Application of the Louvain Community Detection Algorithm to Demographic Migration Data

Ana Paula Paulino
FATEC

Giancarlo Fleuri
UFG

Rodrigo Zerbini
USP

10 Sept 2014

1 Introduction

1.1 Community Structure

Community structure is a common characteristic in a wide range of networks, such as computer networks, social networks and biological networks. It occurs when groups of nodes are highly connected internally, but sparsely connected to other nodes of the network. These groups are called communities. The image below shows a small network with three communities.

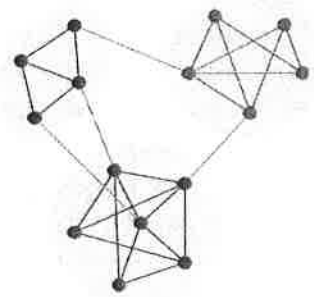


Figure 1: A graph, with highlighted communities

The World Wide Web is another example of network, whose nodes are the webpages and the edges are the hyperlinks between them. We can see a structure community composed of webpages on related topics.

1.2 Our Problem

The Institution called Worldbank has developed a research to help to reduce poverty and support development in the world. In short, in 2010, a research was created in collaboration with the University of Sussex Development Research Centre for the preparation of the South-South Migration and Remittances paper.

As a result of this research, a set containing datas on bilateral migration stocks and bilateral remittance flows worldwide, which quantifies migration between different countries, was created.

In fact, it is possible to represent the bilateral migration in terms of a directed graph, as represented in the following example:

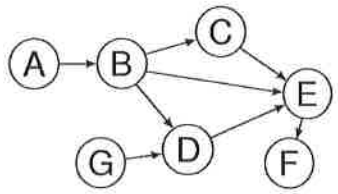


Figure 2: A directed graph

The aim of the project is use Louvain algorithm to investigate the community structure and to try to explain the significance of the comunities in terms of social and geographical concepts.

1.3 Layout of Paper

In section 2 we describe the Louvain method. In section 3 we explain how the Louvain algorithm is written in Java, describing the classes and their main functions and data structures. In section 4 we show the result of apply the Louvain method to the Bilateral Migration Matrix, highlighting its structure. We found 15 communities with one very large (containing more than 70% of the network nodes). In section 5 we briefly interpret these results Sociologically and Geographically. We conclude in section 6 and give suggestions for future work.

2 Louvain Method

To begin with, the Louvain method is an efficient method for identifying communities in networks. These communities are sets of nodes that are

highly connected internally, however the connections between communities are sparse.

This method is based on the modularity function, that measures the quality of a partition of communities. It iteratively searches for new partitions, adopting one every time the modularity is increased. It runs until the modularity can no longer be improved, which means that the optimal partition of communities is reached for the network.

Specifically, we have modelled the world migration as a graph: the countries as the nodes and the migration flows as the link weights. The idea is to find communities of countries that have a dense migration among each other.

Modularity

Modularity Q is a measure of the difference between the sum of the internal edges of a set of nodes and the expected sum of the internal edges. Because we want Q to be between 0 and 1 we need to divide by something proportional to the number of edges (m).

Number of Edges

For a directed graph:

$$m = \sum_{i=1}^{n} \text{out}(v_i) = \sum_{i=1}^{n} \text{in}(v_i)$$

Expected Number of Edges between two Nodes

For a directed graph, the expected number of edges from node v_i to node v_j is:

$$\frac{\text{out}(v_i)\text{in}(v_j)}{m}$$

This is because:

1. The number of expected edges from v to w should be proportional to the product of the out-degree of v and the in-degree of w.
2. if we add up all the expected edges we should get the total number of edges.

$$\text{Sum of all expected edges} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\text{out}(v_i)\text{in}(v_j)}{m}$$

$$= \frac{1}{m} \sum_{i=1}^{n} \text{out}(v_i) \sum_{j=1}^{n} \text{in}(v_j)$$

$$= \frac{1}{m} mm$$

$= m$ as required.

Computing Modularity (directed graphs)

If A_{ij} is the actual number of edges from v_i to v_j, then the actual - expected is:

$$A_{ij} - \frac{\text{out}(v_i)\text{in}(v_j)}{m}.$$

So the modularity of a set of nodes C is

$$Q = \frac{1}{m} \sum_{i \in C} \sum_{j \in C} \left[A_{ij} - \frac{\text{out}(v_i)\text{in}(v_j)}{m} \right]$$

We divide by m to ensure the sum of the modularities of all the communities is between -1 and 1.

Moving an isolated node v into a community C

Doing this, the modularites of all the other communities will remain the same. So before moving the sum of the modularities of the two communities is:

$$Q_{\text{before}} = \frac{1}{m} \left(\sum_{i \in C} \sum_{j \in C} \left[A_{ij} - \frac{\text{out}(v_i)\text{in}(v_j)}{m} \right] + A_{vv} - \frac{\text{out}(v)\text{in}(v)}{m} \right)$$

$$Q_{\text{after}} = \frac{1}{m} \sum_{i \in C \cup \{v\}} \sum_{j \in C \cup \{v\}} \left[A_{ij} - \frac{\text{out}(v_i)\text{in}(v_j)}{m} \right]$$

$$Q_{\text{after}} - Q_{\text{before}} = \frac{1}{m} \sum_{j \in C} \left[A_{vj} - \frac{\text{out}(v)\text{in}(v_j)}{m} + A_{jv} - \frac{\text{out}(v_j)\text{in}(v)}{m} \right]$$

Generalisation to Weighted Graphs

Now A_{ij} can be any real number. Also

$$\text{out}(v_i) = \sum_{j=1}^{n} A_{ij}$$

and

$$\text{in}(v_i) = \sum_{j=1}^{n} A_{ji}$$

Notation

Let v be a vertex and C be a community.

Define $v \to C$ to be the sum of the weights of the edges from v to an element of C.

i.e.

$$v \to C = \sum_{j \in C} Avj$$

Similarly, $C \to v$ to be the sum of the weights of the edges from an element of C to v.

i.e.

$$C \to v = \sum_{j \in C} Ajv$$

Also define the sum of all the weights of edges out of C as $\vec{C}$

i.e.

$$\vec{C} = \sum_{j \in C} out(v_j)$$

Similarly, define the sum of all the weights of edges into C as $\overleftarrow{C}$

i.e.

$$\overleftarrow{C} = \sum_{j \in C} in(v_j)$$

$$Q_{\text{after}} - Q_{\text{before}} = \frac{1}{m}\left[v \to C \quad + \quad C \to v \quad - \quad \frac{1}{m}\left(out(v)\overleftarrow{C} + in(v)\vec{C}\right)\right]$$

Caclulating Change in Modularity moving v from C_1 to C_2

So the change in modularity moving v from C_1 to C_2 is:

$$\frac{1}{m}\left[v \to C_2 \; - \; v \to C_1 \; + \; C_2 \to v \; - \; C_1 \to v \; + \; \frac{1}{m}\left(out(v)(\overleftarrow{C_1} - \overleftarrow{C_2}) + in(v)(\overrightarrow{C_1} - \overrightarrow{C_2})\right)\right]$$

2.1 The algorithm

We have implemented an algorithm to extract communities of countries that present a high flow of migration among them. As explained, the Louvain method works with modularity maximisation and it's known that the quality of the communities identified are very satisfactory.

The algorithm consists of two phases that are repeated iteratively. On the first phase, we create a node for each country and assign a different community to each one. After that, for each country we simulate changing it to the community of each of its neighbours. For each one of these tests, we calculate the change of modularity the network would get if that change was made. After considering all the neighbours of a country, we look at the highest change found. If it is positive, the country is transferred to the community of that neighbour. That means the modularity has improved and the new network presents a better set of communities. The procedure described above is repeated until there are no longer transfers of countries among the communities.

On the second phase, a new network is generated. Each community found on the first phase turns into a node of the new network. The new incidence matrix is calculated. From now on, the network will represent migration flows between communities of countries, instead of singular countries as it was at the beginning.

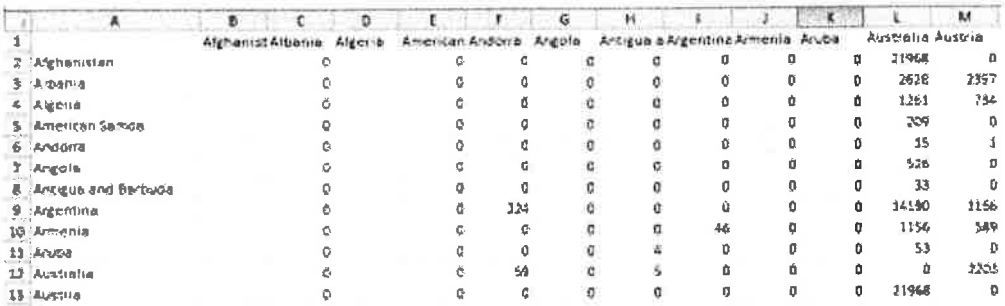
A combination of these two phases can be called "pass". Once a pass is over, another one can be started. At each end of a pass, we get to a new and more compact layer of communities of countries. The algorithm runs until the network can not be contracted anymore.

At that moment, the maximum modularity is obtained and the network is divided in the optimal communities.

3 Explanation of Database extraction and Java implementation of Louvain Method

3.1 Databases Extraction

The Bilateral migration data is a work from The Worldbank Institution, undertaken by a team that combined migration expertise from the Development Economics Vice-Presidency (DEC) and the Poverty Reduction and Economic Management (PREM) Network. Led by DECPG, the team leads the World Bank's contribution to the global policy agenda on migration and development and participation in global initiatives such as the Global Forum on Migration and Development, Global Migration Group, the G8 and the G20. The file, containing all the information about the migration data comes in an .XLS format:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Afghanist	Albania	Algeria	American	Andorra	Angola	Antigua a	Argentina	Armenia	Aruba	Australia	Austria
2	Afghanistan		0		0	0	0	0	0	0	0	21968	0
3	Albania		0		0	0	0	0	0	0	0	2628	2357
4	Algeria		0		0	0	0	0	0	0	0	1261	734
5	American Samoa		0		0	0	0	0	0	0	0	209	0
6	Andorra		0		0	0	0	0	0	0	0	35	1
7	Angola		0		0	0	0	0	0	0	0	526	0
8	Antigua and Barbuda		0		0	0	0	0	0	0	0	33	0
9	Argentina		0		0	124	0	0	0	0	0	14190	1156
10	Armenia		0		0	0	0	0	46	0	0	1156	589
11	Aruba		0		0	0	0	0	0	0	0	53	0
12	Australia		0		0	59	0	5	0	0	0	0	2205
13	Austria		0		0	0	0	0	0	0	0	21968	0

Figure 3: Bilateral Migration Matrix 2010 .xls

Our goal was to read all the numbers of people going from a country to another, with all the 216 countries and regions of the table. Countries with no data, or zero migration were excluded in the pre-processing of the data, implemented in JAVA. This pre-processing has also simplified the way that the Louvain algorithm read the data, reducing the amount of cells and consequently, the quantity reads from 46656 to 30207, providing a better performance to the project. This is basically what the graph file looks like:

Albania	Australia	2628
Albania	Austria	2397
Albania	Belgium	1884
Albania	Bosnia an	63
Albania	Canada	11985
Albania	Chile	48
Albania	Cuba	107
Albania	Cyprus	246
Albania	Czech Rep	180
Albania	Denmark	272
Albania	Dominicar	32
Albania	Ecuador	99
Albania	Finland	69
Albania	France	3037
Albania	Germany	15964
Albania	Greece	676846
Albania	Hungary	117
Albania	Ireland	467

Figure 4: Main information

Tools used for the extraction and pre-processing of the data To manipulate the XLS file reliably and easily, the chosen API was Apache POI[2], very indicated and used for applications such as web spiders, index builders, and content management systems. The IDE used was NetBeans 7.3.1

The Process

The program simply read the XLS file, with 2 loops. One for all the rows and another for all the columns:

```
while (rows.hasNext()) {
   HSSFRow row = (HSSFRow) rows.next();
   System.out.println("\n");
   Iterator cells = row.cellIterator();
   while (cells.hasNext()) {
      Do all the process of saving in the .CSV file)
}
```

Figure 5: code to read the XLS file

The process of reading and filtering all the information, and finally save it all in the new CSV file:

```
HSSFCell cell = (HSSFCell) cells.next();
    if (HSSFCell.CELL_TYPE_NUMERIC == cell.getCellType()) {
     System.out.print(cell.getNumericCellValue() + "     ");
     double a = cell.getNumericCellValue();
         if(a>0){
            HSSFCell origin = row.getCell(0);
            String originToSave;
            originToSave = origin.getStringCellValue();
             HSSFCell destination =
destination2.getCell((cell.getColumnIndex()));
            String destinationToSave;
            destinationToSave = destination.getStringCellValue();
            writer.println(originToSave+ "," +destinationToSave+ ","
+(cell.getNumericCellValue()) );
}}
     else if (HSSFCell.CELL_TYPE_STRING == cell.getCellType()) {

System.out.print(cell.getStringCellValue() + "  ");
     }
    else if (HSSFCell.CELL_TYPE_BLANK == cell.getCellType()) {
    }
    else {
          System.out.print("Unknown cell type");
```

Figure 6: Code to read and filter information

Finally, all the data collected and processed is saved in the .CSV file, ready to be used in Louvain algorithm.

3.2 Java Implementation

The implementation is structured in four classes: Node, Community, Network and Functions.
The class Node is used to represent one node of the network. Each one has an integer id, a list with its neighbours and a name of a country (or a list of countries if the node was generated from a community on the second phase). Also, every node knows the community which it belongs to.
The class Community represents one community of the network. Each one has an integer id and a list of its nodes.
The class Network is the one that contains the main function, as well as the functions for reading the input file and for running both phases of the method. It has data structures for storing the network incidence matrix, as well as its nodes and communities.

The class Functions is the largest one because it contains the majority of the functions needed for the method operation. It is responsible for creating nodes and communities lists, as well as new networks for the second phase. Also, it has functions to calculate a range of different weights: going into and out of a node, going into and out of a community, from a node to a community, from a community to a node. Moreover, there are functions to send a node to a different community and to calculate the variation in modularity as a result of that.
The class Functions is also equipped with some important data structures to store weights related to nodes and communities. Once the algorithm requires a lot of computation, they proved to be very handy for avoiding needless calculation. The data structures count with an auxiliary function to make the necessary updates every time a node is transferred to a new community.

4 Results

In the following figure it is possible to see the community structure through the colours in the map. Each colour indicates a community.

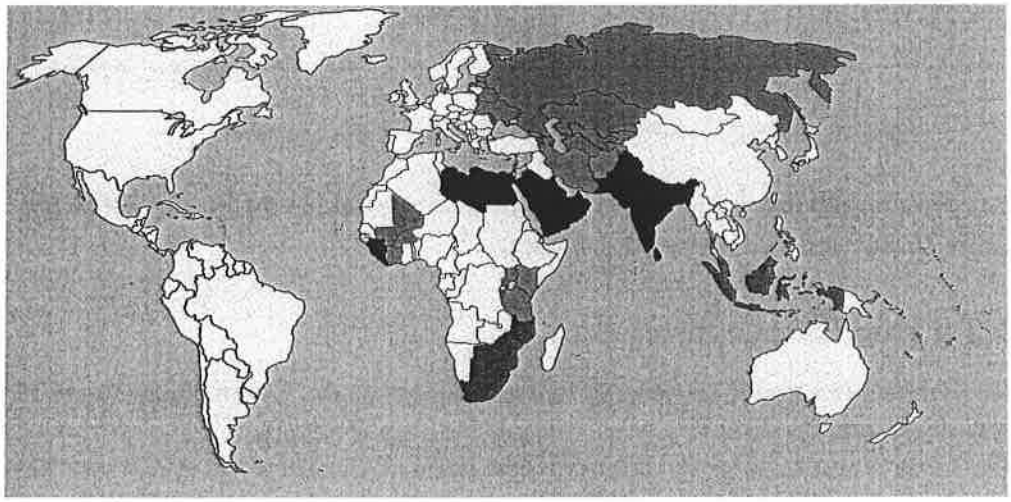


Figure 7: Community structure in the world map

5 Interpretation of the Results

The results showed that most communties are composed by neighboring countries, accordingly [?] it can be due disasters, conflicts or even to take advantage of seasonal weather patterns. Furthermore, it shows a big community

with countries of different parts of the world, this community has included in itself big economies and developing countries, even when we execute Louvain Algorithm only in this community, there was not a different result.

As we can see above, we found 15 communities of countries, which were classified in 4 types: single country, small, medium and big communities.

5.1 Single country communities

They are communities of just one country, all of them being small islands and territories with small populations and without data in the table.

Community 1: Channel Islands Polulation: 163k
Community 2: Gibraltar Polulation: 30k
Community 3: Isle of Man Polulation: 85k
Community 4: Turks and Caicos Islands Polulation: 31k
Community 5: Tuvalu Polulation: 11k

5.2 Small communities (from 2 to 4 countries)

Community 6: Iran Islamic Rep. and Afghanistan

Despite the absence of migration data from Iran to Afghanistan, there is more than 1.7 million migrating in the opposite direction, which corresponded more than 70% of people leaving Afghanistan that year.

Community 7: Coste d'Ivoire, Burkina Faso and Mali

Three countries in northwestern Africa that border each other. There is specially a high flow between Coste d'Ivoire and Burkina Faso. Moreover, 50% of people that leave Mali go to one of the other two countries.

Community 8: Liberia, Guinea and Sierra Leone

Another Community of three countries in northwestern Africa that border each other. The most expressive flows are from Sierra Leone and Liberia to Guinea.

Community 9: West Bank and Gaza, Jordan and Syrian Arab Republic This community in Western Asia encompasses the palestinian territories of West Bank and Gaza as well as Jordan and Syria. A curious cycle is seen here: 51% of people leaving West Bank and Gaza go to Syria, 31% of people leaving Syria go to Jordan, and finally 50% of people leaving Jordan go to West Bank and Gaza.

Community 10: Malaysia, Brunei Darussalam, Indonesia and Singapore Community of close countries situated in Southeast Asia. There is a high flow from all the countries to Malaysia, especially from Singapore and Indonesia. For Malaysia emigrants, Singapore is the destiny for 72% of them.

Community 11: Uganda, Kenya, Tanzania and Burundi Community of close countries in East Africa. The most important flows of migration are: from Uganda to Kenya (70% of the emigrants), from Kenya to Tanzania (20% of the emigrants), from Tanzania to Kenya (29% of the emigrants), from Burundi to Tanzania (42%). Within the community, Uganda is the country that presents the highest number of emigrants and Kenya is the one that receives the most.

5.3 Medium communities (from 7 to 15 countries)

Community 12: South Africa, Botswana, Lesotho, Mozambique, Swaziland, Zimbabwe and Malawi Community in Southern Africa. As expected, most of the migrants go to South Africa as it is the second largest economy in Africa. Malawi being the only exception: 46% of the emigrants go to Zimbabwe.

Community 13: Russian Federation, Armenia, Azerbaijan, Belarus, Estonia, Georgia, Kazakhstan, Kyrgyz Republic, Latvia, Moldova, Tajikistan, Turkmenistan, Ukraine and Uzbekistan This community is formed by Russia and a lot of countries around it. For all the countries, the majority of the emigrants go to Russia. The main destinies for Russian emigrants are Ukraine and Kazakhstan.

Community 14: India, Bangladesh, Bhutan, Kuwait, Nepal, Pakistan, Saudi Arabia, Sri Lanka, United Arab Emirates, Rep. Yemen, Egypt, Libya, Oman, Qatar and Bahrain This community encompasses some countries from North Africa, Arabian Peninsula and South Asia. Two countries receive the majority of the emigrants: India (from Bangladesh, Nepal, Pakistan and United Arab Emirates) and Saudi Arabia (from Kuwait, Sri Lanka, Yemen, Egypt).

5.4 Big communities (more than 15 countries)

Just one big community was found containing 155 countries, which represents 72% of our initial network. The largest community encompasses the entire continents of America and Oceania, the majority of Africa and the majority of Europe.

Differently from the other communities, this one does not have expressive migration flows in general. They are scattered over a lot of countries, what makes the community not highly connected internally. The highest flows are to United States and United Kingdom, which seem to be the factor that ties all the countries in this community.

They are, respectively, the first and sixth gross domestic products in the world and countries of english language, the global *lingua franca*. These seem to be plausible reasons to explain this fact.

Community 15: United States, American Samoa, Antigua and Barbuda, Bahamas, Barbados, Belize, Bermuda, Canada, Cayman Islands, China, Colombia, Cuba, Denmark, Dominica, Dominican Republic, Ecuador, El Salvador, Faeroe Islands, Fiji, Grenada, Guam, Guatemala, Guyana, Haiti, Honduras, Hong Kong, Iceland, Italy, Jamaica, Japan, Kiribati, North Korea, South Korea, Kosovo, Laos, Lebanon, Macao, Marshall Islands, Mexico, Micronesia, Mongolia, Montenegro, Northern Mariana Islands, Norway, Palau, Panama, Peru, Philippines, Poland, Puerto Rico, Romania, San Marino, Somalia, Spain, St. Kitts and Nevis, St. Lucia, St. Vincent and the Grenadines, Sweden, Switzerland, Thailand, Trinidad and Tobago, United Kingdom, Uruguay, Venezuela, Vietnam, Virgin Islands, Andorra, Argentina, Aruba, Australia, Belgium, Bolivia, Brazil, Cambodia, Chile, Cyprus, Djibouti, Finland, French Polynesia, Germany, Ghana, Greece, Greenland, Hungary, Iraq, Ireland, Liechtenstein, Lithuania, Luxembourg, Macedonia, Maldives, Malta, Mauritius, Morocco, Myanmar, Netherlands, Netherlands Antilles, New Zealand, Nigeria, Papua New Guinea, Paraguay, Portugal, Samoa, Seychelles, Solomon Islands, Suriname, Timor-Leste, Togo, Tonga, Turkey, Vanuatu, Albania, Angola, Austria, Benin, Bosnia and Herzegovina, Bulgaria, Cameroon, Cape Verde, Chad, Congo, Croatia, Czech Republic, Ethiopia, France, Gabon, Guinea-Bissau, Israel, Madagascar, Monaco, Namibia, New Caledonia, Nicaragua, Niger, Sao Tome and Principe, Senegal, Serbia, Slovak Republic, Slovenia, Sudan, Tunisia, Zambia, Algeria, Central African Republic, Comoros, Congo Dem. Rep., Costa Rica, Equatorial Guinea, Eritrea, Gambia, Mauritania, May-

6 Conclusion and Future Work

Finally, the resuts shows us that most communities are formed by neighboring countries, regardless their size or how big is their economy. It may be caused by disasters, conflicts with another countries or to take advantage of seasonal weather patterns. Moreover, a big community was created, may be due the high flow between this countries. Also the amount of data available in this work provides several possibilities for research and development of new data. Upon completion of our main objective, possible new research lines based on the same graph used in the current work arose. Below is some of the next steps:

- Calculation of relevance of each country in great community (155 countries) - In this step, each country would be removed from the larger community. This exclusion would change the community in some way, dramatically in the case of countries with large flow (United States and United Kingdom) or not so in the case of countries with lower significance. Our goal here is to create a concrete graph with numbers and information about the relevance of each country, based on the new sub-communities created after these exclusions.

- Remove the direction of the communities to each other – Change the way that the algorithm works and analyse the new results, with a non-directed graph.

- General improvements in the performance of the algorithm.

References

[Erdős and A. Rényi(1961)] P Erdős and A. Rényi. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, 12(1-2): 261–267, 1961. doi: 10.1007/BF02066689.

[Evans and Lambiotte(2009)] T. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1):9, July 2009. ISSN 1539-3755. doi: 10.1103/PhysRevE.80.016105.

[Girvan and Newman(2002)] M Girvan and M E J Newman. Community structure in social and biological networks. *Proceedings of the National*

Academy of Sciences of the United States of America, 99(12):7821–6, June 2002. ISSN 0027-8424. doi: 10.1073/pnas.122653799.

[Paymal et al.(2011)Paymal, Patil, Bhowmick, and Siy] Prashant Paymal, Rajvardhan Patil, Sanjukta Bhowmick, and Harvey Siy. Empirical Study of Software Evolution Using Community Detection. Technical report, University of Nebraska, Omaha, 2011.

[Porter et al.(2009)Porter, Onnela, and Mucha] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in Networks. *Notices of the American Mathematical Society*, 56(9), February 2009.

[Valverde and Solé(2005)] Sergi Valverde and Ricard Solé. Network motifs in computational graphs: A case study in software architecture. *Physical Review E*, 72(2), August 2005. ISSN 1539-3755. doi: 10.1103/PhysRevE.72.026107.

[Šubelj and Bajec(2011)] Lovro Šubelj and Marko Bajec. Community structure of complex software systems: Analysis and applications. *Physica A: Statistical Mechanics and its Applications*, 390(16):2968–2975, August 2011. ISSN 03784371.

[Wasserman and Faust(1994)] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1994. ISBN 0521387078.