Universitat Politècnica de Catalunya

Bachelors in Computer Science and Engineering

Graph and matrix algorithms for visualizing high dimensional data

Director:
Dr. Ricard Gavalda Mestre

Co-Director:
Dr. Marta Arias Vicente

Bachelors Thesis of :
Abhinav Shankaranarayanan Venkataraman

June 20,2016


UPC

To my Mother, Father, Professors and Friends. I owe a lot to My professors Ricard Gavalda and Marta Arias and to Babaji at Gurudwara

Abstract

Motivated by the problem of understanding data from the medical domain, we consider algorithms for visually representing highly dimensional data so that "similar" entities appear close together. We will study, implement and compare several algorithms based on graph and on matrix representation of the data. The first kind are known as "community detection" algorithms, the second kind as "clustering" algorithms. The implementations should be robust, scalable, and provide a visually appealing representation of the main structures in the data.

Acknowledgement

I would like to Acknowledge the support provided by my faculty and admins at my home university – SASTRA University, Thanjavur and UPC Barcelona for supporting me throughout the project.

Contents

Chapter 1

Introduction

In this section an entire overview of the full project is provided. We mention the context of the project we have studied and the goal of the project. We also provide the intended planning, economic estimate and sustainability of the work that has been done.

1.1 Context Of the Project

In the present day scenario, the modern science of algorithms and graph theory has brought significant advances to our understanding of complex data. Many complex systems are representable in the form of graphs. Graphs have time and again been used to represent real world networks. One of the most pertinent feature of graphs representing real system is community structures or otherwise known as clusters. Community can be defined as the organization of vertices in groups or clusters, with many edges joining the vertices of the same cluster and comparatively fewer vertices joining the vertices in another neighbouring cluster. Such communities form an independent compartment of a graph exhibiting similar role. Thus, community detection is the key for understanding the structure of complex graphs, and ultimately deduce information from them.

The networks and highly dimensional data that motivate this problem emerge from the healthcare domain, and particularly from the analysis of complex, chronic disease, which is the major cost factor in modern societies. In the current scenario, a patient does not have one disease but a set of diseases. For example a person with diabetes has a heart disease, kidney

disease, high blood pressure etc. This may vary between sexes, ages etc and thus is a very complex landscape to explore. Visualizing this landscape of diseases would help to analyse the source, the treatment and even the path way of research to done. Thus, such a visualization would be helpful for the medical experts and health planner to understand the landscape of diseases much better.

Within the perspective of the LARCA project, two kinds of networks could be useful to study in this scneario: one in which nodes are patients and edges indicate their similarity, and another one in which nodes are diagnostics/diseases, and edges indicate their association in a population. Hence, we address this visualization of such high dimensional data using the algorithms and visualization technologies. LARCA has been involved in healthcare project by LARCA, a publication of the same has been made in the DSAA conference [16] . A few solutions that are possible to resolve the problem will be analysed and tests will be conducted. The project will also involve study of various algorithms and their respective analysis based on the quality and quantity of data using multiple appropriate experiments. Although, the project is motivated by the real high dimensional data it is not easy to get such data and hence would use simpler ones for testing the project.

1.2 Goal of the Project

The project is built with due recommendations from the director of the project. The project is aimed at using medical domain and thus slides to the side of implemenation which involves faster computation for better visualization. Hence, there are four facets or goals for the project which are enumerated as below:

1. The first objective of the project is to survey a few algorithms that aim in community finding keeping in mind that the input is from the medical domain

2. Next, to choose two algorithms that benefit the purpose of organizing graphs from medical domain and for the purpose of visualization.

3. Implement the algorithms and test the efficiency of the algorithm using variety of graphs.

4. Lastly but more importantly to build a Graphic User Interface (GUI) which enables visualization of the raw input on a web browser by drawing graphs.

1.3 Planning

Planning is essential component of any project. It helps to keep pace with the time. The total duration of the project is 5 months starting from early February 2016 to the end of June 2016. The following describes the tasks that were planned to be performed in the project.

1.3.0.1 Task Description

The tasks for the project have been subdivided into various task phases which are enumerated below :

- **Required knowledge acquisition**
  Necessary knowledge to understand the problem needs to be gained in order to deal with the original topic.In this phase we familiarize with the term of community detection, graph theory and understand all the possible methods that are in practice to deal with the problem. Knowledge about a few visualization methods is also necessary to implement the visualization of the project.

- **Paper Analysis**
  Analysis of paper related to community detection and clustering algorithms over high dimensional graph data is done in this phase of the project. This phase is necessary to understand various functionalities that the project deals with and to assist in the subsequent phases of the project.

- **Design and Implementation**
  The required functionalities are listed and implemented using a programming language. In this phase the methods of the project are designed and programmed using the choose language. The implementation is done for both the community detection algorithm and for the visualization aspect of the project.

- **Testing I**
  In this phase the program is tested using generated test cases and errors are identified and corrected. Multiple recoding is done in this phase of the project In this phase we test the program in order to identify errors in the implementation. It includes the successive recoding.

- **Testing II**
  In this phase we perform tests are performed on the GUI to ensure the limits of GUI.

- **Report Writing**
  In this phase the report of the project is written.

1.4 Economic Budget

1.4.1 An Introduction to Economic Budget

Economic management is primarily based on an estimate of income and expenditure called as budget. Development of a sustainable budget leads to proper economic management of the project. Budget and sustainability is one of the most important phase of the project management. In this phase we analyse the budget for the project. We also aim at providing an estimate of the project budget and optimize the same. We look at the expenditure from various aspects such as software costs, hardware costs, license costs and human resource costs. Additionally we also account the software for its sustainability. One important factor to note is that the budget that we describe in this section is subject to change and it may increase depending on the unexpected obstacles that we may face. For an instance when we don't get the expected results with a particular software we may have to go in for another software that may incur extra installation and operational charges.

1.4.2 Estimation of Economic Budget

We divide the overall expenditure into three categories namely hardware, software and human resources. One very important factor that we need to consider is that we only get an estimate of the total cost. This may vary depending on the systems in use. To calculate the amortization we consider to factors namely, first the overall life of the hardware or software in use.

Second that the project is completed in 5 months. Hence the amortization cost comes one eighth of the actual life of the component.

1.4.2.1 Hardware Budget

Hardware budget accounts for the actual and the amortized costs of the hardware elements used by the project. The cost is fictitious as it has not been developed commercially. Table 1. intents to estimate the economic cost of each of the hardware component of the project.

Table 1 - Hardware Budget				
Sno:	Hardware Component	Useful Life(in years)	Total Cost(in €)	Amortized Cost(in €)
1	PC System	4	1000€	125 €
	Total		**1000€**	**125 €**

1.4.2.2 Software Budget

The software budget shows an estimate for the various software used in the project along with the estimate of the software costs. It is a myth that the software doesn't get old with time just as a software gets but it wears out with time. Thus for every software there is a fixed time during which it gives maximum performance. In addition freeware software and open source software incur no cost. The cost is fictitious as it has not been developed commercially. Table 2 intents to estimate the economic cost of each of the software component of the project.

Table 2 - Software Budget				
Sno:	Software Component	Useful Life(in years)	Total Cost(in €)	Amortized Cost(in €)
1	Linux OS	5	0€	0 €
2	JavaScript Engine	1	0€	0 €
3	Python Components	1	0€	0 €
4	Web.py	1	0€	0 €
5	TexMaker	1	0€	0 €
	Total		**0€**	**0 €**

1.4.2.3 Human Resource Budget

The human resource budget deals with the overall expenditure spent on human resources. Every phase of the project has a cost associated with it in per hour calculation. The cost is fictitious as it has not been developed commercially. Table 3 intents to estimate the economic cost of each of the phases of the project. The cost per hour is intended as an approximation of the current cost per work hour of young analysts and developers in our environment.

Table 3 - Human Resource Budget					
Sno:	Phase	Deadline	Hours	Cost(per hour in €)	Total(in €)
1	Required Knowledge Acquisition	1 Mar 2016	70	15€/h	1050 €
2	Paper Analysis	1 Apr 2016	150	15€/h	2250 €
3	Design and Implementation	30 Apr 2016	230	20€/h	4600 €
4	Testing I	15 May 2016	75	15€/h	0 1125€
5	Testing II	31 May 2016	75	15€/h	0 1125€
6	Report Writing	15 Jun 2016	100	15€/h	1500€
	Total		**600**		**10525 €**

1.4.2.4 Total Budget

The following table, Table 4, summarizes the total budget for the project. This encompasses the hardware, software and human resources budget.

Table 4 - Total Budget		
Sno:	Resource	Total Cost(in €)
1	Hardware Budget	1000 €
2	Software Budget	0 €
3	Software Budget	10525 €
	Total	**11525 €**

1.5 Sustainability

Sustainability is a key factor in any project design. We evaluate the project based on three factors of sustainability namely economic sustainability, social sustainability and environmental sustainability.

1.5.1 Economic Sustainability

In this document we specify the budget estimation of the project. From our estimation it can be said that this will be the maximum bound on the budget for the project. This takes into account all the factors namely the hardware costs, software costs and human resource costs. The cost estimated in the project is the least possible cost and hence is a nonpareil project estimate for any indistinguishable project. The budget may exceed our calculations only during unexpected times. When the proposed plan is precisely followed the estimated lower costs gets achieved. Also the product that we aim at developing here is tested with all kinds of data and we aim at building a very high quality software which in turn provides a durable software that will not wear out easily. Most of the software used in the project is open source which has zero product cost. The The hardware required is nothing but computers that becomes a mandatory part of any project in the present days.

1.5.2 Social Sustainability

The project aims at developing web based platform to perform learning cum visualization analytics. This is indirectly going to analyze the learning char-

acteristics of the patients and provide a feedback both to the medical analyzer and health planner. This is going to improve the quality of health analysis in the state. All this requires is a simple computer connected to the internet. Thus this has a great social responsibility. This is in turn justifies why this project has a great social sustainability.

1.5.3 Environmental Sustainability

From the sections of temporal planning and the budget planning we understand that we have a computer running throughout the project. If we make an assumption that the amount of energy used by a single computer comes to around 250 watts. And given that we spend 500 hours on the project then the energy expended is 125KW. On average, electricity sources emit 1.222lbs CO2 per kWh (0.0005925 metric tons = 0.53750695845 Kg of CO2 per kWh). (Source: EPA eGRID Summary Tables and Data Files). This amounts to a upper bound of 67.1884 kg of CO_2 considering that the energy was produced by using coal lignite. This can be reduced by reducing the code size which is possible by reusing the already existing code. But the project is actually environmentally sustainable.

Chapter 2

Background Knowledge

In this section we present the background knowledge required to understand and solve the problem

2.1 Graph Notion

Many real-world probelms can be solved by describing it by means of a diagram that consists of a set of points in which a few or all the pairs of points are joined together by lines. It is interesting to find whether any two given points are joined by lines or not. A mathematical abstraction of this situation is termed as graphs [4]. In the project of concern where we deal with representing the medical data in this manner it becomes necessary to talk about graphs.

2.2 Graph Definition

A Graph G is formed by two finite sets, the set $V = \{ v_1, v_2, \ldots, v_n \}$ of vertices and the set $E = \{ e_1, e_2, \ldots, e_n \}$ of edges where each edge is a pair of vertices from V, for instance,

$$e_i = (v_j, v_k)$$

is an edge from v_j to v_k represented as $G{=}(V{,}E)$. In other words E $\subset V^2$, which is the set of all unordered edges. The vertices (v_j and v_k) that represent an edge are called *endpoints* and the edge is said to be adjacent to each of its end points.

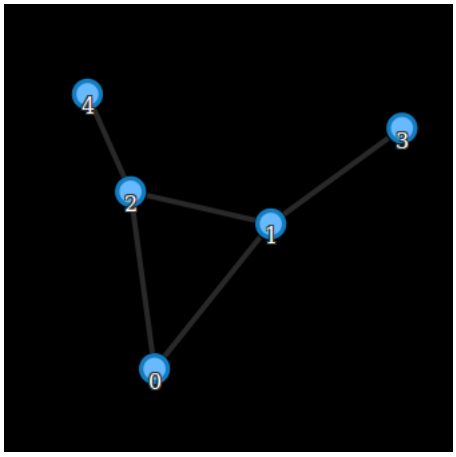


Figure 2.1: Example Graph G=(V,E) Bull Graph (Created using the project)

The neighbourhood of a node v_i is the set of nodes v_i is connected to, $\mathrm{N}(v_i) = \{v_j | (v_i, v_j) \in E, v_i \neq v_j, 1 \leq j \leq n\}$. The degree of a node v_i, or the size of the neighbourhood connected to v_i, is denoted as $\mathrm{d}(v_i) = |N(v_i)|$.

A degree sequence, D, specifies the set of all node degrees as tuples, such that $\mathrm{D} = (v_i, d(v_i)$ and follows a probability distribution called the *degree distribution* with mean d_m [14].

2.3 Graph Matrix Notation

The matrix is commonly use dot represent grpahs for computer processing. The advantage of using matrix is usually that matrix algebra can be readily applied to study the structural property of matrix. There are number of ways in which one can represent the graph in it's matrix form for example, adjacency matrix and Laplacian matrix.
Let G=(V,E) be a simple graph with vertex set **V** and edge set **E**, then the adjacency matrix is square $|V|^2$ matrix **M** such that its element $M_{i,j}$ is *1* when there is an edge from v_i to v_j,where $v_i \in \mathbf{V}$, $v_j \in \mathbf{V}$ and *0* when there is no edge. The adjacency matrix of a graph of order n entitles the entire the topology of a graph. The diagonal elements of the adjacency matrix are all *0* for undirected graphs **M**.

The sum of the elements of i-th row or column yields the degree of node i. If the edges are weighted, one defines the weight matrix **W**, whose element

W_{ij} expresses the weight of the edges between vertices i and j.

The *spectrum* of a graph **G** is the set of eigenvalues of it's adjacency matrix **M**. If **D** is the diagonal matrix whose element $D_{i,i}$ equals its degree of vertex i ($v_i \in V$) [9].

2.4 Approaches

In this section we discuss the various approaches that are involved in dealing with the input to the project for community identification, for clustering and for visualization purposes.

2.4.1 For Community Identification

Virtually in every scientific field dealing with empirical data, primary approach to get a first impression on the data is by trying to identify groups having "similar" behaviour in data. There are numerous methods to achieve this objective of which

- Community Detection
- Clustering

2.4.1.1 Community Detection

2.4.1.2 Definition of a Community

Communities are a part of the graph that has fewer ties with the rest of the system. Community detection traditionally focuses on the graph structugmres while clustering algorithms focuses on node attributes.

Several tyoes of community detection algorithms can be distinguished

2.4.1.2.1 Divisive algorithms Divisive algorithms detect inter-community links and remove them from the network

2.4.1.2.2 Agglomerative algorithms Agglomerative algorithm merges similar nodes or communities in a recursive manner.

2.4.1.2.3 Optimization Methods Optimization methods are mainly based on maximization of an objective function.

2.4.1.3 Clustering

According to the paper "Community detection in graph" [10] there are 4 major traditional clustering methods namely :

- Graph Partitioning
- Hierarchical Clustering
- Partitional Clustering
- Spectral Clustering

2.4.1.3.1 Graph Partitioning This problem deals with dividing graph into groups of predefined size such that the number of edges between the groups is minimized. The paper [10] also defines cut size as the number of edges lying between the clusters. Figure 2.2 shows a problem with 14 vetices and presents a solution for spliting into 2 groups.

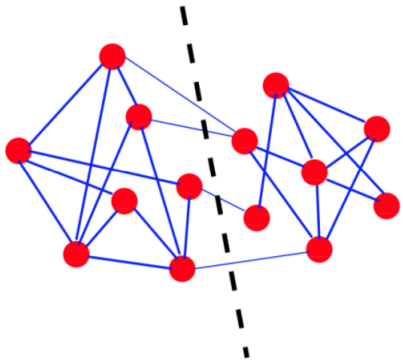


Figure 2.2: Graph Paritioning [10]

Minimum Bisection Problem, is a special problem case that considers partitioning the network into 2 groups of equal size. This problem is an NP-Hard problem. Intutively, to obtain ful partitioning we need to iteratively find all the minimum partition. This is not of significant use in the current problem of finding communities.

2.4.1.3.2 Hierarchical Clustering Hierarchical clustering aims to identify groups of vertices with high similarities. It can be calssifies into two categories:

1. *Agglomerative algorithm* : in one in which Agglomerative algorithms, in which clusters are iteratively merged if their similarity is sufficiently high

2. *Divisive algorithms*, in which clusters are iteratively split by removing edges connecting vertices with low similarity. The figure 2.3 demonstrates the hierarchcal clustering in a diagramatic manner.

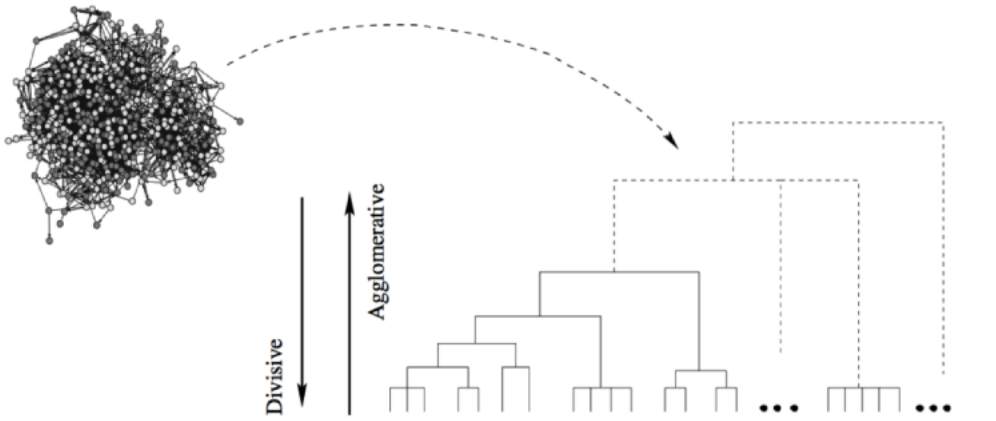


Figure 2.3: From a thickly knit graph to a dendogram [This intuitive diagram was taken from a powerpoint presentation for a data mining class]

2.4.1.3.3 Partitional Clustering Partitional clustering is method to find the clusters as a set of data points. The number of clusters is preassigned. let us call this number as k. The vertex is a point on the metric space with a defined distance measure between the pair of points in the space. The distance represents the difference in dissimilarity between the vertices. The main objective in this method is to separate the points in k clusters such to maximize ((or) minimize) a given cost function based on distances between points and from points to *centroids* that are suitably defined positions in space. Some of the most used functions are : *Minimum k-clustering* , *k-clustering sum*, *k-center* and *k-median*. One of the most popular partitional

technique in literature is *k-means clustering* where the cost function is total intra-cluster distance [10]. This method of clustering is out of scope for the project in concern.

2.4.1.3.4 Spectral Clustering According to the paper [10], Let us suppose to have a set of n objects $x_1, x_2, \ldots, x_n$ with a pairwise similarity function S defined between them, which is symmetric and non-negative (i. e., $\mathrm{S}(x_i, x_j) = \mathrm{S}(x_j, x_i) \geq 0$, $\forall$i,j $= 1, ..n$). Spectral clustering includes all methods and techniques that partition the set into clusters by using the eigenvectors of matrices, like S itself or other matrices derived from it. In particular, the objects could be points in some metric space, or the vertices of a graph. Spectral clustering consists of a transformation of the initial set of objects into a set of points in space, whose coordinates are elements of eigenvectors: the set of points is then clustered via standard techniques, like *k-means clustering*.

2.4.2 For Visualization

Graph visualization is a important task in various scientific application. Visualization of data as graphs Visualizing these data as graphs provides the non-experts with an intuitive means to explore the content of the data, identify interesting patterns, etc. Such operations require interactive visualizations (as opposed to a static image) in which graph elements are rendered as distinct visual objects; e.g., DOM objects in a web browser. This way, the user can manipulate the graph directly from the UI, e.g., click on a node or an edge to get additional information (metadata), highlight parts of the graph, etc. Given that graphs in many real-world scenarios are huge, the aforementioned visualizations pose significant technical challenges from a data management perspective [2].

2.5 Computational Complexity

The estimate of the amount of resources required for by the algorithm to perform a task is defined as computational complexity. The humongous amount of data on the real graphs or real networks that are available in the current scenario causes the efficiency of the clustering algorithm to be crucial.

In a brief, Algorithms that have polynomial complexity describe the Class **P**. Problems whose solutions can be verified in a polynomial time span the class **NP** of *non–deterministic polynomial time* problems, which includes **P**. problem is **NP**-hard if a solution for it can be translated into a solution for any **NP**-problem. However, a **NP**-hard problem needs not be in the class **NP**. If it does belong to **NP** it is called **NP**-complete. The class of **NP**-complete problems has drawn a special attention in computer science, as it includes many famous problems like the Travelling Salesman, Boolean Satisfiability (**SAT**), Integer Programming, etc. The fact that **NP** problems have a solution which is verifiable in polynomial time does not mean that NP problems have polynomial complexity, i. e., that they are in **P**. In fact, the question of whether **NP**=**P** is the most important open problem in theoretical computer science. **NP**-hard problems need not be in **NP** (in which case they would be **NP**-complete), but they are at least as hard as **NP**-complete problems, so they are unlikely to have polynomial complexity, although a proof of that is still missing. Reference to this has been imbibed from the paper "Community detection in graphs" [10].

Many clustering Algorithms or problems related to clustering are **NP**-hard. This makes it irrelavant to use the exact algorithm, in which case we use an approximation algorithm. Approximation algorithm are methods that do not deliver the exact solution but an approximate solution but with an advantage of lower complexity. [10]

2.6 State-of-the-art in Community Detection

Modularity is the objective function that is widely used both as a measure and as a optimizing method for partitioning community. As said before there are various algorithms that can be used for community detection . In reference to the paper [15] discusses six different community detection algorithms namely:

- Louvain Method
- Le Martelot
- Newman's greedy algorithm (NGA)
- Newman's spectral algorithm with refinement

- simulated annealing
- extremal optimization

The following figure 2.4 the average normalized performance rank of each algorithm in terms of partitioning quality and speed. Taken from the paper that proposed Combo algorithm [15]. The main objective of the project is to

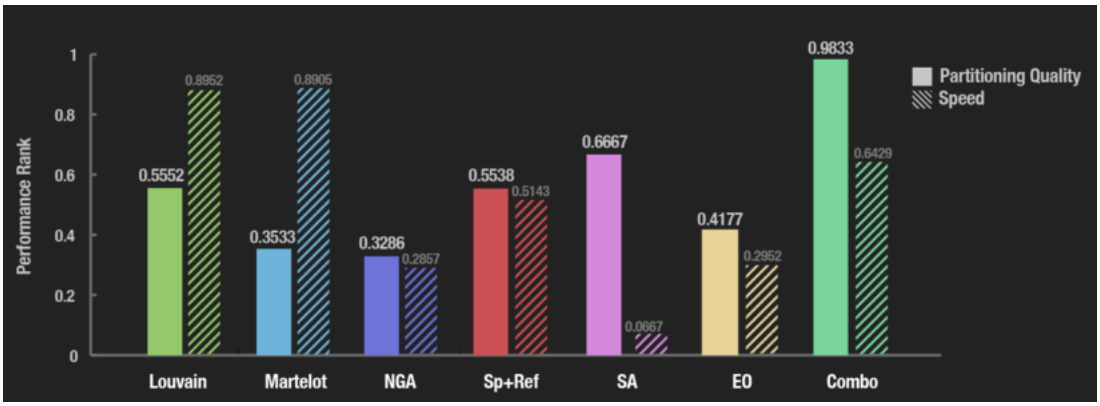


Figure 2.4: Average normalized performance rank of each algorithm in terms of partitioning quality and speed

visualize the data on screen thus needs an algorithm that is fast and should be effective. Hence louvain algorithm was choose for the implementation. The implementation can be found in the later section of the report.

Louvain algorithm algorithm is considered as state-of-the art algorithm for community detection [3]. The algorithm is fast,recursive and is more effective than the other algorithms in real-world graphs. Due to our goal of projecting medical domain we require an algorithm that gives a better trade off between being effective and being fast. Hence Louvain algorithm was chosen.

2.7 Degree Distribution

Degree of a node(v_i) in the graph $G = (V, E)$ where V is the set of vertices and E is the set of edges is the number of edges that a node has to other nodes. Usually denoted as $\deg(v_i)$. *Degree distribution* can be thus described as the probability distribution of there degrees over the entire graph. Degree

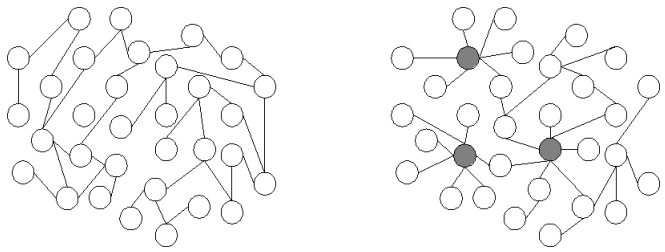
distribution is significant in the study of community networks and hence bringing it into consideration. It is usually denoted as $P(k)$ of a graph which is the fraction of nodes in the network with degree k.

$$P(k) = \frac{N_k}{N} \tag{2.1}$$

where N_k is the number of nodes with degree k and N is the total number of nodes in the graph.

2.7.1 Scale-Free Graph

The graphs whose degree distribution follows power law are called as Scale-Free graphs or scale free networks.
Examples of Scale-Free graphs include Social network graph,protein-protien interation network etc. According to the paper "Resilience of the Internet to Random Breakdowns" [8], removing randomly any fraction of nodes from scale free network will not destroy the network which is in contract to Erdos-Renyi graphs. The figure 2.5 demonstrates how the random graph is different from a scale free graph. The highlighted spots are knon as the hubs. For example: In a large community the celebrities or politicians serve as the hub. In scale free graphs another interesting feature is that as the clustering coefficient decreases with increase in the node degree. This type if distribution will be used in the experimental phase of the project for generating test cases.



(a) Random network (b) Scale-free network

Figure 2.5: Random network (a) and scale-free network (b). In the scale-free network, the larger hubs are highlighted.(Image Source: Wikipedia)

2.8 State-of-the-art in Graph Visualization

Researchers have designed various tool-kits for the purpose of information visualization. Newer visualization techniques are introduced by creating new components or sub-classing the existing ones. The main aim of the project will be analyse the way inwhich the visual frameworks can be used and what kind of intermediate trasitions that are possible between the python program and the visualization framework.

2.8.1 Protovis

[5]

2.8.2 D3

D3 unlike traditional [6]

2.8.3 Gephi

[1]
D3, Alchemy.js, plot.ly, Gephi and graphviz

Chapter 3

Louvain Community Detection Algorithm

In this section we describe the community detection algorithms such as Louvain and various tests that were performed to choose the algorithm.

3.1 Introduction

The problem of community detection requires the graph to be split into communities of tightly packed or in other words densely connected nodes with nodes of different community being sparely connected.

Several algorithms have been proposed for performing good partition in a reasonably good speed. Distinguishably there are several types of community detection algorithms, namely: divisive algorithms, which aims in removal of the inter-community links, agglomerative algorithms, which aim in merging similar nodes and optimization methods which aim in maximizing the objective function.

3.2 Modularity

The quality of partitioning that results from application of method is often measured using modularity. The *modularity* of a partition is hence a scalar value between -1 and 1 that is used to measure the density of the links inside the communities as compared to the density of the links between the communities.

Modularity not only serves as a quality measure for detecting the quality of split or -partition, but also acts as an objective function to optimize. Exact modularity optimization is **NP-Complete** in the strong sense [7].

3.2.1 Definition

Let G=(V,E) be a simple graph,where V is the set of vertices and E is the set of undirected edges. Let $n = |V|$ and $m = |E|$. Let degree of a vertex v be, deg(v) where $v \in$ V. Let C be the community, $C \subseteq V$, be the subset of vertices. A *clustering* $C_s = \{C_1, C_2, \ldots, C_k\}$ of G is a partition of V such each vertex is present exactly in one cluster. We thus define *modularity* as follows: [7]

$$Q(C_s) = \sum_{C \in C_s} \left[\frac{|E(C)|}{m} - \left(\frac{|E(C) + \sum_{k \in C_s} |E(C,k)|}{2m} \right)^2 \right] \tag{3.1}$$

where E(I,J) is set of all edges between vertices in cluster I and J. E(C) = E(C,C). The above equation can be continently rewritten as follows:

$$Q(C_s) = \sum_{C \in C_s} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} deg(v)}{2m} \right)^2 \right] \tag{3.2}$$

In simpler terms the value is of Q can be expressed as

$$Q = \text{(Number of Intra-Cluster Communities)} - \text{(Expected number of Edges)} \tag{3.3}$$

As given in [3]

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - P_{ij} \right) \delta(C_i, C_j) \tag{3.4}$$

$$\delta(C_i, C_j) = \begin{cases} 1, & if C_i = C_j \\ 0, & otherwise \end{cases} \tag{3.5}$$

where, P_{ij} is the expected number of edges between nodes v_i and v_j. P_ij is $\frac{k_i k_j}{2m}$ where k_x is sum of the weights of the edges attached to the vertex v_x for a given random graph G (This is otherwise called as a null model).

3.2.2 Properties of Modularity

1. Q depends on nodes in the same clusters only.
2. Larger modularity implies better Communities.
3. $$Q(C_s) \leq \frac{1}{2m} \sum_{ij} A_{ij} \delta(C_i, C_j) \leq \frac{1}{2m} \sum_{ij} A_{ij} \leq 1 \tag{3.6}$$
4. Value taken by Q can be negative

3.3 Louvain

Louvain algorithm is considered as the state-of-the art algorithm for community detection for identifying community structures [3]. Louvain method developped by Blondel *et al* [3] finds high modularity partitions of large networks in short time. It unfolds a complete hierarchy community detection.

3.4 Implementation of the Louvain Community detection Algorithm

The implementation of the algorihtm is based on the paper "Fast unfolding of communities in large networks" [3]. The implementation is done using basic python packages. The Algorithm has two phases that are repeated iteratively to bring the final solution to the problem. The following figure 3.1 demonstrates the algorithm in the form of a flow diagram,

Algorithm 1 Louvain Algorithm Pseudocode

Require: A graph G = (V,E)
Ensure: Local optimum community split has happened
  while *LocalOptimumReached* **do**
    Phase1 : Split or partition the graph by optimizing modularity greedily
    Phase2 : Agglomerate the found clusters into new nodes
  end while

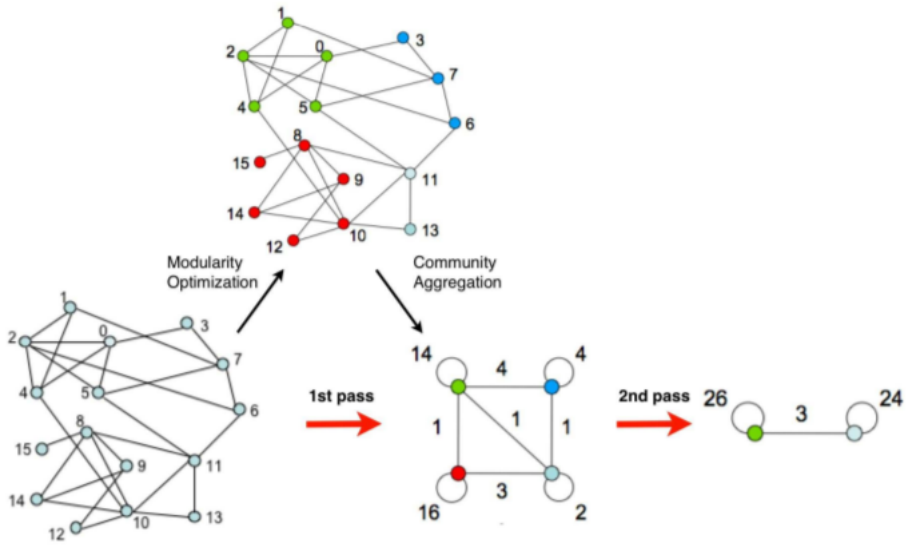


Figure 3.1: Visualization of the steps of our algorithm. Each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the found communities are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible. This was taken from the paper "Fast unfolding of communities in large networks" [3]

3.4.0.1 First Phase : Optimizing Modularity

The first phase of louvain algorithm Let G be a graph with N nodes in the network. The algorithm assigns a different community to each node in the network. The number of nodes is equal to the number of communities in the graph. The report uses the terms node and vertices interchangeably. Let v_i be be a node such that $v_j \in \mathrm{N}(v_i)$. The gain of modularity is then calculated by removing v_i and placing it in community of v_j. If the gain is positive the v_i is moved to the community of v_j else v_i stays in it's original community. This procedure is iterated and the phase one stops when a local maxima of the modularity is achieved, that is when no more move of nodes from one community to another is possible. The ordering of the nodes can affect or effect the computation time which can be a part of future works.

Algorithm 2 Phase 1 in Louvain Algorithm Pseudocode

Require: A graph G = (V,E)
Ensure: Partition network greedily using modularity
  Assign a different community to each node
  while *LocalOptimumReached* **do**
    for all Each node v_i **do**
      For each node $v_j \in \mathrm{N}(v_i)$, consider removing v_i from community of v_i and place it in the community of v_j
      Calculate the modularity gain
      if *ModularityGain* is Positive **then**
        remove v_i from community of v_i and place it in the community of v_j
      else
        No Change
      end if
    end for
  end while

The main algorithm relies on the calculation of modularity. Listing 3.1 demonstrated the calculation using a python snippet. The first phase of the algorithm has been written into a python code snippet and is presented in the Listing 3.2. In the paper it is stated that the gain in modularity as $\triangle$Q

$$\triangle Q = \left[\frac{\sum_{in}+k_{i,in}}{2m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{3.7}$$

where $\sum_{in}$ is the sum of the weights of the links inside C and $\sum_{tot}$ is the sum of the weights of the links incident to nodes in C, k_i is the sum of weights of the links incident to node i, $k_{i,in}$ is the sum of the weights of all the links in the network.

3.4.0.1.1 Second Phase : Agglomerating the communities found in first phase into new nodes In the second phase the algorithm builds the new network. The communities that are found during the first phase are now the nodes here. According to the paper [3], the weights of the links between the new nodes are given by the sum of the weight of the links

between nodes in the corresponding two communities. The edges between nodes of the same community lead to seld-loops for this community in the new network. The resulting new weighted network is then subjected to first phase and this process is iteratively done.

Algorithm 3 Phase 2 in Louvain Algorithm Pseudocode

Require: A graph G = (V,E)

Ensure: Agglomeration of nodes

Every community C_i forms a new node v_i

$W_{ij} = \sum$ {All edges between C_i and C_j} where W_{ij} is the edge between newly formed nodes v_i and v_j

3.4.0.2 Observations of Louvain

1. The final output of the Louvain algorithm forms a complete hierachical structure.

2. Resolution limit problem [11] has been resolved in the algorithm stated in the current paper under discussion [3] due to the multi-level nature of louvain algorithm.

3. Modularity can be redefined for weighted graphs and Louvain works well with weighted graphs.

3.4.0.3 Usage of Louvain in the project

In the project the above algorithm has been implemented in python taking inspiration and reusing some part of the pylouvain program API for implementation [12]. Python was chosen as the programming language as recommended by the project Director. The project work is said to form a part of a major project work on medical domain in LARCA. The director of the project informed that the main project is written in javascipt and python hence, python was choosen as the programming language for the project.

3.4.1 Experiments

[14]

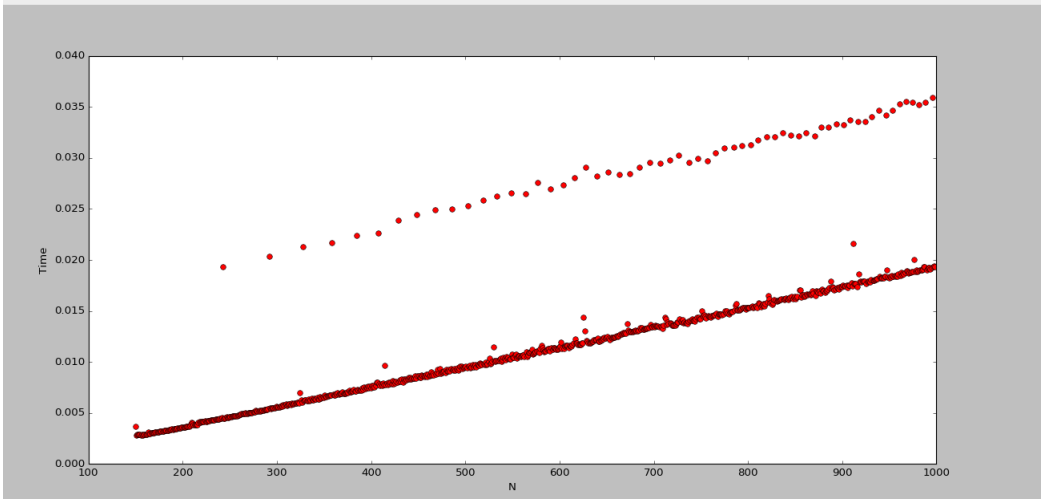


Figure 3.2: Example Graph G=(V,E) Bull Graph

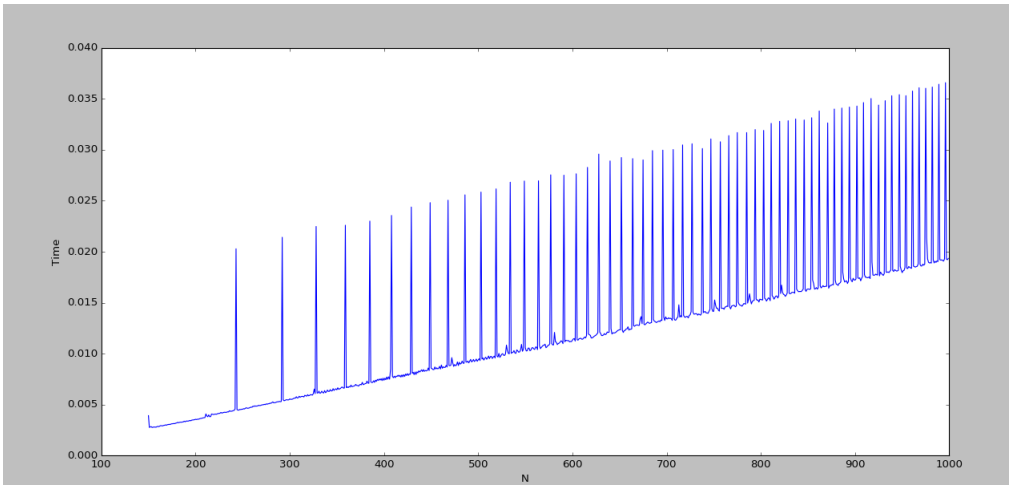


Figure 3.3: Example Graph G=(V,E) Bull Graph

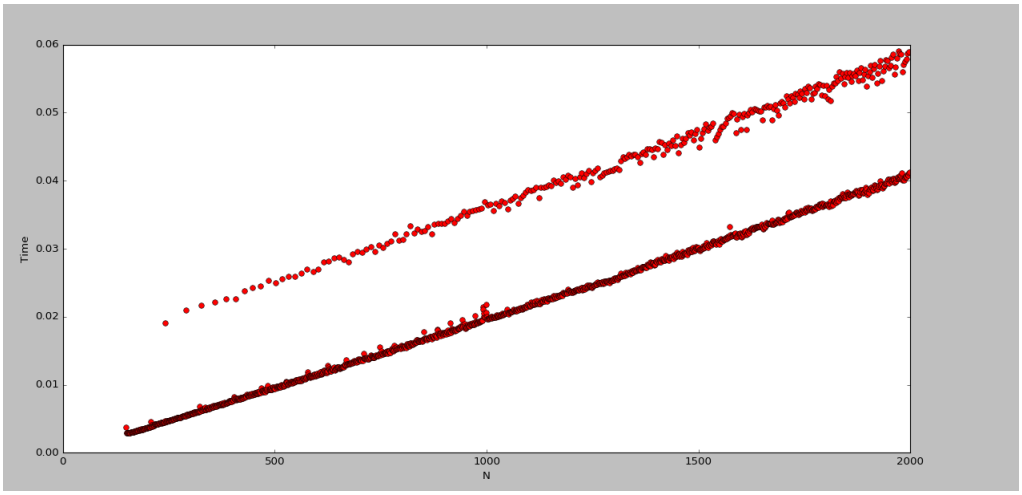


Figure 3.4: Example Graph G=(V,E) Bull Graph

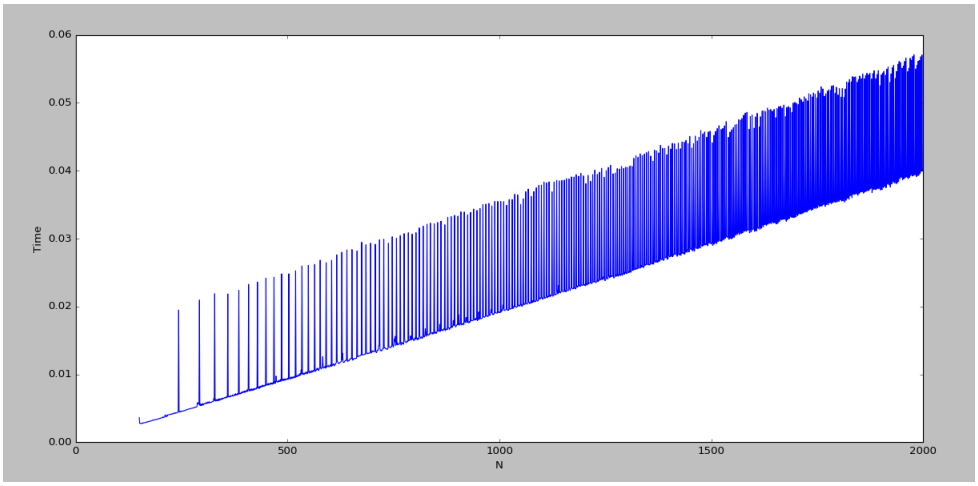


Figure 3.5: Example Graph G=(V,E) Bull Graph

3.4.1.1 Result

3.5 Matrix Based Algorithm

3.5.1 Matrix Algorithm

3.5.1.1 Introduction

3.5.1.2 Reasoning

3.5.1.3 Description

3.5.1.4 Implementation

3.5.1.5 Experiments

3.5.1.6 Result

Chapter 4

Visualization Module

For visual representation, softwares such as D3, Alchemy.js, plot.ly, Gephi and graphviz were tested using example graphs. Graphviz was only used for structural representation and was more useful for only small graphs. Plot.ly is most used for representation over charts and not graphs. Gephi is a tool used for data analysis for understanding and exploring graph based data however it has some drawbacks in the logistics of the project. Softwares such as HyperTree, HyperGraphs were also searched upon but they did not have a python API or was not based on JavaScipt but instead has a Java Code based APIs. I have surely overlooked some of the tools but most of the important tools pertaining to the project were tested in my system. D3 and Alchemy.js were the last ones left. D3 provided the necessary tools and was JavaScript and could be linked to python and was the best option for the project's logistics. Alchemy.js was built using D3. Alchemy.js required minimal code to generate the graphs as most of the customization could be done by just overriding or altering the "config" [Configuration part of the Alchemy.js] instead of implementing it entirely using JavaScript. Alchemy.js also provides a feature in which the core application can be further extended with any other feature of D3. Having D3 to be the base, with minimal code and maximum customization Alchemy.js was chosen for visual representation of the graph in this project. Thus the project director also considered alchemy.js as a visualization software as it fits the bigger project.

4.1 Alchemy.js

Alchemy.js is a graph drawing library built to proivde graph visualization with little overhead. It is built on the d3 library, written in Javascript, which runs in most web browsers.

4.2 Dependencies

Alchemy needs three main units to form as an application namely: *alchemy.css*, *alchemy.js* and *data*. CSS and JavaScript are major dependencies in Alchemy.js. Installation of *jQuery* and *d3* is also useful. *alchemy.min.js*, *alchemy.css* and *alchemy.min.css* will be updated in the CDN (Content Delivery netwrok)

4.3 Steps to use the Alchemy.js

In this project we have uploaded the *alchemy.min.js*, *alchemy.css* and *alchemy.min.css* into the project's file repository http://abhinavsv3.github.io/javascriptsal and hence will be using the link in the following explanation. The following describes the steps that are followed in use of Alchemy.js :

1. *Include the files in this format*
   <link rel="stylesheet" href="http://abhinavsv3.github.io/javascriptsal/alchemy.min.css"/>
   ...
   <script src="http://abhinavsv3.github.io/javascriptsal/alchemy.min1.js"><
   /*script* >

2. *Include an element with "alchemy" ID as the id and class*
   The alchemy class is used to apply styles while the alchemy id is used programatically. By default Alchemy.js looks for the alchemy div but this can be overridden.

3. *Provide Alchemy.js with a JSON dataSource*

4. *Begin Alchemy.js* <script> alchemy.begin({"*dataSource*" : *someData*})
   </script>

4.4 Getting the data from the Louvain Python code to Alchemy

Alchemy.js takes a simple data format called GraphJSON. GraphJSON serves as a light weight and flexible representation of graph data, easily consued locally or over the web.

GraphJSON is a JSON Object which has 2 object namely: *nodes* and *edges* . These are individual arrays that represent the nodes and edges that will be represented in the graph visualization.

Nodes : id key is the only unique value that should be present in the nodes

Edges : source and target key are the only unique value that should be present in the edges.

4.5 Tests

For the purpose of visualization various test were performed for analysising the compatiability of Alchemy.js and to explore and exploit all the functionalities of alchemy.js and choose the best ones for the project and implement them. The folowing will ennumerate the tests that were performed one after another. Alchemy uses a force layout of d3.

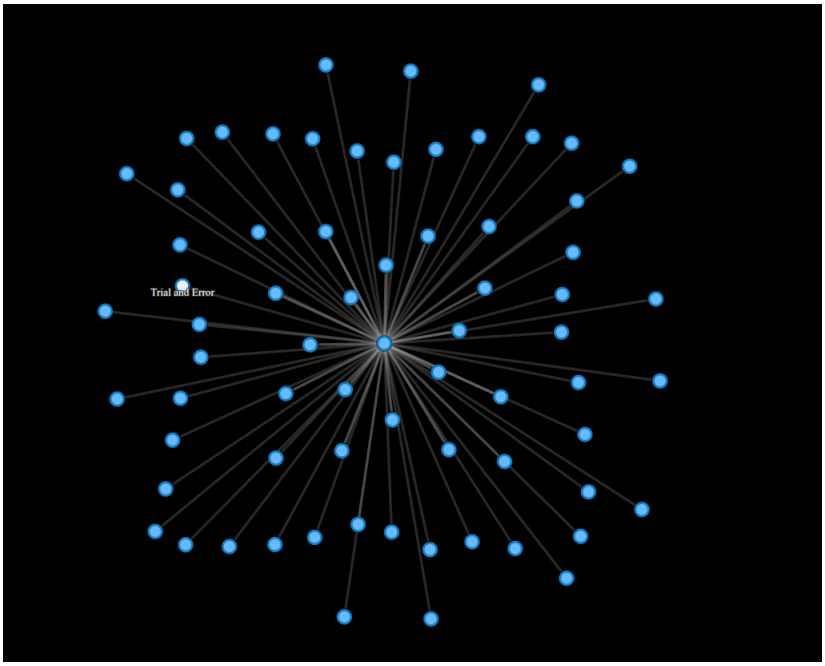


Figure 4.1: Example Graph G=(V,E) Bull Graph

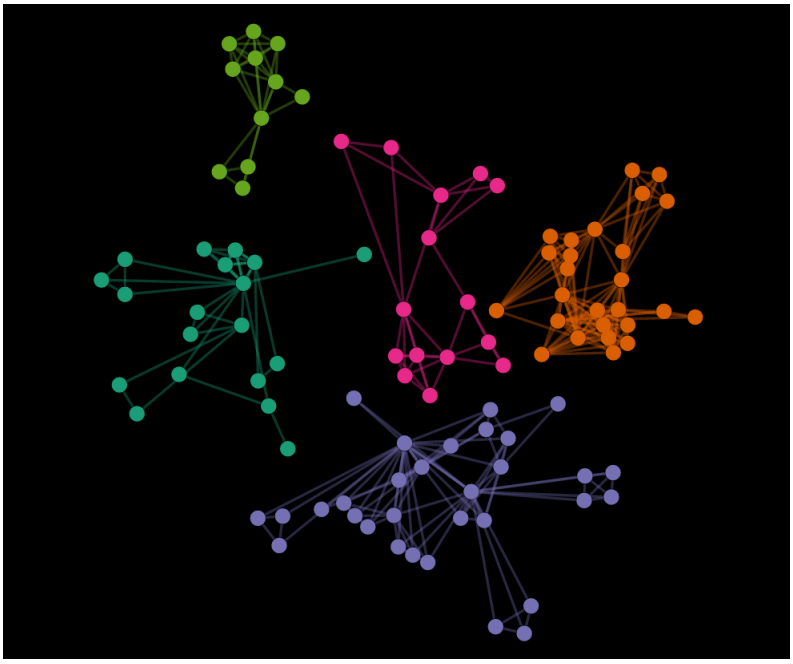


Figure 4.2: Example Graph G=(V,E) Bull Graph

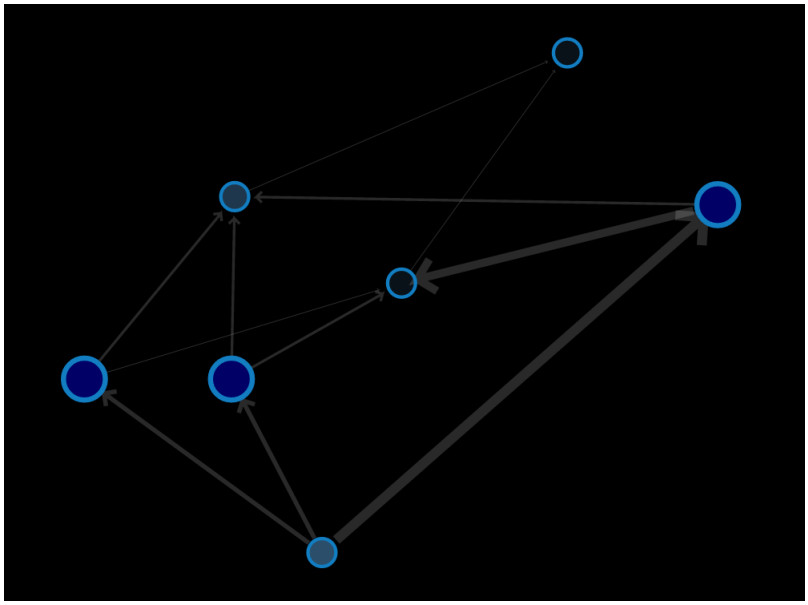


Figure 4.3: Example Graph G=(V,E) Bull Graph

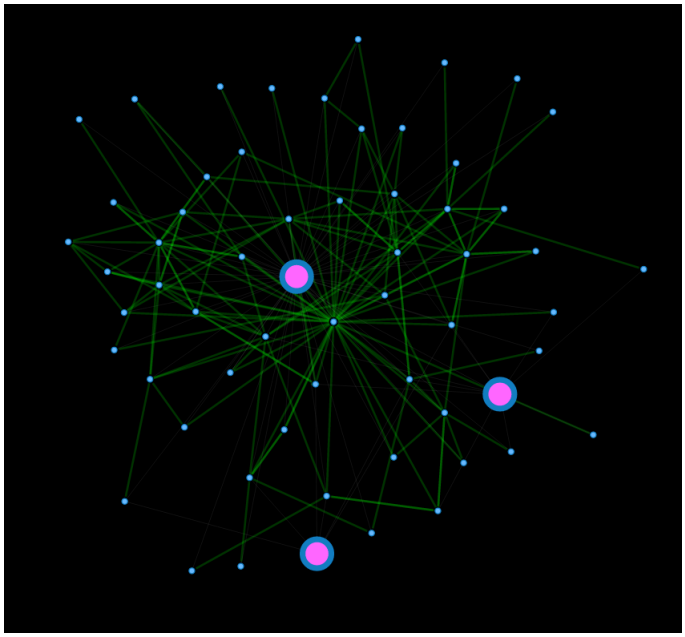


Figure 4.4: Example Graph G=(V,E) Bull Graph

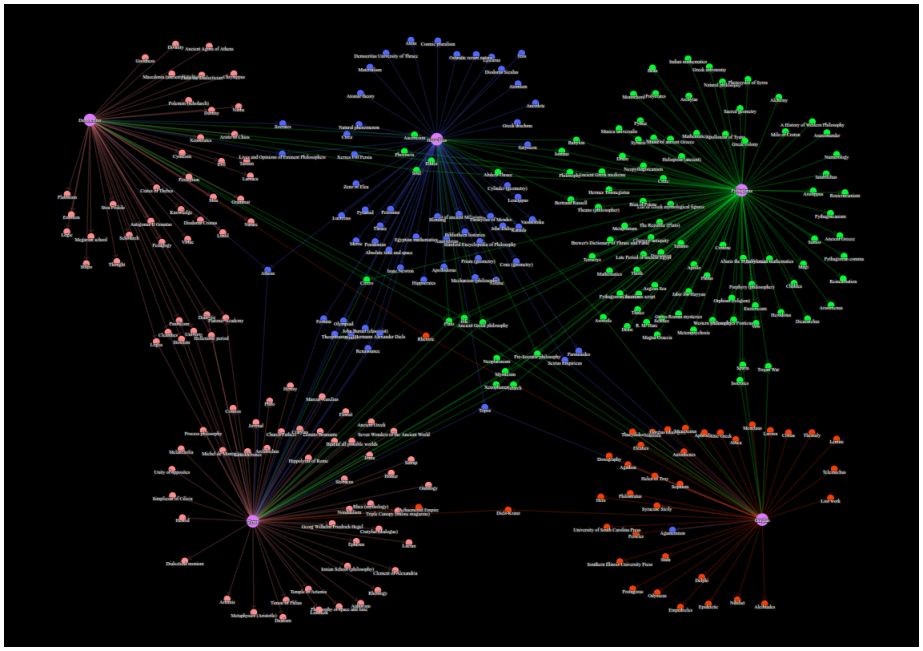


Figure 4.5: Example Graph G=(V,E) Bull Graph

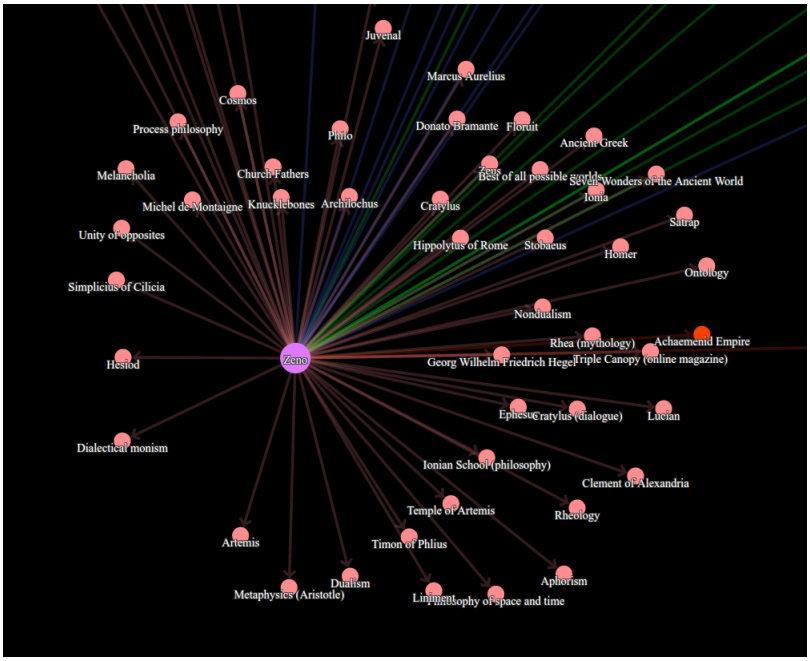


Figure 4.6: Example Graph G=(V,E) Bull Graph

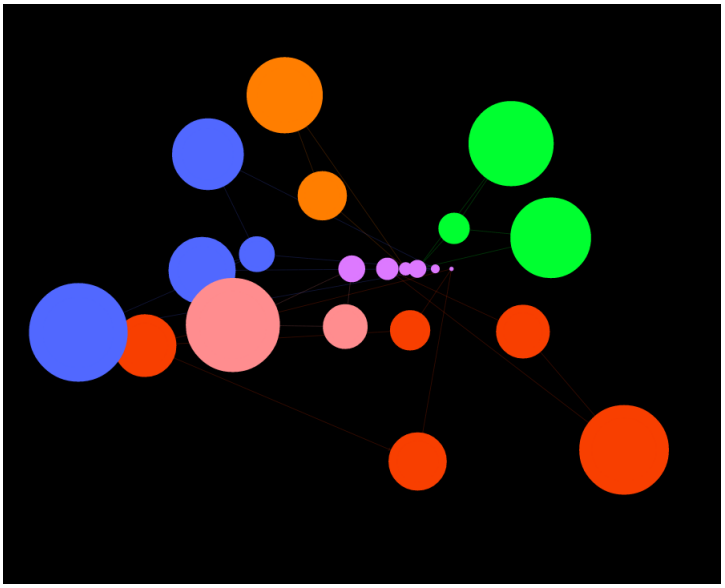


Figure 4.7: Example Graph G=(V,E) Bull Graph

4.5.0.7 Result

Chapter 5

Overall System Description

The project uses two complementary technologies of a raw python code and a JavaScript program that can represent or draw the output of the python code on screen. This lead to a need to include a new element of a web framework for implementing the louvain algorithm and creating a JSON object, and JavaScript to represent the graph on a web browser. A web framework is one that aims to remove the overhead associated with common activities performed in the web development.

5.1 Choice of Web.py

An exploration one a few python web frameworks such as Django, Grok and web.py. A sample application in Django was built to see if Django suits the need of the project. Django was eliminated due to the fact that it was heavier for a simple task that we wish to perform in the project. Web.py was chosen as the web framework for the project as it allowed successful integration of the existing python code with the web framework over grok.We must note that the larger project at the LARCA group does not use web.py instead another framework called Angular.js. However, Angular.js is designed to be used for large, complex projects, and after some evaluation it was clear that the overhead for this small project did not pay off. Thus, Web.py suffices and was easy to used for the current need of the project that is to perform the integration task.

5.2 Frontend Framework

Bootstrap was an intuitive web front-end frame work that has been implemented in the project. Bootstrap allows to divide the screen into various matrix cubes enabling us to place buttons to run the application. Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. It is the second most starred project in github and has more than forty thousand forks [13].

5.3 Using the application

The web application thus developed is intutive enough such that even though it performs tasks of computation and integration into the web it looks simple. Minimal and most necessary elements have been implemented. In the following we describe the entire working flow of the Web Application:

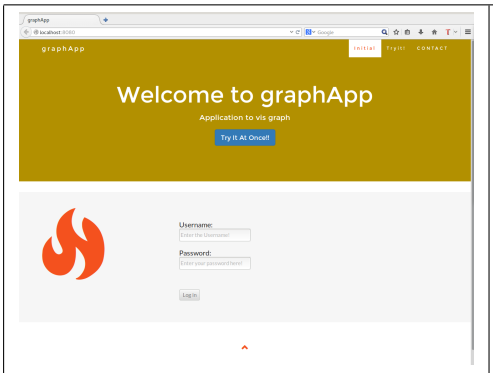
1. **"Login Page"** In this part a simple login password is described.

2. **File Loader Page** After logining the user lands in a file loader page where provision has been provided to upload a text file containing the adjacency matrix of the graphs are present in the "start" "Destination" or "Start" "Weight" "Destination" format. In this space only numerical values can be accepted for supporting the simplity of the background process. Another space to provide a key that maps the names of the node numbers. This helps to present another dimention of seeing the data in the form of names. Viewing the names of the node over nodes helps the user to deepen his vision of analysis.

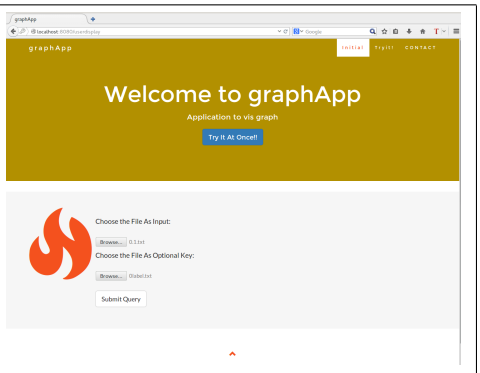
3. **Dash Board** In this part of the web application the user lands in a page that presents to the user 4 different variety of visual representation of the input data : Community graphs (Diplaying the links between various communities), Full graph split into communities represented with different colors, Original graph input(Helps to see if the input was proper) and a Matrix view of the graph

4. **On Click Viz.** On clicking on the visualization the user wants to take the web aplication switches to the graph that the user has clicked.

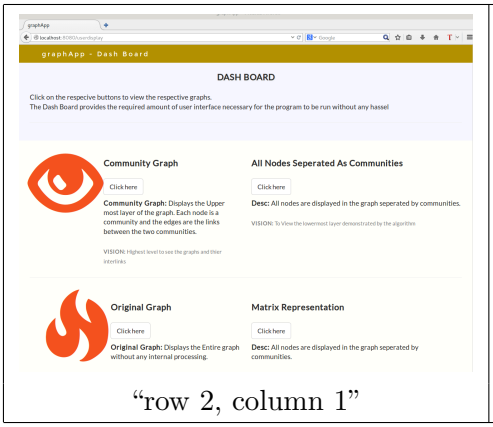
5. **Error Page** In case of mismatch in the format or the password was wrong or the visualization is not possible due the screen size or the browser is not able to handle the large JSON object Errors pages have been genrated to couter act on exceptions.



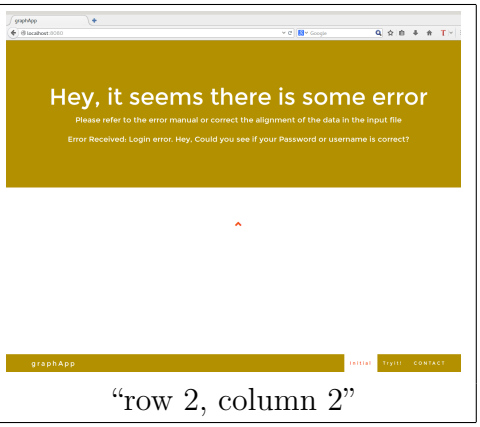
"row 2, column 1"



"row 2, column 2"



"row 2, column 1"



"row 2, column 2"

5.3.0.8 Implementation Benefits

sdgbf akjsfkldsnfksndafnadfa fasfadf adfasfasfadf asfasfadf afadfadf

5.3.0.9 Description

git hub repository : `https://github.com/abhinavsv3/webproject`

5.3.0.10 Result

5.3.1 Benefits to the community

This can be used in places where there is difficulty in visualization of a very complex landscape of data such as medical domain. In Medical domain a patient can be a vector of diseases and visualization of such patients (patients graph–which shows relations of how two patients are similar, a graph in which patient-patient edge weight is the similarity value) would be useful for analyzing and predicting the disease landscape of a region and in turn multiple regions.

Chapter 6

Conclusion and Future Works

6.1 Goals Achieved

6.2 Revision of Planning and Budget

6.3 Future Works

In the span of five months we were able to build a basics project by comparing, contrasting, including the one that the director suggests and chosing the one that is simple and works well. In this project I would like to suggest a few improvemnets that we would have done given more time. We would like to enumerate on that:

1. In the Algorihthm part :
   (a) The order in which the information is presented can affect the computation of the louvain community detection algorithm. Hence the problem of finding specific heuristics to solve this ordering can improve the louvain algorithm computation time.
   (b) Another intersting feature is to analyse whether every step in the louvain algorithm gives the exact or nearly exact hierachy of community detection
   (c) The project relies fully on louvain. One can speculate on whether modularity is the only measure that exists. Thinking about a completely new measure would be interesting.

2. In the visualization Module:

   (a) In the current project the communtiy graph is presented seperately from the main graph. A zoom effect can be introduced to zoom into to community graph to reach different levels of hierachy.

   (b) Double-click on a node to fade out all but its immediate neighbours.

   (c) We deal with large graphs thus it would be nice to have some search functionalities. jQuery can be used to create an autocompleting search box that can be featured on the graph display page which can search the name of the patient or the treatment that is needed.

   (d) Fish eye can be introduced. Since alchemy.js is a framework that runs on d3 it can easily be extended o d3. Hence the fish-eye module can help to view every node along with it's neighbours in a more expanded format.

3. In the Overall structure:

   (a) The current dash board is before the graph board. The Dash board can be included in the graph board itselef to avoid switching back and front in the web application.

   (b) A simple Database can be setup for storing passwords and improvization on sign-ins can deliver a better personlaizaed user experience.

6.4 Availability and requirements

1. **Project Name**: Graph and matrix algorithms for visualizing high dimensional
2. **Project Homepage**: `https://github.com/abhinavsv3/webprojectdimensional`
3. **Operating System**: Platform Independent. Preferably Unix–like operating system
4. **Programming Language**: Python 2.7
5. **Other Requirements** : Alchemy.js, Python Packages, Web.py

6.4.1 Conclusion

This is one of the greatest project experience.

6.4.2 Personal Conclusion

The project has made my mind very innovative. I can produly call myself an engineer. My director, prof.Ricard gave me a freedom to think freely and understand the project and made me do the project the way I have analysed it. This made me develop an new characteristics of learning stuff in the fly and made me feel enthusiatic about working on projects that I have little knowledge on. I am sure given any project I can now make innovation and work hard to bring the project to a better light.

Listings

Bibliography

[1] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

[2] Nikos Bikakis, John Liagouris, Maria Krommyda, George Papastefanatos, and Timos Sellis. graphvizdb: A scalable platform for interactive large graph visualization.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[4] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[5] Michael Bostock and Jeffrey Heer. Protovis: A graphical toolkit for visualization. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2009.

[6] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[7] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv preprint physics/0608255*, 2006.

[8] Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. Resilience of the internet to random breakdowns. *Phys. Rev. Lett.*, 85:4626–4628, Nov 2000.

[9] David Emms, Edwin R Hancock, Simone Severini, and Richard C Wilson. A matrix representation of graphs and its spectrum as a graph invariant. *Electr. J. Comb*, 13(1), 2006.

[10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[11] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[12] Julien Odent and Michael Saint-Guillain. Automatic detection of community strucures in networks, November 26,2012.

[13] Mark Otto and Jacob Thornton. Bootstrap.

[14] Pratha Sah, Lisa O. Singh, Aaron Clauset, and Shweta Bansal. Exploring community structure in biological networks with random graphs. *BMC Bioinformatics*, 15(1):1–14, 2014.

[15] Stanislav Sobolevsky, Riccardo Campari, Alexander Belyi, and Carlo Ratti. General optimization technique for high-quality community detection in complex networks. *Physical Review E*, 90(1):012811, 2014.

[16] M. Zamora, M. Baradad, E. Amado, S. Cordomí, E. Limón, J. Ribera, M. Arias, and R. Gavaldà. Characterizing chronic disease and polymedication prescription patterns from electronic health records. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–9, Oct 2015.