Protein Interaction Prediction

Project Memory

October, 2016

Author: Bernat Huguet Planes

Director: Ulises Cortés García

Co-director: Dario Garcia Gasulla

GEP Tutor: Eguiguren Huerta Marcos

Contents

1 Introduction

Current computer technology enables us to store tremendously large amounts of information at little cost, and due to that our society is experiencing the greatest information explosion ever in history. Moreover, it is expected that this tendency will continue at an exponential rate in the following decades. This situation originates one of the hardest challenges of this century: Transforming data into knowledge in a scalable way.

Amongst the fields that forcefully need to analyze huge amounts of data and the one that can make the greatest impact in human life quality and longevity, there's the field of biomedicine. Biological systems have always been known for their inherent complexity and the technology development currently enables us to build far more complex models that can be used to simulate (In order to research the system behaviour) and/or assist the scientists in their research.

For all these reasons, it is highly desirable to build software to help us speed up the research process and to empower us to examine phenomenons that can only be analyzed with huge amounts of computation and data.

The goal of this project is to build a computer program to assist protein interaction research for the Institute for Research in Bio-medicine of Barcelona. The main functionality of this computer program will be to, given the starting and ending protein of an interaction, predict the paths (Series of proteins) the interaction is most likely to follow. The usage of this feature will help reduce the number of laboratory experiments substantially. Moreover, each new experiment could be added to the system data to further improve the prediction accuracy of the software, guaranteeing a long service life.

2 Scope of the Project

The goal of this section is to make a detailed summary of this project. First we will start defining the problem that we are trying to address. After, we will define the scope of the project and analyze the possible obstacles that we may face during the project development. Finally, we are going to describe the development methodology.

2.1 Objectives

The goal of this project is to develop a research assistance software for the Institute for Research in Bio-medicine of Barcelona.

The problem we are going to help solving is the research in protein interactions, an investigation with the goal to better understand how biology works at a molecular level. Knowledge is power, and the better we understand our bodies the better we can take care of them.

Traditionally, experiments were designed and tested in the laboratory by biological scientists, but that approach is in need of a shift because of the everly increasing problem complexity this field is trying to tackle. The combinatorial explosion is far too high for humans to test in the system itself and, for this reason, model building has increasingly gained popularity in the current bio-medicine investigations.

This is the kind of problem this project aspires to solve, by developing a prediction model that tries to predict the paths a protein interaction will follow. The only information provided to make the prediction will be the starting protein and the ending protein. For example: Given the starting protein A and the ending protein D, we are going to predict the most likely paths the interaction will follow to get from A to D (Ex: A-B-C-D, A-B-D, A-D). To make this prediction we have a graph of theoretically possible protein interactions and a set of paths validated in the laboratory, both provided by the Institute for Research in Biomedicine of Barcelona.

The usage of the resulting program could substantially shorten the length of future researches by guiding the experiments and reducing the number of tests required for complet-

ing the investigation. This would result in more numerous and cheaper research projects in the sub-field of protein interactions study.

2.2 Scope

In order to complete the project we will need to build a computer model to process and explore the data. However, this model has to meet certain requirements. First and foremost the model must perfectly preserve the representation validity of the provided data. Since it will be the base of the software logic it must be both fast and reliable.

Once the model requirements have been met, we will need to implement a set of graph data mining algorithms in order to extract knowledge from the raw data. These algorithms range from computing the centrality indicators of the different nodes of the project (Degree, Betweenness centrality, closeness centrality...) to identifying different communities within the graph data. Regrettably, in order to guarantee efficiency some of this algorithms will need to directly access the model data. Not respecting class boundaries is not the best programming practice most of the time and goes again the maintainability software quality, but in some cases it will be necessary.

After we have extracted knowledge from the data using the algorithms and the model we will have to use data visualization techniques to further analyze this knowledge by looking for useful patterns to make the prediction.

Finally, using the patterns previously found we will be able to design and develop a prediction algorithm. When the algorithm is completed we are going to use techniques as cross validation to adjust its parameters and validate its accuracy. Furthermore, in the case that there is enough time left we could look for more patterns to integrate into the prediction algorithm in order to raise the prediction accuracy.

These are all the core steps of the project, none can be avoided if we want to complete the project. Once the main goal of the project (the proper prediction functionality) is achieved, other less important features could be added (For instance: a graphic interface or alternative predictions). However, all the features that not related to the path prediction will be considered secondary for the project development.

2.3 Methodology and rigour

In order to guarantee the project main requirements are completed within the schedule and in order to satisfy different software quality standards we have defined a strict project methodology, described in the following lines.

Since this project involves a substantial dose of data science and the prediction algorithm has yet to be found the best fit would be a flexible methodology with short interval iterations. For this reason we will follow the SCRUM agile methodology for software development. By dividing the development and data research project into multiple sprints we will be able to keep track of the project advance and make constant incremental advancements.

2.3.1 Project Requirements

In order to assure a certain quality standard its important to focus on the traits we wish to guarantee. We intend this project to meet the following requirements:

Model Validity: To guarantee the prediction quality the model must perfectly reflect the system. The data itself is given in form of a model (graph and paths), so we will assume that the mathematical model validity is assured. Our task is to build an accurate computer model from it. To assure that this quality is accomplished we are going to extensively test the model, focusing on its correctness and efficiency.

Software Quality: Since this software will be used by a third party to analyze large amounts of data the software quality is crucial. It must be perfectly documented, algorithmically efficient and easily maintainable. To assure that this quality is accomplished we are going to extensively test the algorithms efficiency, follow standardized code documentation methods and using widely used software architecture (Model–view–controller).

Prediction quality: This is both the most crucial and the most challenging requirement. The prediction must be as accurate and fast as possible given the available data. To assure that this quality is accomplished we are going we are going to use cross-validation.

2.3.2 Development Tools

In order to develop a stable and fast software we will implement it with the general purpose language C++, coupled with libraries such as OpenMP to improve the software performance.

Since we want a statistically robust tool with rich data visualization options we will use the statistical computing language R for data exploration.

We will also use graph visualization tools, such as the Gephi open source graph visualization software for data exploration.

In order to keep track of the project development and to document the advancements in every SCRUM sprint we will use Git and Github.

2.4 Risks and Obstacles

Even though the objectives, scope, and methodology have been clearly defined there is still room for problems to occur. In the following paragraphs we will analyze both the most threatening and the most probable problems that might arise.

One of the most common and direst problems that data science faces is insufficient data quality. Thankfully, the Institute for Research in Biomedicine of Barcelona has provided us with high quality data to work. For this reason it is unlikely that this project will suffer from that problem.

Another common data science problem is guaranteeing the validity of the prediction model. For this reasons an in depth research of the different existent prediction models will be an invaluable asset. Moreover, both the director (Ulises Cortés García) and co-director (Dario Garcia Gasulla) have great expertise in the field and its guidance will help the project avoid this severe problem.

The most certain challenges the project is going to face are the development difficulties, both the model and the data mining algorithms have to be fast and reliable. Fortunately, there is an extensive list of papers covering this issues that substantially reduces its difficult.

Probably the greatest obstacle is the research component of the project, which makes it hard to predict how the prediction will be made. There is no predefined procedure to make this prediction and an acceptable prediction accuracy must be reached. On the other hand, it's also makes things more interesting and makes the project worthwhile.

2.5 Project Actors

The development of this project involves several actors:

Project director and co-director: The director (Ulises Cortés) and the co-director (Dario Garcia), members of the UPC KEMLG research group. Their role is to provide guidance based on their expertise during the realization of this project.

Institute for Research in Biomedicine (IRB Barcelona): This is the organization responsible for this project. They provided the data that we will use to build the model, established the requirements and will be the users of the resulting software.

Software Programmer: The project will be fully programmed by me. This means that I will be responsible for the code implementation, the code documentation and the validation of the software.

Software Designer: The project software will be fully designed by me. This means that I will be responsible for the software architecture.

Data Scientist: I will build the model and conduct the data analysis. This means that I will be responsible for the data exploration process, model building and model validation.

Technical Writer: I will build the write all the documentation (Project Memory, GEP Deliverables...) of this project. This means that I will be responsible for the project documentation.

3 State of the Art

In this section we are going to discuss the state of the art of the main related areas of the project divided in two subsets: The medical sciences and the data mining groups. This project will also use knowledge from other fields covered in the computer science degree curriculum (Algorithmics, software engineering). However, we will not cover this subjects because of the space restrictions.

3.1 Medical Sciences Group

3.1.1 Medicine

The science and practice of the diagnosis, treatment, and prevention of disease.

It's existence goes back thousands of years, probably to the origin of the human society itself. Originally an art more than a science, its treatments usually included religious rituals. Thankfully, it was rebuild with the scientific method as it's cornerstone. This lead to a steady improvement of this science, that has ended divided into dozens of more specialized branches.

This sub-fields try to understand the whole human health system in different systemic visions (Molecular level, human individual level, society level etc) and from different perspectives (Studying the virus/bacteria behaviour, studying the human immunity defense system etc).

3.1.2 Bio-medicine

Bio-medicine is a branch of medical science that applies biological and biochemistry principles to clinical practice. Thanks to its bottom-up view (Contrary to the traditional up-bottom view that medicine has used for centuries) it is considered the cornerstone of modern health care and laboratory diagnostics. It also has a strong focus on maintaining people healthy instead of the traditional disease/infection/injury responsive approach.

It was traditionally studied by directly observing the system (Ex: Microbiological culture for experimentation), but since those biological systems usually weren't accessible experi-

mentation required a great deal of effort and time. Thankfully, current computer models helped to improve the understanding of the biological systems without direct experimentation, which drastically reduced the required number of biological system experiments to understand how the system actually worked.

The problem that this project helps to address falls into this field, concretely in the sub-field of molecular biology. This sub-field focuses in the molecular basis of biological activity between bio-molecules in the various systems of a cell, and amongst this systems there are the interactions between proteins. The prediction of the path followed in this interactions is the focus of the project.

3.1.3 Bio-informatics

Bio-informatics is an interdisciplinary field that develops methods and software tools for understanding biological data. It uses a wide diversity of mathematics sub-fields, such as statistics, together with computer science.

It can be considered an umbrella term, because currently there is hardly any biological science that doesn't use this kind of assistance. This is the field the project falls into .

3.2 Data Mining Group

3.2.1 Data Mining

Also known and knowledge discovery, data science, Knowledge Extraction and by many others names, is an interdisciplinary sub-field of computer science. Its goal is to discover patterns (Knowledge) in data-sets.

Based in statistical theory, it uses a wide array of techniques from fields shuc as artificial intelligence and machine learning to extract knowledge from data. We understand knowledge as statistical patterns that could help us better understand, modelize and predict how the system behaves. This statistical knowledge can be expressed in different depending on the pattern extraction procedure (Algorithm).

The data we try to "mine for knowledge" is usually stored in data-bases, for this reason database knowledge is considered within this field. Moreover, based on the idea that a system is as good as the quality of its inputs, data cleaning techniques are usually needed because in most cases the original data is incomplete and/or unreliable. It could also happen that changing the data format could improve the data mining algorithms results (For example, extracting different features of an image to use them as an input for a neural net). This process is known as data pre-processing.

The data is traditionally divided in data-sets that can be expressed in a matrix format. Each row is known as an instance/observation and each column as an attribute.

Once the data is revised an correctly formatted, different algorithms can be used to extract knowledge. Most current algorithms focus on:

Summarization: Describing the knowledge using visualization techniques (Ex: Charts, Heatmaps ...) and different metrics (Ex: centrality metrics such as the mean, range of values etc).

Regression: Describing the knowledge with a mathematical function that tries to model the data with the least error.

Classification: Dividing the data instances in a set of previously defined classes. For example, article genre classification by analyzing the words used and its frequency (Bag of words method).

Clustering: Dividing the data instances in a set of previously unknown classes based on the similarity between data instances attributes.

Anomaly detection: Identify unusual data instances (Strange attribute value, strange attribute value combination etc).

Association rule learning: Modeling the relationships between the data instances by rules (Ex: If "cloudy" and "falls water from the sky" <=> "It is raining").

3.2.2 Graph Data Mining

Graph Data Mining is a case of structured data mining. The structured data mining is a sub-field of data mining that focuses on extracting information not from the traditional tabular data sets but from structured data. This requires of more computational power than the traditional approach, still the richer expression capacity that the structured data is capable off enables more in-depth analyses and justifies the efficiency loss. Thanks to the rise of computing power over the last decades large scale analysis to structured data are now possible.

Graph Data Mining is the process of finding and extracting useful information from the structure of a graph. It has a wide variety of sub-domains depending on the different types of networks to be analyzed (For instance the sub-domain of social network mining and the sub-domain of biological network mining). Each of this sub-domains addresses networks with different characteristics and behaviours.

In order to solve the problem addressed in this project we are going to use a variety of general graph data mining techniques and biological network mining techniques.

4 Related Work

In this section we will discuss different works related to this project. Taking into account the current abundance of molecular bio-medicine research, we will focus on the projects that could directly help the development of this project (Ex: Giving access to quality data).

pathwaycommons.org An online platform that provides network biology resources. They provide high-quality data and network biology visualization. It's hard to overestimate the great opportunities that open data offers. It can be used as an extra data source in the case of need.

pantherdb.org Another online plataform that provides network biology resources. It can be used as an extra data source in the case of need, specially if we want to cross genomic interactions with protein interactions.

5 Project Management

This sections aims to clearly define the tasks and the project temporal planning.

Before starting, it's important to remember two points: The tasks are iterative and the temporal planning isn't set in stone. Since this project involves a certain degree of research and trial and error a waterfall model would lack the necessary flexibility, so even though the tasks have a clear defined order we might have to cycle through them. (For instance, to further adjust the prediction changing the parameters might be insufficient and the best option might be to slightly modify the Prediction Algorithm Design) For the same reason, the project temporal planning might be subject to changes depending on the development of the project.

5.1 Tasks Description

In this section we will describe each task. For each tasks there is going to be a concise description of the procedure to follow,the main goal, the resources needed and the previous requirements.

All the tasks will be completed using a PC (Intel Core i7-5820k 330GHz (12 CPUs) CPU) and it's associated peripheral devices (Screen, keyboard...).

5.1.1 Task 1: GEP Project Management Course Completion

An important part of the future success of a project comes from the preliminary work. For this reason this project starts with an in depth research of the state of the art and the related works. Followed by the preparation of a clear temporal planning and the estimation of the resources needed for its completion and its corresponding budget. This documentation will be written using the sharelatex.com web latex editor.

The completion of this task will require the work of a technical writer for 75 hours.

This preliminary task is the starting point of the project. For this reason, this task has no prerequisites.

5.1.2 Task 2: Computer Model Implementation

The first step to trying to understand the protein interaction system is to build a model that will allow us to analyze the interaction structure. Since the data we have access to is a set of protein relationships and protein interaction paths, the most fitting model would be a graph structure. We will build this model using C++ together with the OpenMP library to speed up the knowledge extraction algorithms with parallelization.

At first the idea is to try to implement both the algorithms and model to get a better grasp of graph data mining. However, with a limited implementation time this approach may result too laborious. In which case we will either use a graph processing system such as Apache Gyraph or a high speed performance library such as C++ Boost Graph.

We will assume that the graph representation structure is a valid system representation given that it has been used by the data provider (a domain expert), putting model validity errors out of the picture.

The completion of this task will require the work of a software designer for 25 hours and the work of a software programmer for 125 hours.

This task is the first development phase. Its only requirement is the task 1.

5.1.3 Data Exploration

Once the model is implemented we will be able to use graph data mining techniques to improve our system's understanding. By measuring different node centrality indicators (Such as betweenness centrality) and graph visualization tools (For instance: the open source project Gephi) we will aim to understand what type of network is and how it behaves.

After exploring the graph data the next step would be to make a profile for each protein that appears in the pathways data, focusing on it's apparition positions and frequency.

In order to use data visualization techniques (For example: Heat maps) for numeric indicators we will use the R language and it's high quality visualization packages.

The only problem that we might face in this stage, albeit the most problematic, would be the lack of any kind of pattern useful for prediction. If we find ourselves in this position the best answer would be to ask for more data. Fortunately, this complication is highly unlikely to happen.

The main goal of this phase is to find patterns that will enable us to make the path prediction. The more numerous and the more useful this patterns are, the better.

The completion of this task will require the work of a data scientist for 75 hours.

This task requires a model in order to explore and process the graph data. For this reason, the requirement of this task is a valid computational model (Task 2).

5.1.4 Task 4: Prediction Algorithm Design and Implementation

This step consists in designing and algorithmic solution to exploit all the patterns in the data that can be used for the prediction.

The algorithm is going to be implemented in C++, using the OpenMP library to speed up the process with parallelization.

Taking into the account that the patterns were already found, the only undesirable situation that might appear is that the use of those patterns for prediction can't be parallelized properly. A solution could be to offer different kinds of predictions, ranging from slower to faster predictions of decreasing accuracy.

The completion of this task will require the work of a software designer for 25 hours and the work of a software programmer for 75 hours.

This task requires a set of patterns to exploit in order to make the prediction. Because of this, the requirement for this task is a set of patterns useful for prediction(task 3).

5.1.5 Task 5: Prediction Adjustment

This step consists in using cross validation to adjust the prediction algorithm parameters. In order to maximize the prediction accuracy cycling back to previous steps could be desirable.

Both cycling back to the prediction algorithm design to tweak the algorithm and/or the data exploration to find new patterns may further improve the prediction quality.

The completion of this task will require the work of a data scientist for 75 hours.

This obvious requirement for adjusting a prediction algorithm is a prediction algorithm to adjust. For this reason, this task requires of the previous completion of the forth task.

5.1.6 Task 6: Final Stage

Once the prediction is certain enough, the final stage starts. This phase consists in four sub-tasks: redacting the final version of the project documentation, preparing the final presentation and project polishing.

The goals of the redacting the final version of the project documentation and preparing the final presentation are quite straightforward.

The goal of project polishing is to make some final improvements(As for example: Graphic interface with QT). This sub-task is optional and its execution depends on the project time constraints.

The completion of this task will require the work of a technical writer for 75 hours.

This is the final task and requires all the research assistance software development to be completed. In other words, its previous requirement all are the other tasks.

5.2 Tasks Planning

In this section we address the temporal aspects of the project. The project will last four months, starting in January and ending in July of 2016. We will present an estimation of the time each task will require and a Gantt chart of the whole project.

The idea is to work as planned. However, the planned human resources (hours of work) might not suffice to produce all the desired features. If a task cannot be completed in the planned schedule we will have to decide whether to extend the task finishing date or not. In the case that all the main requirements have been met we will continue with the following

task. However, in the opposite case the finishing date will have to be extended. Being this the case, the future workload will have to be adjusted in order to meet the future project milestones.

Fortunately, the use of the agile methodology SCRUM practically guarantees that all the main requirements will be completed in time, since each task sprints will be completed in order of importance. In other words, our tactic to guarantee the project completion in the set time frame is going to be prioritization of the core functionalities.

The meetings with the director and co-director are going to be arranged every time an important milestone of the project is reached.

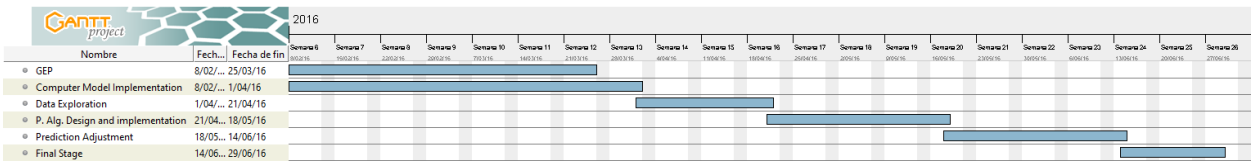
From the following table it can be deduced that the total amount of work is going to be about 600 hours. Taking out the director, co-director and GEP tutors work this results in 550 hours for the project completion. With a time frame of 20 weeks the expected amount of work is 25 hours/week, which is a reasonable workload that makes the project completion achievable in the given period.

5.2.1 Estimation of time required to complete each task

Task	Required Time (hours)
GEP	75.0
Computer Model Implementation	150.0
Data Exploration	75
Prediction Algorithm Design and Implementation	100
Prediction Adjustment	75
Final Stage	75
Total	550

Table 1: Estimation of time required to complete each task.

5.2.2 Gantt Chart



6 Budget and sustainability

In this section we will examine the production costs associated with the different aspects of the project development, divided in human resources, hardware resources and software resources costs. We will also try to anticipate unplanned expenses and analyze the economic, social and environmental sustainability of the project. Because of the project's particular development nature its planned budget is subject to change.

6.1 Cost Estimation

In this section we are going to estimate the resources needed for the development of the project and its associated monetary costs. We will divide the resources needed for the project in three main categories: The hardware, software and human resources. It is worth mentioning that we will not reflect all the office related and all the Internet associated costs in the budget.

For each resource of any type we are going to define the monetary cost, taking into account the useful life of the resource and the duration of use (6 months). At the end we will review the full cost of the project.

6.1.1 Human Resources

The following tables show the costs of the human resources needed in this project. The project manager cost comes from the Director, Co-Director and GEP Tutor in matters of advice and assistance. The work of a data scientist will be needed to complete tasks three (Data Exploration) and five (Prediction Adjustment). For the forth task (Prediction Algorithm Design and Implementation) and second task (Computer Model Implementation) both a software programmer and software designer are going to be required. A technical writer will be needed to concisely write down the project documentation in tasks one (GEP) and six (Final Stage).

Human Resource	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Project Manager	5	10	10	10	10	5
Data Scientist			75		75	
Software designer		25		25		
Software programmer		125		75		
Technical writer	75					75
Total	80	160	85	110	85	80

Table 2: Human resources hours of work divided by task.

Human Resource	Cost per hour (€/hours)	Required Time (hours)	Cost(€)
Project Manager	30.0	50	1500.0
Data Scientist	17.5	150	2625.0
Software designer	15.0	50	750.0
Software programmer	12.5	200	2500.0
Technical writer	10.0	150	1500.0
Total			8875.0

Table 3: Estimation of the human resources monetary cost.

Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
900	2237.5	1612.5	1612.5	1612.5	900

Table 4: Human resources monetary cost (€) divided by task.

6.1.2 Hardware Resources

The only hardware resource used to complete the project is a PC (Intel Core i7-5820k 330GHz (12 CPUs) CPU) and it's associated peripheral devices (Screen, keyboard...). Since this personal computer wasn't bought for the project its current cost is an estimation. We are going to assume a expected service life of four years for the PC project cost. This resource will be used for every task of the project. The next table shows the expenses for each hardware component needed in this project.

Hardware Resource	Power Consumption (W)	Total Cost	Time of use	Project Cost
Personal Computer and peripheral devices	650	1150	550	143.75

Table 5: Estimation of the hardware resources monetary cost.

Associated with the cost of the hardware there is the energy cost. For calculating this cost we will use the following formula:

$$\text{Power Consumption}(€) = cost(e/kWh) * power(kW) * time(h)$$
$$\text{Power Consumption}(€) = cost(0,125e/kWh) * 0,65(kW) * 550(h)$$
$$\text{Power Consumption}(€) = 44.6875\ €$$

6.1.3 Software Resources

Table 3 shows the expenses for each software component needed in this project. R language, together with R ggplot2 Library, R Studio and Gephi are going to be used to visualize an explore the data (Task 3). C++ Language together with the C++ OpenMP Library are going to be used for the second, fourth and fifth tasks. As it can be easily seen, all the software used in this project is free. For this reason, there is no cost related to software.

Software Resource	Cost
R Language	0.0
R ggplot2 Library	0.0
R Studio	0.0
C++ Language	0.0
C++ OpenMP Library	0.0
Gephi	0.0
Total	0.0

Table 6: Estimation of the software resources monetary cost.

6.1.4 Total Cost

Using data shown in tables 1, 2 and 3, we can use table 4 to describe the total project cost:

Resource	Cost
Human Resources	8875.0
Hardware Resources	143.75
Energy Resources	44.7
Software Resources	0.0
Total	9063.45

Table 7: Estimation of the different resources monetary cost.

6.2 Cost Control

As stated previously, the estimated budget is subject to change if the development of the project requires it. Hardware Resources are unlikely to change, given the expected service

life of the PC. Software resources are even less subject to change, since there are a great number of high quality software resources at our disposal for free. Human resources costs are the most likely to change, both for its inherent unpredictability and the very nature of the project, which makes hard to correctly estimate the required manpower for each task.

In order to reduce possible budget deviations we will decompose the requirements of the project in SCRUM sprints. This sprints will be completed following the decreasing order of its importance until all the human resources assigned to the task are expended.

However, taking into account the fact that the project future benefits will increase by every satisfied requirement and the fact that the cost reduction of the resulting software usage is potentially far greater than the project development cost, it would be wise to obtain a line of credit. This would enable us to, in the case of need, increase the human resources in order to better satisfy the project requirements. In a certain light, every extra SCRUM sprint could be seen as a long term investment.

6.3 Sustainability

In this section we are going to evaluate the sustainability of our project in three different dimensions: the economic, social and environmental dimensions.

6.3.1 Economic

In the previous sections we have already assessed the financial costs of the project, taking into account the material, human and energetic costs. All this costs are necessary and none can be substantially reduced.

Because of the very nature of this project, no direct income is expected to be gained by the use of the resulting software. This doesn't hold down the project in any way, since the benefits it tries to produce have not a direct financial nature.

The cost stated in the Budget estimation section of this document could be the only cost spent in the project, since we aim to create a working program. However, it is possible that further development is needed. In order to make things easier for future development of the program (Ex: adding more features or functionality improvements) there is going to be a

strong focus on writing a maintainable code. The work required for achieving this software quality has already been taken into account in the related development tasks.

6.3.2 Social

The application that we are going to develop in this project is going to be used by the Institute for Research in Biomedicine of Barcelona. Since this project isn't going to be used by the general public and the result isn't a specific service or final product its hard to estimate its impact.

Nonetheless, taking into account the fact that research is amongst the human activities with the greatest possible impact in society, all possible assistance is worthwhile. It's hard to overestimate the key importance of investigation in the past human societies development and, in order to advance even further, it is crucial to give proper tools to the researchers.

For these reasons, we consider that this project has a significant, albeit indirect, positive impact in society. Specially taking into account the tremendous life quality improvements bio-medicine can bring to people.

6.3.3 Environmental

Amongst the resources detailed in the cost estimation section, the resource with the most significant ecological footprint is the PC. Taking into account its production cost and the energy cost used for completing the project we can say that it has a significant environmental cost.

The use of the computer could be reduced by using existing libraries, but that would have a detrimental effect on the learning experience. Fortunately, the assistance the resulting software will provide will help reduce the number of experiments. This in turn will reduce future energy and material costs, ending with a positive environmental impact.

6.3.4 Sustainability Matrix

The following table describes the project sustainability. The maximum score is awarded when there are no significant downsides (Ex: No important risks, no large costs ...) and great benefits. The score value decreases by every downside an its significance (Ex: PC production cost, project development cost) and/or the lack of benefits. Each dimension (Environmental, economical and social) is independently scored. The sum of all the scores for each category (PP, Life of Service and Risks) for each dimension is the general sustainability score of the project.

The score range per category is:

PPP From 0 to 10 for every category.

Life of Service From 0 to 20 for every category.

Risks From -20 to 0 for every category.

Each score is awarded based on the dimension analysis (Sections 6.3.1, 6.3.2 and 6.3.3).

Area	PPP	Life of Service	Risks
Environmental	9	18	0
Economical	9	18	0
Social	10	20	0

Table 8: Sustainability Matrix

The general sustainability score of the project is 84/90.

7 Project Development

In this section we are going to describe the project development.

Since the project advanced following the initial plan stages we will organize the narrative by the following tasks: the Computer Model Implementation task, the Data Exploration task, the Prediction Algorithm Design and Implementation task, the Prediction Adjustment task and The Final Stage Task. For each of these tasks we will describe the process followed until their completion, the technologies that were involved and discuss their results.

It is worth noting that the GEP Project Management Course Completion task won't be discussed any further, the reason being that it has already been submitted (And revised) in previous milestones. The previous sections are the result of that task.

7.1 Computer Model Implementation Task

The initial step was to build a basic model with the goal to start analysing the data. The approach taken was to build the most essential classes and progressively extend them with new functionalities whenever it was found necessary.

The Institute of Research in Biomedicine provided us with two different (albeit related) data sets: A graph of protein relationships and a list of tested paths (Protein interactions that were observed in the laboratory).

The graph was expressed with a list of links and the paths were expressed in a condensed format. Each path was an ordered set of unordered sets of proteins. Each position of the path was collection of proteins connected with all the proteins of the next position. The figure number 1 shows an example of a path (The compressed path [a,b],[c,d],[e] contained the paths {a,c,e}, {a,d,e},{b,c,e} and {b,d,e}).

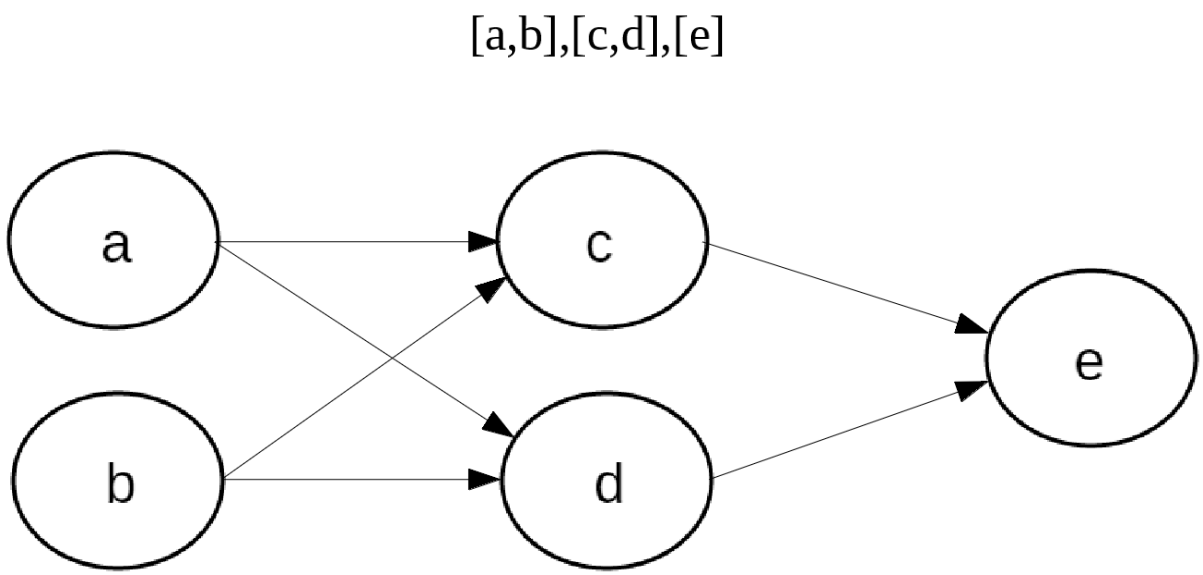


Figure 1: Example of a compressed path

We decided to use the closest representation to the IRB data sets: An undirected weightless graph data structure and a structure to store and manage the compressed paths. The figure number 2 shows the class diagram of the initial model.

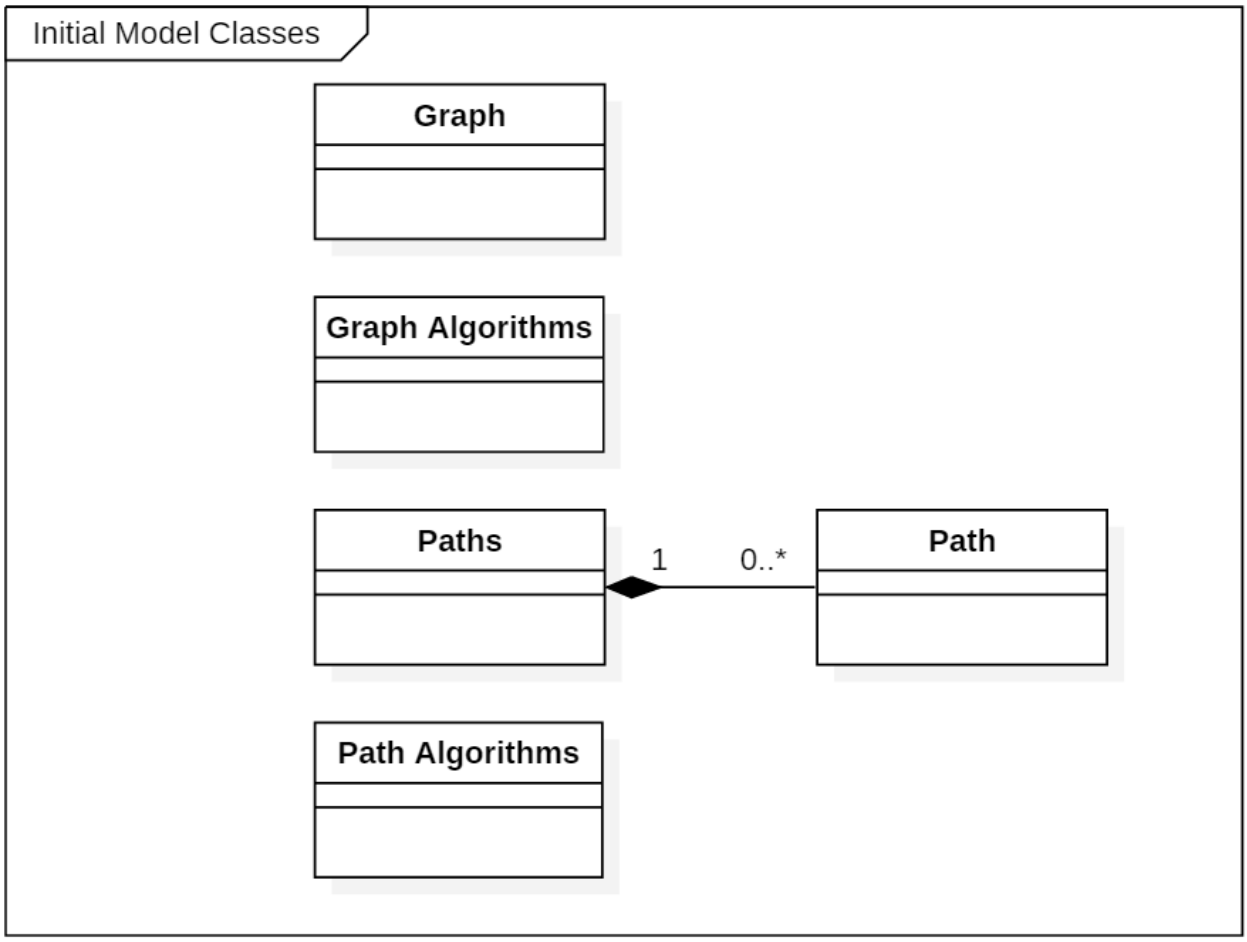


Figure 2: Initial Model with the most essential classes

Initial Model Classes Description:

Graph

An undirected weightless graph data structure. Two implementations based on adjacency lists were made, one using a vector of sets and the other one using a vector of vectors as base data structures. The justification of this duplicity was that different algorithms would perform better using one base data structure or the other.

Paths

A data structure that stored and managed the compressed paths.

Path

A data structure that stores and manages a path (A path example can be seen in figure number 1). Implemented with a vector of vectors as base data structure. We decided to keep the compressed format because the number of decompressed paths would be tremendously large due to combinatorics.

PathsAlgorithms

A collection of path data mining algorithms.

GraphAlgorithms

A collection of graph data mining algorithms.

Initial Model General Description:

Initially all the program's logic was managed by the main method. What is more, all the communication between classes was also managed by the main method. The reason behind this decision is that both the graph algorithm and path algorithm classes were thought to directly use the inner data structures of the graph and paths classes (Reading access only). Although this is not a great practice for building maintainable code, the complexity of some algorithms would probably require the maximum possible efficiency.

7.1.1 Technology Involved in the Task

All the code was written in C++ using the Sublime Text 3 Text Editor in a Linux environment. Additionally, Git technology was used to access the data sets.

7.1.2 Task Results

The initial model was fully implemented with all the basic data structure classes (Graph,Paths and Path). At the end of this task the GraphAlgorithms and PathsAlgorithms classes were still method-less, but functional and ready to store all the algorithms the following step could need.

7.2 Data Exploration

Using the model as a foundation and some other tools we took a series of approaches from different angles in order to find patterns that we could make use of for prediction.

7.2.1 Approach 1: Preliminary Exploration

The first approach was general exploration of the graph and path data sets. We started gathering general information (Number of links, Number of nodes, Number of paths, Mean path distance, etc) about the data sets in order to see if we had enough data. After some global indicators (Ex: Number of Graph Nodes = 12225, Number of Graph Links = 61962, Number Paths = 5274...) we concluded that there was enough data to proceed with a more thorough analysis.

Next, we shifted the focus to analyse each protein present in the data sets. With the goal to measure the relevance in the network of each protein in mind we started measuring the following node centrality indicators(for each protein from the graph data set):

Degree - Number of edges incident to the vertex.

Betweenness Centrality - The number of times a node acts as a bridge along the single shortest path between two other nodes. In the case that there are multiple shortest paths of equal length the value/score is shared by divided by their number.

Closeness Centrality - The average length of the shortest path between the node and all other nodes in the graph.

Shortly after, in order to measure the relevance in the path collection, the following self made indicators were computed for each protein from the paths data set:

Ocurrences - The number of compressed paths where the protein is present. The combinatorics of the path are not regarded. (For example, lets measure the occurrence of 'a' in the path [a,b][c,d][e]: occurrence([a,b][c,d][e]) = 1)

Intensity - The number of uncompressed paths where the protein is present. The combinatorics of the path are taken into account (For example, lets measure the intensity of 'a' in the path [a,b][c,d][e]: intensity([a,b][c,d][e]) = 2x2x1 = 4)

Computing the betweenness centrality of a big graph by brute force was computationally intensive. For this reason the implementation was done based on the paper "A Faster Algorithm for Betweenness Centrality, Ulrik Brandes, University of Konstanz".

The following plots show the visualization of the different indicators previously described. Each of these plots shows the frequency (Number of appearances, ex: There are 100 nodes with degree 27) of an indicator value (y-axis) and it's value (x-axis). With the exception of the histograms of degree (Figure 3 is an example of an histogram without log scale, where the need for a log scale is made apparent) and closeness (Figure 5) all the other plot's axis are in a log scale in order to properly scale the visualization.

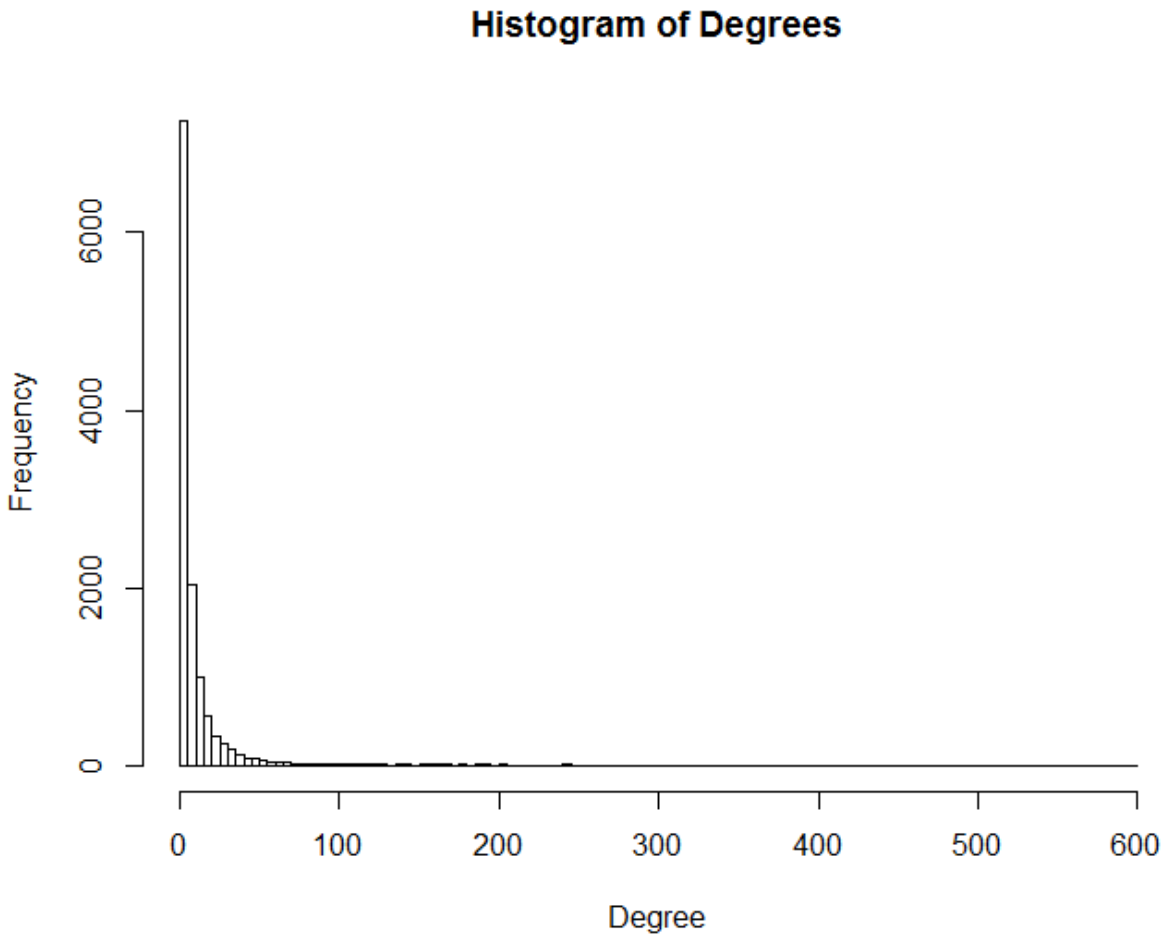


Figure 3: Normal Scale Histogram of Frequency and Degree.

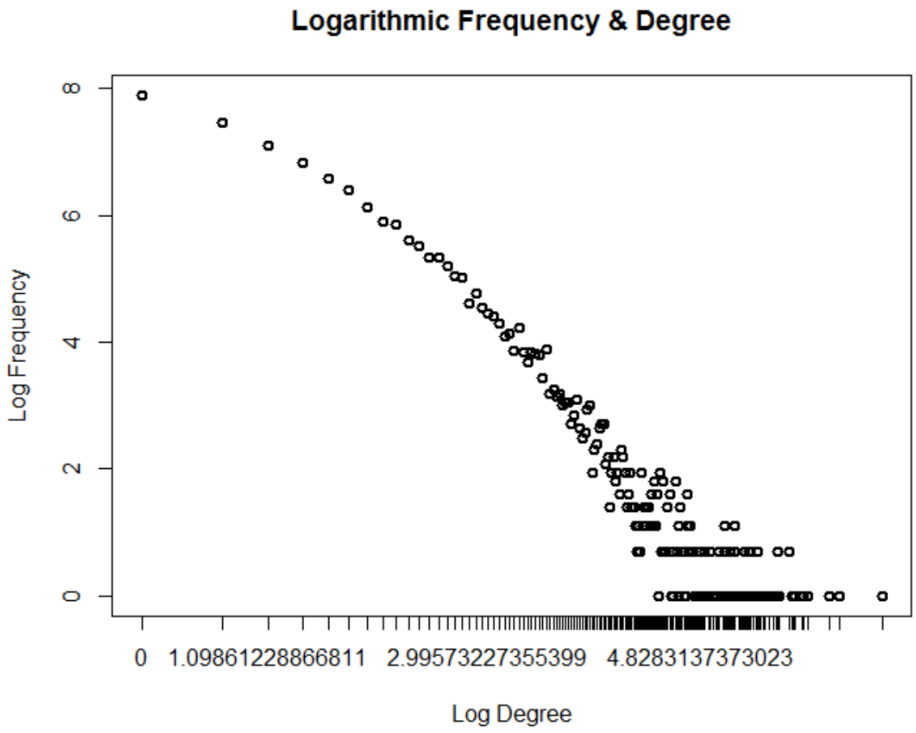


Figure 4: Log scale plot of Frequency and Degree.

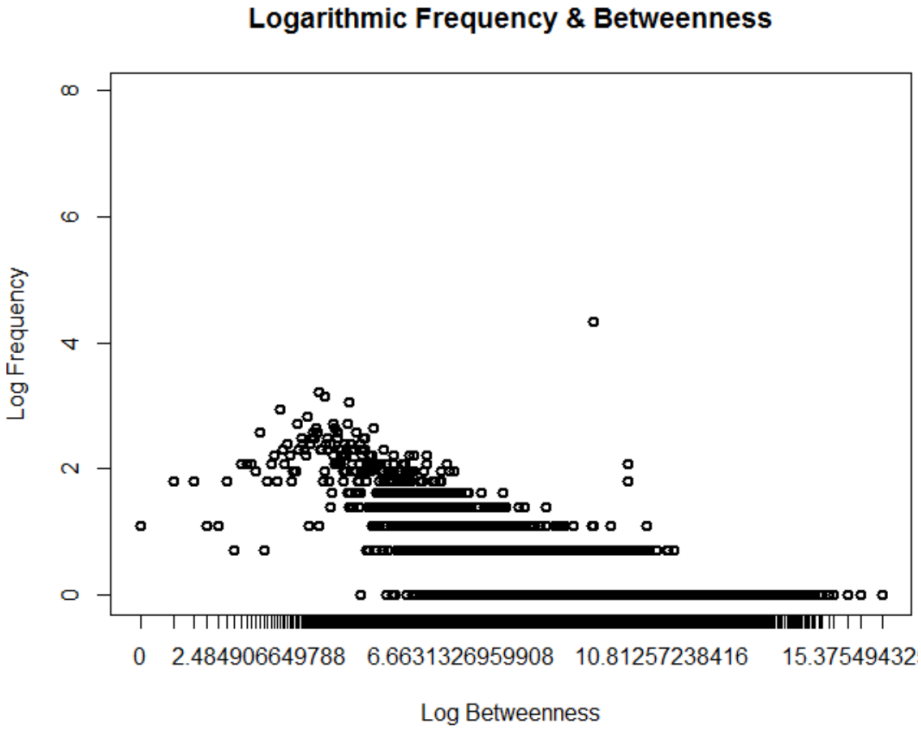


Figure 5: Log scale plot of Frequency and Betweenness.

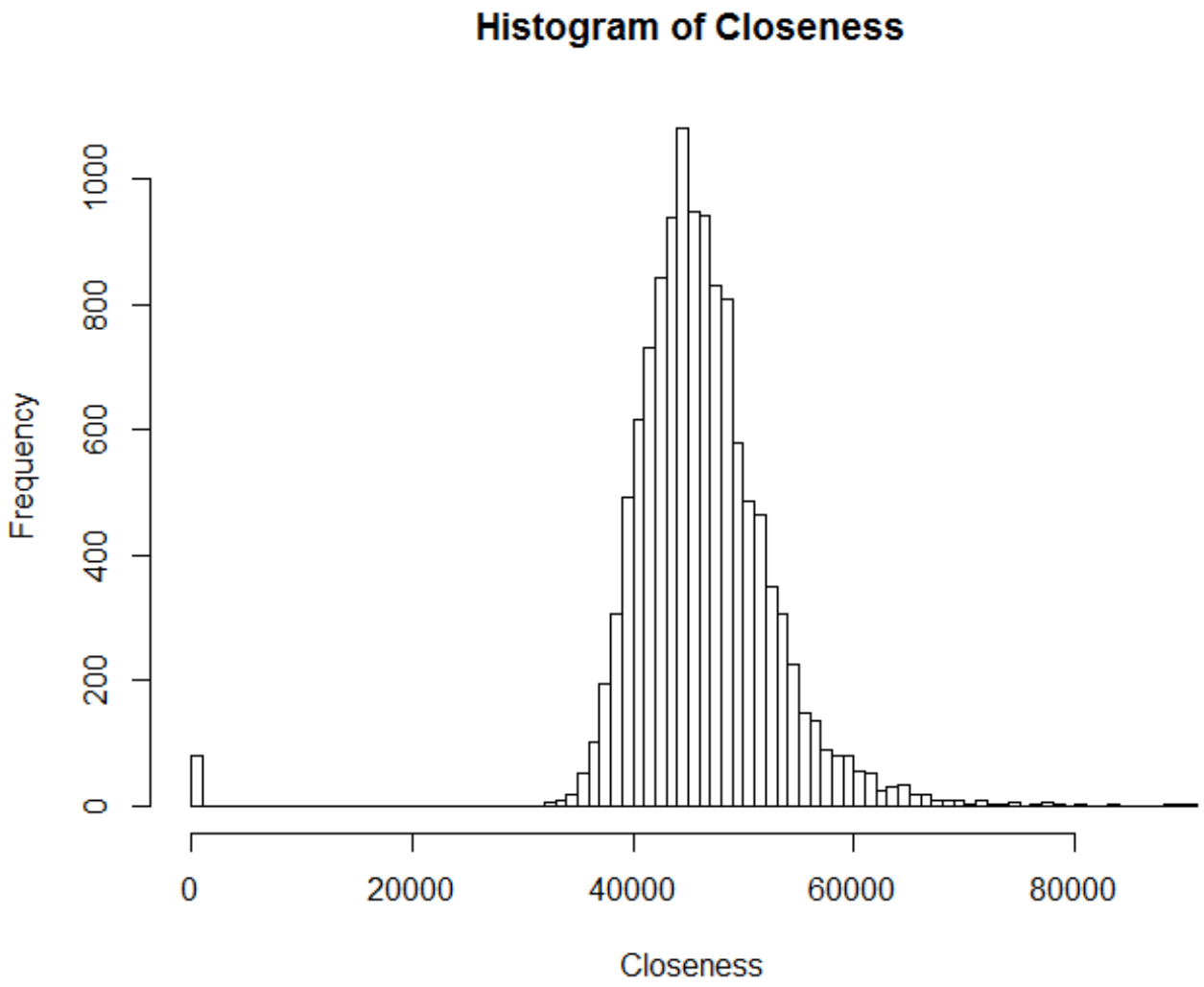


Figure 6: Normal Scale Histogram of Closeness.

Observing the different plots of the graph data we arrived at the conclusion that the protein relationships behaved as a scale-free network (A network whose degree distribution follows a power law).

After discovering this, we hypothesised that the hubs (Proteins with the most number of links) would be the most common in the paths and would define their structure. Furthermore, taking into account the role the transcription factors (A protein that controls the rate of transcription of genetic information from DNA to messenger RNA) we also hypothesised transcription factors were hubs.

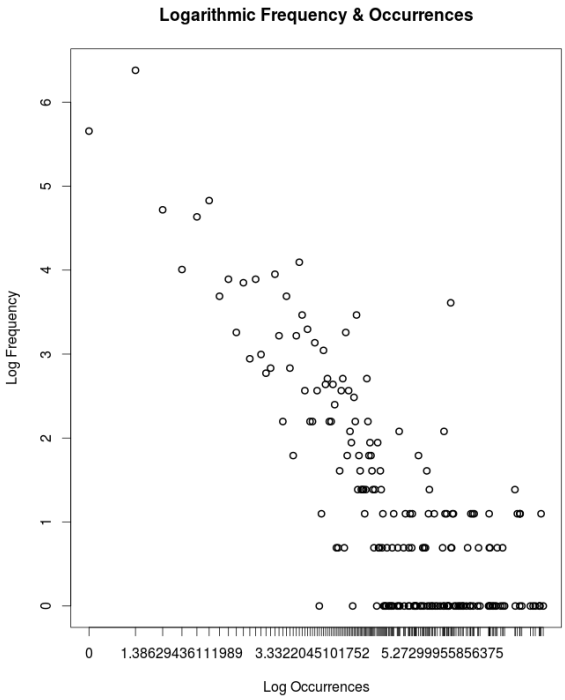
7.2.2 Approach 2: Transcription Factor Identification

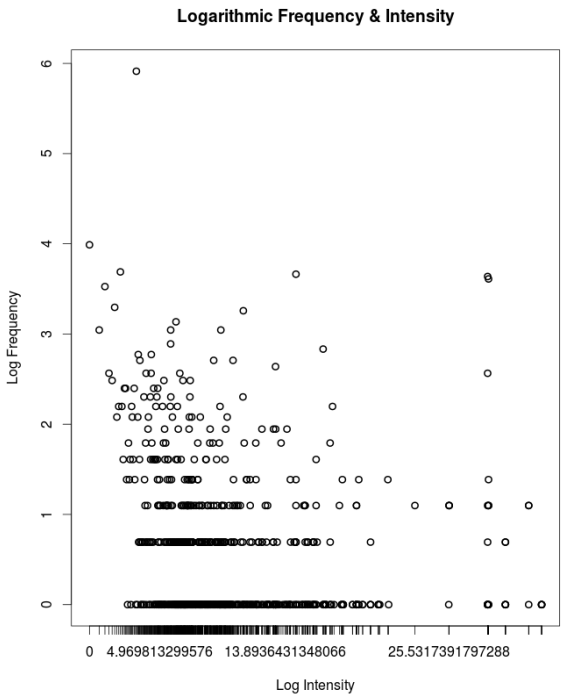
The second approach was the direct result of the first one, it's goal was to validate or refute the hypotheses drawn from the previous approach.

We started by trying to detect what proteins where transcription factors using graph centrality indicators. Amongst them the betweenness centrality (The number of shortest paths from all vertexes to all others vertexes that pass through that node) and closeness centrality (the sum of its distances from all other nodes) and the degree node indicators.

Using this metrics we selected a list of transcription factor candidates amongst the proteins with the highest values in degree, betweenness and closeness centrality.

Before sending the list to the IRB to check how many transcription factors were on the list, we decided to check how these proteins behaved within the paths. Shockingly, there wasn't a clear relationship between the protein node centrality indicators and the number of paths the protein appeared in. What is more, a significant number of proteins and an huge number of links were exclusive to the paths (They didn't appear on the graph) and vice versa. The following plots show the frequency of the path indicators occurrences and intensity of the proteins that appeared in the graph:





In order to face this unexpected outcome we used three graph versions, the original one, a graph constructed reverse engineering the paths and an enriched version (The original one adding the nodes and relationships present on the paths). First, for each of these graphs we generated the different plots and checked their behaviour as a network (All of them behaved as a scale-free one). After, we selected the top outliers for each of the top graph centrality indicators (Expecting most of them to be Transcription Factors) and wrote a list with all the their associated proteins (Transcription Factor Candidates).

Finally, we sent the three lists of candidates (One for each graph) and asked about the discrepancies between the proteins that appear on the graph and the paths. Unfortunately, the sent transcription factor candidates weren't valid. On the other hand, the discrepancies were known by the IRB. They explained us that the data from the graph and paths had been obtained from different sources. Furthermore, they indicated that they didn't want us to directly mix those two data sets. For this reason, the graph version we will utilize for the rest of the project will be the original one.

In the end, all the hypotheses were refuted and we were left without a pattern to make the prediction possible. For this reason, we started taking entirely different approaches.

7.2.3 Approach 3: Graph Visualization

The immediate approach was to visualize the graph and to look for patterns in its emerging structure. We tried several graph visualization algorithms: Fruchterman-Reingold force-directed graph drawing algorithm, Yifan Hu Multilevel force-directed graph drawing algorithm and the forceatlas algorithm 1 and 2. However, no clear structure emerged with any of these algorithms, not even after a several hours of computation (+4h) for each of them.

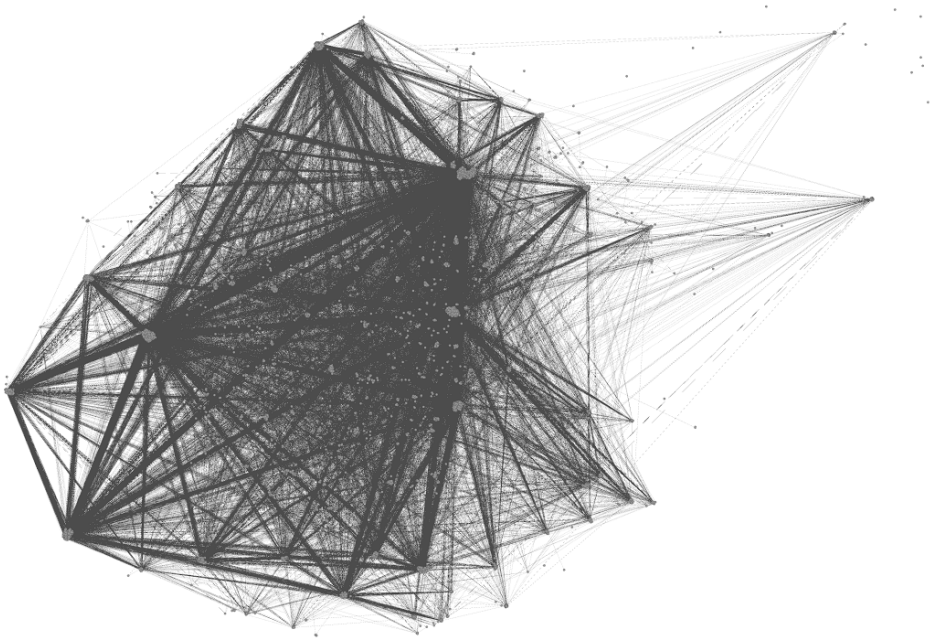


Figure 7: Result of executing the Yifan Hu Multilevel force-directed graph drawing algorithm.

7.2.4 Approach 4: Minimum Path

The next approach was to compare the length of the paths in the path data set with the minimum path distances between it's starting and ending proteins in the graph data set. The relationship between those two can be observed in the following heat scatter, with a colour range of [grey - purple - blue - yellow - orange - red] depending on the number of coincidences (Number of paths of min length Y in the graph with an actual length of x in the known paths).

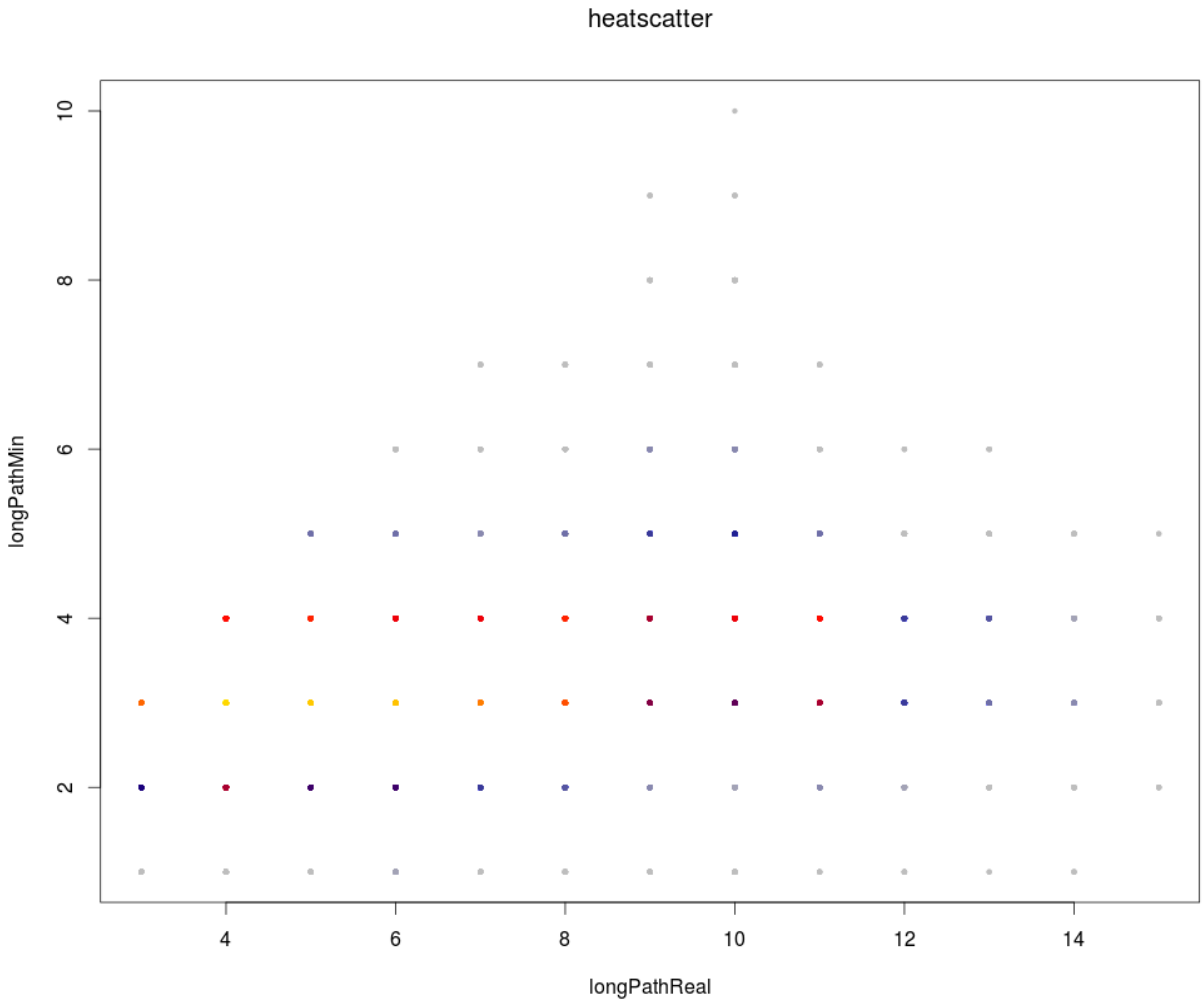


Figure 8: Heat scatter of the length of real known path vs the length of min path

The following heat map shows even more precisely (Although less atheistically) where in the possibility space are located the greatest number of paths (Most paths between two proteins have a minimum of length 3 and 4 and a real length between 3 and 8).

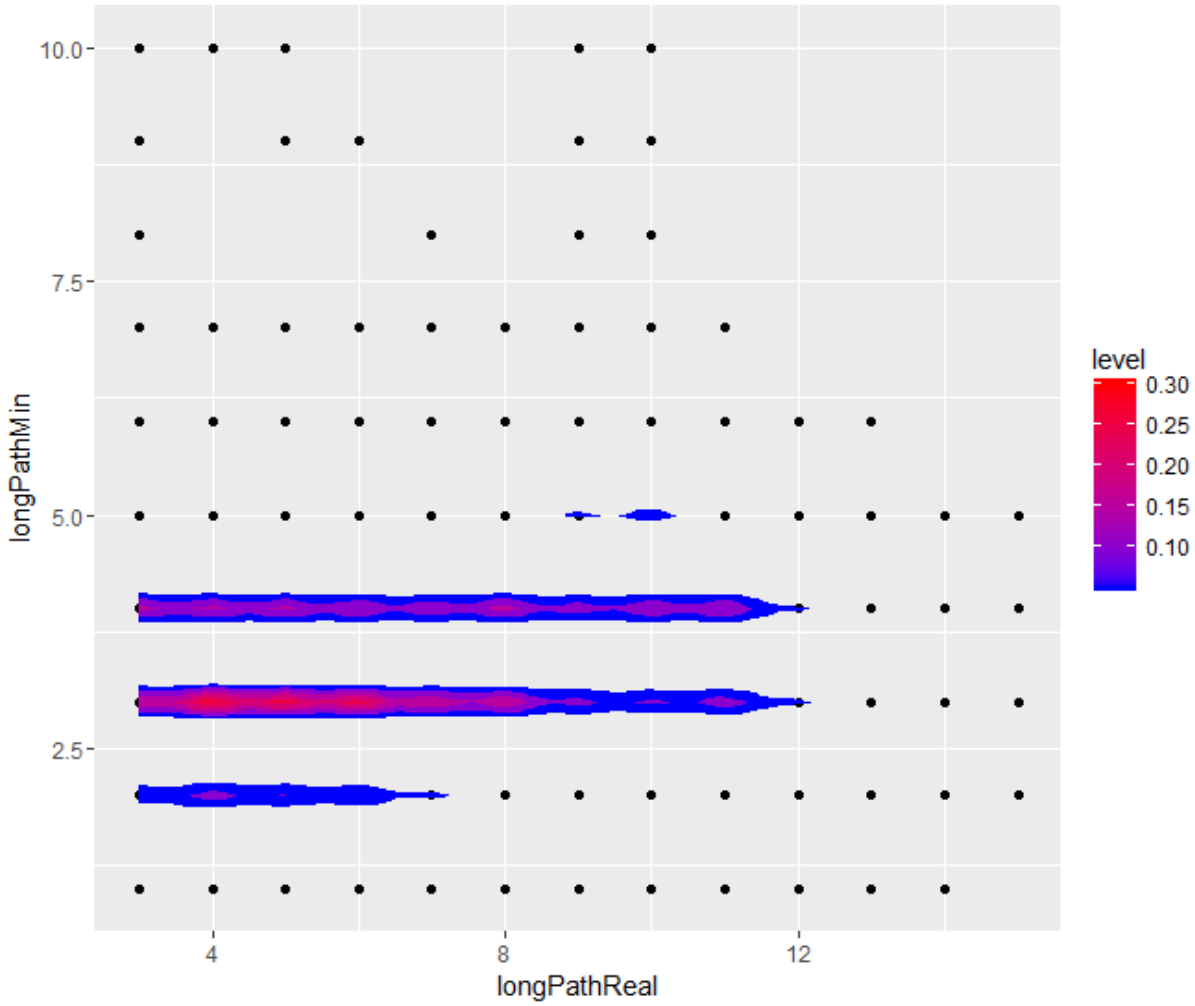


Figure 9: Heat map of length of actual path vs length of min path

Since no clear correlation could be directly extracted from the heat maps (some paths were as short as the shortest option and others were far longer without a way to differentiate them) we used polynomial approximation to fit the probability distribution function of each known path length for each minimum path size. The following plots are the probability distribution functions with different minimum lengths (The x axis representing the known path length and the y axis representing the total number of occurrences for a given minimum path length).

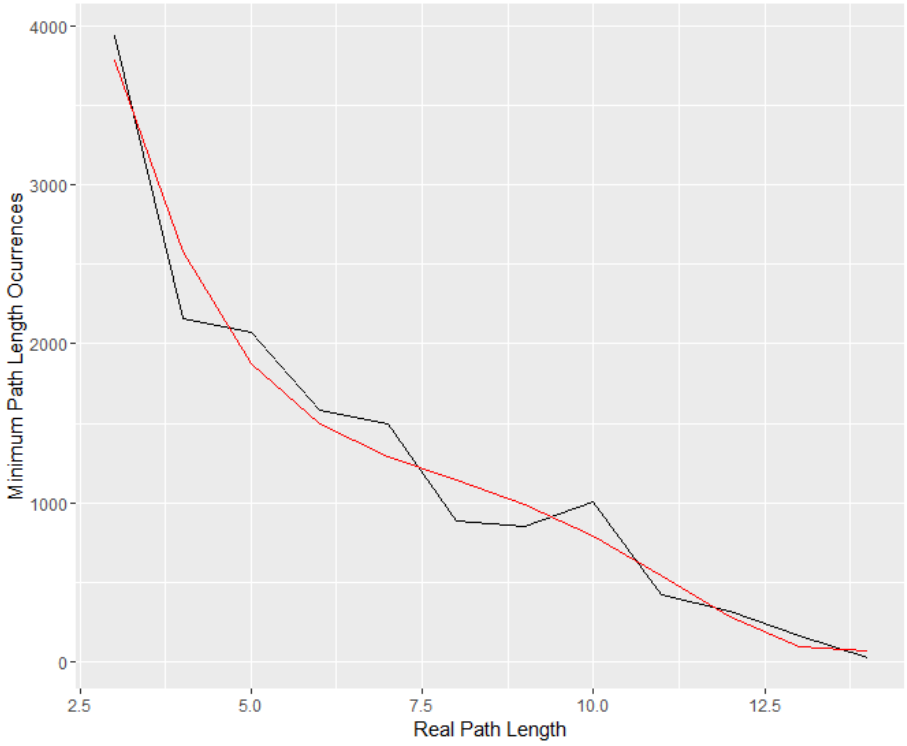


Figure 10: Fitted probability distribution function (red) of the paths with a minimum graph length of 3.

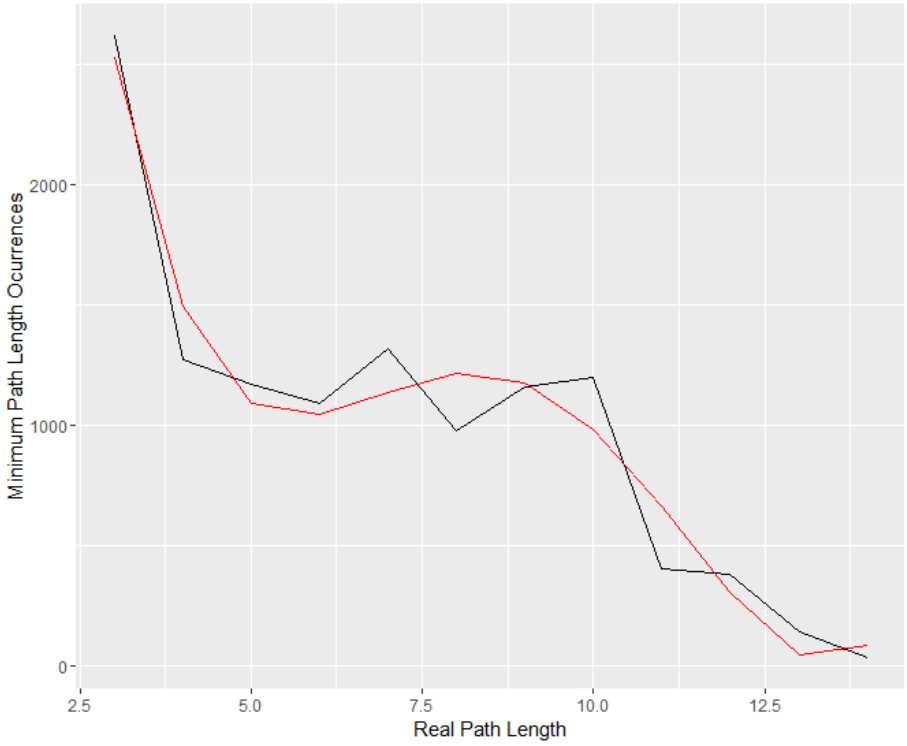


Figure 11: Fitted probability distribution function (red) of the paths with a minimum graph length of 4.

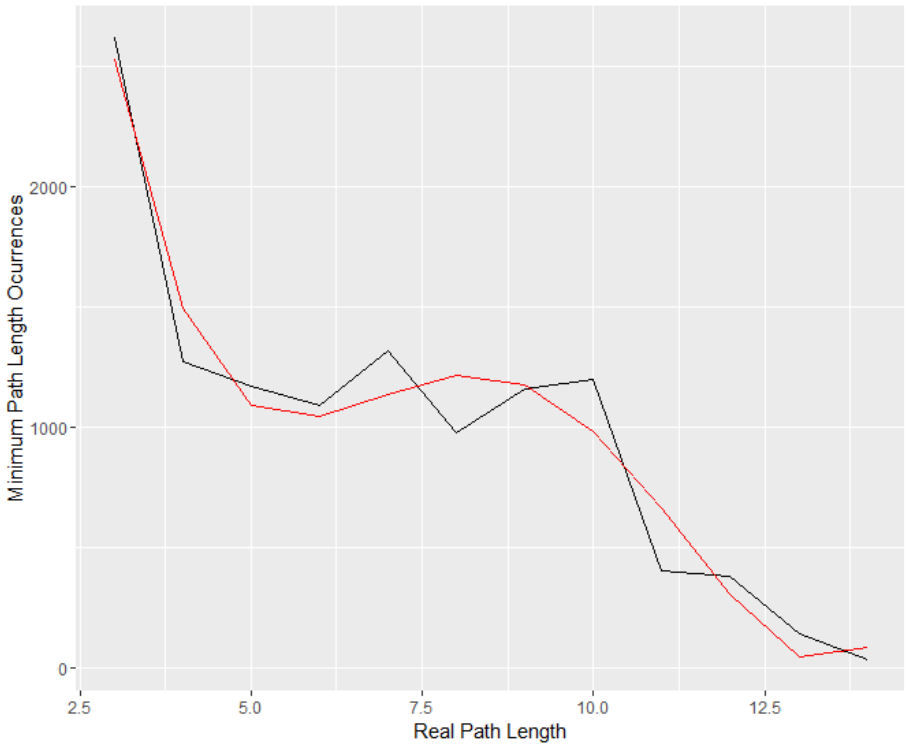


Figure 12: Fitted probability distribution function (red) of the paths with a minimum graph length of 5.

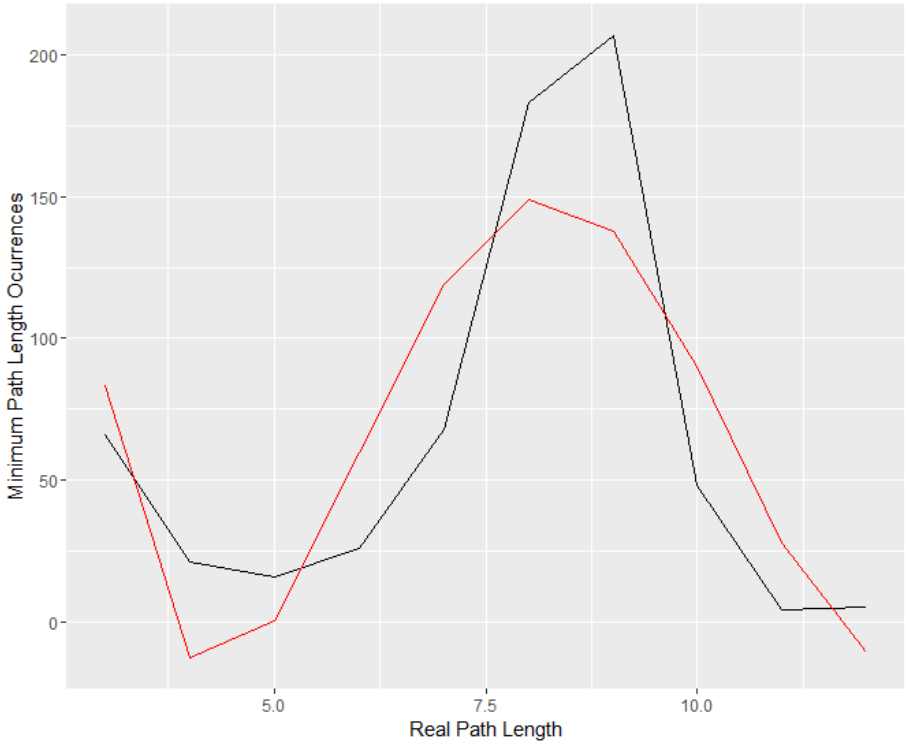


Figure 13: Fitted probability distribution function (red) of the paths with a minimum graph length of 6.

As we had expected, the bigger the minimum path the bigger the known path. However, no additional patterns that could be used to make a proper prediction were found.

7.2.5 Approach 5: Path Position Analysis

The approach that finally worked was to look at the role in the path each protein individually had. The starting idea was to divide each path in three sections: Start, mid and end. Codifying each past position as a number from 0 to (size-1) we defined a two breaking points. The first one marked the start of the mid section and the second one marked the start of the end section. This two points formulas were:

$$\text{Breaking Point 1} = \lfloor pathSize * \tfrac{1}{3} \rfloor$$

$$\text{Breaking Point 2} = \lceil pathSize * \tfrac{2}{3} \rceil$$

After counting the occurrences of each protein in each of this sections we created the following heat map (Using the standard score, also known as z score) to see any emerging pattern.

$$z = \frac{x - \mu}{\sigma}$$

Where μ represents the mean and σ the standard deviation.

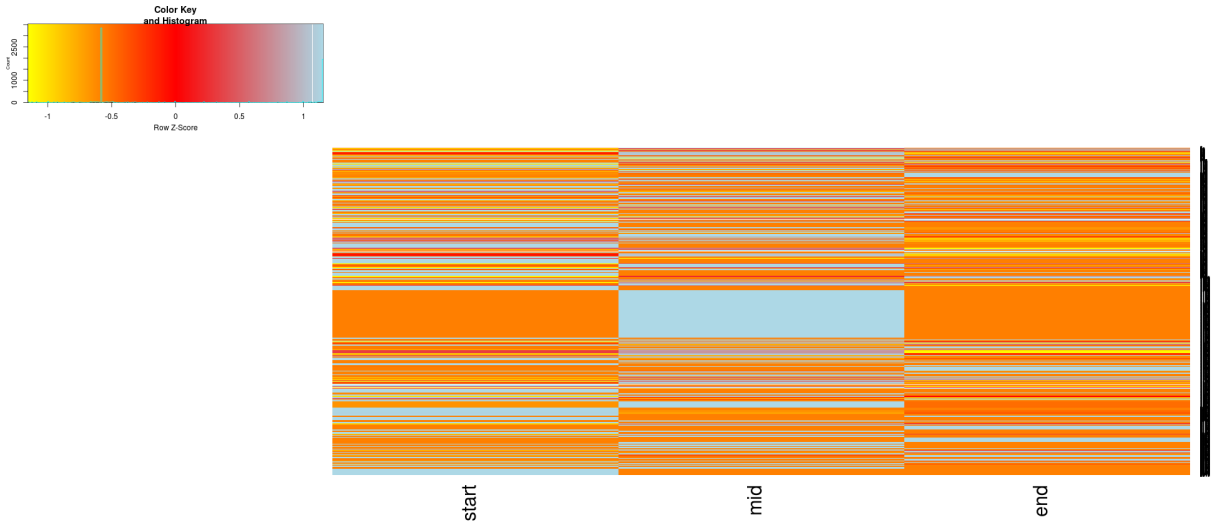


Figure 14: Heat map of the start, mid and end

Even though it may be hard to see in a heat map of thousands of rows, all the rows have a light blue section. Analysing all the path section distribution we discovered that the all the proteins appeared at most in two of the three sections.

After discovering this promising pattern we started analysing the path of lengths 4,5 and 6 (The ones we had enough observations of) separately. Fortunately, 80% of the proteins accumulated more than 95% of its appearances in one position within the paths of the same length, and the other 20% accumulated more than 95% of its appearances in two position within the paths of the same length. This pattern could be exploited to estimate a candidate path likely-hood.

Before proceeding to the prediction algorithm, we decide to discover if we could reliably make inferences between paths of different length. In order to do so, for each path length we grouped the proteins in different clusters depending on which position or combination of positions they appeared (At most 2 in all the cases).

Using the Jaccard Coefficient ($J(A, B) = |A \cap B|/|A \cup B|$) we compared the different clusters of the different path lengths to find possible correlations.

After comparing all of the clusters, we found interesting behaviours. When a protein appeared more than 95% of the time in a position x within the paths of length y, in the case that this same protein also appeared in the paths of length y+1 it would appear in the same position or next to it [x-1,x,x +1] more than 90% of the time (For instance: If the protein occupied the first position in a path of length 3 the same protein would most likely (In more than 90% of the cases) occupy the first and second positions in the length 4 paths). Using this behaviour, we could use the path of close length [x-1,x,x +1] in order to estimate a path likely-hood

Also, some correlation was found between the clusters of proteins that involved more than one single position between the paths of different length, albeit the correlation was considered too low to make reliable predictions.

7.2.6 Technology Involved in the Task

All the code was written in a Linux environment. The code was written in C++ (together with the openMP library to enhance the performance of complex algorithms) using the Sublime Text 3 Text Editor and several Rscripts (together with data visualization packages like ggplot, ggplot2 and LSD) were written using the Rstudio IDE.

Additionally, we used the graph visualization tool Gephi in the third approach. We also tried some efficient and scalable graph libraries (Like the C++ Boost Graph Library) but the already implemented graph class was doing the job fine enough and the additional work of rewriting half the code didn't seem justified.

7.2.7 Task Results

After trying several approaches, we finally found a pattern we could use to make a prediction.

7.3 Prediction Algorithm Design and Implementation

Taking into account the nature of the position patterns we decided that the best option was a fitness prediction algorithm. The algorithm consisted of the following steps:

Step 1 Specify a starting and an ending protein, the maximum amount of path candidates you want to find, and the minimum and a maximum length permitted.

Step 2 Generate paths that go from the starting protein to the ending protein within the length constrains.

Step 3 Calculate for each path a fitness measure and order them from higher to lower fitness score.

Step 4 Return the paths with the higher fitness (At most the maximum amount of path candidates specified in the first step).

The path fitness is measured comparing the positions the proteins take in the candidate path with the positions the same proteins take in all the path (When the paths have different length, the smaller path positions are compared with more than one position of the bigger one). The more the coincidences, the higher the score. In the next page there is a pseudo-code function that does the comparison between a single candidate path and a single known path and returns this comparison score[0,10]. This comparison is made between each candidate path and each known path individually and then all the scores are added together for each different candidate.

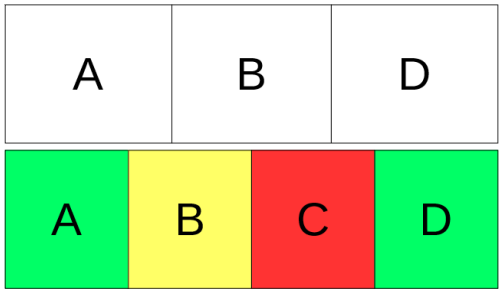


Figure 15: Example of path comparison scoring. A and D have a 100% match, while B has a %67 match. The total score result would be of (1.0+1.0+0.67)/4.0*10.0

```
double compare_paths(const vector<int> & candidatePath, const vector<int> & realPath)
{
        score = 0.0;
        i = 1;
        j = 0;
        divSizeA = 1.0/candidatePath.size();
        divSizeB = 1.0/realPath.size();

        while(i < candidatePath.size() or j < realPath.size())
        {
                if(i*divSizeA >= j*divSizeB)
                {
                        if(isPresent(candidatePath[i-1],realPath[j]))
                        {
                                if(i*divSizeA >= (j+1)*divSizeB)
                                        score += 1;
                                else if(candidatePath[i-1] == realPath[j]))
                                        score += (i*divSizeA - j*divSizeB)/divSizeB;
                        }
                        ++j;

                }
                else
                {
                        if(isPresent(candidatePath[i],realPath[j-1]))
                        {
                                if((i+1)*divSizeA >= j*divSizeB)
                                        score += (j*divSizeB - i*divSizeA)/divSizeB;
                                else if(candidatePath[i] == realPath[j])
                                        score += ((i+1)*divSizeA - i*divSizeA)/divSizeB;
                        }
                        ++i;

                }
        }
        score= score*10 / pathB.size();
        return score;
}
```

7.3.1 Technology Involved in the Task

All the code was written in a Linux environment. The code was written in C++ (together with the openMP library to enhance the performance of complex algorithms) using the Sublime Text 3 Text Editor and several Rscripts(together with data visualization packages like ggplot, ggplot2 and LSD) were written using the Rstudio IDE.

7.3.2 Task Results

The prediction algorithm was designed, implemented and ready to be tested.

7.4 Prediction Adjustment

The preliminary task to adjustment is to measure the precision of the current algorithm. The method used was cross-validation of the path data set. The result of cross-validating (with a train/test proportion of 90%/10%) 20 times can be seen in the next figure, where the average proportion of paths located in each different threshold can be found (Ex: The 80% of the tests paths were found in the top 25% of the list of candidate paths (output of the algorithm)).

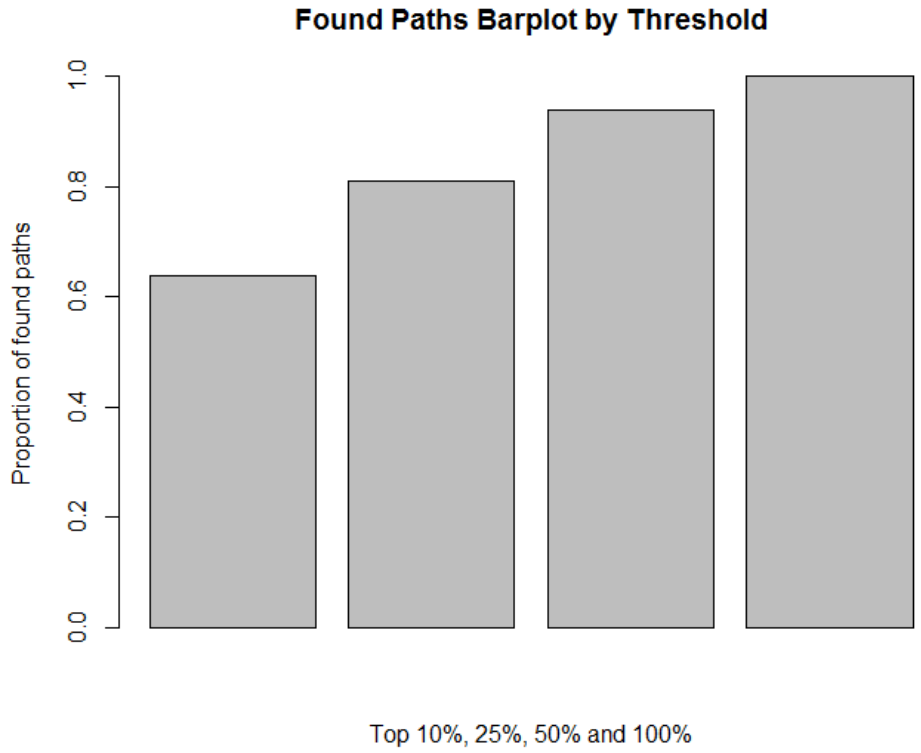


Figure 16: Heat map of the start, mid and end

The algorithm was tested with changes on the fitness formula. However, no modification clearly improved the original algorithm accuracy, while some severely hindered it. Since current results were good already, we proceeded to the final task.

7.4.1 Technology Involved in the Task

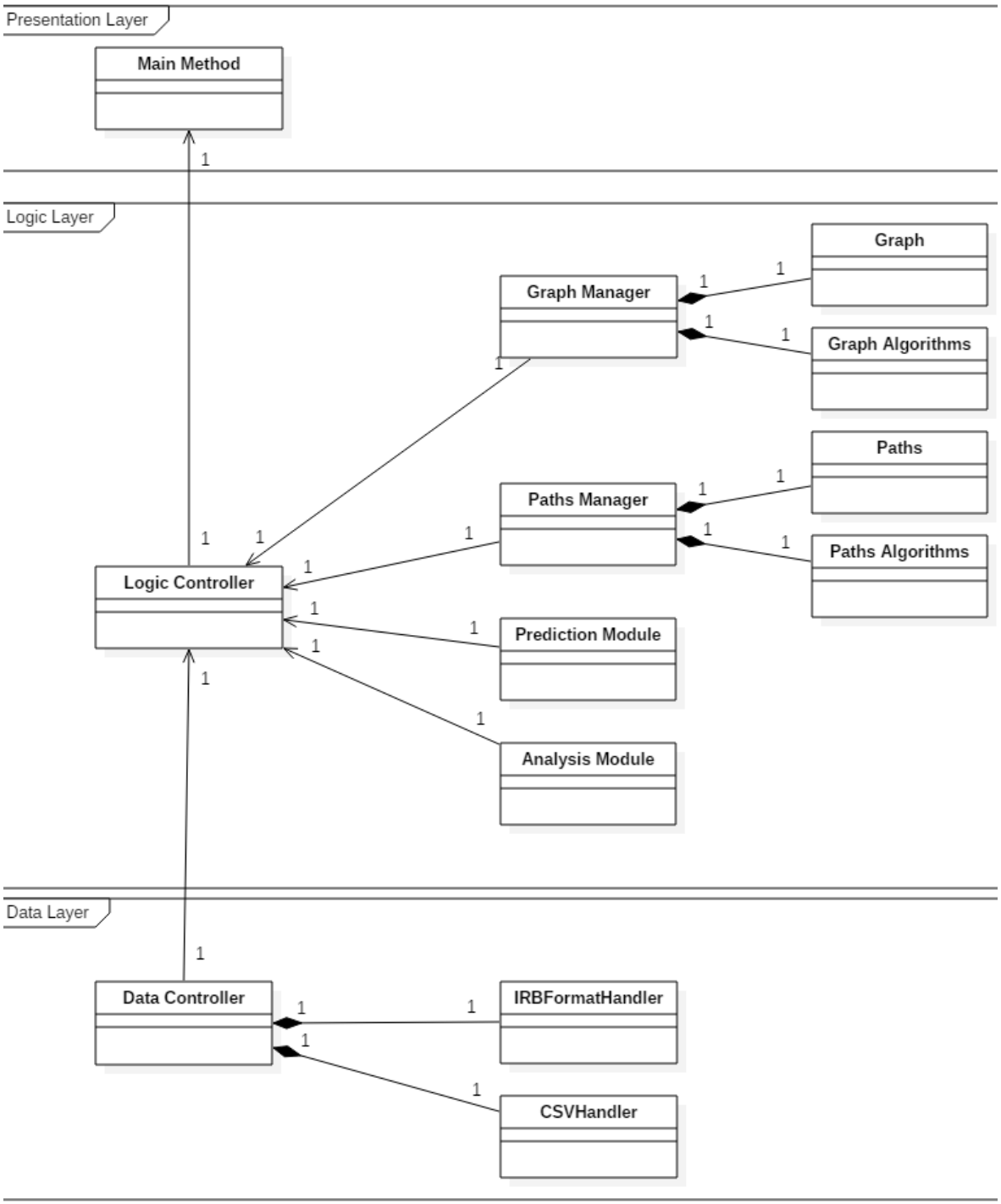
All the code was written in a Linux environment. The code was written in C++ (together with the openMP library to enhance the performance of complex algorithms) with the Sublime Text 3 Text Editor and several Rscripts written with the Rstudio IDE.

7.4.2 Task Results

The prediction algorithm was validated with good results.

7.5 Final Stage Task

The first step of the final task was to develop the predictor program. It's architecture is based on the Model–view–controller software architecture pattern. The following diagram shows the architecture of the predictor software:



Predictor Software Classes Description:

Main

This method that acts as the presentation layer controller.

Logic Controller

This class that handles the logic of the program.

Graph Manager

This class that handles all the logic that makes use of the Graph and GraphAlgorithms classes.

Graph

This class implements an undirected weightless graph data structure that uses a vector of vectors as a base data structure.

GraphAlgorithms

This class that is collection of graph data mining algorithms.

Paths Manager

This class that handles all the logic that makes use of the Paths and PathsAlgorithms classes.

Paths

This class that stores and manages the paths.

PathsAlgorithms

This class that is collection of Paths data mining algorithms.

AnalysisModule

This class that is a collection of analysis algorithms.

PredictionModule

This class handles the prediction logic.

Translator

This class that handles the translation from the system identifiers to the protein names and vice versa.

DataController

This class that handles the data layer.

IRBFormatHandler

This class that handles the data with the IRB format.

CSVHandler

This class that handles the data with CSV format.

Final Details

Once the predictor software model was built the only things left to do were to write down this memory and to upload the code to a git repository (The direction of this public repository can be found in the next section, "Code").

7.5.1 Technology Involved in the Task

All the code was written in a Linux environment. The code was written in C++ (together with the openMP library to enhance the performance of complex algorithms). The sharelatex web latex editor and the github git hosting services were used to write down the project memory and enable acces to the project's code. Also, the starUML software was used to generate all the UML diagrams.

7.5.2 Task Results

The prediction software and the memory were finished.

8 Code

All the code of the predictor software, most of the Rscripts and tests batteries for the critical classes are present (After 19th October 2016) on the following repository:

github.com/Bernat417/TFG

The legacy code (Different Dropped Data Exploration Approaches) is not present on the repository. During the project, there was a lot of uncertainty about the direction the prediction algorithm would take. For that reason the code left out doesn't fully fit into the quality levels set at the start of the project (Right architecture or properly documented). In case that someone wants to revise the legacy code, contact me and I will provide it. The project data sets can't be uploaded to a public repository for legal reasons.

9 Conclusion

Even though the pattern that enabled us to make the prediction took lots of different approaches and work to find, the positive ending makes it all worth it. What is more, every last approach taken helped me learn a wide variety of techniques that might prove useful in the future. For these reasons, I can sincerely affirm that this project has been a great first experience in graph data mining.

It is worth noting that the resulting predictor software can and should be improved. More patterns can be found, especially with more data. There are still many approaches left to try. All the effort put into documenting and defining a maintainable architecture was made to help anyone who tries to improve this prediction system in the future

10 Bibliography

Websites:

- Pathway Commons − Search and visualize public biological pathway information. Single point of access. URL: *http* : *//pathwaycommons.org*. Visited on April 1, 2014.
- Panther CLassification System. URL: *http* : *//pantherdb.org*. Visited on April 1, 2014.
- IRB − Institute for Research in Biomedicine website. URL: *http* : *//www.irbbarcelona.org/es*. Visited on April 1, 2014.
- Wikipedia. Artificial Intelligence − Wikipedia, The Free Encyclopedia. Version: 19:47, 5 April 2016. URL: *https* : *//en.wikipedia.org/wiki/Artificial_intelligence*. Visited on April 1, 2014.
- Wikipedia. Data Mining − Wikipedia, The Free Encyclopedia. Version: 19:41, 5 April 2016. URL: *https* : *//en.wikipedia.org/wiki/Data_mining*.
- Wikipedia. Structured Mining − Wikipedia, The Free Encyclopedia. Version: 13:41, 27 February 2015. URL: *https* : *//en.wikipedia.org/wiki/Structure_mining*. Visited on April 1, 2014.
- Wikipedia. Medicine − Wikipedia, The Free Encyclopedia. Version: 14:03, 26 March 2016. URL: *https* : *//en.wikipedia.org/wiki/Medicine*. Visited on April 1, 2014.
- Wikipedia. Biomedicine − Wikipedia, The Free Encyclopedia. Version: 19:18, 5 March 2016. URL: *https* : *//en.wikipedia.org/wiki/Biomedicine* . Visited on April 1, 2014.
- Wikipedia. Bioinformatics − Wikipedia, The Free Encyclopedia. Version: 02:23, 6 April 2016. URL: *https* : *//en.wikipedia.org/wiki/Bioinformatics* . Visited on April 1, 2014.

Books:

- Artificial Intelligence: A Modern Approach. Third Edition − 2009. ISBN: 978-0136042594. Author: Stuart Russell and Peter Norvig.

- Managing and Mining Graph Data. First Edition − 2010. ISBN 978-1441960443 Author: Aggarwal, Charu C., Wang, Haixun.

Papers:

- Clustering and Summarizing Protein-Protein Interaction Networks: A Survey. Author: Sourav S. Bhowmick. Nanyang Technological University, 2015..
- A Survey of Frequent Subgraph Mining Algorithms. Author: Chuntao Jiang, Frans Coenen and Michele Zito. University of Liverpool, 2004.