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 (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.

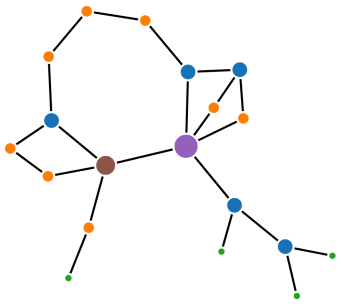


Figure 1. Example of social network graph

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, 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 university. With PageRank algorithm we obtain a "ranking" of all conferences' participants; it 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 the institutions' 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.

After the first useless attempt to use Mendeley API, I decide to extract the information from some online conferences' programs; the idea is that if two authors made a conference together, in my social network it turns out that they have collaboration.

I crawl the conferences' 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, Analysis of Partial Differential Equations.

I use Apache Nutch that is a highly extensible and scalable open source web crawler and it allow you 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 putting there 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 each page in a distinct file in a repository.

Thanks to Apache Nutch I manage to create a text file with the html of all the saved conferences' pages. Obtained this file ,the next step is studying the structure of the page and how to extract the information needed: where are the information I need? Are they in a particular html tag? How we can extract them? Here the parser do its job.

2.2 Parsing

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

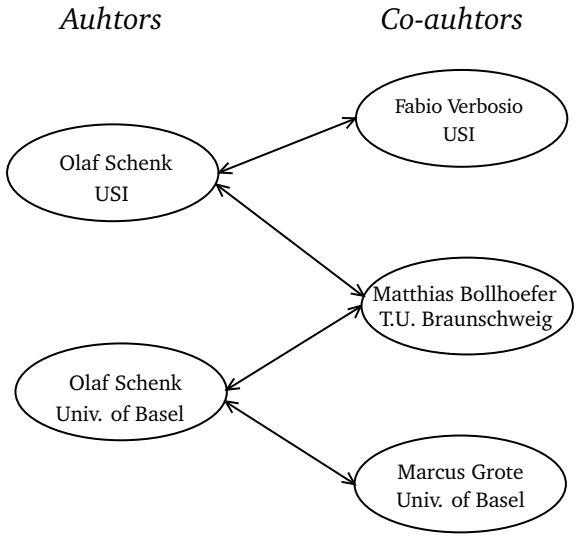


Figure 2. Example of author with two different universities and different coauthors

In this project I personally write a parser, using Python programming language, to extract the information from the conferences' html pages. The parser go through the structure of the page and extract only the pieces where there are useful information; clearly different page structures require different parser. I write two of them because all the SIAM conferences have the same structure but the PASC ones are different.

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

Obtained the desired lines 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.

I run into some problems like an author with two different institution and I decide to consider it as two different authors 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 has two different universities, Università della Svizzera Italiana and Univerisity of Basel: he collaborates with Matthias Bolhoefer with both institutions but he works with Fabio Verbosio only in USI and with Marcus Grote only in the University of Basel.

I make two different databases, one for the Switzerland and one for all the world. The first one contains only Swiss authors and their collaboration with members of the same nation, instead in the second there are other nations authors together with the Swiss ones but 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 these information that 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'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.

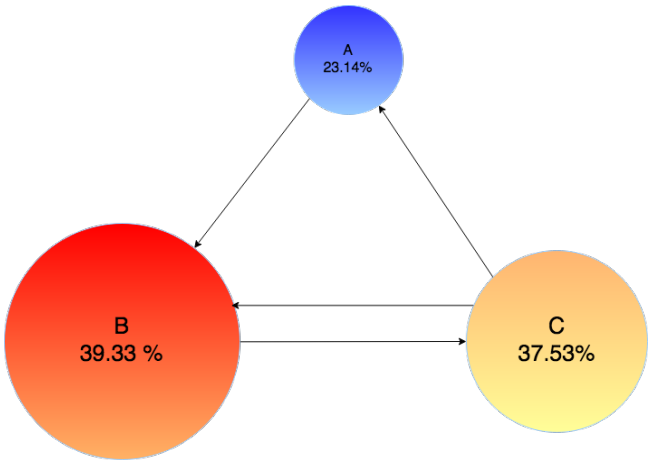


Figure 3. PageRank example

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 of 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,
- 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 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: $c_i = \sum_{j=1}^{n} g_{ji}$

Since my graph is undirected, 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 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

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

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 chian, all its eigenvalues are real, between 0 and 1, and 1 is an eigenvalue itself.

We can now compute PageRank values by solving the problem.

$$A\,x = x$$

It is a particular eigenvalue problem because usually they are of the form $Ax = \lambda x$. Finding the solution of eigensystems is complicated and doesn't exist direct methods; the process has to be iterative such as the power or Jacobi methods.

In our specific problem we have $\lambda = 1$ so the power method seems to be the most effective. It computes the eigenvector relative to the greatest eigenvalue and it is our problem since A is a matrix of a Markov chain and the highest eigenvalue is 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.,

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

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.

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)

```
1: procedure PAGERANK
2:     U ← list of authors
3:     G ← adjacency matrix
4:     p ← dumping factor
5:     G ← G − diag(G)  (remove self collaborations)
6:     c ← sum(G,1)  r ← sum(G,2)  (Compute out-degree and in-degree of each node)
7:     D ← Diagonal matrix with reciprocals of out-degrees
8:     (I − pGD)x = e
9:     x ← x/sum(x)  (normalize the result x)
10:    x ← 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.

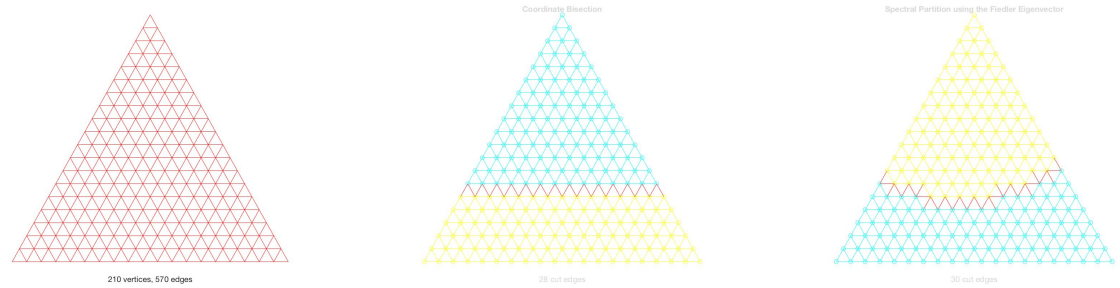
A partitioning algorithm is said to be efficient if it divides a given graph into smaller components with about the same size, cutting as few edges as possible. A valid and fast method is the coordinate bisection, which simply chooses a partitioning plane perpendicular to one of the coordinate axes but spectral algorithm is more efficient. It finds a partition based on the eigenvalues and eigenvectors of the Laplacian matrix of the graph and doesn't need the nodal coordinates.

As we can see in Figure 4 sometimes coordinate and spectral partitioning behave similarly but usually the second one is more effective even if it cuts more edges.

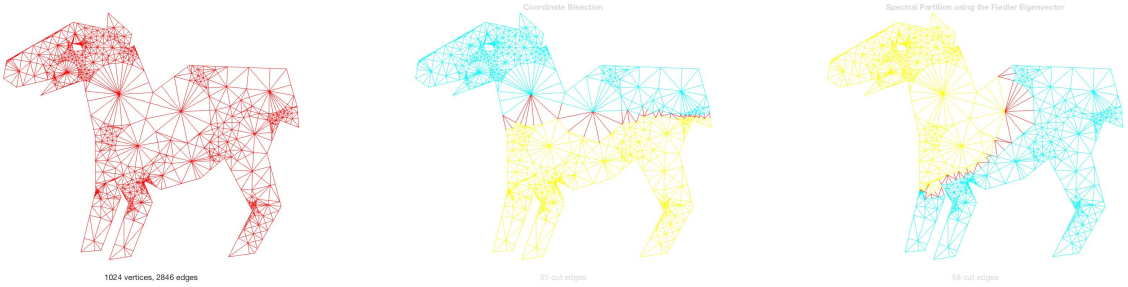
4.1 How to apply spectral graph partitioning

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.



(a)



(b)

Figure 4. 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

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:

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

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** $G1$ = (V1,E1) and $G2$ = (V2,E2)

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 5).

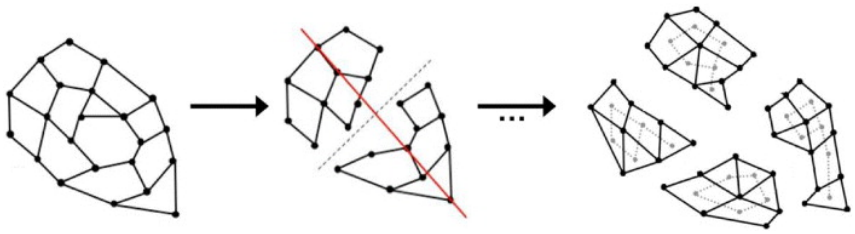


Figure 5. Example of k-way graph partitioning

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)

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 $G_1 = (V_1, E_1)$ ... $G_K = (V_K, E_K)$

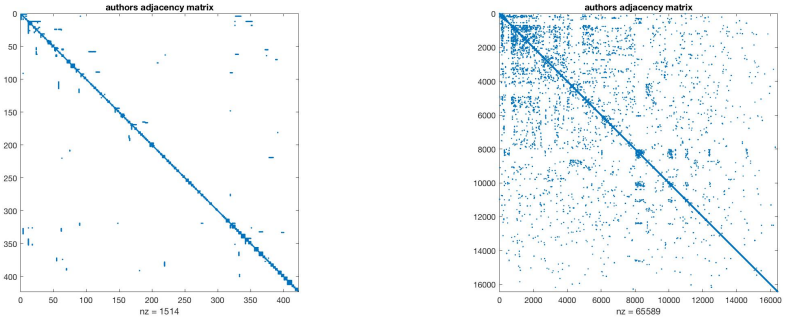
5 Numerical analysis

After retrieving all the necessary information and organizing the data in adjacency matrices, I applied PageRank and Graph Partitioning algorithms, obtaining the following results. Since the Swiss authors' names obtained are only 422, I decide to extend the research to the institutions from all over the word, reaching 16425 names. So 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 universities; for the universities I also build a weighted matrix which keeps the number of collaborations between each university. I consider that each author collaborates with himself so the diagonal of the matrix will have all ones. We can see in Figure 6 the authors' matrices, instead in Figure 7 the one of the Swiss universities.

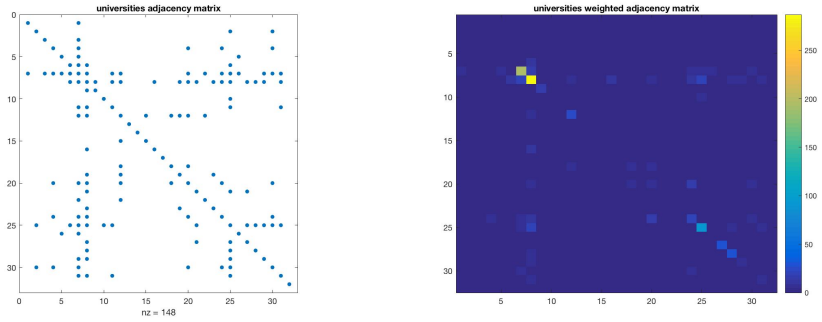
The sparsity of a matrix is the number of zero-valued elements divided by the total number of elements; the Swiss authors adjacency matrix has sparsity equal to 0.0085 instead the one of the world authors matrix is $2.4312 \cdot 10^{-4}$.

In Figure 7(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 yel-



(a) (b)

Figure 6. (a) is the Swiss authors' adjacency matrix and (b) is the one of the world authors



(a) (b)

Figure 7. (a) is the Swiss university adjacency matrix, (b) is the universities weighted graph

low. The universities which collaborate more in Switzerland are École Polytechnique Fédérale de Lausanne and ETH Zürich which correspond to the 7th and 8th columns in the matrix. 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 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 university 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 8.

For the matrix of Swiss authors in 8(a) the bandwidth is 48, instead the one of the authors from all the world 8(b) has a bandwidth of 1210.

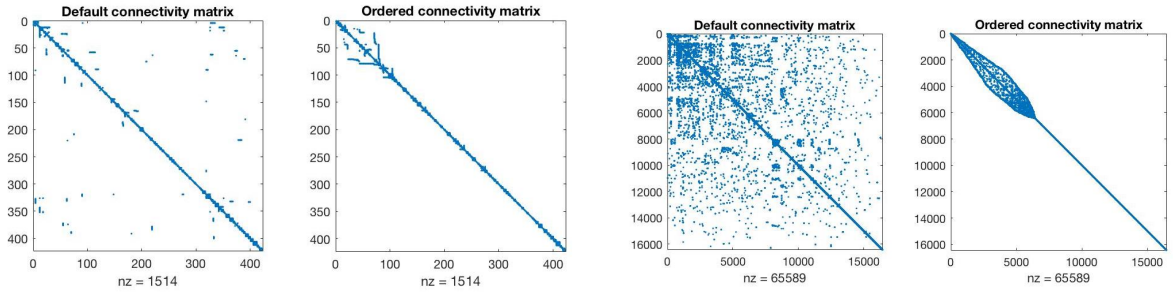
We can notice that in both figures the number of non-zero elements, written below the graph, doesn't change; in fact, the reverse Cuthill-McKee ordering consists only in rows and columns permutations and don't change any entry of the matrix.

5.1 PageRank

Applying the PageRank algorithm I obtain a sorted list of authors and in Figure 9(a) we can see all the 422 Swiss authors with their corresponding PageRank value.

In Figure 9(b) we can see the first 10 most "popular" authors sorted by the algorithm; we can see that the *In_degree* column, which correspond 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 in Figure 10 that contains the first 10 authors sorted only by degree-centrality (i.e. number of collaborations). The most



(a) (b)

Figure 8. The Reverse Cuthill McKee ordering for only Switzerland in (a) and for the authors of all the world in (b)

collaborative author is still *Rolf Krause* because it has 30 associations but from the second we can already see some differences.

I have also applied the PageRank algorithm to the universities' adjacency matrix obtaining the results in Figure 11.

We can observe that the most present universities in the area of computational science are École Polytechnique Fédérale de Lausanne, ETH Züric and Università della Svizzera italiana. Here the universities sorted by PageRank corresponds to the sorting by number of collaborations (*In_degree* column).

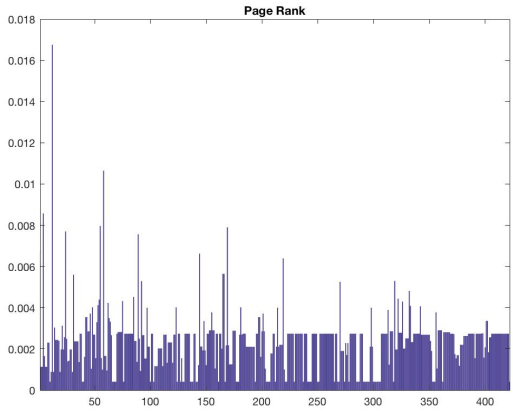
About what concerning world information we can better see the difference between PageRank and degree centrality sorting in Figure 12; the Swiss *Rolf Krause*, member of USI, has the 7th PageRank value with only 38 collaborations but doesn't appear in the first ten positions of the degree centrality list.

In Figure 13 we can see the first 10 universities with higher PageRank value: in the computational science area the most important are the University of California and the Massachusetts Institute of Technology. At the last position we can find the Swiss institution, École Polytechnique Fédérale de Lausanne, which was in the first position in the Swiss list.

5.2 Graph Partitioning

Applying the graph partitioning algorithm I would like to obtain a division between different Swiss universities based on the fact the the members mainly work with people of the same institution. The algorithm, cutting the edges, probably cut the collaboration between different universities and keep the ones internal to 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 Züric, collaborate with almost all other institutions and for this reason appear in all the groups obtained. This results in four groups which are not divided by universities otherwise we can see that all members of the University of Basel are in the second group, instead the ones of USI are almost all in the third group.



(a)

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

(b)

Figure 9. (a) is a bar graph of PageRank values, (b) are the first 15 people with higher PageRank value

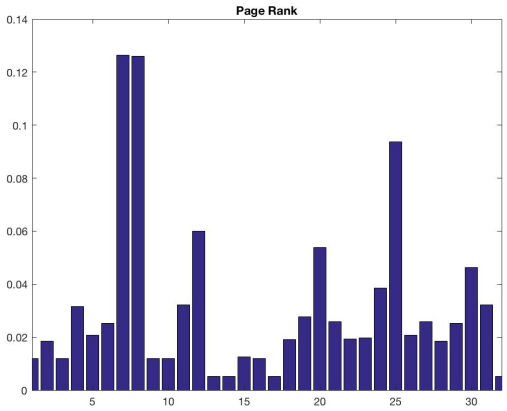
6 Bibliography Temporary

- 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

7 Conclusions

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

Figure 10. Degree Centrality



(a)

Index	Pagerank	In_degree	Out_degree	University
7	0.12632	16	16	'EPFL'
8	0.12601	16	16	'ETHZ'
25	0.09368	12	12	'USI'
12	0.059924	7	7	'IBM'
20	0.053813	7	7	'SIB'
30	0.046199	6	6	'University of Geneva'
24	0.038438	5	5	'UNIL'
11	0.032109	4	4	'HES-SO Valais-Wallis'
31	0.032109	4	4	'University of Neuchatel'
4	0.031569	4	4	'CHUV'

(b)

Figure 11. (a) is a bar graph of PageRank values, (b) are the first 15 universities with higher PageRank value

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

(a)

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

(b)

Figure 12. (a) is the words authors sorted by PageRank values; (b) is the words authors sorted by degree centrality

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

Figure 13. PageRank list of universities from all over the world

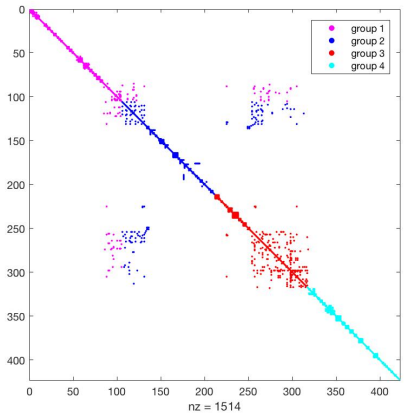


Figure 14. The graph partitioning algortihm