracts only the useful information. Clearly, different page structures require different parser, then I write two of them because the SIAM conferences and the PASC have different structures.

The parser extracts only the lines of the pages which contain the names of the authors, their institutions and nationality; the institutions members that present a work together at a conference appear in group in the page so I keep them together to later extract the relations. Thanks to the nation expressed in the program I can differentiate the Swiss authors from the rest of the the world.

Once that the desired lines are collected, I take the groups one after the other, and I extract name, institution, nation and coauthors for each member. With all this information I manage to construct a social network and its corresponding adjacency matrix.

In case a single author belongs to two distinct institutions I decide to include two different entries because I'm interest in institutions collaborations: so the same author can have some coauthors with one university and completely different coauthors with the other one. We can see an example in Figure 2 where professor Olaf Schenk is affiliated to two different universities (Università della Svizzera Italiana and Univerisity of Basel) and collaborates with Matthias Bolhoefer from both institutions but he works with Fabio Verbosio only from USI and with Marcus Grote only from the University of Basel.

One of the big issues of my dataset was that the institutions were labeled differently. For example, the Swiss university *ETH Zürich* can also be named *ETHZ* or *Swiss Federal Institute of Technology Zurich*. Moreover there can be some organizations of the university such as the

ETHZ Computational Laboratory or the *Institute of Fluid Dynamics*. All these names have to be represented by the same institution label.

To solve this problem I construct an hash map putting the label to use as key and a list with all names represented by that label as value. For example the one for ETH Zürich will have:

- label: "ETHZ"
- value: ["ETH Zürich","Swiss Federal Institute of Technology Zurich", "ETHZ Computational Laboratory", "Institute of Fluid Dynamics"]

For each university I scan this map and reassign the corresponding name.

I make two different databases, one for Switzerland and one for the whole world. The first one contains only Swiss authors and their collaborations with members of the same nation, while in the second there are also authors from other nations and their collaborations are extended to all over the world. For each author I save the name, the university, the nation, a list of his coauthors and I assign to each of them a unique index.

From this data set I easily create the adjacency matrix and thanks to the indexes I can switch between the matrix and the database; from the graph I can read the index and go in the database to see to which author it corresponds.

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 founders) in 1996 as part of their research project at Stanford University [4].

PageRank is used, together with other algorithms, to measure the importance of web pages and sort them by popularity in the results; it can be used on any graph to compute the importance of each node by 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 even more likely from important websites.

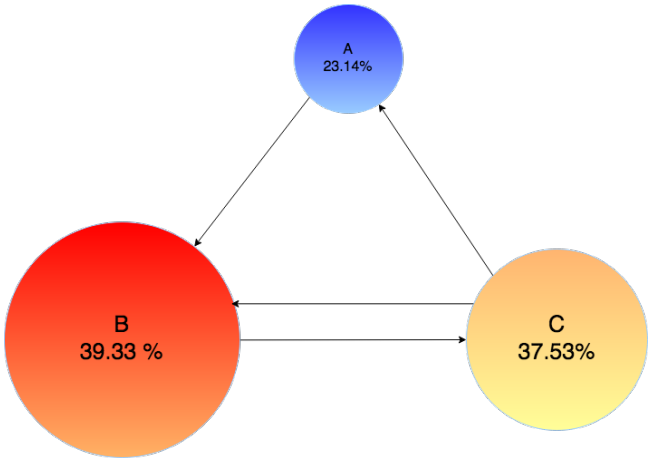


Figure 3. PageRank example

As we can see in Figure 3, node C has higher PageRank than A even if it has the same number of inner links; this is because the only inner link of C is from the most important node of the graph, B, so it is considered to be more important than the inner link on node A.

While surfing websites, going from one page to another, I want to avoid problems with pages without outer links so I introduce two different methods to choose the next page. The surfer can:

- randomly choose an outgoing link from the page, or
- choose a random page from the network.

We assume that the surfer follows the first option with probability p (typically $p = 0.85$) and the second with probability $1 - p$. This theoretical random walk is known as a Markov chain or Markov process.

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

Since the collaboration is a symmetric relation, my graph is undirected, hence the number of inner-links is equal to the number of outer-links.

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 the elements are positive and smaller than one, and the sum of each column is equal to one. The last property comes from the fact that column j represents the probabilities to go from page j to all other pages. So, we set

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

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. Since matrix A is a transition matrix of a Markov chain [2, Chapter 7], all its eigenvalues are real, between 0 and 1, and 1 is an eigenvalue itself

$$0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = 1.$$

We can now compute PageRank values by solving the problem

$$A\,x = x.$$

This is a particular eigenvalue problem (for more details see [1, Chapter 8]) of the form $Ax = \lambda x$, where, in our specific case, $\lambda = 1$. We compute the eigenvector relative to the greatest eigenvalue, which is 1 as just mentioned.

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.,

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

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. For more details about this, see [**?**, Chapter 7] (https://ch.mathworks.com/moler/chapters.html).

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

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

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

Algorithm 1 (PageRank)

procedure PAGERANK
    $U \leftarrow$ list of *authors*
    $G \leftarrow$ adjacency matrix
    $p \leftarrow$ dumping factor
    $G \leftarrow G - \text{diag}(G)$ (remove self collaborations)
    $c \leftarrow \text{sum}(G,1)$ $r \leftarrow \text{sum}(G,2)$ (Compute out-degree and in-degree of each node)
    $D \leftarrow$ Diagonal matrix with reciprocals of out-degrees
    Solve $(I - pGD)x = e$
    $x \leftarrow x/sum(x)$ (normalize the result x)
    $x \leftarrow$ sort x in descending order
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. 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). 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 2.

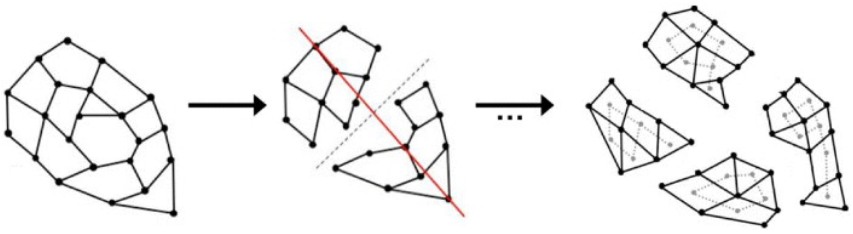


Figure 4. Example of k-way graph partitioning

Algorithm 2 (k-way Partitioning)

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

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 based on the eigenvalues and eigenvectors of the Laplacian matrix of the graph and doesn't need the nodal coordinates.

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

[1] A graph G with vertexes V and edges E.

that:

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

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

$$L = D - A$$

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, in the general case, the multiplicity of the the first eigenvalue corresponds to the number of connected components of the graph.

The eigenvalues of L are ordered as follows:

$$0 = \lambda_1 = \ldots = \lambda_m < \lambda_{m+1} \leq \ldots \lambda_n$$

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 $v = (v_1, v_2, ..., v_n)$ gives two partitions V_1 and V_2 such that

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

Such a partition is called the Fiedler cut and leads to two parts of G with nearly equal number of vertexes, minimizing the number of cut edges.

Fiedler theorem said that if the graph G is connected, L the Laplacian matrix, N_+ and N_- a partitioning such that

$$x(i) = +1 \qquad \text{if } v_i \text{in } N_+$$

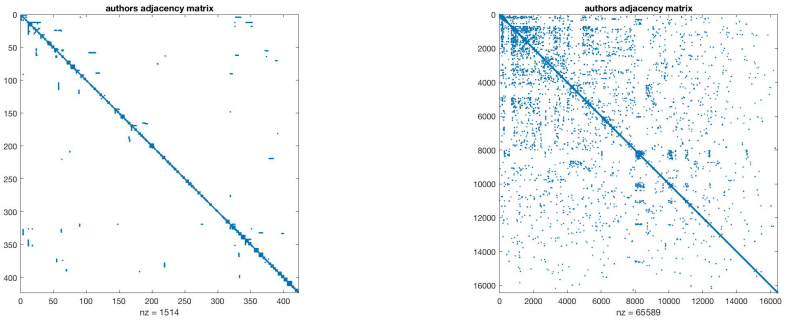
$$x(i) = -1 \qquad \text{if } v_i \text{in } N_-$$

Then the number of edges cut is:

$$\#\text{edge\_cut} = \frac{1}{4} x^T L x$$

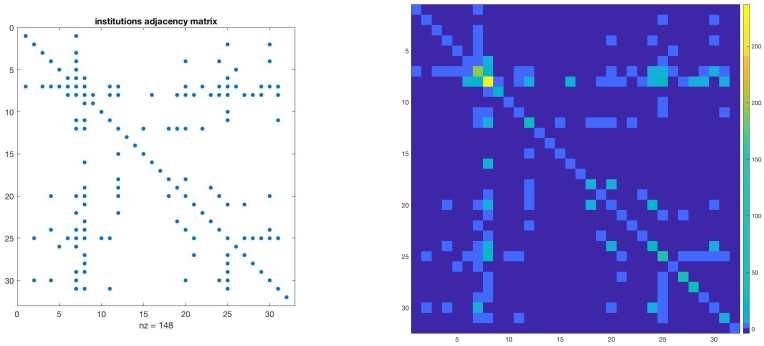
This problem of minimizing the number of cut edges is NP-hard and can be written as a constrained minimization problem:

$$\min_z f(z) = \frac{1}{4} z^T L z \qquad \text{subject to} \qquad \begin{cases} z^T z = n & \text{z is a real vector} \\ z^T e = 0 & \text{with } e = [1, 1, ..., 1]^T \end{cases}$$



(a) (b)

Figure 5. (a) is the Swiss authors' adjacency matrix and (b) is the one of the world authors



(a) (b)

Figure 6. (a) is the Swiss institutions adjacency matrix, (b) is the organizations weighted graph

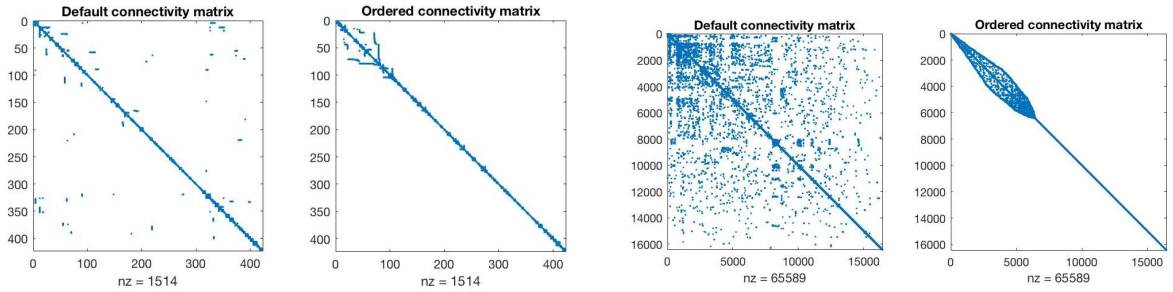
5 Numerical analysis

After retrieving all the necessary information and organizing the data in adjacency matrices, I applied the PageRank algorithm, obtaining the following results. Since the Swiss author names obtained are only 422, I decide to extend the research to the institutions from all over the word, reaching 16425 names. The analysis is done in parallel for both Swiss collaborations and authors from all over the world.

I first construct the adjacency matrices of the authors and their corresponding institutions; for the organizations I also build a weighted matrix which keeps stored the number of collaborations between them. I consider that each author collaborates with himself so the diagonal of the matrix will contain ones. We can see in Figure 5 the authors matrices, while in Figure 6 the one of the Swiss institutions.

The sparsity of a matrix is the number of zero-valued elements divided by the total number of elements; the Swiss authors adjacency matrix has sparsity equal to $8.5 \cdot 10^{-3}$ instead the one of the world authors matrix is $2.4 \cdot 10^{-4}$ (for more on sparse matrices, see [5, Chapter 3]).

In Figure 6(b) (lets call this matrix U) we can see the matrix representing the weighted collaborations between Swiss universities: as we can see in the bar at the right, blue corresponds to no collaboration and while increasing the number of collaborations the color changes to yellow. The institutions which collaborate more in Switzerland are École Polytechnique Fédérale de Lausanne and ETH Zürich which correspond to the 7th and 8th columns/rows in the ma-



(a) (b)

Figure 7. The Reverse Cuthill McKee ordering for only Switzerland in (a) and for the authors of all the world in (b)

trix. These two universities collaborate both internally in the university (see the yellow squares of $U_{7,7}$ and $U_{8,8}$ entries of the matrix U) but also with other institutions (see the 7th and 8th columns/rows of U where we have a lot of light-blue squares): for example ETH Zürich has 19 collaboration with Università della Svizzera italiana (25th column). We can see that most of the light-blue squares appear in the diagonal of the matrix; it means that institutions' members usually works with authors of the same organization instead of ones from a different institution.

I applied the Cuthill-McKee algorithm to the two authors matrices: it permutes a sparse matrix that has a symmetric sparsity pattern into a band matrix form with a small bandwidth. A band matrix is a sparse matrix whose non-zero elements are confined to a diagonal band; if the bandwidth of a symmetric matrix A is n, it means that

$$A_{ij} = 0 \quad \text{if} \quad |j-i| > n$$

This means that all the non zero elements are put as near as possible to the diagonal of the matrix, as we can see in Figure 7.

For the matrix of Swiss authors in 7(a) the bandwidth is 48, instead the one of the authors from all the world 7(b) has a bandwidth of 1210.

We can notice that in both figures the number of non-zero elements, written below the graph, doesn't change; in fact, the reverse Cuthill-McKee ordering consists only in rows and columns permutations and don't change any entry of the matrix.

The reverse Cuthill Mckee ordering may reduce the fill in of the matrix in some iterative methods. For example, we can apply the LU factorization to the original matrix and then to ordered one. Considering M to be the adjacency matrix of Swiss authors, we run the following code.

```
[L,U] = lu(M);
nn = nnz(L)+nnz(U);
r = symrcm(M);
[L1,U1] = lu(M(r,r));
nn1 = nnz(L1)+nnz(U1);
```

It computes the LU factorization and counts the non zero elements of the result, then applies the reverse Cuthill McKee ordering and figures out the factorization again.

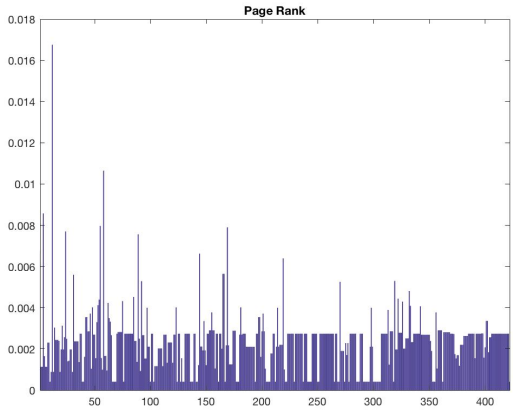


Figure 8. Bar graph of Swiss authors PageRank values

We obtain that with the original matrix the result has 2344 non zero elements, while with the ordering only 1422.

Increasing the size of the matrix, the result can be even more evident. We can consider the world author matrix that is more than four times larger than the Swiss one, examined before. Applying the LU factorization we obtain a matrix with 3297712 non zero elements but with the reordering there are only 72637.

5.1 PageRank

Applying the PageRank algorithm I obtain a sorted list of authors and in Figure 8 we can see all the 422 Swiss authors with their corresponding PageRank value.

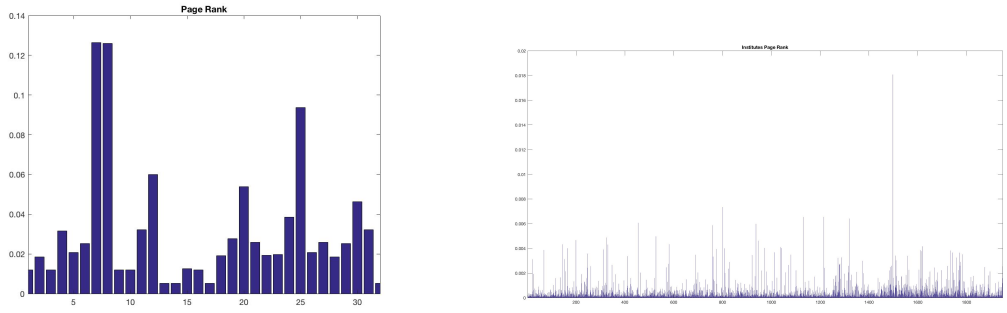
Table 1 (left) shows the first 10 most “popular" authors sorted by the algorithm. We can see that the *In_degree* column, which corresponds to the number of collaboration, it is not sorted because PageRank takes into account also the quality of the collaborations. For example, the third author *Simone Deparis* has more collaborations than the second *Christoph Schwab*. To see it better we can compare this table with the one on the right, containing the first 10 authors sorted only by degree-centrality (i.e., number of collaborations). The most collaborative author is still *Rolf Krause* because it has 30 associations but from the second we can already see some differences in the ranking.

Table 1. First 10 Swiss authors ranked by PageRank (left) and degree centrality (right)

Index	PageRank	Degree Centrality	Name	Institution
12	0.016757	30	Rolf Krause	USI
58	0.010648	12	Christoph Schwab	ETHZ
4	0.0085765	14	Simone Deparis	EPFL
55	0.0079721	12	Costas Bekas	IBM
169	0.0078995	8	Bruno Sudret	ETHZ
24	0.0076977	12	Peter Arbenz	ETHZ
89	0.0075607	7	Fabio Nobile	EPFL
1	0.0067923	8	Alfio Quarteroni	EPFL
144	0.0066262	7	Olaf Schenk	USI
219	0.0063917	10	Torsten Hoefler	ETHZ

Index	Degree Centrality	Name
12	30	Rolf Krause
4	14	Simone Deparis
24	12	Peter Arbenz
55	12	Costas Bekas
58	12	Christoph Schwab
219	10	Torsten Hoefler
332	10	Nicolas Salamin
319	9	Michael Afanasiev
1	8	Alfio Quarteroni
169	8	Bruno Sudret

About what concerning world information we can better see the difference between



(a) (b)

Figure 9. (a) is a bar graph of Swiss institutions PageRank values, while (b) is the one of world institutions

PageRank and degree centrality sorting in Table 2; the Swiss *Rolf Krause*, member of USI, has the 7th PageRank value with only 38 collaborations but does not appear in the first ten positions of the degree centrality list.

Table 2. First 10 authors ranked by PageRank (left) and degree centrality (right)

Index	PageRank	D.Centrality	Name	Institution
1343	0.00066775	47	Youssef M. Marzouk	Massachusetts Institute of Technology
3794	0.00064672	42	Andrea L. Bertozzi	University of California
2719	0.00055678	42	Stanley J. Osher	University of California
115	0.00052919	49	Omar Ghattas	The University of Texas at Austin
2478	0.00052283	23	Stefan Ulbrich	Technical University Darmstadt
2333	0.0005056	33	Gianluca Iaccarino	Stanford University
25	0.00049204	38	Rolf Krause	USI
758	0.00047899	52	Xiaoye Sherry Li	Lawrence Berkeley National Laboratory
701	0.00047145	51	Eric Phipps	Sandia National Laboratories
773	0.00045236	41	James W. Demmel	University of California

Index	Degree Centrality	Name
715	52	Mark S. Shephard
758	52	Xiaoye Sherry Li
701	51	Eric Phipps
115	49	Omar Ghattas
1343	47	Youssef M. Marzouk
188	44	Siva Rajamanickam
189	43	Erik G. Boman
957	43	Lois Curfman McInnes
784	42	Emmanuel Agullo
2719	42	Stanley J. Osher

I have also applied the PageRank algorithm to the universities adjacency matrices obtaining the results in Figure 9. We can observe that, in Switzerland, the most present universities in the area of computational science are École Polytechnique Fédérale de Lausanne, ETH Zürich and Università della Svizzera italiana. Here we have a special case where the universities sorted by PageRank corresponds to the sorting by number of collaborations (*In_degree* column).

Table 3 shows the first 10 universities with higher PageRank value, the Swiss ones on the left and world information on the right: in the computational science area the most important are the University of California and the Massachusetts Institute of Technology. At ranking 10th we can find the Swiss institution, École Polytechnique Fédérale de Lausanne, which was in the first position in the Swiss list.

Table 3. First 10 Swiss institutions (left) and world institutions (right) ranked by PageRank

Index	PageRank	D.Centrality	Institution
7	0.12632	16	EPFL
8	0.12601	16	ETHZ
25	0.09368	12	USI
12	0.059924	7	IBM
20	0.053813	7	SIB
30	0.046199	6	University of Geneva
24	0.038438	5	UNIL
11	0.032109	4	HES-SO Valais-Wallais
31	0.032109	4	University of Neuchatel
4	0.031569	4	CHUV

Index	PageRank	D.Centrality	Institution
1498	0.018077	256	University of California
801	0.0073476	114	Massachusetts Institute of Technology
1215	0.0065502	101	Stanford University
1132	0.0065372	103	Sandia National Laboratories
1319	0.0064138	97	The University of Texas at Austin
454	0.0060685	92	Georgia Institute of Technology
937	0.0059833	96	New York University-NYU
760	0.0058694	87	Lawrence Berkeley National Laboratory
527	0.0049737	80	IBM
326	0.0048789	69	EPFL

5.2 Graph partitioning

Applying the graph partitioning algorithm, the expected result was to obtain groups divided according to institution: members of the same organization will be in the same partition. This is based on the fact the the members mainly work with people of the same institution.

I decide to apply 4-way partitioning, obtaining four groups of about 105 members.

Unfortunately the result was not what I expected because the main universities, such as École Polytechnique Fédérale de Lausanne and ETH Zürich, collaborate with almost all other institutions and for this reason appear in all the groups obtained.

Despite all, the members of the University of Basel are almost all in the second group and the ones from USI are in the third partition.

6 Conclusions

In this project I constructed a Swiss society for industrial and applied mathematics social network. Starting from crawling some conference programs, I manage to construct a data set containing some Swiss authors coupled with their institutions. Two members are connected in the social network graph if and only if they present a work together at a conference.

Retrieved the information and constructed the social network, I created the corresponding adjacency matrices and analyzed them. The implementation of the PageRank algorithm helped in ranking both the authors and the institutions: considering the number of collaborations and their quality, the algorithm compute the PageRank value, with which we can sort the data. I found which are the most relevant authors and institutions in the computational science area and how they interact with each other. I implemented the graph partitioning algorithm even if it does not lead to meaningful results.

Even if the original project changed and we had to find a new way to take the information, the results provide an interesting picture of the different organizations in Switzerland and how they interact with each other.

I want to thank my project advisor, prof. Olaf Schenk, and my assistant, Fabio Verbosio, for suggesting this project. I really enjoyed working on it although the initial information retrieval issue. The project was challenging and pushed me to give the best of myself.

References

[1] Uri M. Ascher and Chen Greif. *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2011.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Introduction to Probability*. Massachusetts Institute of Technology, second edition, 2008.

[3] Apache Software Foundation. Apache Nutch, 2017. [Online; accessed 11-June-2018].

[4] Larry Page, Sergey Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web, 1998.

[5] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.