UdG Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Anàlisi estadístic i visualització de la robustesa de xarxes

Document: Memòria

Alumne: Jordi Capdevila Mas

Director/tutor: Josep Lluís Marzo Lázaro
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): setembre/2015

Agraïments

Potser no s'estila molt una secció d'agraïments en aquest tipus de treballs. Sé que això no és una tesi doctoral de 3 anys, és simplement un projecte d'uns quants mesos, però a mi em venia molt de gust escriure aquestes línies, de manera lliure i amb el meu estil.
Tinc la sort de tenir moltes coses a agraïr i ho vull fer aquí i ara, a la primera pàgina i en català, perquè les persones que s'hi puguin sentir aludides ho puguin veure i saber des d'un inici.

De fet, això no és un agraïment només per aquests últims mesos. Vaig arribar a Girona fa 6 anys i encara recordo amb alegria i amb nostàlgia els primers dies de moltes coses: el dia de la matrícula, el primer dia d'arquitectura, el primer dia que vaig entrar al pis, la primera classe d'informàtica, el primer viatge amb la L-8... Aquí puc dir que m'hi sento com a casa, que m'hi han fet sentir com a casa, que no és poc. Només això ja resumeix tot el que puc arribar a agraïr.
Gràcies a tots els professors que he tingut, des del primer dia, a tots els companys amb els que m'he creuat. **Gràcies** als amics que he tingut la sort de fer aquests anys, per les experiències i els bons moments viscuts.
Gràcies a en Josep Lluís Marzo i a en Víctor Torres per haver-me donat fa 2 anys una cosa que ja no s'estila massa, una primera oportunitat. **Gràcies** per la paciència, la confiança i l'aprenentatge.
Gràcies en concret a en Josep Lluís, no només per haver-me tutoritzat el projecte, sinó també per tenir sempre la porta del seu despatx (metafòricament) oberta, per tractar-nos sempre d'igual a igual.
Gràcies també a en Diego Rueda, que va aportar la seva ajuda en l'inici d'aquest projecte.

Estic deixant el més important pel final. Òbviament, **gràcies** a la meva família, als de casa, pel suport i pels ànims incondicionals, sempre. **Gràcies** als meus pares, sense ells no estaria aquí, sense ells no seria la persona que sóc. He tingut la sort de créixer en una família estable, de rebre una educació i uns valors que tothom desitjaria. Sense això res no seria com ara és.

Dedicat a tu, pare,
des d'allà on estiguis mirant-nos

List of Figures

List of Tables

List of Symbols

In order to be more precise, the following basic notation will be used throughout the whole project. Please, refer always to this page for any notation doubt:

Graph-related notation

$\mathbf{G}$ = graph
$\mathbf{V}$ = node set
$\mathbf{E}$ = edge set
$|\mathbf{V}|$ = number of nodes
$|\mathbf{E}|$ = number of edges
$(\mathbf{i}, \mathbf{j})$ = undirected edge between i and j nodes
$\delta_{\mathbf{i}}$ = degree of node i

PCA and robustness surface notation

$\mathbf{P}$ = set of percentage of failures
$\mathbf{M}$ = number of failure configurations, i.e., different elements that fail
$\mathbf{N}$ = number of metrics
$\mathbf{t_0}$ = vector of metrics results without failures
$\mathbf{t_p}$ = vector of metrics results when p% elements fail
$\mathbf{A_p}$ = matrix of $M \times N$ metrics results when p% elements fail
$\mathbf{C_p}$ = covariance matrix of A_p
$\bar{\mathbf{C}}$ = averaged covariance matrix
$\mathbf{v}$ = eigenvector as principal component
$\hat{\mathbf{v}}$ = normalized eigenvector v
λ = eigenvalue
$\mathbf{V}$ = matrix containing N eigenvectors v
$\mathbf{D}$ = diagonal matrix containing N eigenvalues λ
$\mathbf{l}$ = number of relevant eigenvectors
$\omega_{\mathbf{p}}$ = vector of *R*-values*
$\omega'_{\mathbf{p}}$ = decreasing-ordered ω_p
$\mathbf{\Omega}$ = *robustness surface*

Contents

Chapter 1

Introduction

It is well known that we live now in a society more intercommunicated than ever. The clearest example that comes to mind is Internet, which is the huge network that connects many devices and people around the world and is changing and growing rapidly every day.
But many other types of services are required to ensure the modern lifestyle: electricity, running water, gas, public transports...

All these services are distributed through their own network, each with its characteristics, and they have become essential in our everyday life. Unfortunately, all these networks have to deal against failures in their elements everyday, which could endanger the correct service provision: a cut on a power line could provoke a blackout, a blockage on the train tracks could lead to several delays.
Due to the omnipresence of these networks, a failure could involve a big problem upon a portion of the population, that is why it is necessary to have robust networks against these failures, capable to fight against them and minimize their impact.

The study of the network robustness tries to find methods to measure this robustness, thus, being able to design better networks or to enhance the existing ones.

1.1 Motivations

Nowadays, most of the presented projects used to be about software implementation, mainly centered in web or mobile applications. This project offers the possibility to go in depth into a research area, it is therefore a kind of project different to most of the others.

Moreover, the area of network robustness is focusing several lines of research in the last decades and it has a growing importance nowadays. Most of the documentation encountered and listed in the bibliography has been done since year 2005.

Finally, being able to learn and expand the work of a PhD thesis has also been a great motivation.

1.2 Starting point and objectives

As it has been said in the last section, **this project starts from a recent PhD thesis** [10] made by a former student of the Universitat de Girona, Marc Manzano, in collaboration with the research group BCDS[1] (Broadband Communications and Distributed Systems). Its main objective is to find a way to measure the network robustness using several metrics summed all together. In order to refer to this project more easily, from this point on it will be named as *ALPHA project*.

Starting from this, the objectives of the current project are:

- Replicate the functionalities, methodologies and concepts introduced in *ALPHA project*.
- Once achieved the latter, thoroughly analyse its functioning and enhance and correct all the necessary aspects.
- Automatize and generalize the whole process to make it applicable to any kind of network, attack, metric...
- Implement a method to visualize the implied data and networks in the process.

1.3 Project outline

The project structure is basically the same as the one proposed in the official document, except for the viability study, which has not been done because this project is settled into a research area and a viability study is not that necessary or suited in this case.
Moreover, chapter 8 includes a section with a complete test example, which also serves as a user manual. That is why there is not an exclusive user manual section.

Chapter 1 introduces the project scope, its starting point and its final objectives. **Chapter 2** and **chapter 3** describe the methodology and the planning used for the project development, respectively. **Chapter 4** defines the necessary previous concepts and describes *ALPHA project* in depth. **Chapter 5** describes the functional and non-functional requirements of the project. **Chapter 6** presents deeper analysis and decisions made on the basic concepts, and also defines the chosen software solutions. **Chapter 7** specifies the project design, the data model and the use cases. **Chapter 8** describes problems and their solutions found during the implementation. It also includes a complete test example. **Chapter 9** presents the results obtained with the implemented project, and they are compared to the results which could be obtained with *ALPHA project*, in order to draw conclusions about them. **Chapter 10** analyses the accomplishment of the initial requirements and objectives and presents the deviation of the final result respect to these requirements. **Chapter 11** proposes possible future work lines.

[1] `http://bcds.udg.edu`

Chapter 2

Methodology

The followed methodology throughout the project could be classified as a typically agile methodology, given the fact that it is composed of an iterative and incremental development process.
Agile methodologies are focused on improve the quality and the efficiency of the corresponding project development. Therefore, the whole process is planned in short iterations in order to be able to deal with unpredictable changes and readjust the development orientation.

The methodology phases can be seen more easily in figure 2.1. Basically, once the project is selected, the process starts with a previous study of the specific research area and its basic concepts. From here, the study and definition of objectives and requirements of the project start, and the decision of which development tools will used.

At this point, the project development begins with the iterative process. Each iteration begins with a set of new tasks to do. Once chosen, they imply a deeper study about the possible new concepts involved.

At the moment when this analysis is good enough, the task development begins, ending with a testing phase about the new implemented functionalities. If all the tests are satisfactory, the iterative process is closed and starts again. Otherwise, it is logically necessary to go back and analyse and correct the errors found in the tests.

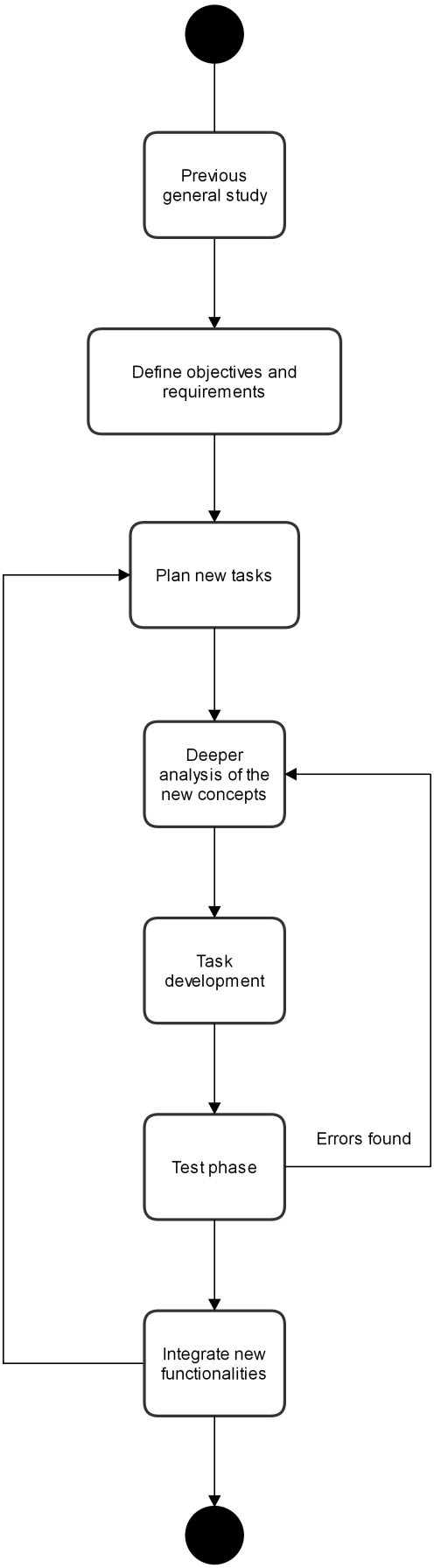


Figure 2.1: Methodology workflow diagram

Chapter 3

Planning

The project began last March 2015. As a project settled in a research area, it had a first study phase as introduction to the field where it belongs to, which in my case was mostly new and unknown.
This first study phase started with the comprehension of the implied basic concepts and a study of the *ALPHA project* as well. The goal was to understand its content and limits.

From here, the first development phase was to achieve the functionalities of the *ALPHA project*. With this first version, the first primary results could be obtained and helped to make a deeper analysis. It helped to correct and improve the general functioning of these first features.
It has to be said that in this phase, Diego Rueda, who is a Colombian PhD student in collaboration with the BCDS group, collaborated with the project giving some help. He provided a basic implementation of a portion of the project (more concretely, the generation of the attacks and the calculation of some metrics). Therefore, this phase was focused in the implementation of the other part (concretely, PCA and *robustness surface* processes). Do not worry about these parts of the project now, they all will be introduced and explained at the right moment throughout this document.

In a second phase, the functionalities provided by Diego were implemented, as well as new functionalities to expand and complement the process.

The documentation was elaborated along the project, also taking advantage of the intermediate analysis which were made. The bulk of the documentation has been made at the end of the process, when I had a more global view of the whole project.

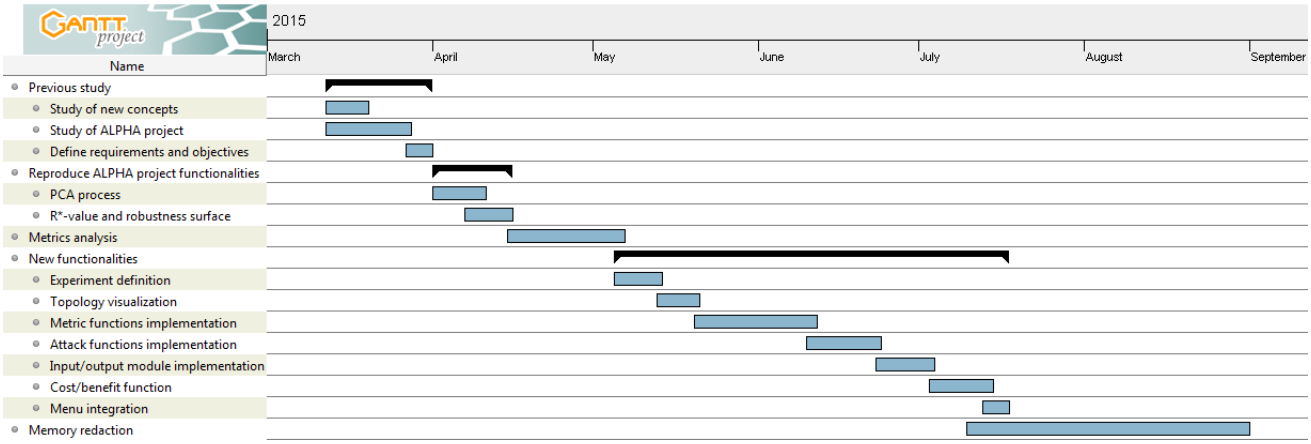


Figure 3.1: Project planning: Gantt chart

Chapter 4

Framework and previous concepts

4.1 Previous concepts

In this section, the basic, main concepts that the reader needs to know are introduced. It may seem that some of them are unrelated to each other but they are all necessary for the whole project comprehension.

4.1.1 Network / topology

Formally, **a network is defined as a set of interrelated elements** with a given purpose. This wide definition could be materialized into countless examples of daily life: the power grid, the telephone, water or gas distribution networks, city streets or subway tracks, the network formed by our neurons or the social networks, which are so trendy nowadays.

In this project, the term *"topology"* will be used. It is a more used term in mathematics area and it will be used here as a synonym of network, although it has more meanings.

4.1.2 Graph

A graph is a mathematical structure used to represent relations between elements. Formally, a graph G is represented as a pair $G = (V, E)$, where V is a node set (representing the elements) and E is an edge set (representing the relations). Actually, an edge is an unordered pair of nodes as well.
This definition describes simple, undirected graphs (in contrast to multigraphs and directed graphs), which will be the ones used in this project.
Graphs can also have labels in its nodes and/or edges which give metadata about these elements. In these cases, they are known as labelled (usually for node labels) or weighted (usually for edge labels) graphs.

4.1.3 Network theory

As it can be deducted from the latter points, **networks are represented by graphs**[1] in mathematics and network or graph theory is the area of mathematics in charge of analysing and studying their

[1] Note that graphs can represent either real networks, such as those defined in 4.1.1, or more abstract ones, for example relations between actors and films

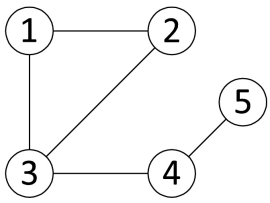
properties.
The range of problems studied by graph theory is so wide and diverse, and among them there is the study of network robustness. Thus, **network theory is the general framework of this project, and more precisely it is included in the study of network robustness.**

4.1.4 Graph theory background

Next, some basic concepts of graph theory are introduced. It is not intended to be a complete introduction to graph theory, the most basic concepts are assumed to be known and only the involved concepts in the project will be explained.

Adjacency matrix

The adjacency matrix is a way of representing which nodes of the graph are adjacent to which other (see figure 4.1). The adjacency matrix of a graph G is denoted by $A(G)_{|V|\times|V|} = (a_{ij})$, where $a_{ij} = 1$ if $(i, j) \in E$, otherwise $a_{ij} = 0$.
For a weighted graph, $a_{ij} = x$ where x is the (i, j) edge weight.



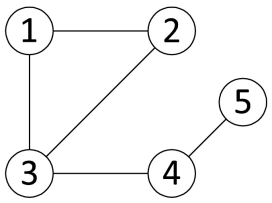
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 4.1: A labelled graph and its corresponding adjacency matrix

Laplacian matrix

Laplacian matrix is known as a matrix representation of a graph (see figure 4.2). Given a graph G, it is denoted by $L(G)_{|V|\times|V|} = D(G) - A(G)$, where $D(G)$ is the degree matrix of the graph which has node degrees on its diagonal and zeroes elsewhere.
The elements of $L(G)$ are defined as following:

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



$$\begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Figure 4.2: A labelled graph and its corresponding Laplacian matrix

Eigenvectors and eigenvalues

In linear algebra, an eigenvector v of a square matrix A is a special vector which does not change its direction when multiplied by A. In other words, Av is a scalar multiple of v, this condition could be written as:

$$Av = \lambda v$$

where λ is a number known as the eigenvalue associated with the eigenvector v.
Eigenvectors and eigenvalues are very used in linear algebra for many problems involving matrices.

Graph spectrum

The classical study of graphs is centred on its nodes and edges, which capture the topological graph characteristics. However, spectral graph theory is the area of mathematics which studies the properties of a graph in relationship to the eigenvalues and eigenvectors of its associated matrices, such as its adjacency matrix or Laplacian matrix.
The set formed by these eigenvalues and these eigenvectors is called the graph's spectrum. Graph spectrum is proved to be useful to detect topological features of the graph or to check the connectivity [14].

4.1.5 Robustness

Robustness is defined as the property of being strong and solid. As an anecdote, robustness means *"oak"* in Latin, which was the symbol of strength and longevity in the ancient world.

Robustness applied to a network is a concept without a unique and standard definition. In the scope of this project, robustness could be defined as

the ability of a network to maintain its throughput under node or edge failures [10, pg. 1].

Thus, **the classical study of network robustness tries to identify how well a network will behave when it is under attacks or failures**. The way to evaluate this behaviour (or throughput, as is in the last definition) is usually to quantify it with a value, which will be the result of calculating a metric on the given network.

The study of network robustness is important in order to evaluate the network itself and try to improve its weak points. This will lead to better networks against failures and it will ensure no interruptions nor affectations to the service they are providing.

4.1.6 Metric

Metric is simply the specific name received by a topological graph property. The concept *"metric"* points out that it gives a quantifiable, comparable measure about this graph property. Every metric has its calculation process or function which is applied to the graph to get the result. Every metric express a different feature of the graph and its meaning could be used as the situation requires.

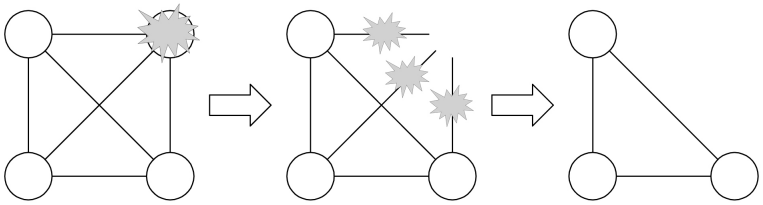


Figure 4.3: A failure on a node affects also its edges

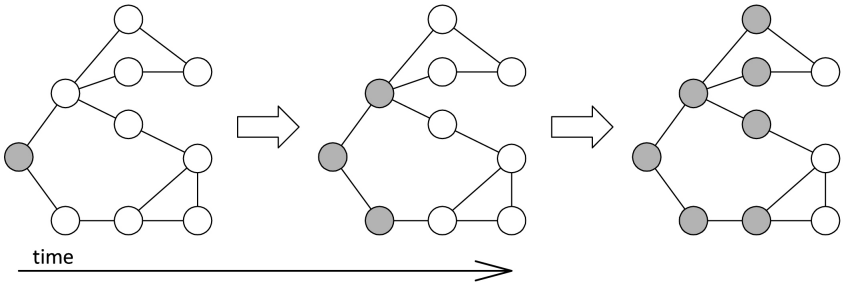


Figure 4.4: Shadowed nodes represent the spreading of a dynamic failure with the time

The simplest metric examples go from the most basic concepts of graph theory (graph diameter or the average degree of its nodes are possible metrics) to other more complex or specific which could be the main object of a whole scientific project.

The metrics used in this project are, in a generic sense, topological metrics which measure a static feature of the graph. There are other metrics which quantify dynamic, time-varying features, such as the flow generated in the topology, the number of collisions, the occupation in the nodes, the spreading of a virus...

4.1.7 Failure / attack

In this scope, a failure on a network element (node or edge) is defined as the physical loss of this element. In a graph, a failure is done by eliminating the given element of the structure.
The concept of *"failure"* is closely related to the concept of *"attack"*, given that the former is provoked by the latter; that is why from here on both terms could be used.

As it can be seen in figure 4.5, attacks can be classified according to different characteristics (extracted from [11]):

- **The attacked element:** it could be either nodes or edges. It has to be mentioned that a failure on a node affects not only the node itself, but also its edges (figure 4.3).
- **The temporal behaviour:** it could be either static or dynamic.
  - Static attacks are like a snapshot of the attack on the topology. Every attack is independent from the others.
  - Dynamic attacks have a continuity in time, the same attack evolves and grows within the topology (figure 4.4). Two clear examples are a failure which produces another failures in cascade, or a virus which first appears in an element and then begins to spread.

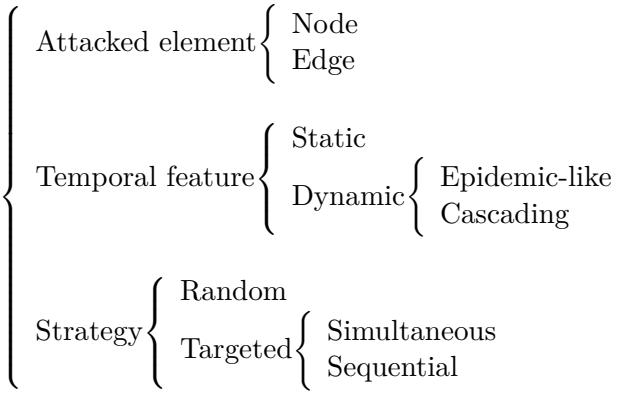


Figure 4.5: Attack classification

This characteristic of the attacks will not be considered in the project implementation. All the attacks will be considered as static.

- **The strategical behaviour:** it could be either random or targeted [10, pg. 1].

  - Random attacks are usually the result of an unintended accident or a natural disaster, one could think of a subway station closing due to a flood or a worker wrecking a water pipe by accident.

  - Targeted attacks are intentionally directed to certain chosen elements in order to maximize the total affectation of the attack on the network throughput (usually using centrality measures A.4). One could think of an attack to a server (a Denial of Service attack) or a terrorist attack to an institutional building. Targeted attacks could be more precisely defined, according to the strategy used to select which elements are attacked:

    * **Simultaneous:** centrality measures are calculated for all the topology elements and then the specified fraction of them are removed in order, from highest to lowest.

    * **Sequential:** centrality measures are calculated for all the initial topology elements and only the highest one is removed. In the resulting topology, centrality measures are re-evaluated again and the new highest central element is removed. This process is repeated iteratively.

	Variables			
	N_1	N_2	$\cdots$	N_N
M_1	x_{11}	x_{12}	$\cdots$	x_{1N}
M_2	x_{21}	x_{22}	$\cdots$	x_{2N}
M_3	x_{31}	x_{32}	$\cdots$	x_{3N}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
M_M	x_{M1}	x_{M2}	$\cdots$	x_{MN}

⇨

	Reduced variables	
	ω_1	ω_2
M_1	x'_{11}	x'_{12}
M_2	x'_{21}	x'_{22}
M_3	x'_{31}	x'_{32}
$\vdots$	$\vdots$	$\vdots$
M_M	x'_{M1}	x'_{M2}

Figure 4.6: PCA example with M observations. The original data passes from N variables to 2

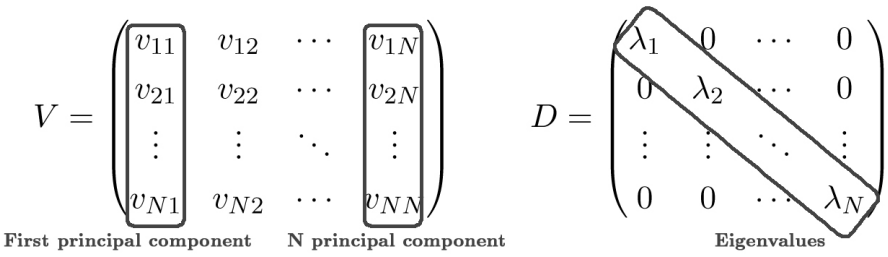


Figure 4.7: V and D matrices

4.1.8 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure used to identify the most relevant information in a data table formed by a set of observations each one defined by a set of variables [10, pg. 6].
PCA is used to analyse and find patterns which allow to **reduce the data table dimensionality** and define each observation with less variables than the original ones, everything without losing relevant information (figure 4.6).
PCA procedure is the following:

- Let $A_{M\times N}$ be a data set of M observations and N variables.
- Compute $\bar{A}$ normalizing and scaling the data in A, denoted by

  $$\bar{A}_j = (A_j - \mu_j)/\sigma_j$$

  where $j \in [1..N]$, μ_j is the mean and σ_j is the standard deviation of jth column of A.
- Let $C_{N\times N}$ be the covariance matrix of $\bar{A}$, denoted by

  $$C_{N\times N} = (c_{i,j} | c_{i,j} = cov(\bar{A}_i, \bar{A}_j))$$

  where $i, j \in [1..N]$ and $cov(\bar{A}_i, \bar{A}_j)$ is the covariance function between column i and column j.
- Compute the eigenvectors and eigenvalues of C. Let $v_{i_{N\times 1}}$ and λ_i be the eigenvectors (also called principal components) and eigenvalues respectively. Let $V_{N\times N}$ be the matrix with all v_i as columns and $D_{N\times N}$ be the matrix with all λ_i in its diagonal and zeroes elsewhere (figure 4.7).
- Finally, let $\tilde{V}_{N\times l}$ be the matrix which only contains the most important l columns of V, where $l \leq N$. Therefore, we can obtain $\omega_{M\times l} = A\tilde{V}$, where ω will be the transformed and reduced data of the original data set.

By this, the original N dimensions have been reduced to l, and the original data can be expressed in l dimensions without losing relevant information.
But there is still an open question, which criterion is used to choose the value of l:

- V and D matrices must be column-sorted in decreasing order, according to the values of the eigenvalues in D.

- The relevance of each eigenvector is characterized by its energy quantum g. The energy quantum of the jth eigenvector is defined by:

$$g[j] = \sum_{k=i}^{j} D[k,k]$$

where $j \in [1..N]$.

- The goal is to define l as a value as low as possible but maintaining a high value of relevance. This value of relevance is up to the situation and can be chosen as you want. Let α be the desired threshold of the relevance wanted to be kept, l will be the minimum value which accomplishes:

$$\frac{g[l]}{g[N]} \geq \alpha$$

Thus, the relevance kept by the l eigenvectors is defined as a percentage between 0 and 1.

4.2 ALPHA project

ALPHA project [10] is the result of the PhD thesis of a former student of the Universitat de Girona, Marc Manzano, in collaboration with the research group BCDS[2] (Broadband Communications and Distributed Systems). As it has been already said, **this project has served as a starting point** of the current project you have in your hands.

ALPHA project addresses two main goals:

- The first one is to **find a general framework to deal with all the existing metrics** related with robustness.
  The fact is that there exist a lot of projects about network robustness and much different metrics are used, but they are usually used alone.
  The goal is to try to find a mechanism which allows to work with several metrics all together to evaluate robustness. This presents two issues: **how it is possible to sum different metrics**, and once found this summation procedure, **which percentage of the total sum brings each metric**.
  The proposed solution in this project is named *R*-value*.

- The second one is to **find a way to graphically visualize the network robustness** variability against the increase of failures.
  Using the *R*-value* to measure the robustness and making different failure experiments above the same network, the goal is to get a graphical tool which allows to visualize the robustness variability between all these experiments and to compare the obtained results on different networks as well.
  The proposed solution in this project is named *robustness surface*.

[2] `http://bcds.udg.edu`

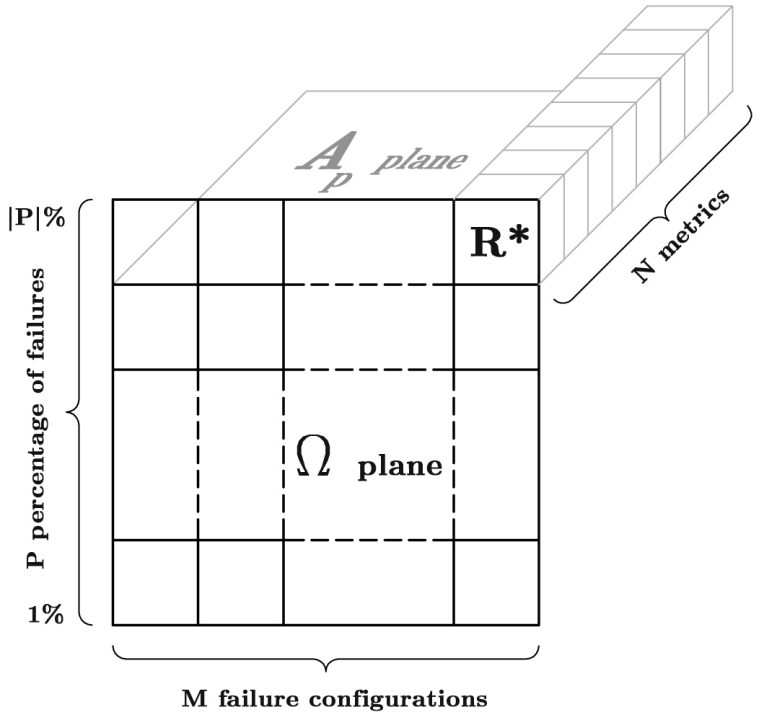


Figure 4.8: *Robustness surface*: graphic scheme

4.2.1 R*-value

The concept *R*-value* comes from Trajanovski et al.[13], who defined a framework to measure the robustness based on the sum of several metrics. This concept was named *R-value* and it was denoted by:

$$R = \sum_{k=1}^{N} s_k t_k$$

where $t_{N\times 1}$ is a vector with the result of several robustness metrics, $s_{N\times 1}$ is a vector with the weight of each of them and N is the number of metrics. Thus, Trajanovski et al. proposed a method to sum multiple metrics, but **they still leave undefined how to assign vector s weight values**.
Starting from this definition of *R-value*, Marc Manzano proposed the ***R*-value* to make use of PCA to find these values of** s. Instead of assigning weights, these are obtained from the first principal component of PCA result, which is applied to all the metric values. *R*-value* is denoted by:

$$R^* = \sum_{k=1}^{N} \hat{v_k} t_k$$

where $\hat{v_k}$ is the obtained principal component. See section 4.2.2 for detailed information on how to obtain $\hat{v_k}$.

4.2.2 Robustness surface

Robustness surface **(Ω) is a matrix where rows are percentage of failures (P) and columns are distinct situations of failures (also called failure configurations) on the same network (M).** $P \in [1\%..100\%]$ denotes the percentage of network elements which fail. The different failure configurations M depict the different subsets of elements that fail for a given percentage of failures.

All the robustness values in $\Omega[p, m]$, where $p \in [1\%..|P|\%]$ and $m \in [1..M]$, are given by the corresponding R^* of the network in that case. Look at figure 4.8 to see a graphic scheme.
This is the procedure described to obtain the *robustness surface* (refer to figure 4.9 to see a graphic example of the process):

- Let $A_{p_{M\times N}}$, where p is the percentage of failures, be the matrix with the results of N metrics in M different failure configurations. Thus, we have $|P|$ A matrices.
- Compute the covariance matrix C_p of each A_p matrix. Average all the $|P|$ covariance matrices to get $\bar{C}$ matrix. Thus, a unique covariation matrix for all the data is obtained.
- Perform PCA over $\bar{C}$ and keep only the first eigenvector, v.
- Obtain $\hat{v}$ normalizing v:

  $$\hat{v_j} = \frac{v_j}{\sum_{k=1}^{N} t_k^0 v_k}$$

  where $j \in [1..N]$ and t_k^0 is the vector with the metric results on the network without failures. **This normalization is performed to ensure that the network without failures will have an *R*-value* = 1**. A greater value of *R*-value* will mean a better robustness, and a lower one will mean a worse robustness, which is the expected with the increasing of failures. It also makes easier to compare results between different networks.
- With the normalized principal component $\hat{v}$, it can be multiplied with every A_p matrix to obtain a vector $\omega_{p_{M\times 1}}$. Thus, the N dimensions of A_p have been reduced to 1. Now, each value of ω_p is indeed the corresponding R^**-value* of the failure configuration, and there are M failure configurations in each ω_p. A set of |P| vectors ω_p is obtained.
- Let ω'_p be the vector ω_p sorted in decreasing order. Finally the *robustness surface* is obtained by concatenating all ω'_p vectors, $\Omega_{P\times M} = [\omega'_1..\omega'_p]$. The vectors ω_p are first sorted to get an Ω surface with smooth changes.

Therefore, *robustness surface* is an ordered matrix with the robustness results of different attacks on a network calculated via *R*-value*. **All these values can be, for example, translated to a scale of colours to be able to visually analyse the variability of the robustness**.

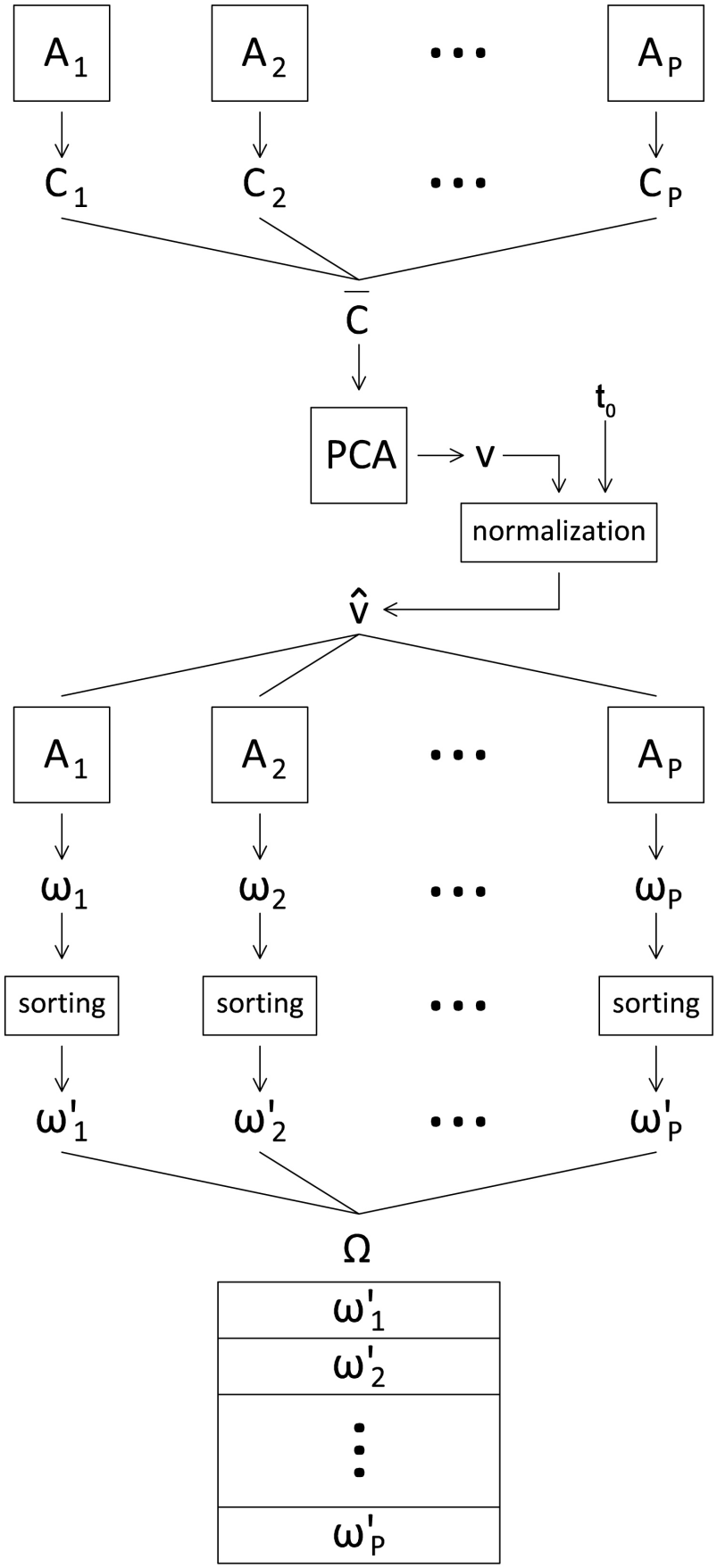


Figure 4.9: *Robustness surface* process: graphic scheme

Chapter 5

System requirements

According to the project objectives (1.2), these are the functional requirements needed to accomplish them, and the non-functional requirements which have to be taken into account:

5.1 Functional requirements

- **Read topologies:** be able to read topologies, saved as graph structures in a file, for its further manipulation.
- **Generate basic topologies:** generate parametrized, basic topology structures (bus, tree, grid...) to be able to experiment with them.
- **Calculate metrics:** implement needed functions to be able to compute all the required metrics on a graph.
- **Generate attacks:** be able to generate different types of attack (4.1.7 section) on network elements, either nodes or edges, and affect either a percentage or a given number of graph elements.
- **Visualize topologies:** find a way to graphically represent topologies with the proper layout and also visualize the attacked elements to see the topology affectation.
- **Compute *robustness surface*:** be able to reproduce the process to calculate the *robustness surface* on a given topology. This requirement includes the previous ones.
  It also includes the process to calculate PCA on a data set and the function to calculate the *R*-value*.
- **Choose the proper metrics:** find a way to choose the metrics to be calculated, according to a cost/benefit criterion.

5.2 Non-functional requirements

- **Functional and adaptable code:** make a code as functional as possible and pay attention to its quality.

Make the whole process customizable and adaptable to different types of experiments, i.e., different types of attacks, percentage of failures, different chosen metrics, different topologies...

- **Easy interaction:** provide the system with an easy input and with a general and reusable output methods. Keep the user interaction easy but sufficient to control all the process features.
- **Efficient code:** the process involves expensive mathematical and statistical calculations, given the fact that it could work with large graphs or matrices. Give special attention to code efficiency and especially avoid unnecessary repetitive processes or minimize their cost.

Chapter 6

Studies and decisions

This chapter presents an introduction to all the implied metrics in the project, and a deep analysis in order to classify and clarify relations between them.
Finally, chosen software-related solutions for the project are explained and justified in relation to the previously presented requirements.

6.1 General metrics classification

As a starting point of the project, it was basic to understand all the metrics in a deep sense. That is why this analysis is one of the first things which was done.
All the metrics presented here were recollected from different scientific projects, where they were used as a robustness metric. Obviously, there exist much more robustness metrics (more complicated ones, more modern or more classical, even time-varying metrics...) but, in the scope of this project, only the ones presented here will be considered.
Firstly, a general classification of all the metrics is proposed in figure 6.1. Although the metrics are introduced here, none of them will be defined throughout this document, given that it will take much space and it is an information better suited in an appendix. Moreover, from this point on, **all the metrics will be referenced by its acronym.** Please, **refer to appendix A to consult any metric definition**.

As it can be seen in figure 6.1, metrics are grouped in two main sets:

- **Structural:** structural measures quantify topological features of the graph. Most of these metrics are classical concepts of graph theory. They can be subdivided in:
  - **Basic structure** (A.1): they are based in concepts like nodal degree, shortest paths and graph spectrum (4.1.4), basic concepts of the graph theory.
  - **Fragmentation** (A.2): fragmentation explains how much is the graph disconnected or how many components does it have. All these metrics give a fixed value while the graph is connected; otherwise, they all begin to give information of the fragmentation at the moment when the graph becomes disconnected.
  - **Connectivity** (A.3): connectivity explains how difficult is to disconnect the graph or how far is this from happening. All these metrics give information of the connectivity while the graph is connected; otherwise, they all give zero when the graph becomes disconnected.

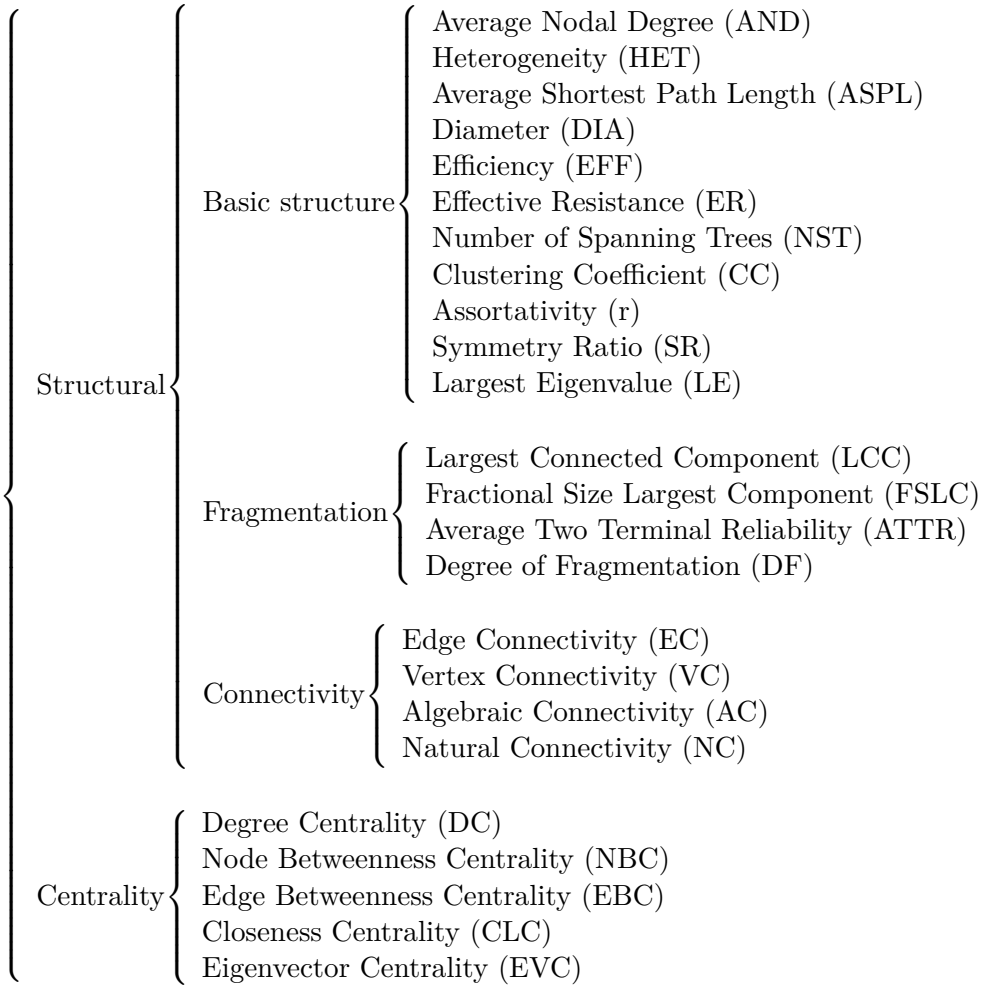


Figure 6.1: Proposed metrics classification

- **Centrality** (A.4): centrality measures are slightly different to the other, because they are usually used to be applied to graph components (nodes or edges). Centrality explains how much important is a given component within the graph structure. That is why these metrics are used to sort graph components by importance, for example, to be used to choose attacked elements in targeted attacks.
  To obtain a centrality measure for the whole graph, all the centrality measures of its elements can be averaged and normalized and the resulting value could be used as a robustness metric.

6.2 Metrics analysis and comparison

Due to the high number of available metrics, it was necessary to make comparisons between them to find special relations like:

- One metric could replace another because they are expressing the same feature.
- One metric has a lower computation cost than another one.
- One metric is harder to compute but it gives more information.

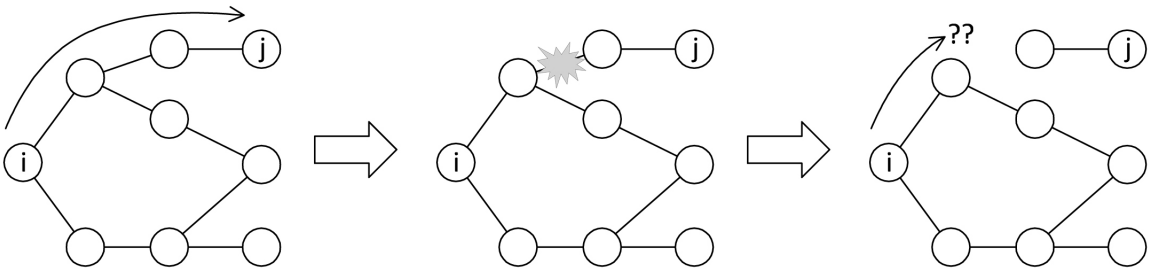


Figure 6.2: Distance between disconnected nodes becomes ∞

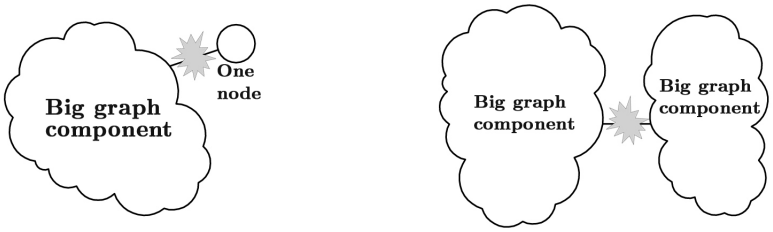


Figure 6.3: Path-related metrics lose its utility in both cases, it does not matter the differences in fragmentation

- Two metrics are expressing the same feature but they give paradoxical results.

The goal of these comparisons was to clarify relations between metrics and to provide criteria to choose between them in a given situation. All these comparisons are available in appendix B, because it is not a crucial information and it is better suited there.
Instead, here are presented some deeper analysis of certain metrics, which are more relevant because they will change the calculation process of the involved metrics.
All these comparisons and analysis were done on basic topologies, such as grids, full graphs, trees... which make it easier to comprehend the problems and study the solutions.

6.2.1 Path-related metrics

Detected issue

All the metrics related with paths between nodes lose part of their meaning at the moment when the graph is disconnected. Look at the figure 6.2, which is the shortest path between nodes i and j when they are not connected? And which is the distance of this path?

Graph theory says that this distance is ∞. This value is mathematically and formally correct but, in practice (or in this project), it becomes useless.
All the implied metrics lose their utility at the moment when the graph (even only one node) becomes disconnected. Their value goes from a discrete number to ∞ or zero. This behaviour is so *"extreme"*, it wastes a lot of possible information (6.3).

Proposed strategy to deal with this issue

Instead of removing elements to represent failures as it has been explained (4.1.7), penalize these elements by moving them away an ∞ distance, which will produce the same effect (see

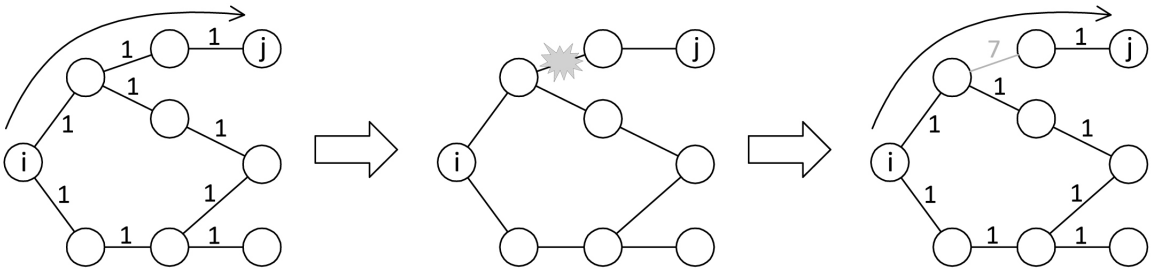


Figure 6.4: Path-related metrics: proposed solution. Initial $d_{max} = 6$

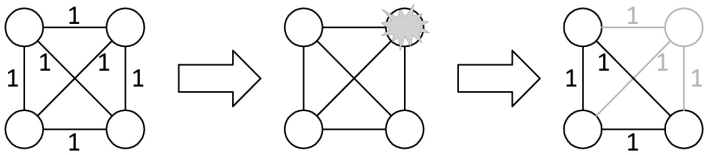


Figure 6.5: Special case of full graphs. Initial $d_{max} = 1$, thus, this value does not represent a penalization without adding one more unit

the example in the figure 6.4). But obviously, ∞ has to be understood as a particular finite value. And *"moving away"* has to be understood as assigning a weight as described following:

- Instead of removing an edge, give it an ∞ cost
- Instead of removing a node, give an ∞ cost to their edges
- All the other unaffected edges would have a cost of 1

This strategy is applicable to ASPL, DIA and EFF metrics. Thus, with this solution, these metrics have become weighted metrics (because they will work on weighted graphs). However, it has many benefits:

- They now work properly in every case, because there will not be disconnected graphs.
- They will continue to give information, even when the graph would become disconnected with the old strategy.
- They now distinguish differences in graphs that would be both disconnected with the old strategy.
- Their trends always follow the same direction with the number of failures and have a smooth profile.

The last question is which is the penalization value. In this case, the chosen value is the initial graph diameter $d_{max} + 1$, given that it is a value with a certain sense, not a randomly one.
Note: it is $d_{max} + 1$ because there are special cases, see figure 6.5. Every full graph has an initial $d_{max} = 1$. Thus, in these cases this value will not represent a penalization without adding one more unit.

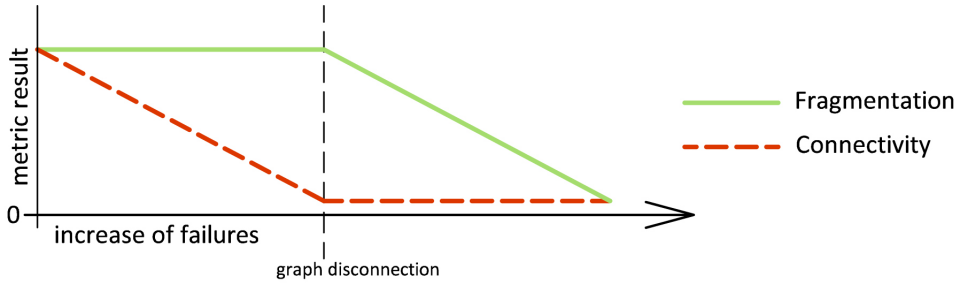


Figure 6.6: Fragmentation and connectivity metrics behaviour

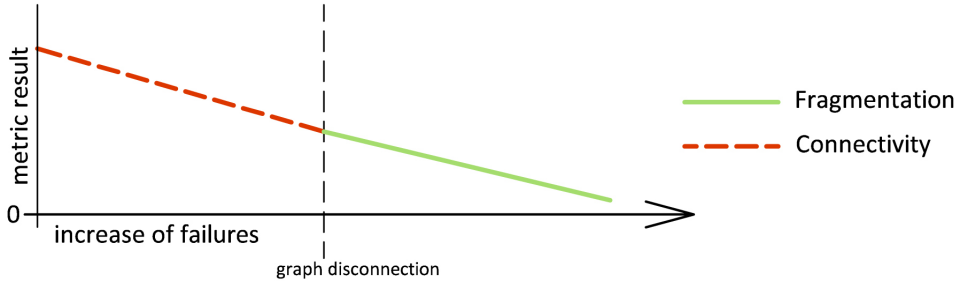


Figure 6.7: Fragmentation and connectivity: proposed solution

6.2.2 Fragmentation and connectivity metrics

Detected issue

As it can be deducted from the definitions in 6.1, **fragmentation and connectivity metrics have a sort of complementary behaviour**.
Taking the increase of failures as a time-line (see figure 6.6), while the graph is connected, one set of metrics gives variable results respect to the growth of failures and the other gives always a fixed result. At the moment when the graph becomes disconnected, roles are changed.

Proposed strategy to deal with this issue

Looking at these behaviours, **it could be possible to get a hybrid metric linking the results of a fragmentation metric and a connectivity metric**, depending on whether the graph is connected or not. Only as a example, VC and ATTR can be linked together (their domains do not overlap): if the graph is connected, connectivity VC results are taken; if the graph is disconnected, fragmentation ATTR results are taken (see figure 6.7).
Thus, with this solution, a hybrid metric will give robustness information in every case.

6.3 Software-related decisions

Next, all the decisions taken respect to software, libraries and formats to be used are described and justified.

Figure 6.8: Project R logo

6.3.1 R language

R[1] **is a programming language oriented to statistical computing and graphics.** It was first created by Ross Ihaka and Robert Gentleman at the Statistics Department of the University of Auckland in 1993. Its current development is in charge of the *R Development Core Team*.
It was mainly influenced by S language (an older statistical language) and Scheme, and its core is written in C and Fortran.

R language has been chosen for the project mainly due to:

- **R is an open-source project** and is freely distributed under the GNU General Public License (GPL). It can provide the same performance as other equivalent commercial languages, like Matlab.
- **R is a cross-platform software**. It is available either in Windows, Linux or Mac OS. Once installed, it can be executed simply via command line interface, although there exist other graphical front-ends. It can be edited with any text editor as well.
- **R is easy extensible via packages**. It has a large community of contributors which develop specific packages to extend R basic configuration. They are all freely available in an official repository.

For these reasons, R is one of the most used languages within the scientific community.
In this project, **release 3.2.1** (June 18, 2015) of the language will be used.

6.3.2 igraph package

igraph[2] **is an R package**, although it has also other versions for Python and C/C++ languages. **It is focused on network analysis** and it is optimized to deal with large graphs. It provides several tools to:

- **Create and manage graph structures**. There exist a lot of methods to create graphs from file sources or also methods to directly create the most classical, famous graphs.
  It gives the possibility to manage graph, node or edge attributes, it gives operations over these components and also provides graph visualization tools.
- **Compute graph algorithms**. A lot of classical graph calculations are already available in the package. Some of the most popular metrics which have been already presented are also available.

This package was chosen because it covers most of the project needs (5.1 section) related with topologies and it is recently maintained and updated. In this project, **release 1.0.1** (June 26, 2015) of the package will be used. The full reference manual is available in [1].

[1] `https://www.r-project.org/`
[2] `http://igraph.org/r/`

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
                 http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
        <key id="d0" for="node" attr.name="color" attr.type="string">
                <default>yellow</default>
        </key>
        <key id="d1" for="edge" attr.name="weight" attr.type="double">
                <default>1.0</default>
        </key>
        <graph id="G" edgedefault="undirected">
                <node id="n0"/>
                <node id="n1">
                        <data key="d0">red</data>
                </node>
                <node id="n2"/>
                <node id="n3"/>
                <edge source="n0" target="n2"/>
                <edge source="n1" target="n2">
                        <data key="d1">2.0</data>
                </edge>
                <edge source="n3" target="n2"/>
        </graph>
</graphml>
```

Figure 6.9: Basic GraphML: equivalent code

6.3.3 GraphML format

GraphML[3] is a comprehensive and easy-to-use file format for graph structures. It is based on XML, thus, it is ideally suited for exchanging graph data between different processes or protocols. There is not an official, standard format for representing or saving graphs, but GraphML is one of the most accepted (as an example, *igraph* package is ready to work with GraphML). The extension of a GraphML file is *.graphml*. GraphML format is licensed under a Creative Commons License.
As it could be deducted from the latter reasons, GraphML format will be used in this project to store the involved topology structures.

Basic example

The following example shows basic GraphML semantics. A more complete document is available in `http://graphml.graphdrawing.org/primer/graphml-primer.html`.

This is a simple example but it contains most of the basic GraphML features (compare figures 6.9 and 6.10):

- The first two lines are the header. The first line is typical in all the XML-type files. It declares the XML version and the encoding of the document.

[3] `http://graphml.graphdrawing.org/index.html`

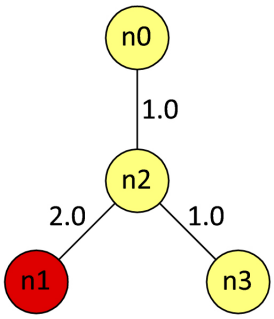


Figure 6.10: Basic GraphML: equivalent graph

The second line is specific for GraphML. It only defines the namespace and the schema of the format.

- Before the graph structure itself, some metadata can be defined. Every *<key>* entry defines a different attribute, with an *id* to reference it, a name, a type and the elements which it will be applied on.

- The graph structure starts with the *<graph>* entry. In this case, it is defined as undirected but it can be directed too.

- Nodes are defined first. Every entry simply contains its *id* to be referenced later. Nodes (as well as edges) can contain *<data>* entries specifying the value of a certain attribute, defined previously in the *<key>* section. Those elements which do not define an attribute will have the default value.

- Edge entries contain two node *id*s, their source and their target node.

6.3.4 CSV format, VBA and Microsoft Excel

Comma-separated values (CSV) files are plain text files used to store tabular data. Each line of the file is called a data record, each consisting of one or more fields, separated by commas (actually, the separator character could be commas, semicolons, even spaces or tabs). The extension of a CSV file is *.csv*.
Due to its simplicity, **CSV is easily compatible and it is a common format to exchange tabular data** between different programs, protocols, processes, operating systems...
CSV will be used in this project for the generated data in the different calculations which needs to be saved for a future use.

ID	Name	Age	Qualifications
1234	Jane	18	7.1, 4.0, 9.5
5252	Peter	24	7.2, 7.9, 6.5
7547	Carl	23	10.0, 9.5, 8.7
7771	Helen	19	5.1, 5.0, 5.5

```
ID,Name,Age,Qualifications
1234,Jane,18,"7.1,4.0,9.5"
5252,Peter,24,"7.2,7.9,6.5"
7547,Carl,23,"10.0,9.5,8.7"
7771,Helen,19,"5.1,5.0,5.5"
```

Figure 6.11: A tabular data and its corresponding CSV representation

The amount of generated results for each experiment could be difficult to be analysed for a person, therefore, it is necessary to **find a graphical tool which presents these results in a more human-friendly way.** Spreadsheet applications provide a proper tool to make it easier as they are very suited for tabular data. Moreover, the user can interact with these results.
In this project, Microsoft Excel[4] will be used to visualize some of these results of the calculation processes.

Visual Basic for Applications (VBA) is an implementation of Microsoft's Visual Basic programming language. It could be **used in some specific programs to build user-defined functions and to automatize processes.**

Thus, data exchanging between R and Microsoft Excel will be done through CSV files, and VBA will be used to automatize the process of importing and preparing the data in Microsoft Excel spreadsheets.

[4]It has to be noted that **this solution has a drawback, as it is only compatible with Windows and Mac OS** operating systems, which have the Microsoft Office suite. Instead of this, an open source solution (like Open Office suite) could has been chosen, but it was not done due to time limitations.
The aim of the project has always been to make it as much compatible as possible. Actually, this is one of the possible future work lines included in chapter 11 to enhance the project.

Chapter 7

Analysis and design

In this chapter, the analysis of the implemented project is presented, explaining in detail the design of its content and functioning.

7.1 General analysis

According to the functional requirements exposed in 5.1, the implemented software environment is designed as shown in figure 7.1.

Functionalities are grouped into different modules and **all the user interaction is done through a general interface** module, which has the proper options to access to all the other modules functionalities. Basically, the other modules are classified in:

- **Auxiliary modules:** they basically manage the input and output data, which is specified below.
- **Functional modules:** they implement the main processes and calculations to satisfy the requirements.

The involved data is organized within the project folder structure, which is basically composed by:

- ***"resources"* folder:** it includes data related to the topologies, which are:
  - topology structure files (6.3.3)
  - results of the metrics calculated without failures. Note that these values are independent from the experiment, they are always the same for a given topology

  It also includes the CSV file with all the experiment declarations, defined in CSV format (6.3.4) as it will be explained in 7.2. In order to make it easier for the user, it is not necessary to define the experiment directly in CSV format. Instead of this, there is a Microsoft Excel spreadsheet where the experiments could be declared in, and a VBA macro could be easily executed to export them into a CSV file.
  For the correct functioning of the whole system, **it is only required that the user provides the topology source files and declares the desired experiments correctly.**

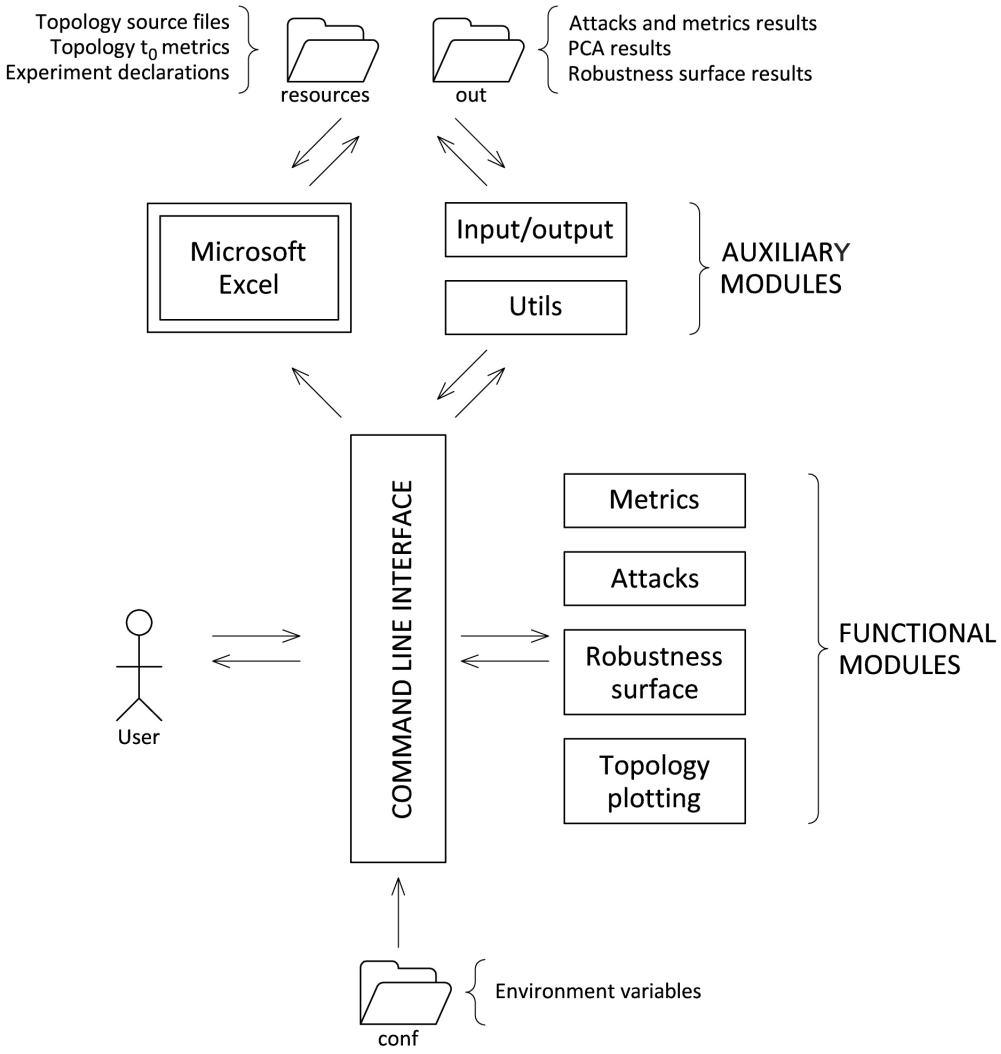


Figure 7.1: Project design: modular scheme analysis

- ***"out"* folder:** it contains a subfolder for every experiment. Each subfolder contains the saved data related to the results of the main calculation processes (PCA process, robustness surface process, generated attacks and metrics...) in CSV format.
  It also contains a Microsoft Excel spreadsheet with all these results presented in a more human-friendly way, that is why the interface module also provides the necessary communication between the user, the data and Microsoft Excel itself.

- ***"conf"* folder:** it contains a configuration file with the basic environmental and path variables. In the beginning, it is necessary to set the value of some variables of the configuration file.

Look at 8.2 to see a step-by-step practical example of the system functioning.

7.2 Data model

As it has been introduced above, **the basic working object of the project is what is called *experiment***. An experiment contains all the attributes to define the topology, the attack, the metrics...

Once the working experiment is defined, any of the functionalities presented on the requirements can be applied to it.

Experiment definition

A experiment is defined by the following attributes:

- **id:** unique identifier composed by a two-digit number.
- **topology:** name of the topology where the experiment will be performed on. This name must be related to the source file which contains the topology structure, as it is described in the step 0 of 8.2.
- **attack**: attack attributes. See 7.2.
- **metrics**: list of metric identifiers to be calculated, see 7.2. This list implicitly brings the N attribute, which is the number of metrics.
- **M:** number of failure configurations, i.e., every configuration is a different subset of attacked elements for a given percentage of failures.
- **P:** upper limit of the failure range.
- **P_mode:** related with the latter, it defines whether P should be taken as a percentage of elements or a discrete value of them.

Attack definition

An attack is defined by the following attributes (refer to 4.1.7):

- **attack_type:** name of the attack type. Possible values: *{random, targeted}*
- **attack_parameter:** extra parameter to define the attack type, if needed. Possible values: *{simultaneous, sequential}*.
  Note: if "attack_type" parameter is different of "targeted", this attribute must be "NA".
- **element_type:** name of the attacked graph element. Possible values: *{node, edge}*

Metric definition

A metric is defined by the following attributes:

- **id:** unique identifier of the metric, composed by a two-digit number concatenated with the acronym of the metric, to make it more human-friendly. See the possible identifier values in table 7.1.

01-AND	Average Nodal Degree	13-FSLC	Fractional Size Largest Component
02-HET	Heterogeneity	14-ATTR	Average Two Terminal Reliability
03-ASPL	Average Shortest Path Length	15-DF	Degree of Fragmentation
04-DIA	Diameter	16-EC	Edge Connectivity
05-EFF	Efficiency	17-VC	Vertex Connectivity
06-ER	Effective Resistance	18-AC	Algebraic Connectivity
07-NST	Number of Spanning Trees	19-NC	Natural Connectivity
08-CC	Clustering Coefficient	20-DC	Degree Centrality
09-r	Assortativity	21-NBC	Node Betweenness Centrality
10-SR	Symmetry Ratio	22-EBC	Edge Betweenness Centrality
11-LE	Largest Eigenvalue	23-CLC	Closeness Centrality
12-LCC	Largest Connected Component	24-EVC	Eigenvector Centrality

Table 7.1: Metric identifiers

id	**topology**	**attack_type**	**attack_parameter**	**element_type**	**metrics**	**M**	**P**	**P_mode**
01	tree16Nodes	targeted	sequential	node	01-AND, 03-ASPL, 17-VC	10	8	discrete

Table 7.2: Example of experiment declaration

Experiment declaration example

With the previous definitions, look at tables 7.2 and 7.3 to see two examples of possible experiment declarations.

- In the first example, a targeted sequential attack is performed on node elements. The attack will affect from 1 to 8 nodes and there will be 10 different configurations, hence a total of 80 different attack configurations.
- In the second example, a random attack is performed on edge elements. The attack will affect from 1% to 50% of the topology edges and there will be 10 different configurations for every percentage, hence a total of 500 different configurations.

Remember that it is not mandatory to make these declarations directly in CSV format. Instead of this, there is a Microsoft Excel spreadsheet where the experiments could be easily declared in, and a VBA macro is executed to export them into a CSV file.

7.3 Process design

In the following section, the basic functioning of the main processes available for the user is described. They are all available from the interface module and involve the functionalities of the other modules, as it will be seen in 8.2.

id	**topology**	**attack_type**	**attack_parameter**	**element_type**	**metrics**	**M**	**P**	**P_mode**
02	networkX	random	NA	edge	04-DIA, 14-ATTR	10	50	percentage

Table 7.3: Another example of experiment declaration

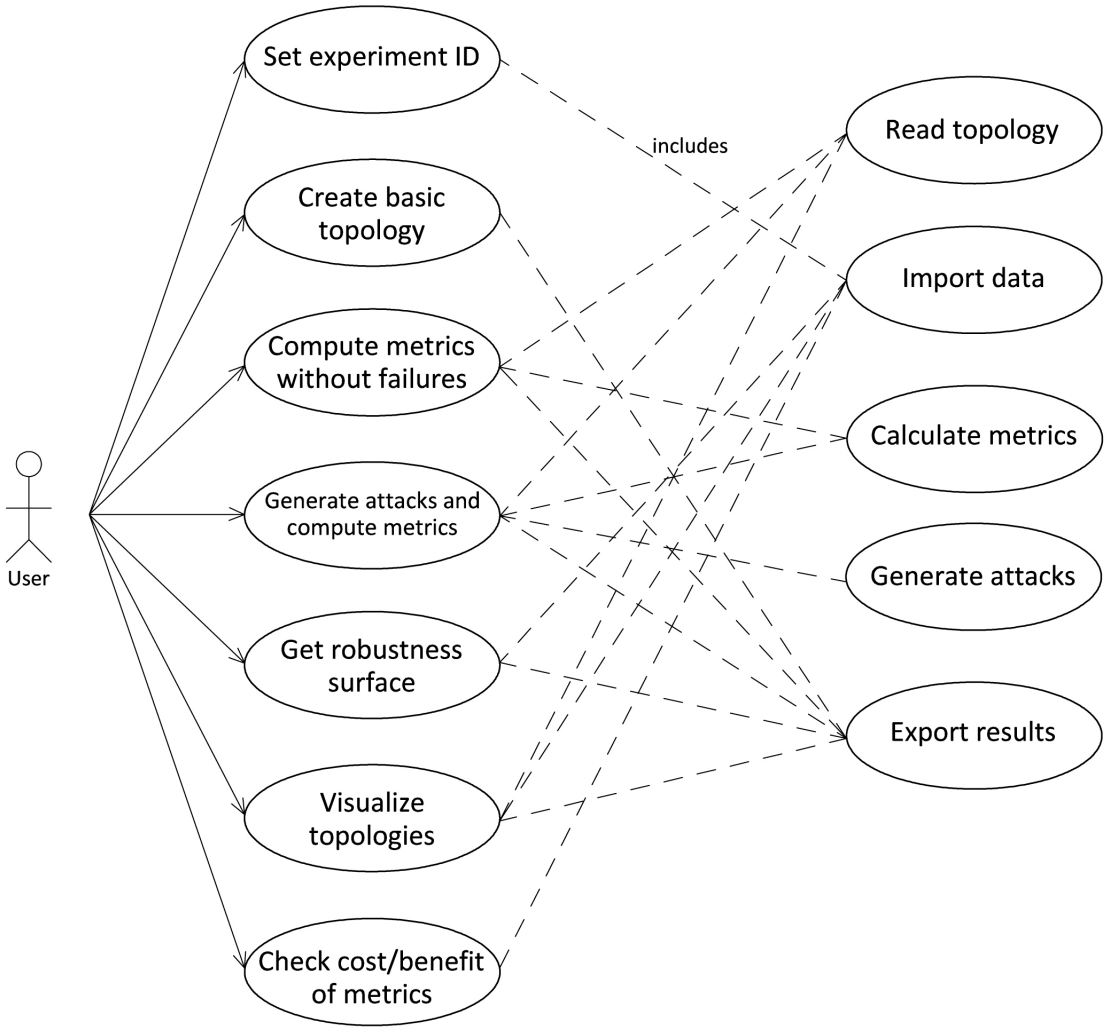


Figure 7.2: Use cases diagram

First, a use case diagram is presented with all the involved use cases. Then, only the most relevant of them are extensively explained with a use case table for each one.

7.3.1 Use cases diagram

The functionalities available from the main interface module are defined in the figure 7.2. They basically cover the functional requirements in 5.1.

7.3.2 Compute metrics without failures

Recalling the process to calculate *R*-value* and *robustness surface* (4.2.2), it is needed to calculate before the metrics on the topology without failures to obtain the normalized vector $\hat{v}$ (see table 7.4). Remember that vector $\hat{v}$ is normalized to ensure that the topology has an *R*-value* $= 1$ when no failures occur, which is supposed to be the more robust situation. Thus, *R*-value* is expected to decrease when failures increase.

COMPUTE METRICS WITHOUT FAILURES	
Description	User wants to calculate a set of metrics over a topology without any failures
Precondition	Experiment ID must be set Experiment attributes linked with this ID must be properly declared A topology source file linked with this ID must be available
Principal flow	1. Read the experiment attributes 2. Read the topology 3. Compute metrics on the topology 4. Export results 5. If there already exist previously exported results 5.1 Ask whether they are wanted to be overwritten 6. End if
Postcondition	Generated results with the metric values are saved

Table 7.4: Compute metrics without failures: use case table

GENERATE ATTACKS AND COMPUTE METRICS	
Description	User wants to generate a set of failure configurations and calculate a set of metrics on them
Precondition	Experiment ID must be set Experiment attributes linked with this ID must be properly declared A topology source file linked with this ID must be available
Principal flow	1. Read the experiment attributes 2. Read the topology 3. For every failure configuration in [1..M*P] 3.1 Generate a attack on the topology 3.2 Compute metrics on the attacked topology 4. End for 5. Export results 6. If there already exist previously exported results 6.1 Ask whether they are wanted to be overwritten 7. End if
Postcondition	Generated results with the metric values and the corresponding failure configurations are saved

Table 7.5: Generate attacks and compute metrics: use case table

7.3.3 Generate attacks and compute metrics

Recalling the process to calculate *R*-value* and *robustness surface* (4.2.2), it is also needed a set of failure configurations with the results of the calculated metrics (see table 7.5).
According to the experiment attributes, a set of $P \times M$ different failure configurations is generated, depending on the attack attributes, and the N metrics are calculated for each configuration.

7.3.4 Get robustness surface

With the corresponding data obtained from 7.3.2 and 7.3.3, *robustness surface* can be obtained following the process described in 4.2.2 (see table 7.6).

GET ROBUSTNESS SURFACE	
Description	User wants to get the *robustness surface* of a set of failure configuration results
Precondition	Experiment ID must be set Experiment attributes linked with this ID must be properly declared Failure configuration results linked with this ID must be already available
Principal flow	1. Read the experiment attributes 2. Read the failure configuration results 3. Perform PCA over this data 4. Perform *robustness surface* process 5. Export results 6. If there already exist previously exported results 6.1 Ask whether they are wanted to be overwritten 7. End if
Postcondition	Generated results with the *robustness surface* and related data are saved

Table 7.6: Get *robustness surface*: use case table

VISUALIZE TOPOLOGIES	
Description	User wants to visualize a certain failure configuration of a topology
Precondition	Experiment ID must be set Experiment attributes linked with this ID must be properly declared Failure configuration results linked with this ID must be already available A topology source file linked with this ID must be available
Principal flow	1. Read the experiment attributes 2. Read the topology 3. Read the failure configuration results 3. While user wants to visualize topologies 3.1 Choose a failure configuration to be visualized 3.2 Visualize topology 4. End while
Postcondition	The desired failure configurations of the topology are visualized

Table 7.7: Visualize topologies: use case table

7.3.5 Visualize topologies

Once data from 7.3.3 and *robustness surface* from 7.3.4 are obtained, it is possible to visualize any failure configuration which is desired. To choose a certain failure configuration, it is needed to choose the corresponding M and P (see table 7.7).
Along with the saved metric results, every failure configuration has also the elements which have been attacked. Thus, the affectation of the attack can be visualized on the topology.

Chapter 8

Implementation and tests

In this chapter, the most relevant problems found during the implementation process and the applied solutions are explained. Actually, they are the implemented solutions of some already presented issues. Finally, there is a complete test example to show the functioning of the previously analysed functionalities.

8.1 Problems found and implemented solutions

8.1.1 PCA process differences

The implemented process to reproduce the PCA calculation has some changes respect to the one which is described in the *ALPHA project* ([10, pg. 2]). In that case, some characteristics of the process were described as follows:

- The original data is not normalized, which causes that the PCA analysis is not correct indeed because it is sensitive to the data scaling.
- The covariance matrix $\bar{C}$ is obtained from the average of $|P|$ matrices of covariance, each corresponding to a percentage of the initial data.
- According to the paper ([10, pg. 7]), the contribution of the first principal component is, at least, 90% of the relevance.

After making a study and comparing results of different strategies, it was decided to implement PCA process as it has been described previously in 4.1.8:

- **The original data has to be normalized to obtain valid results.** Instead of using a covariance matrix, a correlation matrix can be used. It is equivalent to first normalize the data and then calculate the covariance matrix.
- The correlation matrix is calculated over all the initial data (a matrix with $P \times M$ configurations with N metrics each one), it is not split depending on the percentage P. Thus, **the correlation matrix contains the variability of the whole data.** It seems to be more rigorous and better suited in PCA process.

- With the data normalization, **the first principal component cannot achieve 90% of the relevance**, but a maximum of 80%. This happens because the PCA process is very sensitive to data scaling, i.e., a variable with a larger range could monopolize much more relevance. However, with the normalization, all the variable ranges are shrunk to the same domain and the relevance of each variable is more equal to the other.
  Even with the loss of relevance, the PCA calculation process is considered to be more rigorous with this normalization.

A comparison of the relevance of these changes in the PCA process calculation can be seen in 9.1.

8.1.2 Weighted metrics

As it has been seen in section 6.2.1, some metrics work now on weighted topologies with the adopted solution. The whole implemented process works with unweighted graphs, except for these cases.
As described in the proposed solution (6.2.1), before calculating a weighted metric, a default cost of 1 is assigned to all the topology edges, and the attacked elements are penalized with an extra cost.
Thus, **different strategies for treating the metrics and applying the failures in the attacked elements has been implemented for weighted and unweighted metrics.**
A comparison of the relevance of these changes can be seen in 9.2.

8.1.3 Targeted attacks

As it has been explained, **the procedure to evaluate the robustness of a topology includes M different configurations** (i.e. different subsets of elements which fail) for each percentage of failures P. This is done to have more variability and to avoid that special cases could spoil the results.
The strategy to select M different failure configurations is the following, depending on the type of the attack:

- In random attacks it is trivial. Simply M subsets are selected in a random way for every percentage. It has to be said that there is no control about whether the chosen subset is different from the others.

- In targeted attacks it is more complicated. Remember that targeted attacks affect the given percentage of most important elements, according to a centrality measure (4.1.7).
  If you want to be strict, the different M subsets which maximize the importance of its elements must be found, but this is a non-trivial combinatorial problem with a high computational cost.
  To solve this, these conditions have been relaxed a bit, and **a mix between a targeted strategy and a random one** is implemented as the following (see figure 8.1):

  - First, the vector with the centrality measures of all the components is calculated.
  - To ensure M different subsets, the vector is randomized per parts. This is, the elements are split in intervals and the intervals are internally randomized. After this, the percentage of elements which are in the first positions are selected.
  - Thus, the strategy incorporates a part of randomization to obtain different subsets, but it is still oriented to a targeted strategy, because it is not a pure randomization.

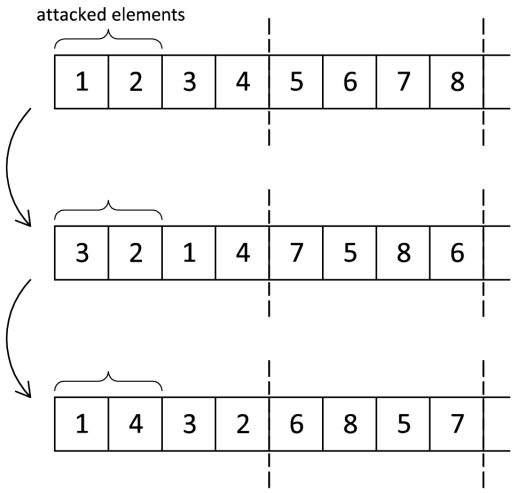


Figure 8.1: Targeted attacks: strategy to choose different subsets

In fact, apart from splitting the vector in intervals to maintain the global position of the elements, the randomization is conditioned to the value of the centrality measure. This is, the elements with higher centrality value have more probability to continue in the first positions of its interval when performing the randomization.

8.1.4 Cost/benefit metrics measure

Due to the high number of available metrics, it is a reasonable idea to find a way to quantify the convenience to calculate a given metric. This convenience measure has to serve as a indicative value to decide whether to incorporate or not the metric in the robustness calculation process.
The implemented solution is based in a cost/benefit function, defined as following:

- **The cost to compute a metric is defined by the required time to calculate it**, which is basically the main drawback of the expensive calculation processes nowadays.
  Once generated the attacks and calculated the metrics of an experiment, the cost measure c_k of each metric is obtained from the mean time of the $P \times M$ calculations of the metric:

  $$c_k = \frac{1}{P \times M} \sum_{i=1}^{P \times M} t_i$$

  where t_i is the required time taken to calculate the metric.
  In order to obtain easily comparable results, these cost values are normalized as described:

  $$\hat{c_k} = (c_k - c_{min})/(c_{max} - c_{min})$$

  where c_{min} and c_{max} are the minimum and maximum cost values of the experiment metrics respectively.
  Thus, cost values are within the range [0..1], where greater values mean greater cost.

- **The benefit to compute a metric is defined by its contribution to the corresponding *R*-value***.
  Once performed the PCA process over the results of the attacks and metrics of an experiment,

the benefit measure b_k of each metric is simply obtained from its corresponding absolute value of the principal component vector v:

$$b_k = abs(v_k)$$

Thus, benefit values are within the range [0..1], where greater values mean greater benefit.

Therefore, **the cost/benefit value is the ratio between the latter variables.** Cost/benefit values are within the range [0..∞], where smaller values mean higher benefit and lesser cost, and high values mean lesser benefit and higher cost.
As it can be deducted, it is first necessary to compute the whole process, at least once, to obtain cost/benefit values of the metrics, which could be used then as an indicative measure on further experiment executions.

8.1.5 R and Excel connection

As it has been explained in 6.3.4 and in 7.1, some of the results of the whole process can be visually analysed with Microsoft Excel, which provides a proper tool to analyse and interact with tabular data in a more human-friendly way. This implies that **the executed process in R and Microsoft Excel must communicate in some way.**
As it has been said, all the necessary results produced by R are saved in its corresponding folder in CSV files, which can be easily imported to Microsoft Excel.

The communication process between R and Excel has been done by the following (see related figure 8.2):

- Whenever the user wants to visualize these data results, R process gives the order to the operating system to execute Microsoft Excel.
- The order is done through the *shell()* R function, which executes a OS native command within R scope.
- In the command, it must be defined the Microsoft Excel path, the Microsoft Excel file to be executed (these last parameters are defined in the configuration file of the project) and some necessary attributes of the experiment.
- Once Microsoft Excel file is opened, a VBA script is automatically executed. It receives the passed parameters from R, which define which is the data to be imported. This script imports and prepares the data for the user and saves the result in a spreadsheet in the corresponding subfolder of the experiment.

8.2 Test example

The following section includes a basic test example to show the most relevant project functionalities. The example shows the steps to perform a complete robustness calculation of an experiment. Therefore, **this section is also intended to be a user manual.**
The basic workflow of the test example is the following:

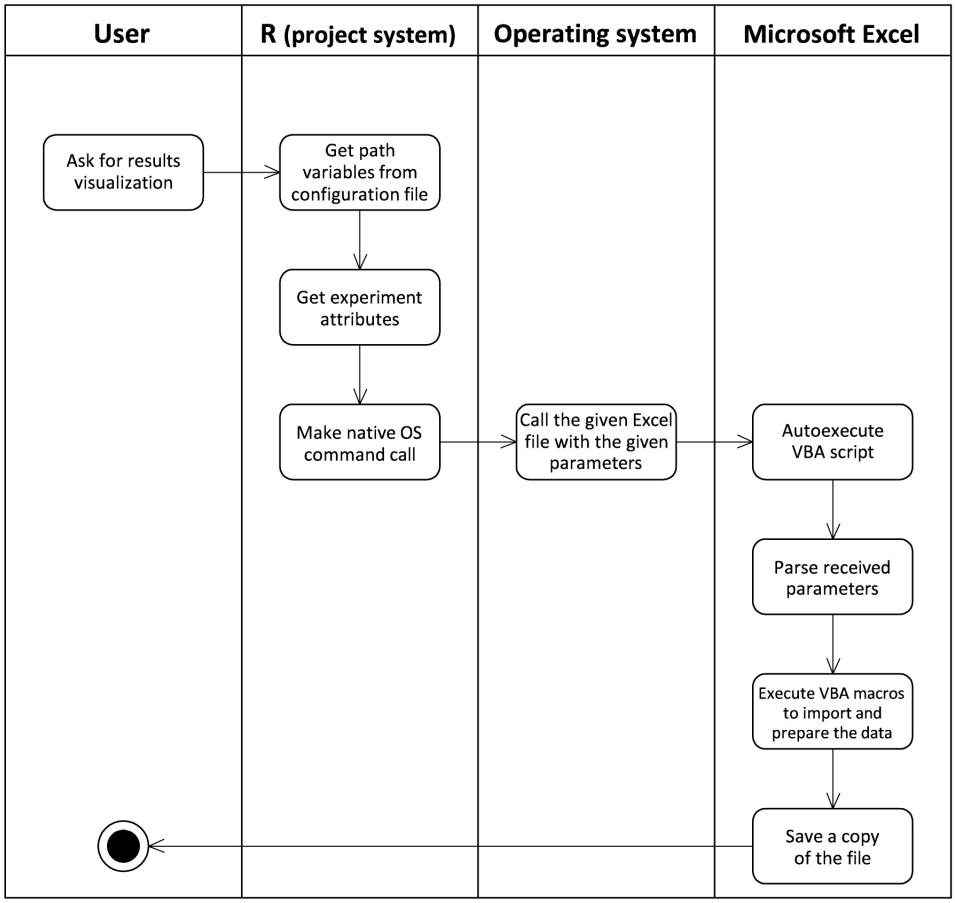


Figure 8.2: R and Excel communication: schematic diagram

Step 0

Before starting the calculation process indeed, the user has to provide the necessary initial data:

- **The experiment has to be declared** in the experiment declarations file, as explained in 7.2. As seen in figure 8.3, the experiment could be more easily declared in a Microsoft Excel spreadsheet and then exported to a CSV file with the highlighted button. More than one experiment could be declared in this file, each one in a different row.
  In this example, the experiment attributes which will be used are defined in table 8.1.

- **A file with the topology structure has to be saved** in the *"resources"* folder of the project. There are two options for the structure file format: either GraphML format, and the file name must be *"{topologyName}.graphml"*; or the adjacency matrix saved in CSV format, and the file name must be *"{topologyName}-adjacency_matrix.txt"*.
  In this example, the chosen topology is named *"GpENI_L2"* and it is available on the website [12], along with other networks. The adjacency matrix of these topologies can be freely downloaded.

- Before executing the project for the first time, **some configuration variables must be properly set.** As seen in figure 8.4, the user has to set the Microsoft Excel path of its operating system in the configuration file.

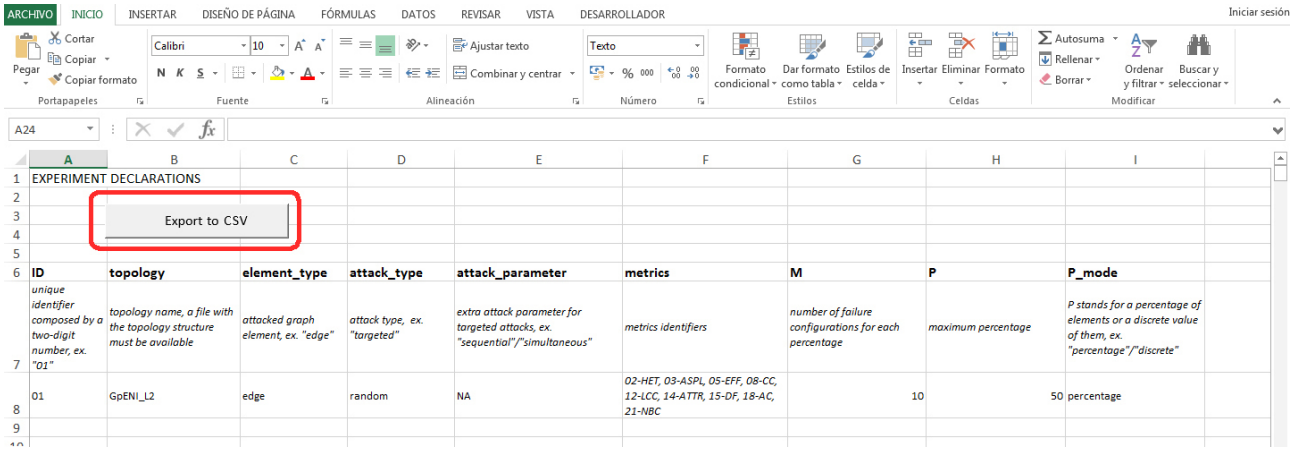


Figure 8.3: Basic test: step 0

```
1  ########################################
2  # ENVIRONMENT PARAMETERS
3  LIBRARIES        <- c("RCurl", "igraph")
4  RESOURCES_PATH   <- "resources"
5  OUT_PATH         <- "out"
6  TEMPLATE_RESULT <- "templateResult.xlsm"
7  ###
8  # Set the corresponding Microsoft Excel path
9  EXCEL_PATH       <- "C:/Program Files/Microsoft Office/Office15/EXCEL.EXE"
10 ###
11
12 ########################################
13 # INPUT PARAMETERS
14 INPUT_SEP        <- ','
15 DECLARATIONS_FILE                    <- "experimentDeclarations.csv"
16 delayedAssign("T0_FILE",             paste0(TOPOLOGY_ID, "-t0_metrics.csv"
```

Figure 8.4: Basic test: step 0

id	topology	attack_type	attack_parameter	element_type	metrics	M	P	P_mode
01	GpENI_L2	random	NA	edge	02-HET, 03-ASPL, 05-EFF, 08-CC, 12-LCC, 14-ATTR, 15-DF, 18-AC, 21-NBC	10	50	percentage

Table 8.1: Basic test: step 0

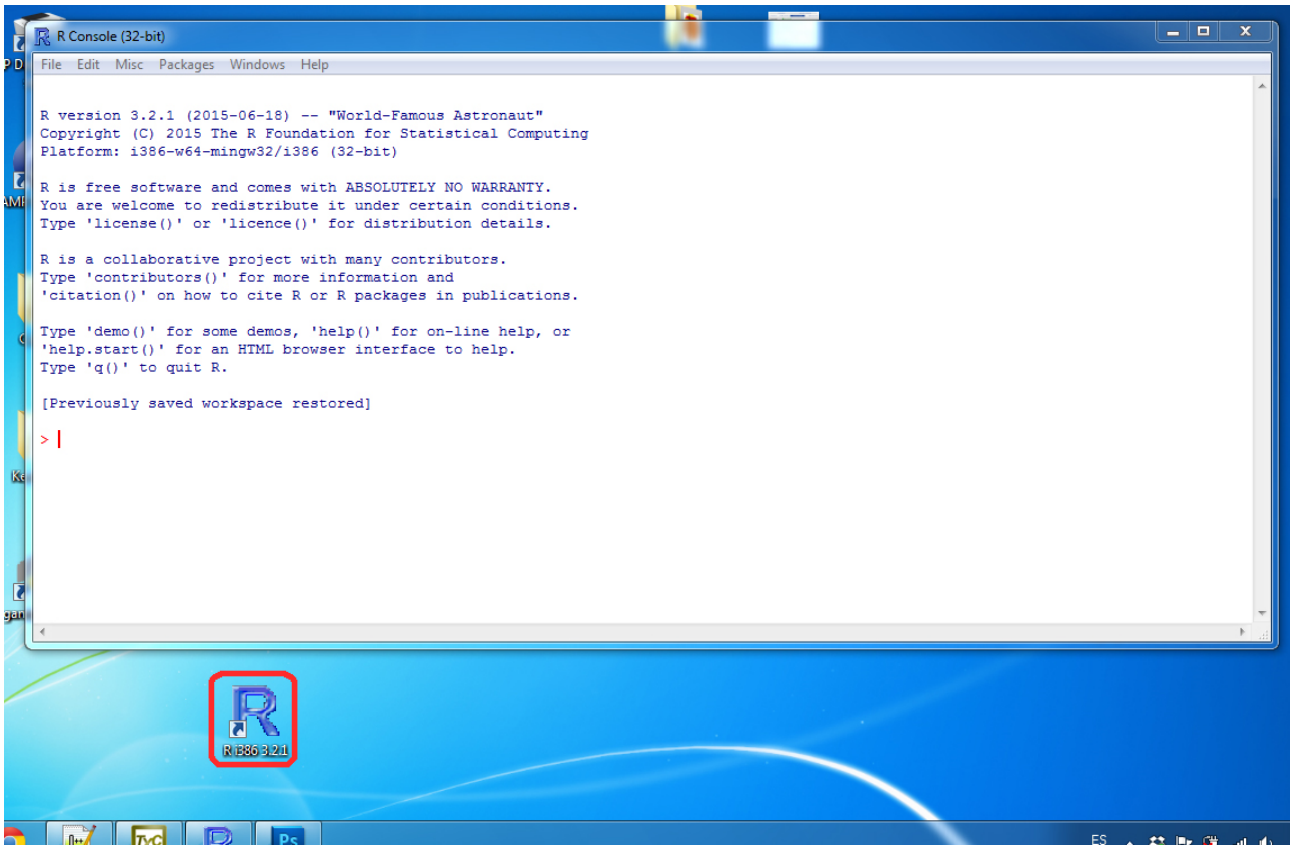


Figure 8.5: Basic test: step 1

Step 1

First of all, the R command interface has to be launched (see figure 8.5). The official page to download it and the version used in this project are the ones cited in 6.3.1.

Step 2

The working directory of R has to be changed to the root folder of the project, labelled with number 2 in figure 8.6. Follow the steps shown in the figure.

Step 3

At this moment, the project is ready to be executed. To launch it, the main interface module has to be executed, using the command seen in figure 8.7. When the project is started, it prompts the user to set the desired experiment ID. This ID could be changed later during the execution using the option [1] of the menu, if wanted.

Step 4

From here on, any of the menu options could be selected, but it is preferred to execute them in the order which will be executed here, because some menu options need previous results from other previous options.
So, the main flow of the process starts with the option [3], which computes the previously declared

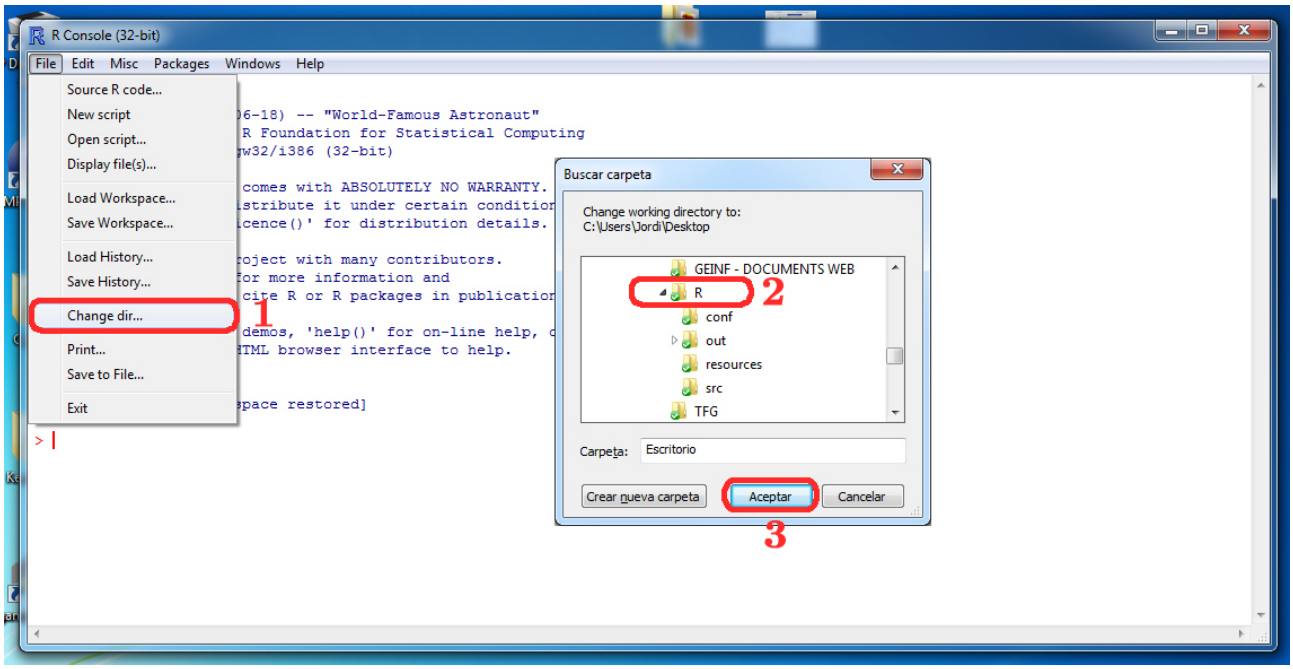


Figure 8.6: Basic test: step 2

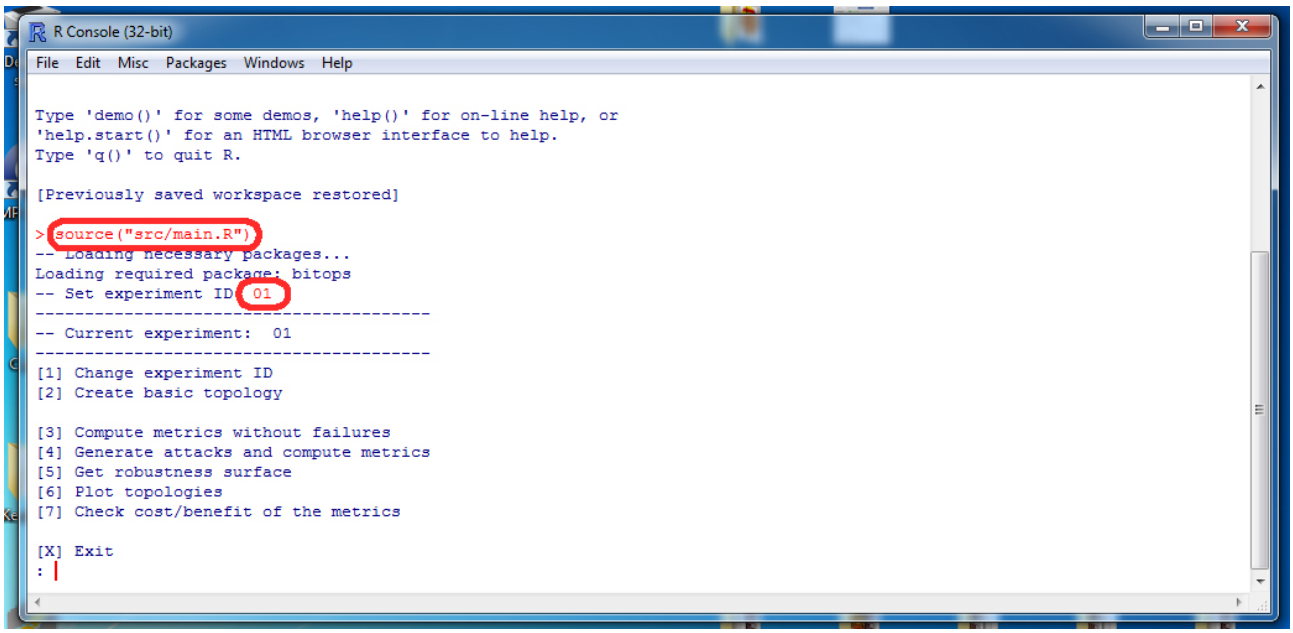


Figure 8.7: Basic test: step 3

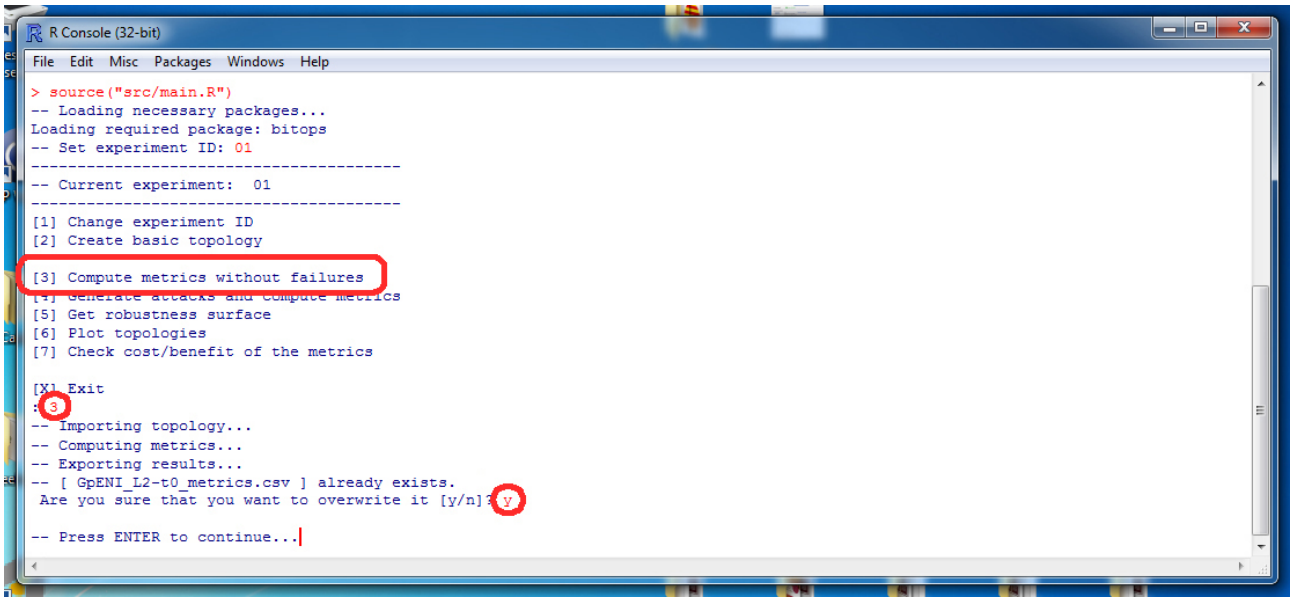


Figure 8.8: Basic test: step 4

metrics over the topology without failures and saves the results in a file. Note that the execution warns the user if there is already the same file with previous results (see figure 8.8).

Step 5

The following step is done with the menu option [4], which generates $P \times M$ different attacks, computes the metrics over each one and saves the results and the required times to compute them. These results are saved in the corresponding subfolder of the *"out"* folder of the project. This step is, by far, the most computationally expensive of the whole process.
With the two previous options executed, the option [5] of the menu computes the *robustness surface* of the experiment data. It also saves the results of the PCA process and the *robustness surface* process. Note that the relevance captured in the PCA process is shown in a message in the command line (see figure 8.9).

Step 6

Once computed the *robustness surface*, the results and the implied attacks can be visualized, using the menu option [6]. This option prompts the user to define a specific failure configuration of the experiment, defining the P and M variables. This will show the corresponding attack over the topology. In order to choose the P and M variables, a Microsoft Excel spreadsheet is automatically launched when this option is selected, as explained in 8.1.5 (this spreadsheet can also be manually opened by the user whenever it is wanted, it is located in the corresponding experiment subfolder). From here, the user can choose whatever failure configuration of the experiment and check the attack over the topology (see figure 8.10).
Note that P and M could make reference either to the initial data or to the Ω matrix of the *robustness surface*, remember that this happens because the failure configurations are sorted in Ω matrix. The title of the plotted topology contains both Ms, in order to be able to relate the failure configuration in both data sets.

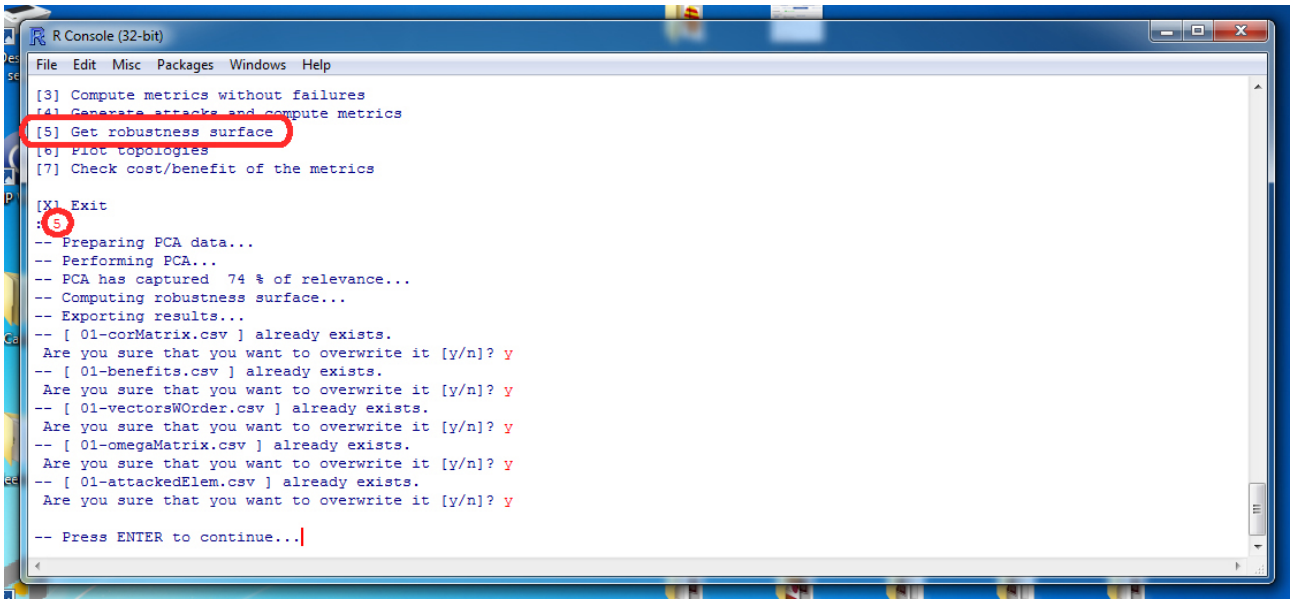


Figure 8.9: Basic test: step 5

In the figure, two screen captures of the content of the Microsoft Excel spreadsheet can be seen. The one on the left corresponds to the initial data of the generated attacks, the one on the right corresponds to the *robustness surface* and in the middle there is their corresponding topology with the attack represented on it.
Many topologies can be visualized at the same time, if wanted.

Step 7

This step is only for the user to obtain additional information of how the calculation process is performing. The menu option [7] prints in the command line the cost/benefit values of the implied metrics in the process, as well as the normalized cost values and the benefit values, as described in 8.1.4. This information can be used to choose a subset of more appropriated metrics (see figure 8.11).

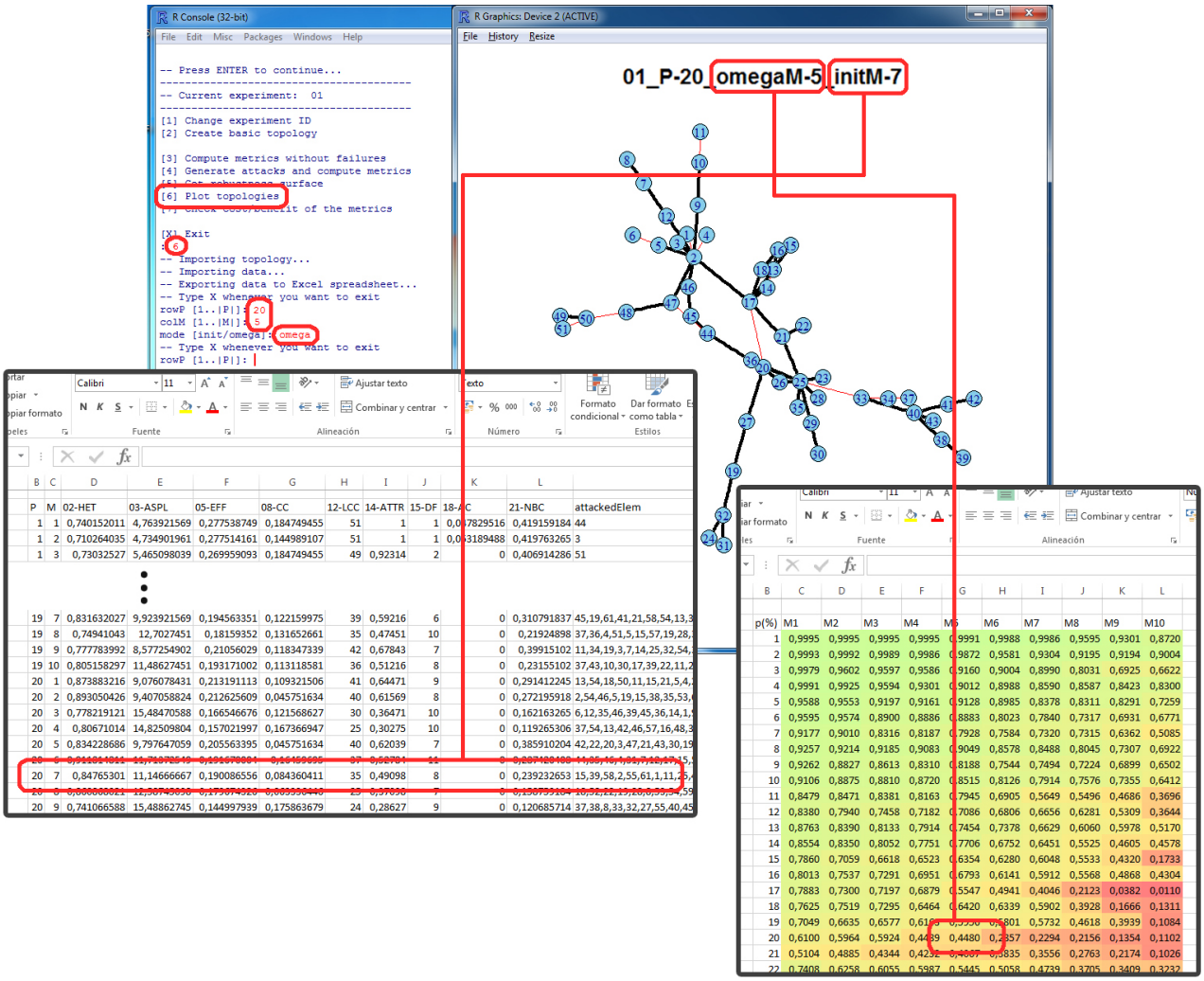


Figure 8.10: Basic test: step 6

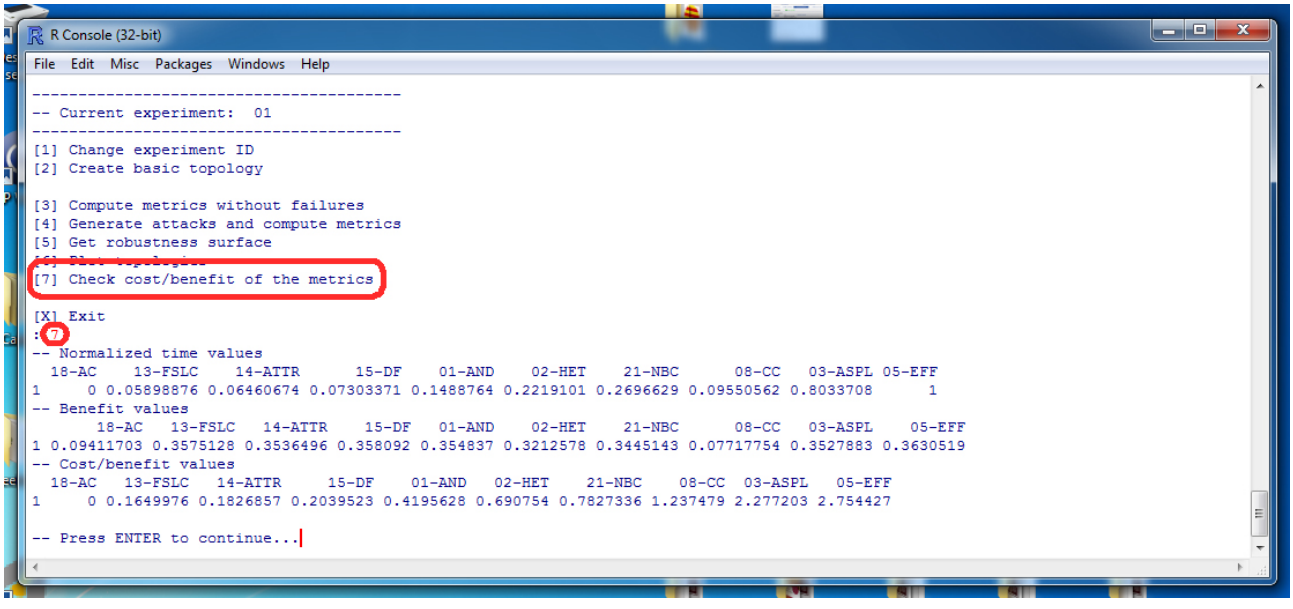


Figure 8.11: Basic test: step 7

Chapter 9

Results

To conclude, in the following chapter some results which could be obtained with the whole process execution are presented. Therefore, the process is performed on a particular experiment and the obtained *robustness surface* is analysed, which is in fact the desired result of the whole process.
In order to check the satisfaction degree of the results and be able to draw conclusions of the introduced changes, **the same experiment is executed both with the process implemented in this project and with the process as it is described in *ALPHA project*** (actually, there is no access to its source code, instead of this, the code of this project is adapted to work as it is described in *ALPHA project*).
Thus, the usefulness of *robustness surface* to compare different results of experiments could be proved.

The chosen experiment is the same which has been used before in section 8.2, look at table 8.1 to see its attributes. *"GpENI_L2"* is a proper topology because it is a real network, but it is not too large (only 51 nodes and 61 edges) and it makes easier to perform tests on it.
The process to obtain the following results is the same which is described in 8.2: calculate the metrics without failures, generate attacks and calculate metrics on them and perform the *robustness surface* calculation with these latter results. From here, visualize the corresponding spreadsheet with the experiment results, which contains the *robustness surface* Ω matrix. The Ω matrix values are translated to a scale of colours to visually analyse the results.
Once executed the two compared experiments, the obtained pair of *robustness surfaces* are analysed and compared in the following sections. First, each section includes a comparison between the implemented project and *ALPHA project*, but only with one of the introduced changes each time, in order to analyse them separately.

9.1 PCA differences

As it is explained in 8.1.1, the most significant change between the PCA process described in *ALPHA project* and the one implemented here is the data normalization. In order to analyse the implication of this difference, the experiment is evaluated with the following strategies:

- **PCA without normalization (figure 9.2):** this is the strategy described in *ALPHA project*. PCA process is performed with a covariance matrix and without normalizing the data. The

relevance kept in this experiment with this strategy is 92%.

- **PCA with normalization (figure 9.1):** this is the strategy implemented in this project. PCA process is performed with a correlation matrix, which normalizes the data at the same time. The relevance kept in this experiment with this strategy is 74%.

In both cases, weighted metrics are used (see next section), thus, **the only difference relays on PCA process.**
As it has been already said, **with the data normalization the first principal component achieves less relevance.** This happens because the PCA process is very sensitive to data scaling and a variable with a larger range could monopolize much more relevance. In contrast, with the normalization, all the variable ranges are shrunk to the same domain and the relevance of each variable is more equal to the other. But it has to be recalled that the correct way to perform PCA is with the normalization.

Differences on PCA process imply that the resulting principal component could be slightly different, and as a consequence, the obtained *R*-values* too. Figures 9.1 and 9.2 show the resulting *robustness surfaces* of the latter strategies. It can be observed that the robustness decreases faster in 9.1.

9.2 Metrics differences

As it is explained in 6.2.1 and in 8.1.2, some of the path-related metrics (ASPL, DIA, EFF...) have been implemented with a weighted strategy. Before presenting the results of this solution, it has to be understood which differences does it imply when applied. To make it easier, it is done with the DIA metric as a example. The differences begin at the moment when the graph becomes disconnected:

- **Classical strategy (figure 9.4):** this is the strategy used in *ALPHA project*. The diameter of each connected component is calculated, and the greatest one is taken as the result of the metric. Therefore, the metric decreases with the growth of failures, because the connected components sizes become smaller.
  Referring to A.1.4, the diameter should increase with the growth of failures, or at least, this is the expected behaviour to take it as a valid robustness metric.
  To sum up, **this metric was giving increasing results while the graph was connected, but at the moment when it becomes disconnected, it changes its trend and it gives decreasing results.**

- **Proposed strategy (figure 9.3):** this is the strategy implemented in this project. The diameter is computed with the already explained strategy. **This solution has corrected the contradictory behaviour explained above.** Now, the diameter always increases with the growth of failures.

The latter example is applicable to every path-related metric. This was the behaviour that have these metrics and this is the solution provided with the proposed strategy, which seems to correct some

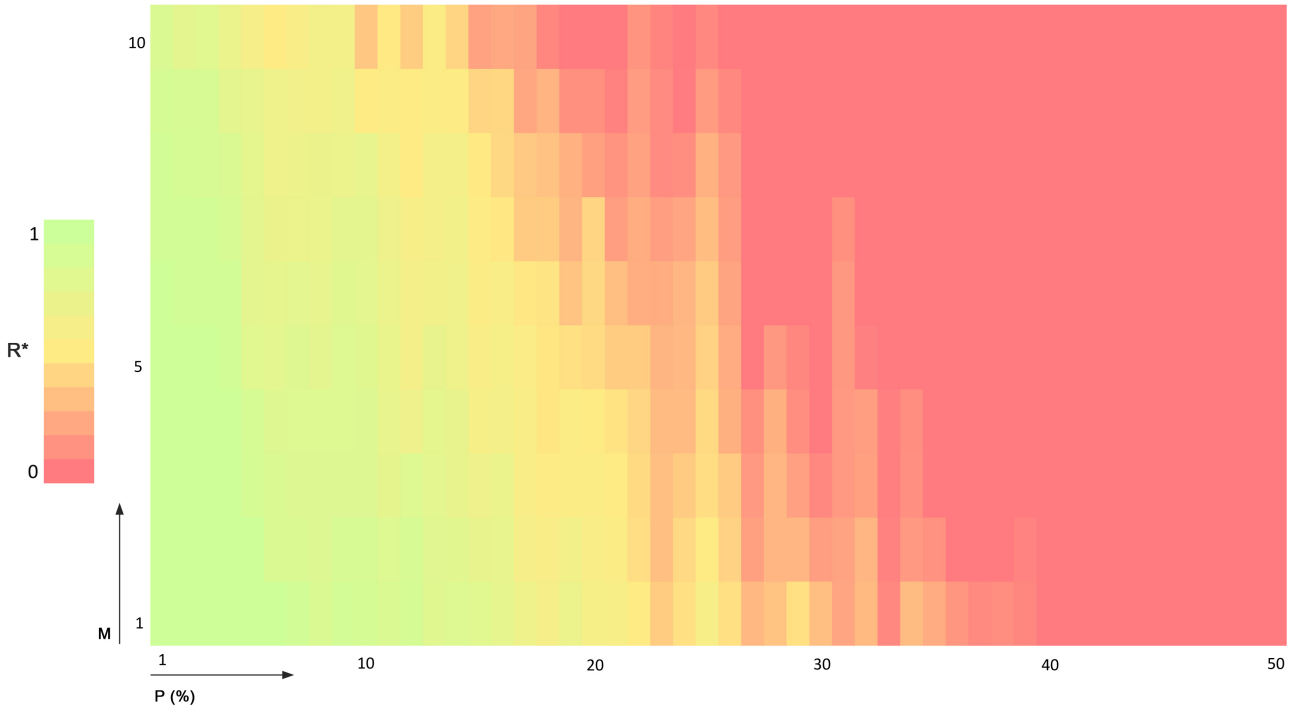


Figure 9.1: Results 1: PCA with normalization and weighted metrics

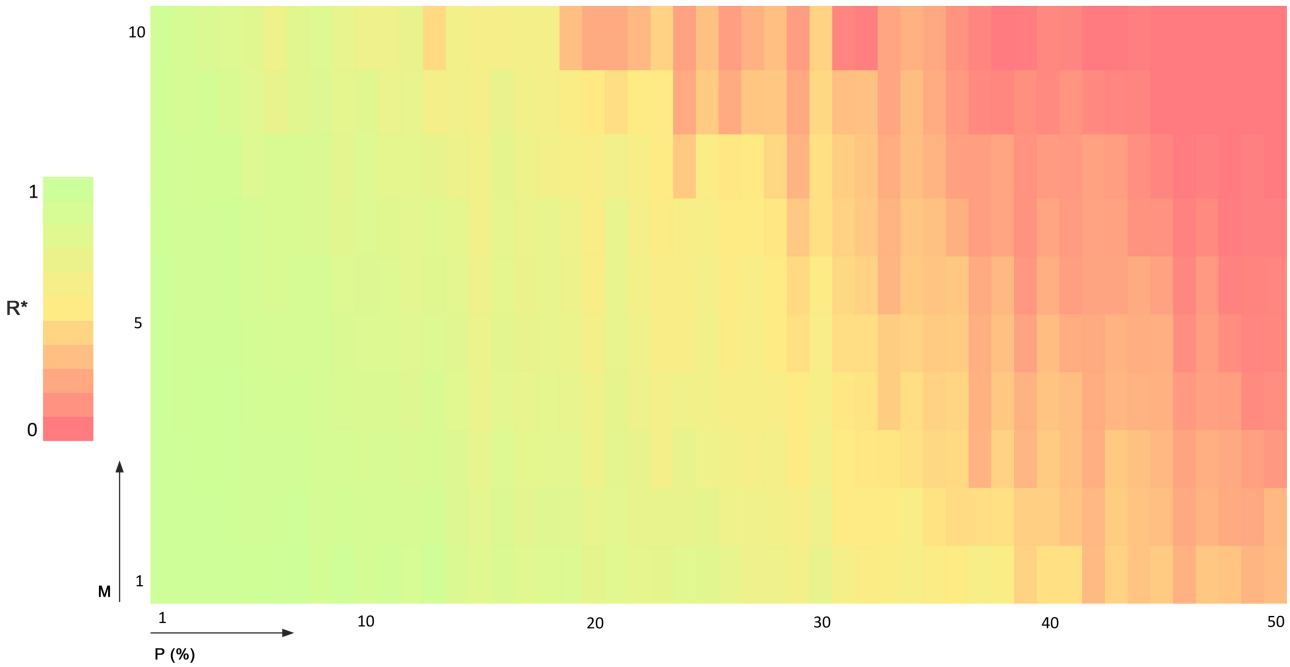


Figure 9.2: Results 1: PCA without normalization but with weighted metrics

contradictory behaviours in terms of robustness.

In order to analyse the implication of these differences, the experiment is evaluated with the latter strategies, see figures 9.3 and 9.4. In both cases, the PCA process is performed with the same strategy, thus, **the only difference relays on the weighted metrics.**
It can be observed that the robustness decreases faster in 9.3, that is because the path-related metrics are penalizing the robustness in a correct way, and as a result, the robustness decreases more. With the classical metrics calculations this is not done properly, as explained.

9.3 Final result comparison

Finally, the differences depicted in latter sections are joined together. Figure 9.5 is the robustness surface obtained with the process implemented in this project (this is, with the normalized PCA and weighted metrics), and figure 9.6 is the robustness surface obtained with the process as it is described in *ALPHA project* (this is, with the non-normalized PCA and classical metrics). It can be easily observed that the changes introduced in the project have a greater penalization on the robustness.
This does not have to be bad, all the introduced changes have been extensively and properly justified, and these are their implications in the final result.

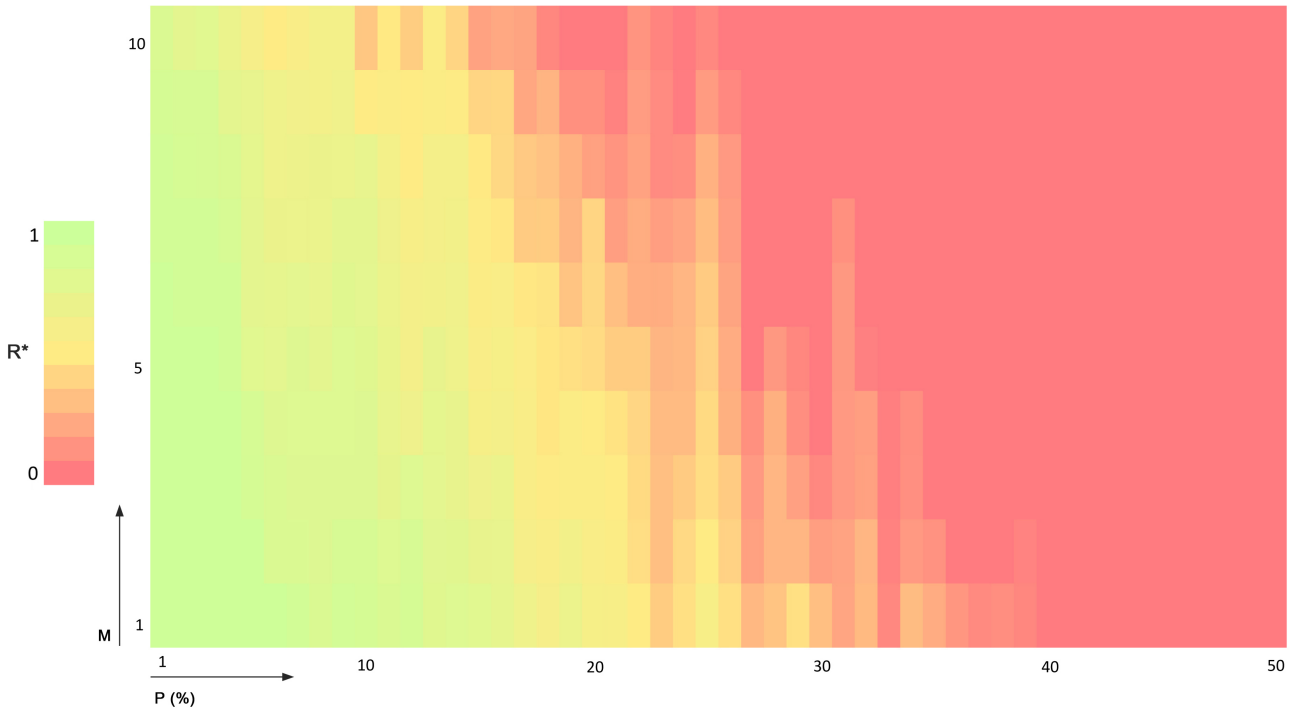


Figure 9.3: Results 2: robustness with weighted metrics

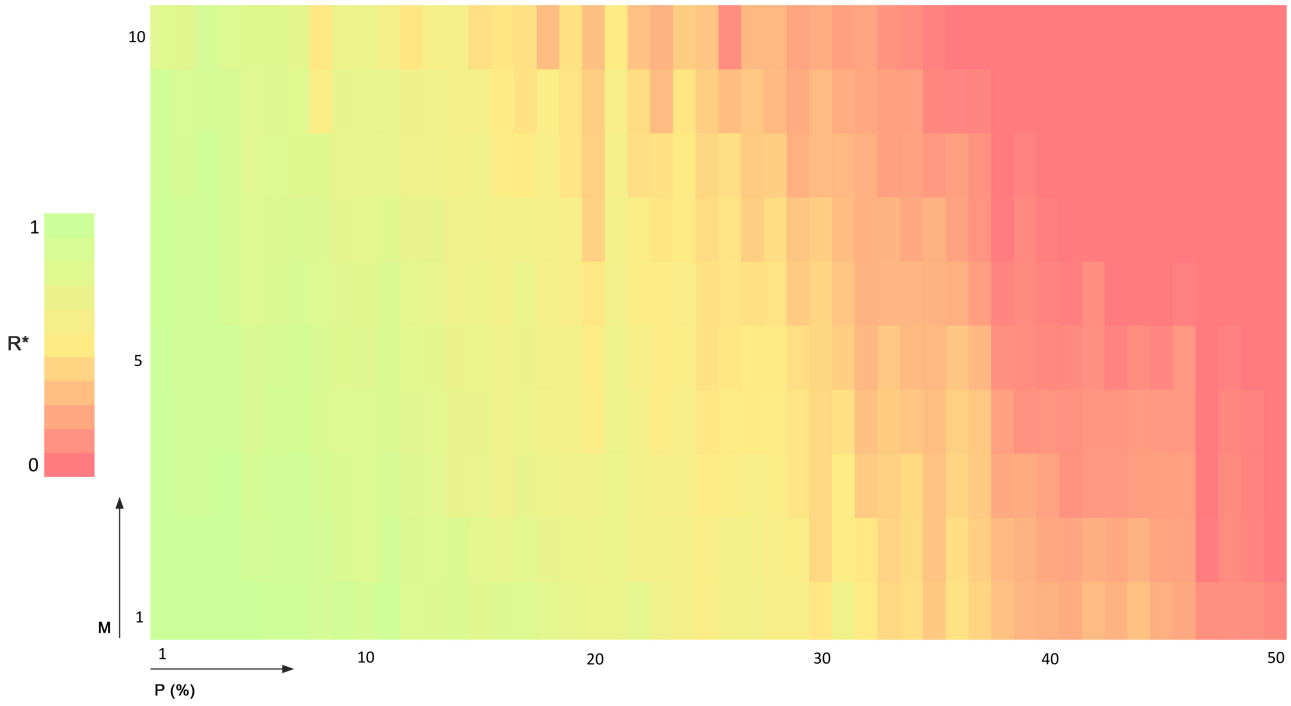


Figure 9.4: Results 2: robustness with classical metrics

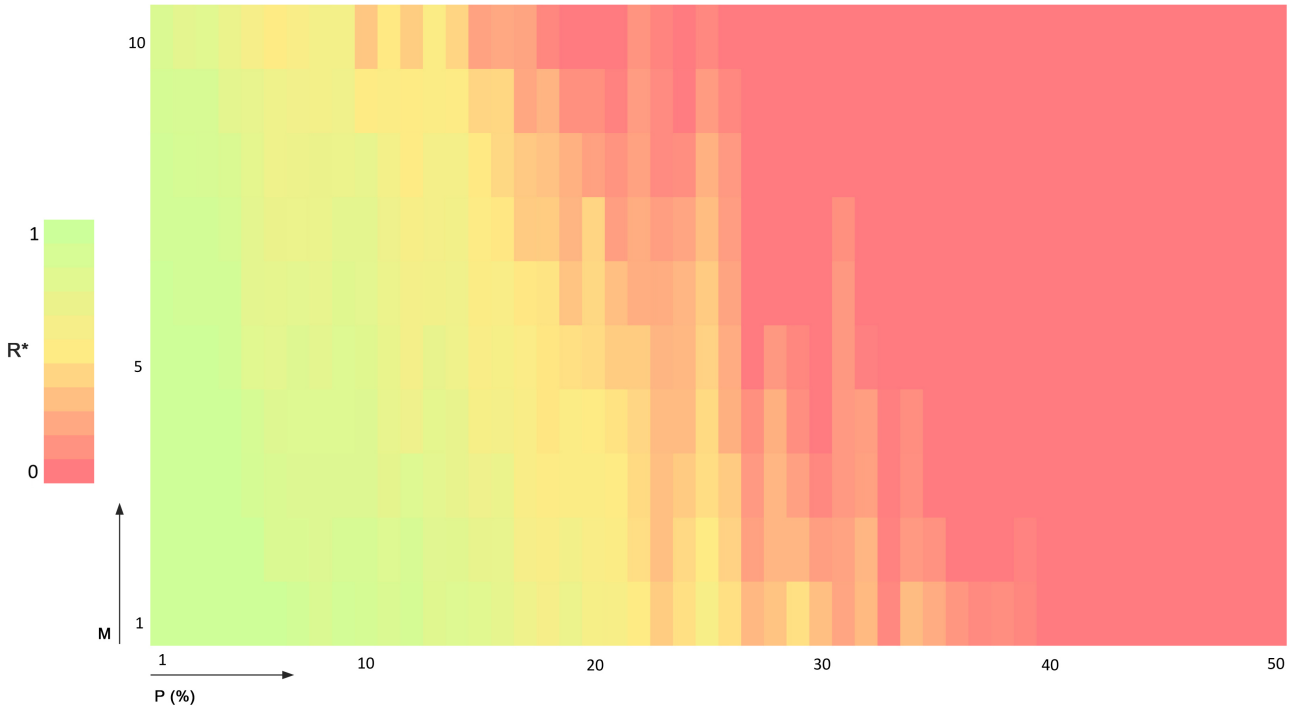


Figure 9.5: Final results: this project result

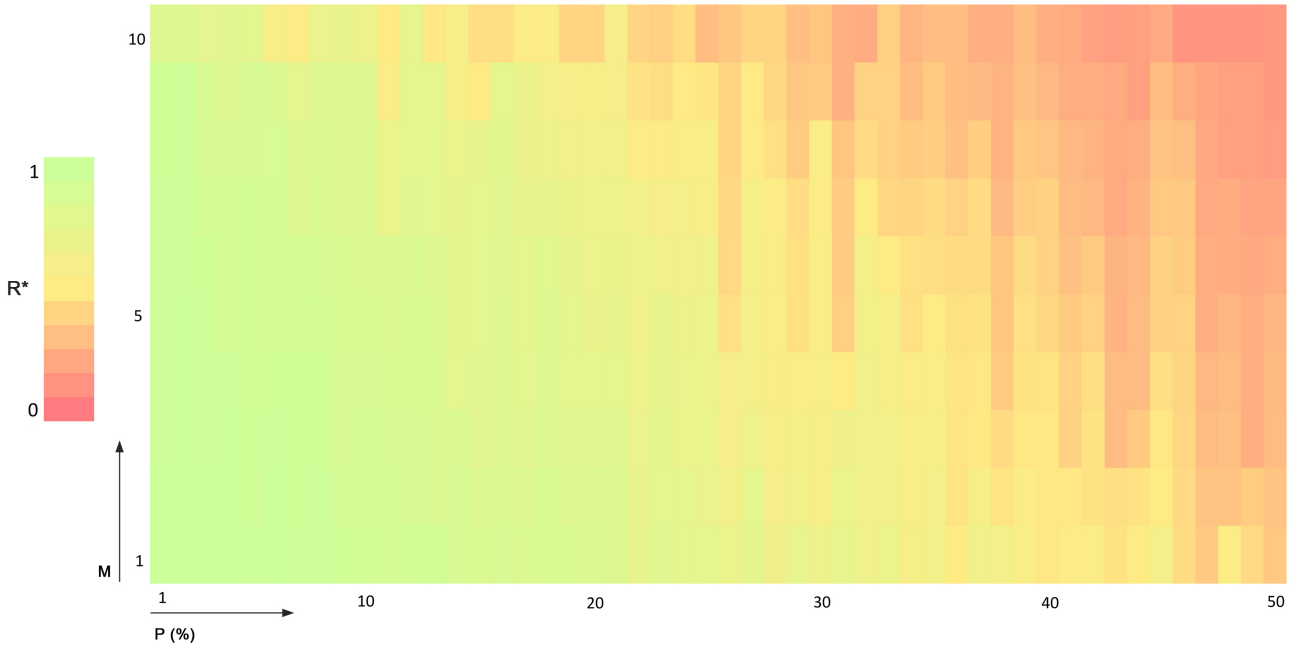


Figure 9.6: Final results: *ALPHA project* result

Chapter 10

Conclusions

Recalling the objectives which were set at the beginning of the project (1.2), these are my general conclusions about their achievement:

- It has been possible to **implement a software tool that emulates *ALPHA project***. The calculation processes to get *R*-value* and *robustness surface* have been fully understood and analysed, and they have been implemented with success too.
- Furthermore, the whole process has been provided with **new functionalities and methodologies**, in order to make it more generic, simple and customizable for the end-user.
- It has to be noted that **an extensive analysis work has been done** over the related concepts in robustness area, especially with all the implied metrics. A deep study of all them, their behaviour and their implication in terms of robustness has been done and all these results can be seen in the project annexes.
- The project has introduced me into a research area, providing **new concepts and knowledge.** Moreover, I have been able to apply some of the concepts which I have learnt throughout these last years, such as functional programming, software engineering patterns, statistical and graph related concepts... Even with the redaction of this document I have learnt new things, because it has been done in LaTeX.

In general, the resulting project has fulfilled the main initial expectancies and, personally, it has a satisfactory result.
However, it also has to be said that some aspects have not been achieved, or at least partially achieved, most of them due to time limitations. Maybe some of them were not main objectives, but it was also a matter of personal ambition to achieve them:

- The project has been focused first, in a theoretical research area, and then with the implementation of the latter. The next step would be to apply all of this on a practical example or a real case, in order to **translate the work to real results and obtain clearer conclusions.**
- Once the results of an experiment are obtained, the project lacks of a work to **take advantage of the potential of this data.** For the moment, the results can be *"manually"* visualized, but a set of methodologies would be appropriate to analyse this data and get conclusions to could enhance the network, for example.

- Some results comparing the performance between this project and *ALPHA project* have been introduced in 9. They show that the introduced solutions in this project have changed the resulting *robustness surfaces* in some way. **A deeper analysis of the differences and the correctness of the results** would be appropriate.

Chapter 11

Future work

Analysing the whole implemented project, some future work options are:

- **Graphic interface:** The scope of this project has included the whole design and implementation of the back-end part of the procedure. The front-end part has not been a priority nor an objective, that is why this project works through a command-line interface.
  In the future, some kind of graphic interface could be implemented to communicate with the functionalities in this project, and also to replace the result presentation which is done by Microsoft Excel now.
  It could be also recommendable to implement a solution to save the input and output data in a database.

- **Implement dynamic attacks:** As it can be seen in classification 4.5, this project has implemented all the attack characteristics except for dynamic attacks, like cascading or epidemic-like attacks.
  Actually, the implemented code has been designed in order to accommodate new attack types like these mentioned, but they are not implemented yet.

- **Implement dynamic metrics:** As it can be seen in 4.1.6, there are other kinds of metrics which measure time-varying features of the topology. One example could be the metrics related to the traffic generated by a distributed service within the topology, they could measure features like the occupation of the elements, the number of collisions... Actually, these features depend on the type of service.
  The implemented code is ready to accommodate these kind of extensions, to implement traffic in the topology and new metrics, for example.

- **Make the process parallelizable:** most of the implemented code related to attacks generation and metrics calculation is highly suited to be parallelized, i.e., the calculation of a given metric on a network is totally independent from all the other metrics.
  These processes are the most computationally expensive in the project, thus, parallelizing them would be a great improvement in order to obtain a valuable tool to deal with large topologies and many metrics as well.

Generally, all the future implementations, especially those related with calculation processes, have to be careful with the efficiency of the resultant code, as it has been done so far.

Bibliography

[1] Package 'igraph': reference manual. `https://cran.r-project.org/web/packages/igraph/igraph.pdf`, June 26, 2015.

[2] Marc Manzano Castro. Metrics to evaluate network robustness in telecommunication networks. Technical report, University of Strathclyde, May 25, 2011.

[3] Bryn Mawr College. Network centrality. `http://cs.brynmawr.edu/Courses/cs380/spring2013/section02/slides/05_Centrality.pdf`, 2013.

[4] Paolo Crucitti, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. Efficiency of scale-free networks: Error and attack tolerance. *Physica A 320*, 2003.

[5] Anthony H. Dekker and Bernard Colbert. The symmetry ratio of a network. *Australasian Symposium on Theory of Computing*, 2005.

[6] W. Ellens, F.M. Spieksma, P. Van Mieghemc, A. Jamakovic, and R.E. Kooij. Effective graph resistance. *Linear Algebra and its Applications*, 2011.

[7] Wendy Ellens. Effective resistance and other graph measures for network robustness. Technical report, University of Leiden, April 29, 2011.

[8] Arpita Ghosh, Stephen Boyd, and Amin Saberi. Minimizing effective resistance of a graph. *International Symposium on Mathematical Theory of Networks and Systems*, 2006.

[9] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Xenofontas Dimitropoulos, and Amin Vahdat. The internet as-level topology: Three data sources and one definitive metric. *SIGCOMM Comput. Commun. Rev.*, 2006.

[10] Marc Manzano, Faryad Sahneh, Caterina Scoglio, Eusebi Calle, and Jose Luis Marzo. Robustness surfaces of complex networks. *Scientific Reports, Nature*, 2014.

[11] Josep Lluís Marzo. Network robustness under multiple failures: relevant metrics analysis and robustness surfaces. (not published), CITS 2015.

[12] James Sterbenz, Justin Rohrer, Egemen Cetinkaya, Mohammed Alenazi, Austin Cosner, and James Rolfe. Ku-topview network topology tool. `http://www.ittc.ku.edu/resilinets/maps/`, 2010.

[13] S. Trajanovski, J. Martín-Hernández, W. Winterbach, and P. Van Mieghem. Robustness envelopes of networks. *Complex Networks*, 2013.

[14] Madeleine Udell. Introduction to spectral graph theory, March 14, 2011.

[15] W. N. Venables, D. M. Smith, and the R Core Team. An introduction to r. `https://cran.r-project.org/doc/manuals/R-intro.pdf`, 2015.

[16] Jun Wu, Mauricio Barahona, Yuejin Tan, and Hongzhong Deng. Robustness of regular graphs based on natural connectivity, 2009.

Appendix A

Metrics dictionary

This appendix is intended to be a summary of all the implied metrics, a sort of metrics dictionary. All the metrics involved in this project are presented and explained here in detail. No metric is introduced along the rest of the document so, refer to this section whenever is needed.

Every metric definition is intended to have the same structure, which is:

- **Complete name**
- **Acronym** to refer to the metric more easily
- Informal text **definition**
- Formal definition with a **formula** whenever is possible, in some cases there is no formula or it is not needed
- **Range** of the possible values of the metric
- **Robustness meaning**, i.e., which results indicate better robustness when applied on a topology
- **References** to other consulted documents or scientific papers which make use of or mention this metric in terms of robustness
- **Practical example** to show one or two results of the metric applied to simple topologies

Apart from these definitions, the equivalent R code of all the metrics can be consulted in the attached source files of the project.

A.1 Basic structural metrics

A.1.1 Average Nodal Degree (AND)

The degree of a node is the number of incident edges (in an undirected graph). Thus, this metric is the average of all node degrees.

$$\bar{\delta} = \frac{1}{|V|} \sum_{i=1}^{|V|} \delta_i$$

or alternatively:

$$\bar{\delta} = \frac{2 \times |E|}{|V|}$$

Range: $\bar{\delta} \in [0..|V|]$

More robust when: larger. Topologies with higher values are more connected on average and, consequently, are likely to be more robust.

References: [2], [9]

Example: see figure A.1

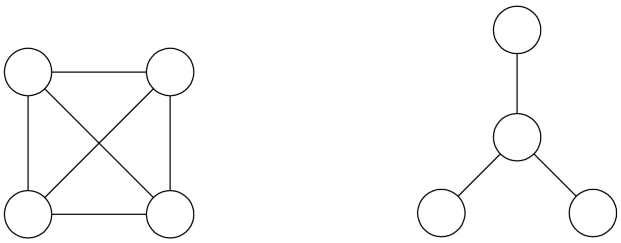


Figure A.1: AND example

A.1.2 Heterogeneity (HET)

It is the standard deviation of the nodal degrees divided by the average nodal degree.

Range: $HET \in (0..\infty]$

More robust when: smaller. More homogeneous topologies, i.e., topologies with less differences in their node degrees, are likely to be more robust because every node is more evenly connected. Thus, there is no node whose failure affects the topology more significantly.

References: [2]

Example: see figure A.2

Figure A.2: HET example

A.1.3 Average Shortest Path Length (ASPL)

The shortest path between two nodes is the path within the graph which minimizes the number of its edges. The length of this path is indeed the number of edges, which is called the distance. The average shortest path length is the average of the distances between every node pair of the graph.

$$\bar{d} = \frac{2}{|V|(|V|-1)} \sum_{i=1}^{|V|} \sum_{j=i+1}^{|V|} d_{ij}$$

where d_{ij} is the distance between i and j nodes.

Range: $\bar{d} \in [1..|V|]$

More robust when: smaller. This is slightly related with AND (A.1.1), topologies with small values are likely to be more closely connected, which could mean that have greater AND as well.
References: [2], [9], [7]
Example: see figure A.3

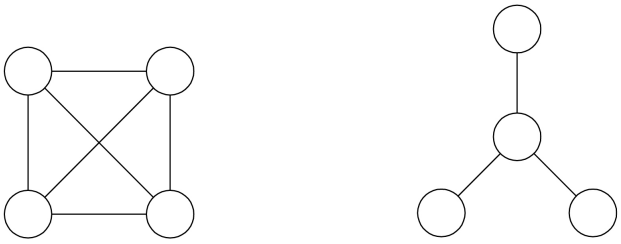


Figure A.3: ASPL example

A.1.4 Diameter (DIA)

The diameter of a graph is the longest path between all the shortest paths.
Range: $d_{max} \in [1..|V|]$
More robust when: smaller, for the same reason that ASPL (A.1.3).
References: [2], [7]
Example: see figure A.4

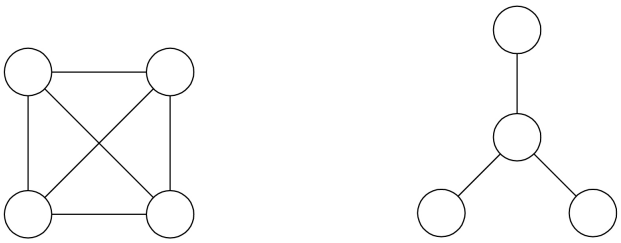
Figure A.4: DIA example

A.1.5 Efficiency (EFF)

It is the average of the inverse of all the distances between all node pairs.

$$EFF = \frac{2}{|V|(|V|-1)} \sum_{i=1}^{|V|} \sum_{j=i+1}^{|V|} \frac{1}{d_{ij}}$$

Range: $EFF \in [0..1]$
More robust when: larger. As it can be seen in the formula, EFF is the inverse of ASPL (A.1.3), so the robustness meaning is the opposite.
References: [4]
Example: see figure A.5



$EFF = 1$ $EFF = 0.75$

Figure A.5: EFF example

A.1.6 Effective Resistance (ER)

Seeing the graph as an electrical circuit, you can see an edge as a resistor. The edge weight would be equivalent to its resistance. The effective resistance between two nodes can be calculated by series and parallel manipulations, see figure A.6:

- Two resistors with resistances r_1, r_2 in series can be replaced by one resistor with resistance $r_1 + r_2$
- Two resistors in parallel can be replaced by one resistor with resistance $(r_1^{-1} + r_2^{-1})^{-1}$

Resistors in series Resistors in parallel

Figure A.6: Effective resistance between a and b

The whole effective graph resistance is the sum of the effective resistances over all node pairs. It has been proved that it can be written as a function of the non-zero Laplacian eigenvalues:

$$ER = |V| \sum_{i=2}^{|V|} \frac{1}{\lambda_i}$$

where λ_i are the Laplacian matrix eigenvalues.
ER is considered a robustness metric mainly for these reasons [6, pg. 9]:

- First, ER is the sum of pairwise effective resistances. Every pairwise resistance, with the series and parallel manipulations shown above, takes both the number of paths between the two nodes and their length into account. Therefore, it is capturing the number of different paths and their quality.
- Second, ER can be approximated by AC (A.3.3), and AC is a robustness metric indeed.

Range: $\frac{|V|}{\lambda_2} < ER \leq \frac{|V|(|V|-1)}{\lambda_2}$ [6, pg. 4], where λ_2 is AC (A.3.3).
More robust when: smaller. ER is proved to decreases when edges are added on a graph [6, pg. 9], resulting in a more robust topology.
References: [7], [6], [8]
Example: see figure A.7

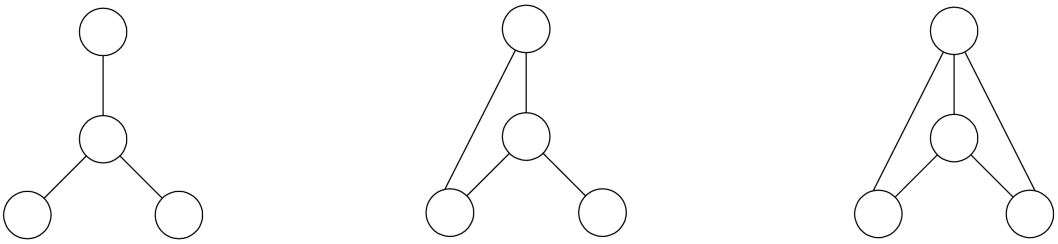


Figure A.7: ER example

A.1.7 Number of Spanning Trees (NST)

A spanning tree is a subgraph containing $|V| - 1$ edges, all $|V|$ nodes and no cycles. The total number of spanning trees is the set of all different spanning trees. In big graphs, this number could become very large.

The number of spanning trees is calculated via Laplacian matrix:

$$NST = \frac{1}{|V|} \prod_{i=2}^{|V|} \lambda_i$$

where λ_i are the Laplacian matrix eigenvalues.

Range: $NST \in [1..|V|!]$

More robust when: larger. The greater the number of spanning trees, the more possibilities to get all the topology connected in different ways.

References: [7]

Example: see figure A.8

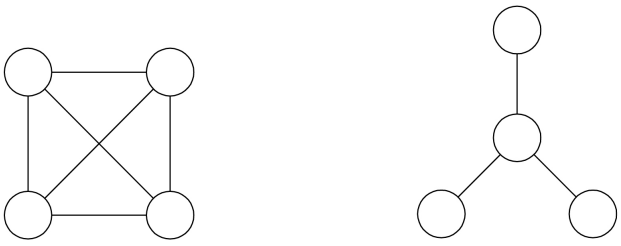


Figure A.8: NST example

A.1.8 Clustering Coefficient (CC)

A triplet consists of three nodes that are connected by either two (open triplet) or three (closed triplet) undirected edges.

Clustering coefficient captures the presence of triangles, which compares the number of closed triplets over the total number of triplets (both closed or opened). It measures the probability that the adjacent nodes of a node are connected between them, see figure A.9.

The local clustering coefficient of a node i is defined as the number of edges among neighbours of i divided by $\delta_i(\delta_i - 1)/2$, the total possible number of edges among its neighbours, see figure A.10.

The whole clustering coefficient of a graph is the average over the clustering coefficients of the nodes:

$$CC = \frac{1}{|V|} \sum_{i=1}^{|V|} c_i$$

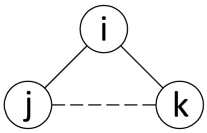
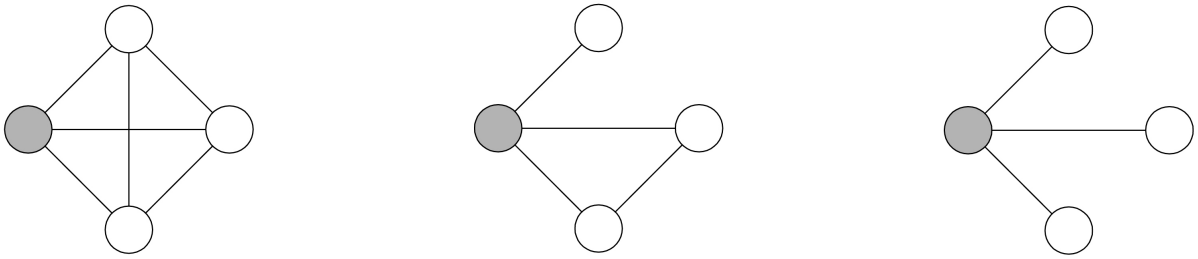


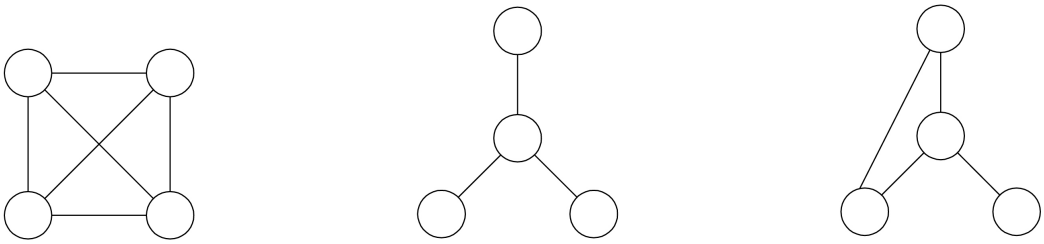
Figure A.9: Nodes j and k sharing a neighbour i may or may not be neighbours themselves



$c_i = 1$ $c_i = 1/3$ $c_i = 0$

Figure A.10: Local clustering coefficients of the shadowed node

where c_i is the local clustering coefficient of node i, defined as above.
Sometimes clustering coefficient is also called transitivity.
Range: $CC \in [0..1]$
More robust when: higher. High values indicates topologies with the presence of many triangles, which help to increase the growth of possible paths.
References:[7], [9]
Example: see figure A.11



$CC = 1$ $CC = 0$ $CC = 0.58$

Figure A.11: CC example

A.1.9 Assortativity (r)

It is the preference of nodes to attach to others which have a similar degree. It is similar to a Pearson correlation measure. Negative results mean that low degree nodes often connect to high degree nodes (dissassortative topology). Oppositely, positive results mean that nodes of equal or similar degree are often connected (assortative topology).
Range: $-1 \leq r \leq 1$
More robust when: larger. Dissassortative topologies are more hierarchical and are proved to be more vulnerable ([2, pg. 29]), rather than assortative topologies, which are more regular.
References: [2], [9]
Example: see figure A.12

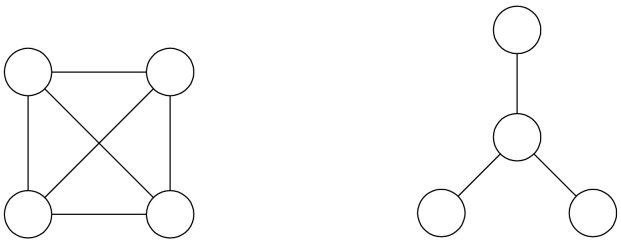


Figure A.12: r example

A.1.10 Symmetry Ratio (SR)

It is the quotient between the number of distinct eigenvalues of the graph (obtained from the adjacency matrix) and the diameter.

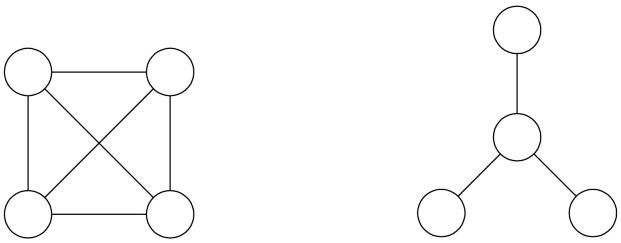
$$SR = \frac{\epsilon}{d_{max}+1}$$

where ϵ is the number of distinct eigenvalues of the adjacency matrix and d_{max} is the diameter.
Range: $SR \in (0..\epsilon]$
More robust when: larger, as DIA is used dividing in the formula.
References: [2], [5]
Example: see figure A.13

Figure A.13: SR example

A.1.11 Largest Eigenvalue (LE)

Graph spectrum is one of the most important global characteristics of a topology and the largest eigenvalues are particularly important.
Range: $\lambda \in [0..2]$
More robust when: larger. Most topologies with high values for this largest eigenvalue have small diameter and are more robust. In general, topologies with larger eigenvalues have more node and link disjoint paths to choose from [2, pg. 29].
References: [2], [9]
Example: see figure A.14



$LE = 3$ $LE = 1.7$

Figure A.14: LE example

A.2 Fragmentation metrics

Fragmentation explains how much is the graph disconnected or how many components does it have. Every metric explains these concepts in its own way.
All these metrics give a fixed value while the graph is connected, which is the situation where the topology is more robust. That is why the robustness meaning section of every definition has been changed here for a fragmentation meaning.
Alternatively, all the metrics begin to give information of the fragmentation at the moment when the graph becomes disconnected.

A.2.1 Largest Connected Component (LCC)

Number of nodes of the largest connected component of the graph.
Range: $LCC \in [0..|V|]$
No fragmented when: |V|
Example: see figure A.15

$LCC = 4$ $LCC = 3$

Figure A.15: LCC example

A.2.2 Fractional Size Largest Component (FSLC)

Fraction of nodes of the largest connected component of the graph related to the total number of nodes. It actually measures the same feature as LCC (A.2.1), only that this is in percentage.
Range: $FSLC \in [0..1]$
No fragmented when: 1
Example: see figure A.16

$FSLC = 1$ $FSLC = 0.75$

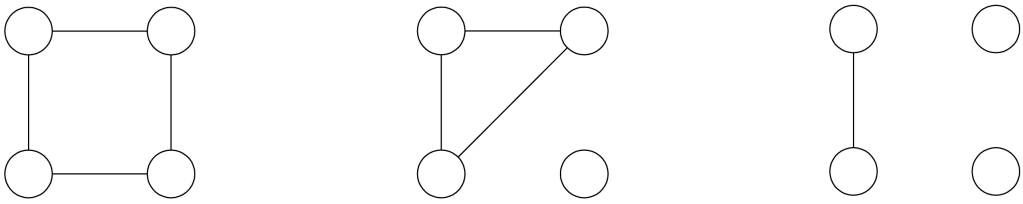
Figure A.16: FSLC example

A.2.3 Average Two Terminal Reliability (ATTR)

It is the probability that a randomly chosen pair of nodes remains connected after a failure. It gives the ratio between the node pairs within the different connected components and the total number of node pairs:

$$ATTR = \frac{\sum_{i=1}^{x}(|V|_x \times (|V|_x - 1))/2}{(|V| \times (|V| - 1))/2}$$

where x is the number of connected components of the graph and $|V|_x$ is the number of nodes of each component.
Range: $ATTR \in [0..1]$
No fragmented when: 1
References: [2]
Example: see figure A.17



$ATTR = 1$ $ATTR = 0.5$ $ATTR = 0.17$

Figure A.17: ATTR example

A.2.4 Degree of Fragmentation (DF)

Number of connected components.
Range: $DF \in [1..|V|]$
No fragmented when: 1
Example: see figure A.18

A.3 Connectivity metrics

Connectivity explains how difficult is to disconnect the graph or how far is this from happening. Every metric explains these concepts in its own way.
All these metrics give information of the connectivity while the graph is connected, which is the situation where the topology is more robust. That is why the robustness meaning section of every

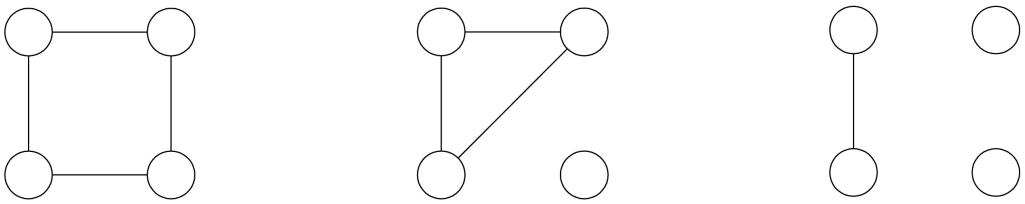


Figure A.18: DF example

definition has been changed here for a connectivity meaning.
Alternatively, all the metrics give zero when the graph becomes disconnected.

A.3.1 Edge Connectivity (EC)

Minimal number of edges which have to be removed to disconnect the graph.
Range: $EC \in [0..|E|]$
More connected when: larger.
References: [2], [7]
Example: see figure A.19

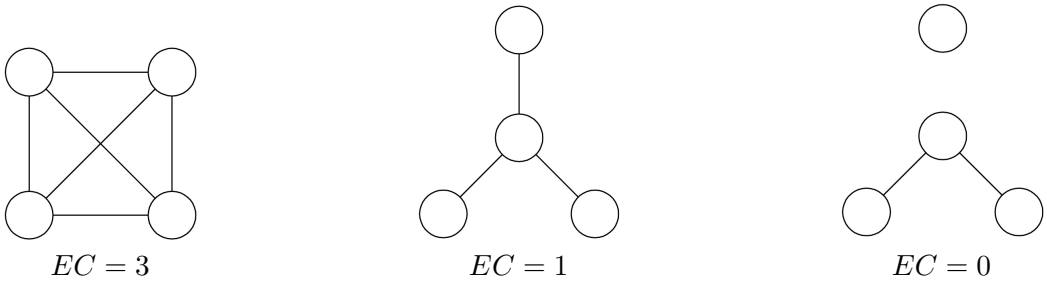


Figure A.19: EC example

A.3.2 Vertex Connectivity (VC)

Minimal number of nodes which have to be removed to disconnect the graph.
Range: $VC \in [0..|V|]$
More connected when: larger.
References: [2], [7]
Example: see figure A.20

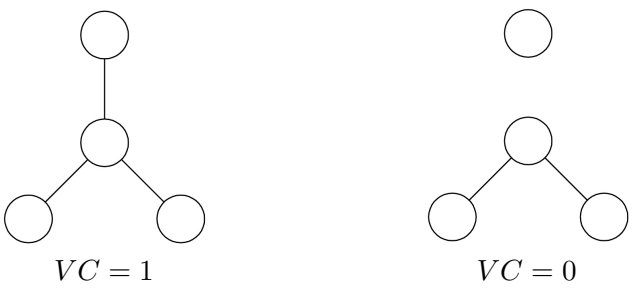


Figure A.20: VC example

A.3.3 Algebraic Connectivity (AC)

Algebraic connectivity is defined as the second smallest Laplacian eigenvalue. It has been proved that its value is zero if and only if the graph is disconnected, and that it cannot be greater than the vertex connectivity [7, pg. 16]. That is why this metric is used as a connectivity measure.
Range: $\lambda_2 \leq VC \leq EC \leq \delta_{min}$ [7, pg. 17]
where δ_{min} is the minimum nodal degree.
More connected when: larger.
References: [2], [7]
Example: see figure A.21

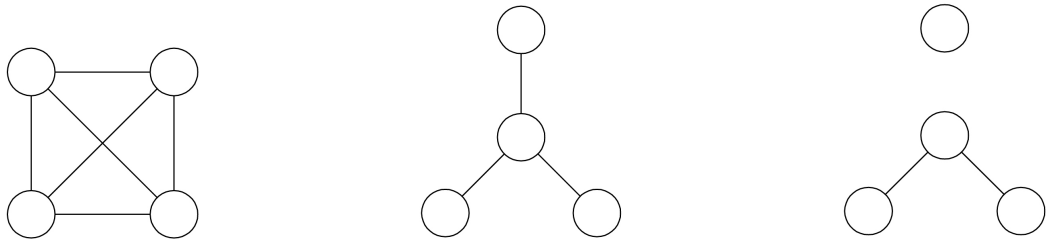


Figure A.21: AC example

A.3.4 Natural Connectivity (NC)

A walk of length k is a path over the nodes and links of the graph, starting at v_0, passing $k-1$ nodes and ending at v_k. If $v_0 = v_k$ this path is called a closed walk.
The number of closed walks is an important index for complex networks and recently, it has been proposed that the number of closed walks of all lengths quantify the redundancy of alternative paths in the graph and can therefore serve as a measure of network robustness [16, pg. 5].
The number of closed walks can be related to the sum of eigenvalues, and this is how it is calculated:

$$\bar{\lambda} = ln\left(\sum_{i=1}^{|V|} e^{\lambda_i}/|V|\right)$$

where λ_i are the adjacency matrix eigenvalues.
Range: $\lambda_{|V|} \leq \bar{\lambda} \leq \lambda_1$ [16, pg. 6]
More connected when: larger.
References: [16]
Example: see figure A.22

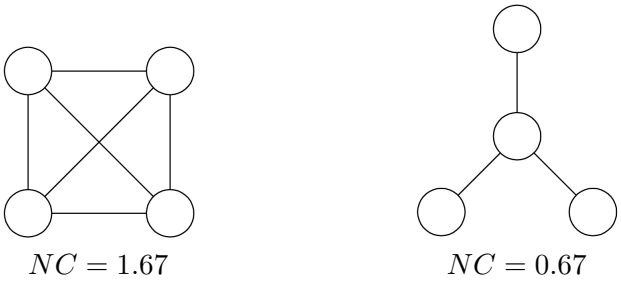


Figure A.22: NC example

A.4 Centrality metrics

Centrality measures are slightly different to the other presented, because **they are usually used to be applied to each graph component** (nodes or edges). Centrality explains how much important is a given component within the graph structure, and every metric explains this concept but in its own way.

That is why these metrics are used to sort graph components by importance, for example, to be used to **choose attacked elements in targeted attacks**.

Every centrality measure has a normalized (whose range is within [0..1]) and a non-normalized version. To obtain a centrality measure for the whole graph, all the centrality measures of its elements can be averaged and normalized. The result is called *centralization* and is denoted by:

$$\bar{C} = \frac{\sum_{i=1}^{x} C_{i_{max}} - C_i}{(x-1)(x-2)}$$

where C_i is a certain centrality measure, $C_{i_{max}}$ is the maximum value of this measure in the graph and x is the number of elements. This process can be applied to all the centrality measures and the resulting **centralization value will be used as a robustness metric in this project**.

However, the fact is that any reference document clearly relating centrality with robustness has not been found. Therefore, the robustness meaning of these metrics has been proven from the experiment results of this project.

A.4.1 Degree Centrality (DC)

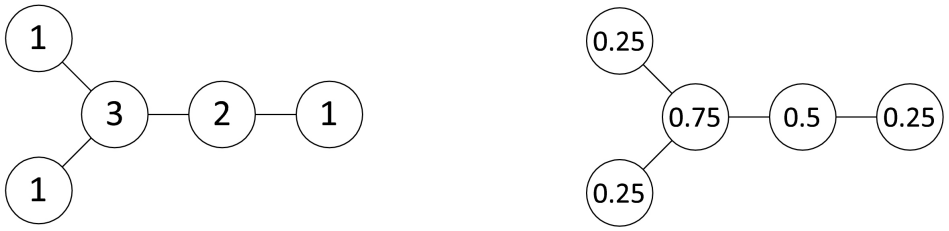
The centrality measure of each node is given by its nodal degree. It is similar to the concept *"I am important because I know a lot of people"*. It is a local centrality measure, it only considers the characteristics of each node individually, in front of the other centrality measures, which take into account the characteristics of the rest of the topology.

The normalized measures are obtained dividing by the maximum possible value, $|V| - 1$.

More robust when: larger.

References: [3]

Example: see figure A.23



Non-normalized measures Normalized measures

$Centralization = 0.35$

Figure A.23: DC example

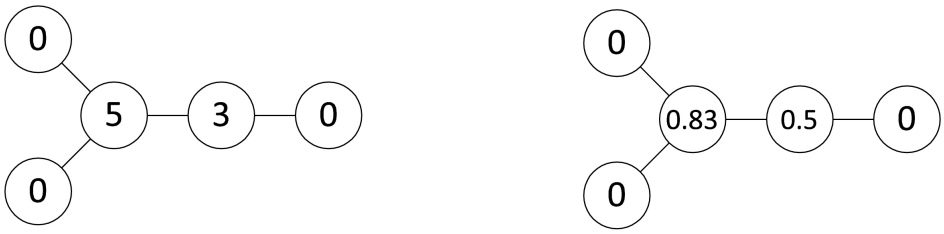
A.4.2 Node Betweenness Centrality (NBC)

The centrality measure of each node is given by the number of shortest paths which pass through it. It is similar to the concept *"I am important because I am in the connection between a lot of people"*.

Sometimes, this is called simply betweenness.
The centrality measure for node x is defined as:

$$NBC_x = \sum_{i=1}^{|V|} \sum_{j=i+1}^{|V|} \frac{\sigma_{ij}(x)}{\sigma_{ij}}$$

where σ_{ij} is the total number of shortest paths between i and j nodes and $\sigma_{ij}(x)$ are the shortest paths which pass through node x. If there exist more than one shortest path between two nodes, then each of these k paths is counted $1/k$ times.
The normalized measures are obtained dividing by the maximum possible value, $(|V|-1)\times(|V|-2)/2$, which is the maximum number of shortest paths excluding those which involve the given node.
More robust when: larger.
References: [3], [7]
Example: see figure A.24



Non-normalized measures Normalized measures

$Centralization = 0.71$

Figure A.24: NBC example

A.4.3 Edge Betweenness Centrality (EBC)

It is the same concept as NBC (A.4.2), but applied to edges.
In this case, the normalized measures are obtained dividing by the maximum possible value, which is $|V| \times (|V|-1)/2$, which is the maximum number of shortest paths.
More robust when: larger.
References: [3], [7]
Example: see figure A.25

Non-normalized measures Normalized measures

$Centralization = 0.67$

Figure A.25: EBC example

A.4.4 Closeness Centrality (CLC)

Closeness is based on the length of the average shortest path (the distance) (A.1.3) between a node and all the other nodes in the graph. Actually, the concept closeness is the opposite of the distance, which measures the farness. It is similar to the concept *"I am important because I am not far from the rest"*.

The centrality measure for node i is defined as:

$$CLC_i = \frac{1}{\sum_{j=1}^{|V|} d_{ij}}$$

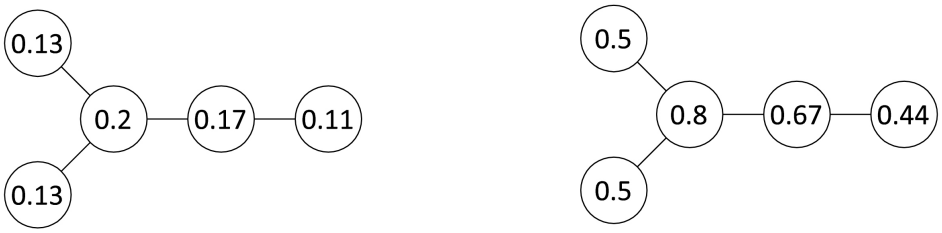
where d_{ij} is the distance between i and j nodes.

The normalized measures are obtained dividing by $|V| - 1$.

More robust when: larger.

References: [3]

Example: see figure A.26



Non-normalized measures Normalized measures

$Centralization = 0.64$

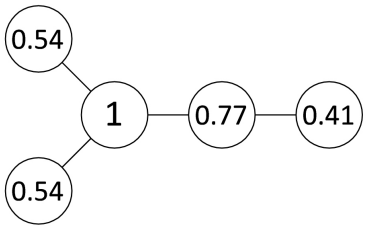
Figure A.26: CLC example

A.4.5 Eigenvector Centrality (EVC)

The centrality measure of each node is given by the corresponding value of a vector component, where the vector is the largest eigenvalue of the adjacency matrix. It measures the way in which important nodes are connected to important nodes. It is similar to the concept *"I am important because I know important people"*.

More robust when: larger.

Example: see figure A.27



$Centralization = 0.58$

Figure A.27: EVC example

Appendix B

Comparisons and paradoxes between metrics

This appendix gathers up a set of comparisons between pairs of metrics. The objective is to analyse pairs of similar metrics and find whether there is a special relation between them or not.
Every comparison is intended to have the same structure, which is:

- **Concept comparison:** what features do the two metrics represent? Do they have some kind of relationship? Do their results have correlation with the increase of failures?
- **Computation cost comparison:** is there a metric which is easier to compute than the other, in terms of time of computation?
- **Differences:** is there any difference which could make one of the metrics *"better"* for some reason?

Apart from these comparisons, there are also presented some paradoxes between metrics. A paradox is a special situation which shows that some metrics that are measuring the same characteristic are giving contradictory results. These paradoxes are only presented here as a set of observed cases with academic purposes only.

B.1 Path-related metrics

Comparison DIA & ASPL

- **Concepts:** DIA and ASPL are two metrics with clear correlation. It is completely logical, given that DIA is a special case of a shortest path.
- **Computation cost:** The process to obtain them is practically the same, because all the shortest paths have to be computed. Thus, there is not a reason about computation cost to choose one of them.
- **Differences:** ASPL has a more amortized behaviour against changes, because it is a mean. It could be more suitable than DIA, whose behaviour could be more abrupt.

Comparison ASPL & EFF

- **Concepts:** EFF represents the inverse of ASPL (see the formulas in A.1.3 and A.1.5), actually they have inverse correlation.
- **Computation cost:** In both cases, the process to obtain them is nearly the same, because all the distances have to be computed.
- **Differences:** EFF is normalized within the range [0..1]. This could sometimes be an advantage, because it is easier to ponder a value within this range.

B.2 Nodal-degree-related metrics

Comparison AND & HET

- **Concepts:** They have inverse correlation, actually AND is used in the calculation of HET.
- **Computation cost:** The process to obtain them is practically the same, because all the node degrees have to be computed.
- **Differences:** HET may give more information, because it captures the degree variance apart of the degree mean. For example, in some cases, it could help to distinguish graphs with the same AND but with different structures indeed. See example in figure B.1, any tree with the same number of nodes has the same AND, but a different structure.

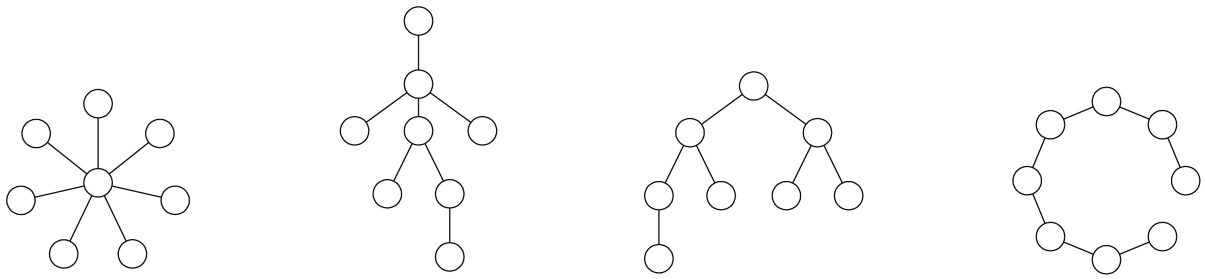


Figure B.1: Comparison between AND & HET

B.3 Fragmentation metrics

Comparison FSLC & LCC

- **Concepts:** They express exactly the same idea (see A.2.1 and A.2.2), so there is total correlation.
- **Computation cost:** The process to obtain them is practically the same.

- **Differences:** FSLC is within the range [0..1], which would be more suited for a fragmentation metric. Moreover, LCC could have a very large value compared to the ranges of the other metrics.

Comparison ATTR & FSLC

- **Concepts:** FSLC only captures information about the largest component, nothing about the others. However, ATTR gives information about the number of connected components and all their sizes as well.

- **Computation cost:** Both metrics have a similar process, all the connected components and their sizes have to be found.

- **Differences:** ATTR gives a more complete idea about the graph fragmentation (similar to what has been explained in B.2), see example in figure B.2.

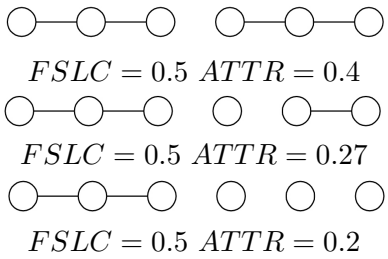


Figure B.2: Comparison between ATTR & FSLC

B.4 Connectivity metrics

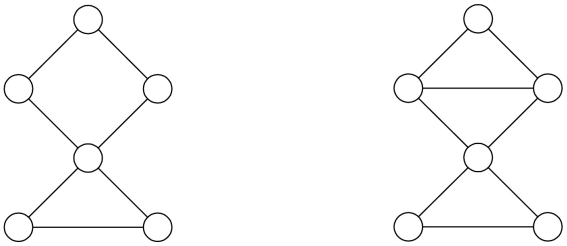
Paradox between EC & AC

In some special cases, two connectivity metrics could lead to a contradiction. It is, given a pair of graphs, one of the metrics could say that one graph is more robust (or is more strongly connected), while the other metric could say the opposite.
All connectivity metrics express the same idea but, in certain cases, they could lead to this contradictions, especially if their values are picked alone.

Figure B.3: Paradox between connectivity metrics

Paradox between EC, AC & NC

In this case, given two graphs, EC and AC gives the same connectivity result whereas NC do not (see figure B.4). Thus, NC helps to distinguish different situations of graph connectivity in these cases, it seems to capture information that the other two metrics could not.



$EC = 2$
$AC = 0.76$
$NC = 1.09$

$EC = 2$
$AC = 0.76$
$NC = 1.35$

Figure B.4: Another paradox between connectivity metrics

B.5 Centrality metrics

Paradox between DC & NBC

As seen above (B.4), with a given graph, different centrality measures could give a different importance to the same element, because every measure express it in a different way.

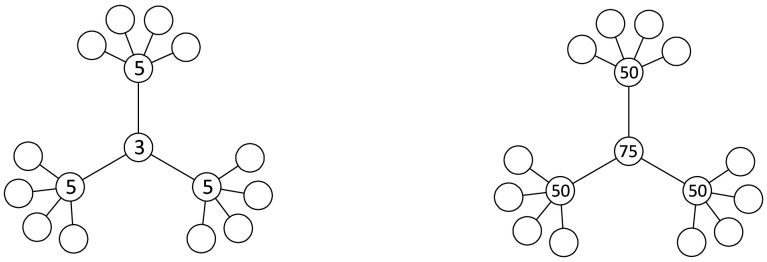


Figure B.5: Paradox between centrality metrics