ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

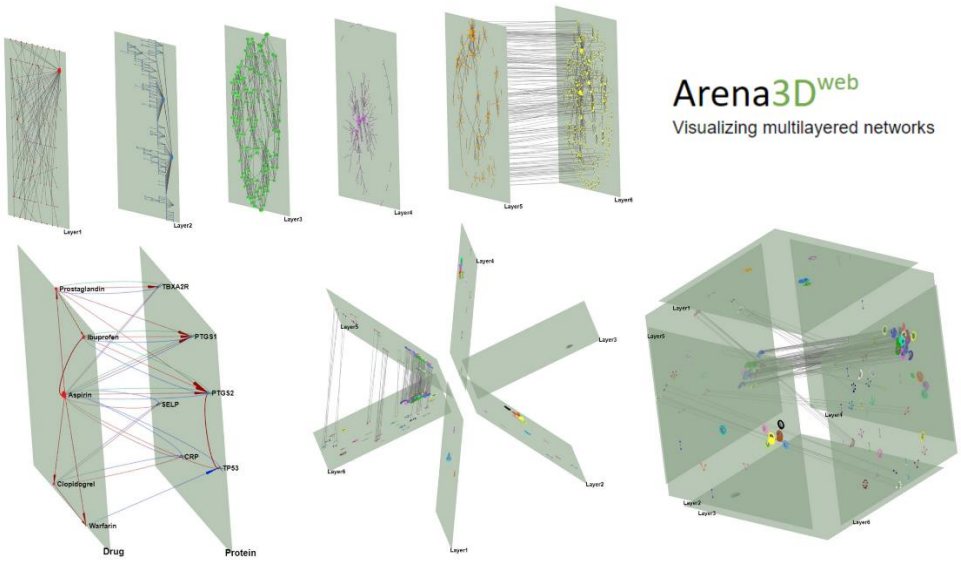
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό
Πανεπιστήμιο Αθηνών

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΒΙΟΛΟΓΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ – ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΛΟΓΙΑ»

Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η Ε Ρ Γ Α Σ Ι Α

«Οπτικοποίηση και ανάλυση τρισδιάστατων πολυεπίπεδων βιολογικών δικτύων με πολλαπλές ακμές»



Μαρία Κοκόλη

Πτυχιούχος Τμήματος Πληροφορικής, Ο.Π.Α.

ΑΘΗΝΑ (2022)

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

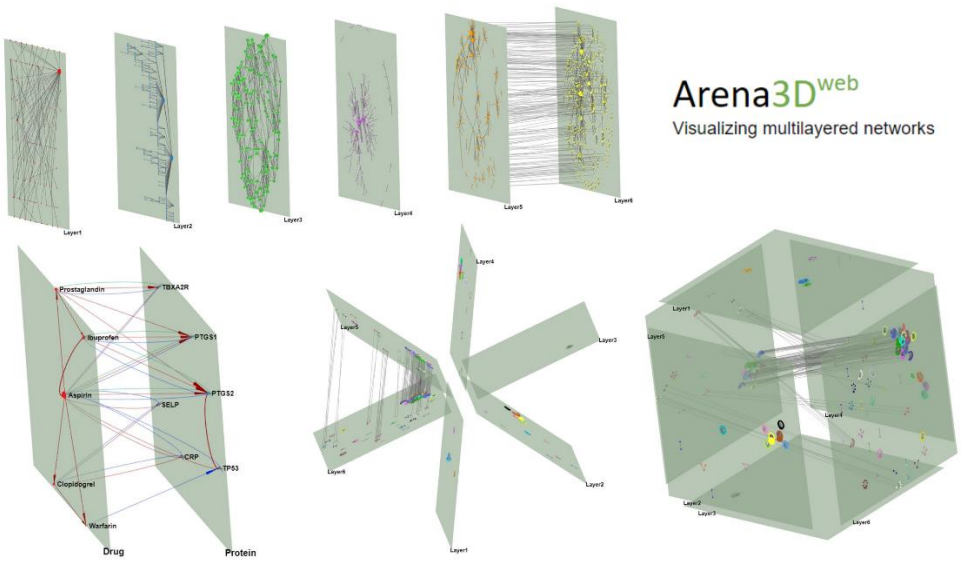
HELLENIC REPUBLIC
National and Kapodistrian University of Athens

SCHOOL OF SCIENCE
FACULTY OF BIOLOGY

MASTER IN
"BIOINFORMATICS – COMPUTATIONAL BIOLOGY"

Master Diploma Thesis

«Visualization and analysis of 3D multi-edged and multi-layered biological networks»



Maria Kokoli

Computer Science, A.U.E.B.

ATHENS (2022)

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

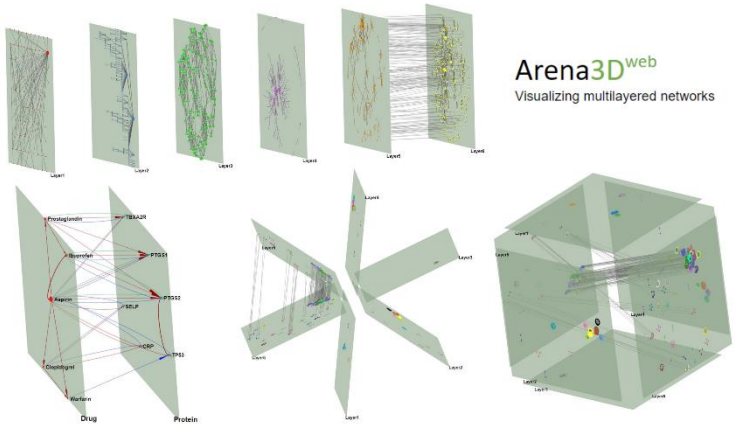
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΒΙΟΛΟΓΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ «ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ – ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΛΟΓΙΑ»

Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η Ε Ρ Γ Α Σ Ι Α

«Οπτικοποίηση και ανάλυση τρισδιάστατων πολυεπίπεδων βιολογικών δικτύων με πολλαπλές ακμές»



Τριμελής εξεταστική επιτροπή

Καθηγητής Δρ. Παντελής Μπάγκος (Επιβλέπων ΠΜΣ)
Καθηγητής Τμήματος Πληροφορικής με Εφαρμογές στη Βιοϊατρική, Πανεπιστήμιου Θεσσαλίας

Ερευνητής Δρ. Γεώργιος Παυλόπουλος (Επιβλέπων Εργαστηρίου)
Κύριος Ερευνητής Β',
Ε.ΚΕ.Β.Ε «Αλέξανδρος Φλέμινγκ»

Καθηγήτρια Δρ. Βασιλική Οικονομίδου
Αναπληρώτρια Καθηγήτρια Βιοφυσικής – Μοριακής Βιοφυσικής

ΠΕΡΙΛΗΨΗ

Η αποτελεσματική ενσωμάτωση και οπτικοποίηση ετερογενών πληροφοριών σε μία απεικόνιση αποτελεί βασική πρόκληση. Σε αυτή τη μελέτη, παρουσιάζουμε το εργαλείο Arena3D^{web}, την πρώτη, πλήρως διαδραστική και χωρίς εξάρτηση από άλλα εργαλεία, διαδικτυακή εφαρμογή που επιτρέπει την οπτικοποίηση πολυεπίπεδων δικτύων στον 3D χώρο και τις καινούριες δυνατότητες της. Με την Arena3D^{web}, οι χρήστες μπορούν να ενσωματώσουν πολλαπλά δίκτυα σε μία μόνο προβολή, μαζί με τις συνδέσεις τους, εντός και μεταξύ επιπέδων. Για σαφέστερες και πιο ενημερωτικές οπτικοποιήσεις, οι χρήστες μπορούν να επιλέξουν ανάμεσα σε μια πληθώρα αλγορίθμων διάταξης και ομαδοποίησης και να τις εφαρμόσουν σε ένα σύνολο επιλεγμένων επιπέδων, είτε μεμονωμένα, είτε σε συνδυασμό. Οι χρήστες μπορούν επίσης να ευθυγραμμίσουν τα δίκτυα και να επισημάνουν τοπολογικά χαρακτηριστικά κόμβων, ενώ κάθε επίπεδο, καθώς και ολόκληρη η σκηνή, μπορούν να περιστραφούν και να τοποθετηθούν οπουδήποτε στον 3D χώρο. Ο χρήστης μπορεί να επιλέξει χρώματα, τόσο για τις ακμές και τα διαφορετικά κανάλια σε περίπτωση πολλαπλών ακμών, όσο και για τους κόμβους, ενώ χρώματα μπορούν επίσης να χρησιμοποιηθούν για την επισήμανση σημαντικών διαδρομών και μονοπατιών. Στην τρέχουσα έκδοση, η Arena3D^{web} υποστηρίζει κατευθυνόμενους ή μη κατευθυνόμενους, βεβαρημένους ή βεβαρημένους μη γράφους, με πολλαπλές ακμές. Η εφαρμογή είναι γραμμένη σε R, Shiny και JavaScript. Κάνουμε μια εισαγωγή στην θεωρία των γράφων, στα διάφορα μοντέλα και αλγορίθμους, καθώς και σε ανταγωνιστικά εργαλεία και αναλύουμε την υπάρχουσα και νέα λειτουργικότητα της Arena3D^{web} χρησιμοποιώντας διαφορετικά σενάρια χρήσης.

Η Arena3D^{web} είναι διαθέσιμή στο σύνδεσμο http://arena3d.org ή http://arena3d.pavlopouloslab.info.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω για την παρούσα διπλωματική τον επιβλέποντα ερευνητή, κ. Δρ. Γεώργιο Παυλόπουλο, ερευνητή στο Ε.ΚΕ.Β.Ε "Αλέξανδρος Φλέμινγκ'', για την πολύτιμη καθοδήγηση και βοήθεια του.

Επιπλέον, θα ήθελα να αποδώσω τις ευχαριστίες μου στα μέλη της τριμελούς επιτροπής, τον επιβλέποντα καθηγητή κ. Δρ. Παντελή Μπάγκο, Καθηγητή στο Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική, του Πανεπιστημίου Θεσσαλίας και την κα. Δρ. Βασιλική Οικονομίδου, Αναπληρώτρια Καθηγήτρια Βιοφυσικής – Μοριακής Βιοφυσικής του Τμήματος Βιολογίας Ε.Κ.Π.Α, για την συμβολή τους στην εκπαίδευση μου πάνω στον τομέα της Βιοπληροφορικής και για τα εφόδια που μου έδωσαν.

Επίσης θα ήθελα να ευχαριστήσω θερμά τους μεταδιδακτορικούς ερευνητές του ΕΚΕΒΕ "Αλέξανδρος Φλέμινγκ'' Ευάγγελο Καρατζά και Φώτη Μπαλτούμα για την βοήθεια, τις συμβουλές τους και μια από τις καλύτερες συνεργασίες, αλλά και τους Ερευνητές του Novo Nordisk Foundation Center for Protein Research, Lars Juhl Jensen και Nadezhda T. Doncheva για την συνεργασία που είχαμε για την διασύνδεση με το Cytoscape.

Τέλος θα ήθελα να ευχαριστώ τα γύρω μου αγαπημένα πρόσωπα που με ενθάρρυναν να εγγραφώ στο μεταπτυχιακό πρόγραμμα και με στήριξαν κατά την διάρκεια του.

Μαρία Κοκόλη

Αθήνα, Νοέμβρης 2022

ABSTRACT

Efficient integration and visualization of heterogeneous biomedical information in a single view is a key challenge. In this study, we present Arena3D^{web}, the first, fully interactive and dependency-free, web application which allows the visualization of multilayered graphs in 3D space, as well as its new features. With Arena3D^{web}, users can integrate multiple networks in a single view along with their intra- and inter-layer connections. For clearer and more informative views, users can choose between a plethora of layout and clustering algorithms and apply them to a set of selected layers either individually or in combination. Users can align networks and highlight node topological features, whereas each layer, as well as the whole scene, can be translated, rotated, and scaled in 3D space. User-selected edge colors can be used to highlight important paths, while node positioning, coloring, and resizing can be adjusted on-the-fly. In its current version, Arena3D^{web} supports weighted and unweighted, directed and undirected multi-edge networks. The application is written in R, Shiny, and JavaScript. We demonstrate the old and new functionality of Arena3Dweb using different use-case scenarios.

Arena3D^{web} is available at http://arena3d.org or http://arena3d.pavlopouloslab.info

CONTENTS

1 GRAPH AND MATHEMATICAL MODELING

1.1 The concept of graphs

Networks or **graphs**, as they are called in discrete mathematics, are a way to represent and describe relationships between entities.

A graph is a pair *G = (V, E)* where **G** is the graph name, **V** is a set of vertices/points/ nodes and **E** is a set of edges/lines/arrows/links. A graph can be divided into **subgraphs** *G' = (V', E')* where V' is a subset of V and E' is a subset of E, and G' is the name of the subgraph.

Two different graphs that have the same number of vertices and the same edges are called **isomorphic** and expressed $G1 \simeq G2$.

In our everyday life, we have different graphs such as the water supply network, the road network, telecommunications, the internet, social media, etc. In biomedical sciences, graphs are used to represent biomolecules, proteins, DNA, or RNA and how these are connected or interact. [1]

1.2 Graph categories

The most common graph categories are undirected graphs, directed graphs, weighted graphs, multi-edge, bipartite graphs, and trees (Figure 1.1).

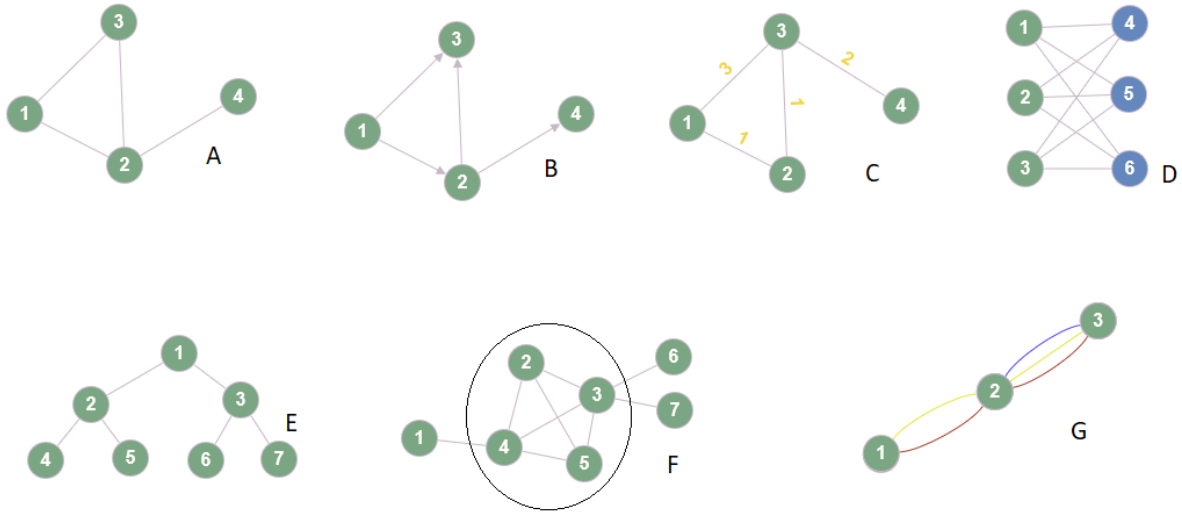


Figure 1.1 Graph categories. A) Simple graph. **B)** Directed graph. **C)** Weighted graph. **D)** Bipartite graph. **E)** Tree graph. **F)** Cluster. **G)** Multi- edge graph. Graphs were visualized using graphonline opensource tool.

- A graph is called **undirected** if the edges between the vertices are singled connections.
- A graph is called **directed** if the edges between the vertices are arrows that show the direction of the graph.
- A graph is called **weighted** if every edge has a different weight that shows the importance of the connection. A weighted graph can be undirected or directed.
- A graph is called a **multi-edge** if it contains multiple edges for two same vertices. A multi-edge graph can be undirected, directed, and/or weighted.
- A graph is defined as a **bipartite graph** if the vertices can be divided into two disjoint and independent sets and every edge connects a vertex from the first to the second set.
- A **tree** is an undirected graph, in which two vertices are connected by one path or a connected acyclic undirected graph.
- A **cluster** is a subgraph with a set of vertices with common characteristics.

1.3 General network properties

1.3.1 Degree

One of the most important topological features is the degree deg_i, which is the total number of edges that are adjacent to the vertex in an undirected graph. In a directed graph, degree is the sum of indegree deg^{in}_i , which is the number of arcs incident from the vertex, and outdegree deg^{out}_i , which refers to the number of arcs incident to the vertex $deg_i = deg^{in}_i + deg^{out}_i$. The average degree of the network is

$deg_{avg} = \frac{\Sigma\, degi}{|V|}$.

1.3.2 Density

Density is the ratio between the number of edges in a graph and the number of possible edges in a graph. The number of those in a full connected graph is $E_{max} = \frac{V(V-1)}{2}$, so the density is calculated $\frac{E}{Emax} = \frac{2E}{V(V-1).}$. According to the density, the graph is considered dense *if* $E \simeq V^k$, $2 > k > 1$, or sparse if $E \simeq V$.

1.3.3 Clustering Coefficient

Another important network topological feature is the clustering coefficient, which is a measure that shows if a network, or a node, forms (or has the tendency to) clusters or tight communities. This can be defined as $C_i = \frac{2e}{/\, k(k-1)}$, where e is the number of edges between the k neighbors, and C_i can be between 0 (lower tendency to form clusters) and 1 (high tendency to form clusters).

1.3.4 Distance

The shortest path length between two nodes is called the network's distance ($dist_{ij}$). The length of the shortest path is defined as the smallest number of edges between two vertices. In case two nodes are not connected, then the $dist_{ij} = \infty$.

1.3.5 Diameter

The network diameter is the length of the longest path between two vertices and is related to distance with the following $diam_m = max(dist_{ij})$. [1]

1.4 Models

Models are used to better understand a network's topology. Some of them are the following:

1.4.1 Erdős–Rényi

One of the most popular models in graph theory is the **Erdős–Rényi** model (Figure 1.2) [2] and it was introduced to describe the properties of a random graph. When we have V nodes, then the random connection probability is $p = \frac{2E}{V(V-1)}$ and the degree distribution is given by a binomial distribution. The possibility of a node having a specific degree can be calculated as $p(deg) \simeq e^{-degavg} \frac{degavg^{deg}}{deg!}$. When p is small, then the network is not connected, but if $p \approx \frac{1}{V}$ the network has a bigger component containing most of the network's connections. If $p \geq \frac{\log(V)}{V}$, then we assume that all vertices are connected homogeneously and randomly. The clustering coefficient ($C = p = \frac{degavg}{V}$) shows, that the possibility of two nodes with the same neighbor being connected, is the same as the connection possibility of two random nodes. [1]

1.4.2 Watts-Strogatz

Watts Strogatz is a model that is used to describe random networks, where every node can be reached by any other node through a small path (Figure 1.2) [3]. This model is used in networks that are described by local structures and generally small path lengths. A good example is metabolic networks, where metabolites are linked to each other[2]. If all the nodes are placed on a circular ring, each one would be connected to its $\frac{V}{2}$ neighbors in a Watts Strogaz network [1].

1.4.3 Barabási–Albert

The **Barabási–Albert** model is used to describe random scale-free networks, where the degree distribution follows a power law considering their inhomogeneous degree distribution or otherwise networks with nodes that do not have a typical number of neighbors (Figure 1.2) [4]. According to this, networks evolve, new edges do not appear randomly, and the new nodes follow the existing degree distribution. An easy example of this model is the social networks, where a person/node, who has a lot of friends, is likely to get more, compared to others that have fewer friends. [1]

When comparing networks of the afformentioned models with Barabási–Albert, assuming all of them have the same density and size, the latter one was found to have shorter average path lengths[5] [6].

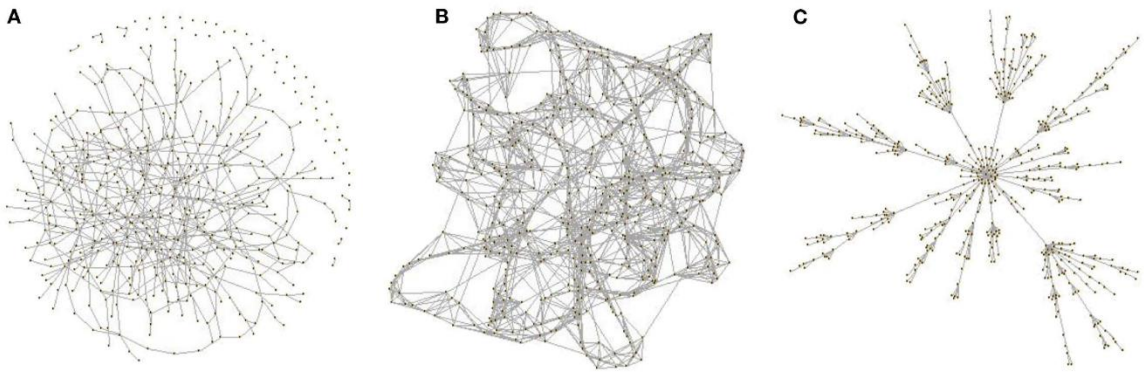


Figure 1.2 Network models. A) An Erdos–Rényi random network. **B)** A Watts-Strogatz network. **C)** Barabási–Albert scale-free network. Graphs were visualized using R.

2 BIOLOGICAL AND BIOMEDICAL NETWORKS

Graphs and networks are great tools in biomedical research because they can capture the associations between the biological entities that are studied. Some examples of those entities are proteins, genes, metabolites, small molecules, ligands, diseases, drugs, literature, and general database records. There are dissimilar categories of biological and biomedical networks according to the items and the associations that they model. Some of them are:

2.1 Protein-protein interaction networks

Protein-protein interaction networks, or PPIs, hold information about how different proteins interact with each other and how they operate to enable a biological process. These interactions can be physical or predicted. Usually, PPIs are small world and scale-free networks, where central hubs often represent the evolutionarily conserved proteins, while cliques have a high functional significance [7]. Some of the databases that have PPIs are BIND [8], BioGRID [9], and DIP [10]. (Figure 2.1 B).

2.2 Sequence similarity networks

Sequence similarity networks that are used in widely used tools[11], capture the similarity between amino acids or nucleotide sequences depending on node use (proteins or genes). When two nodes are connected to each other, it means that their sequences - whether they are proteins or genes - have a similarity percentage greater than one specific value defined by the user. In this way, a small network is created with weights, representing possible functional associations, between biomolecules. These networks are weighted, small world, and scale-free and usually contain hubs. It is common to use clustering algorithms in those to detect protein families[12]. The most-well known tools for determining sequence similarity are BLAST[13], LAST[13], and FASTA3 suite[13]. Sequence similarity network clustering helps to identify protein families, where proteins have similar functions or participate in biological processes.

2.3 Gene regulatory networks

Gene regulatory networks are, usually, directed dynamic networks and express the relationship between transcription factors (TFs) and TF-binding sites, or between genes and their regulators. Often in those networks, the nodes have few interactions, and a small number of hubs have a higher connectivity degree. They also follow a power law degree distribution $p(k) \sim k^{-\gamma}$, $\gamma \approx 2$ [14].

Some databases that have information about gene regulatory networks are the KEGG[15], GTRD[16], TRANSFAC[17], and TRRUST[18] (Figure 2.1 C).

2.4 Signal transduction networks

Cell signaling is a series of molecular events within a cell, or from the exterior to its interior[19]. This is captured in the signal transduction networks. Most of the time, these networks are directed and sparse and follow a power law degree distribution with small-world properties. These networks can be found at KEGG [20], Reactome[19] MiST[21], NetPath[22], and other databases (Figure 2.1 D).

2.5 Metabolic networks

Metabolic networks capture the metabolite interactions in an organism. Metabolites (nodes) are amino acids or polysaccharides. These networks are directed, scale-free with small-world properties[2] and hierarchies[23]. KEGG[20] along with Reactome[19] host metabolic networks (Figure 2.1 A).

2.6 Gene co-expression networks

Gene co-expression networks are undirected weighted networks, where the nodes are genes and if there is a connection between two nodes, there is a significant co-expression relationship between them[24]. These networks can be expressed as a gene-gene similarity matrix. These networks have the ability to show which genes are active at the same time, or in the same biological processes. They cannot distinguish regulators from regulated genes but can specify which genes tend to show a coordinated pattern of expression in a group of samples. The generation and analysis of gene expression networks are described as follows:

1. Defining a co-expression measure and calculating a similarity score for each pair. Generally, different correlation measures are used for the construct networks, including Pearson or Spearman correlations.
2. Determination of threshold in order to compare the degree of similarity with a threshold value. Those pairs that have a degree of similarity greater than the threshold, are considered to have a significant co-expression relationship and are connected in the network.
3. Gene co-expression network generation, where each gene is represented by a node and the relationship between two gene-nodes is represented by an edge.

4. Identifying groups of co-expressed genes using clustering tools. Depending on the overall clustering factor, the network can gather to detect functional modules. The clustering method must be chosen with care because it can have a significant impact on the result and the meaning of the analysis.

Some databases with co-expression data are GEO[25], ArrayExpress[26] and COXPRESdb[27].

2.7 Phylogenetic networks

Phylogenetic networks describe the evolutionary relationship between nucleotide sequences, genes, chromosomes, genomes, or even species[28] (Figure 2.1 F). Their representational structure is not yet clear, as it is questionable whether the presentation of them as a tree is correct or not. The claim that phylogenetic networks differ from phylogenetic trees is based on their modeling consisting of richly connected networks, by adding hybrid nodes (nodes with two parents) as opposed to node trees, where each node has a single parent (a node hierarchy)[29]. One more difference is that phylogenetic trees are only suitable for the study of vertical evolution processes, while phylogenetic networks are more general and can be used for the study of both horizontal and vertical evolutionary processes. The horizontal processes are represented by meshes in the network, which do not appear in the tree[30]. Phylogenetic trees are considered a subset of the networks.

Based on Darwin's theory, which claims that all species living today descended from a common ancestor, the relationships between each group of individuals, even those from different species, can appear in a phylogenetic tree. Thus, the goal of phylogenetics is the use of biological data for a collection of individuals or species to create a tree describing how they are related[31]. The most well-known methods for tree construction are Neighbor-Joining (NJ), UPGMA, and Maximum Parsimony (MP)[32].

Some common applications are SplitsTree[33], DendroScope [34], and phangorn (R package)[35] can be used to visualize these networks.

2.8 Ecological networks

Ecological networks are used to describe the ecosystem, where each species (nodes) interact with each other. There are four different interactions: trophic, symbiotic, mutualistic, and competitive[1]. These networks describe the functioning of the ecosystem, and their modeling helps to study possible impacts in case of change of some element of the system. Food

Interactions can be described as directed or undirected graphs in the case of binary food webs, whereas quantitative food chains can be shown as weighted graphs[36] (Figure 2.1 E).

The presence of grouped species in ecological networks is debatable but is being strengthened by the analysis of small, not well-resolved, clustered networks. High-resolution chains, where species are not clustered into "trophic" species, show a higher degree of clustering than their random counterparts[32]. More generally, a species in the middle of a cluster may have the role of a cornerstone, and its loss could have large effects on the network.

2.9 Epidemiological networks

Epidemiological networks are used to study disease transmission [37]. These networks can help find the way that disease is transmitted and are used by biological, biomedical, and social scientists, as epidemic diseases are contagious diseases that directly affect social development, with the current outbreak of the coronavirus as a typical example. In the above two scientific fields, there are several analogies in the way an epidemic spreads, which has led to the development of infection spread models from biology and their application to computer networks.

The study of an epidemiological network helps to identify the routes of transmission of disease. Also, through such a network we can access information on epidemiological dynamics. By connecting network dynamics with real-life data, patient data can be a useful basis for developing hypotheses about how a disease works and prove useful in drug creation and treatment development. A co-morbidity network, i.e., the simultaneous occurrence of diseases or pathological conditions in the same patient, is an example of an epidemiological network.

Epidemiological networks can be found at KEGG[38] or HPRD database[39].

2.10 Literature co-occurrence networks

Literature co-occurrence networks capture the bio entities' connection between different texts, papers, books, etc. When reading a text, capturing and mapping different entities such as genes/proteins, phenotypes, diseases, environments, chemical compounds, and terms using Name-entity recognition (NER) it is possible to identify these entities at public databases such as Wikipedia, PubMed, or PubMed Central[1].

2.11 **Knowledge networks**

Knowledge networks capture information and meta-data from various sources and are usually multi-edge graphs. These sources are usual public biological or literature databases [1]. Knowledge networks can be found at STRING[40] (known and predicted protein interactions), STITCH[41] (known and predicted interactions between proteins and chemical compounds), and PICKLE[42] (meta-data database).

2.12 **Expression Quantitative Trait Loci (eQTL) Network**

Data obtained from genotyping and/or transcriptomic experiments are used as a locus (eQTLs) in explaining a fraction of the genetic variance of a gene expression [43]. For this purpose, eQTL networks are suitable for summarizing this information[44,45,46]. Genome-wide association studies (GWASs) are used for an association between common genetic variants and phenotypic traits based on many variants of relatively small effect size. Those single nucleotide polymorphisms (SNPs) are measured by expression quantitative trait locus (eQTL) analysis and are represented by eQTL networks with significant associations, as edges. Findings provide unique insight into the genotype-phenotype relationship [e.g., Enhanced tissue-specific heritability of type 2 diabetes (T2D) was identified by eQTL networks].

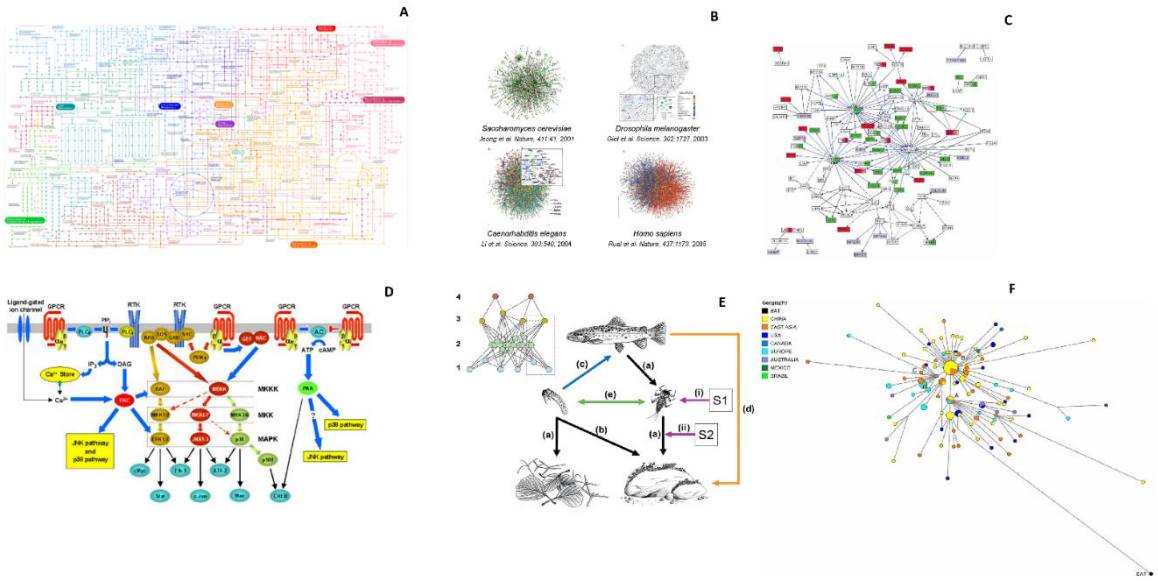


Figure 2.1 Biological and Biomedical Network types. A) A Metabolic Network (KEGG | Reactome). **B)** Protein Interaction Networks (IntAct database). **C)** Gene regulatory networks **D)** Signal Transduction Networks **E)** Ecological Network (A simplified representation of biotic interactions in a stream food web based on leaf litter and periphyton and affected by multiple stressors) **F)** Phylogenetic network of 160 SARS-CoV-2 genomes.

3 LAYOUTS

Graph layouts help analyze and understand networks, especially large ones, and avoid problems such as hairball[47]. Using the correct layout for the data that are represented in the graph, it is possible to depict its structure, and symmetries and emphasize the key features in a clear aesthetic way [48].

Some of the most successful layouts are the **force-based layout** approaches. In those approaches, the nodes are modeled metaphorically as particles connected with spring, and between those, there are acting repelling forces. The layout is finalized when the network has minimized its energy model (Figure 3.1).

Other approaches are the **spectral layout**, **orthogonal layout**, **tree layout**, and **circular layout** (Figure 3.1).

The **spectral layout** uses eigenvectors, usually of a Laplace matrix of the graph, and graph's vertices as coordinates of the graph's vertices[49]. In addition, with the **orthogonal layout**

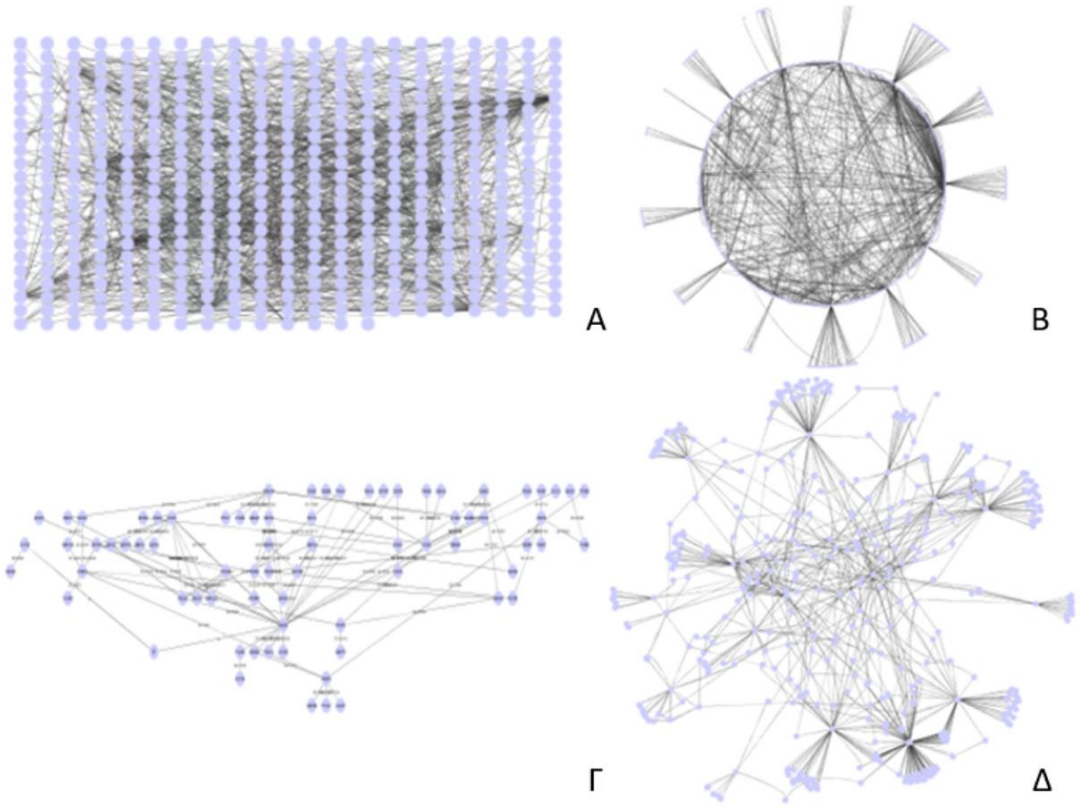


Figure 3.1 Network layouts. A) Grid layout. **B)** Circular layout. **C)** Hierarchical layout **D)** Force-based layout. Networks have been generated with Cytoscape

methods, a graph can have horizontal or vertical edges parallel to the coordinate of the layout. **The tree layout** method is useful when there is a need to show hierarchy because it creates tree-like structures. Lastly, **circular layout** approaches create circular structures, while watchfully choosing the vertices' order, to avoid as many crossings as possible[1].

It is important to understand the advantages of each approach to choose the best algorithm to have the best visual results at the most efficient time. For example, OpenOrd algorithms can handle networks with over a million nodes for less than half an hour[50], Yifan Hu[51] can give aesthetic layouts that are easily comparable to the ones made with the most known Fruchterman Reingold [52] algorithm.

With **edge bundling** methods it is possible to have even more aesthetic layouts and emphasize the information that the network has, reducing the visual clutters. The main idea is to group edges that have a similar path and merge them like cables (Figure 3.2). These methods are following hierarchical or force-directed methods [53] and are still computationally expensive.

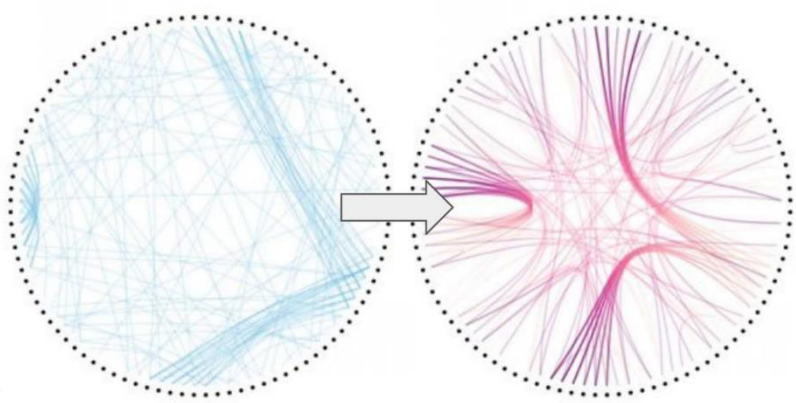


Figure 3.2 Edge Bundling. Networks have been generated with Cytoscape.

These and more algorithms are usually embedded in visualization tools such as Gephi [52], Cytoscape[54], and igraph library[53]. Gephi contains circular, contraction, dual circle, random, MDS, Geo, Isometric, GraphViz, and Force atlas layouts, along with the aforementioned ones. Cytoscape contains grid, random, circular, force-directed and hierarchical layouts, and edge bundling.

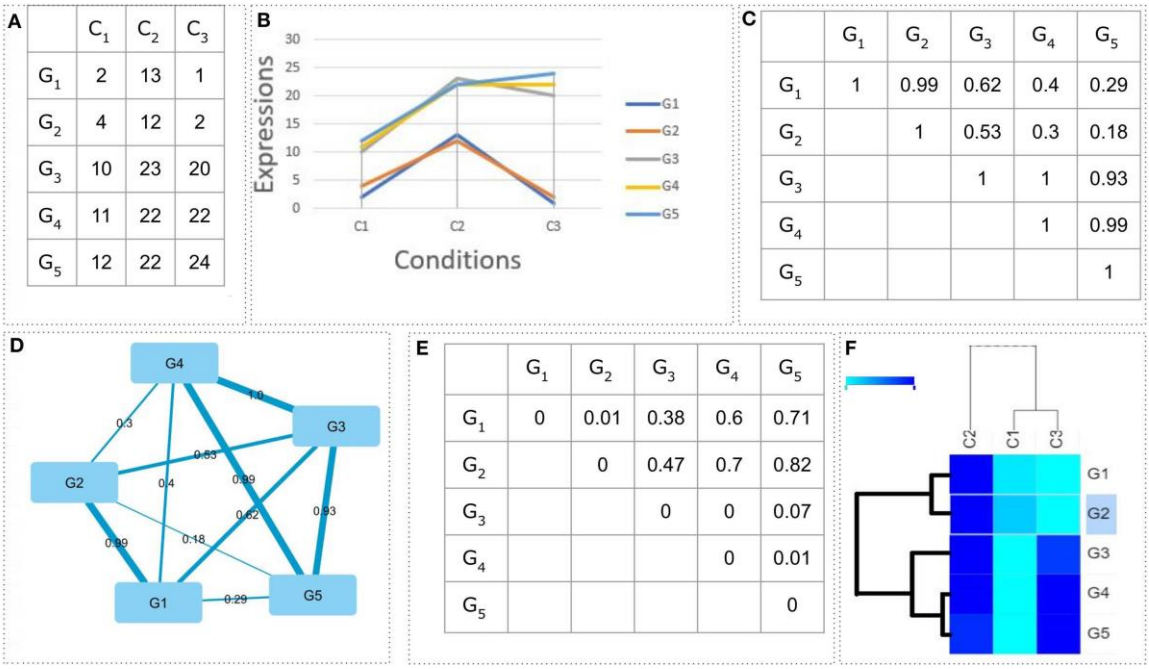
4 CLUSTERING

In large networks, it is often that more than one node has the same properties as another. Grouping those are the next comprehensive step for the best understanding of a network. This grouping process is called clustering and there are multiple algorithms to make it happen. A lot of these take into account the network's topology (for example, find dense areas and try to normalize them). Even though there is a plethora of clustering algorithms, only some of them can handle large networks [55] [56]. Some common clustering algorithms are described below.

4.1 Hierarchical clustering

Hierarchical clustering is a non-graph-based way of data clustering, which accepts a distance matrix containing all pairwise distances between the nodes as input, and outputs a dendrogram showing the hierarchical relationship between the clusters. The standard hierarchical algorithm has $O(n^3)$ time complexity and requires $O(n^2)$ memory, thus making this method inappropriate for large data sets. Hierarchical clustering is divided into three main categories. These are: Single linkage, which calculates the smallest distance between objects in each iteration step, Complete linkage, which calculates the longest distance between objects in each iteration step, and Average linkage, which uses the average distance between all pairs of objects in every iteration step. For more details, a survey explaining how hierarchical clustering algorithms work and what their variations are can be found elsewhere[57].

Notably, all calculations are based on a distance matrix (fully connected graph) which can be generated by a correlation matrix as $D_{ij} = 1 - PCC_{ij}$. D is the distance matrix and PCC is a Pearson Correlation Matrix (e.g., gene co-expression networks). Figure 4.1 shows an example of how five genes can be hierarchically clustered according to their expression values/patterns measured in three hypothetical conditions or time points. The final output is a heatmap accompanied by a dendrogram showing how genes are grouped. Notably, in cases where it is not straightforward which cutoff to apply on the tree in order to define the number of clusters, statistical methods to automate such task, are available[57].



A	C_1	C_2	C_3
G_1	2	13	1
G_2	4	12	2
G_3	10	23	20
G_4	11	22	22
G_5	12	22	24

C	G_1	G_2	G_3	G_4	G_5
G_1	1	0.99	0.62	0.4	0.29
G_2		1	0.53	0.3	0.18
G_3			1	1	0.93
G_4				1	0.99
G_5					1

E	G_1	G_2	G_3	G_4	G_5
G_1	0	0.01	0.38	0.6	0.71
G_2		0	0.47	0.7	0.82
G_3			0	0	0.07
G_4				0	0.01
G_5					0

Figure 4.1 Example of hierarchical clustering. (A) The expression values of five genes in three conditions. **(B)** The chart showing the genes' expression values as patterns. **(C)** The Pearson correlation coefficient (PCC) matrix showing all pairwise PCC values. **(D)** The Pearson correlation matrix in the form of a fully connected graph. **(E)** The distance matrix as a product of the PCC matrix (*Dij* = 1 – *PCCij*). **(F)** A 2D average linkage hierarchical clustering. Genes *G1, G2* as well as genes *G3, G4, G5* are clustered together.

4.2 Graph-based clustering algorithms

4.2.1 SPICi

Speed and Performance in Clustering algorithm (SPICi) is one of the fastest algorithms for large-scale networks. The algorithm finds the nodes with large connectivity according to their degree. It starts from the local nodes with large densities and then chooses the most appropriate neighbors and adds them to the cluster. SPICi has a running time complexity of O(VlogV + E) and needs O(E) memory since it uses heuristics, where V is the number of vertices and E is the number of edges.

More thoroughly this heuristic algorithm finds the first seed - node u and its neighbors. These are divided into 5 groups according to their vertices' weights: (0, 0.2], (0.2, 0.4]. (0.4,0.6], (0.6, 0.8], and (0.8, 1]. The procedure starts from the last group (0.8,1] and moves forward to the first. For each group, the algorithm checks if the group is empty and if this is not true a vertex v with the highest degree is selected from there as the second seed. The pair (u, v) is called the seed edge. This implementation is based on two points, the first is that there is a positive

correlation between the weighted degree of a node and a measure of the overall functional enrichment among the nodes, and the second is that two nodes are more possible to be in the same group if they relate to a high – weight edge. For the cluster expansion, initially, the u and v nodes are added in a set S and then search at the unclustered vertices for the ones that have the best support (u, S). The search continues by adding the u vertices that are founded and updating the cluster's density until the support (u, S) or the density value after the addition is smaller than the threshold. After the algorithm finishes, the S is outputted and continues to the next group[58].

SPICi algorithm, as noted, is significantly faster than other clustering algorithms, especially for biological networks and it works better in dense networks[58].

4.2.2 MCL

Markov Cluster Algorithm (MCL) started at the computational graph clustering field[59] for simple and weighted graphs, but since it is possible to represent biological sequence similarity relationships with these graphs[60], MCL is used for biological purposes as well. Another advantage of this algorithm is that it is not affected by edges that relate to different clusters.

MCL is using a mathematical bootstrapping procedure to find the clusters. The algorithm deterministically calculates probabilities of random walks of the sequence similarity graph. Then, it uses expansion and inflation operators and through them, it transforms one set of probabilities to another, using stochastic (Markov) matrices to capture the mathematical concept of a random walk.

More thoroughly for biological networks, nodes must represent a set of proteins ready to be assigned to a family, edges as a similarity between the proteins, and their weights as the BLAST similarity score. At first, a Markov matrix is created, where the columns and rows are the proteins (nodes) and the values are the similarity score between them, meaning that the diagonal elements are neutral. After that, the initial matrix is seeded to MCL to create the initial random walks to measure the graph's flow and find the areas where there are a lot of random walks (areas with high traffic). The algorithm iterates rounds of expansion and inflation to promote flow in the areas where it is strong and remove flow in the ones where it is weak. The algorithm ends, when the graph is at an equilibrium state and any other expansion and inflation rounds do not affect the graph[61].

4.2.3 Louvain

The Louvain algorithm is a simple and elegant algorithm that optimizes a quality function in two phases. First moves the local nodes and second aggregates the network[62].

Specifically, the Louvain algorithm finds a large-scale network, and groups with large modularity in a brief time. This algorithm also develops a hierarchical community structure for the network, which is important because it gives access to different resolutions of community detection. Modularity is the value between -1 and 1 that evaluates the quality of partitions [63].

Having a weighted network with N nodes, the Louvain algorithm assigns a community per node *i* to the network and finds the neighbors of each one. Then, each node calculates the gain that the modularity would have, if node i was removed from its cluster and added to the neighbor cluster that has the most positive gain. Otherwise, node i remains in its cluster. The algorithm terminates when there is no further improvement. That means that a node can be revisited more than one time. After the end of the first phase, the algorithm continues to the second phase, network aggregation. At this point, a new network is built, where its nodes are the communities from the old network. The sum of the old edges between the nodes in the old communities is now the weight of the new edges that connect the new nodes. If there were edges between the nodes inside the community, then there is a self-loop at the new nodes. After the end of the second phase, it is possible to rerun a pass. A pass is both Louvain's phases[64] (Figure 4.2).

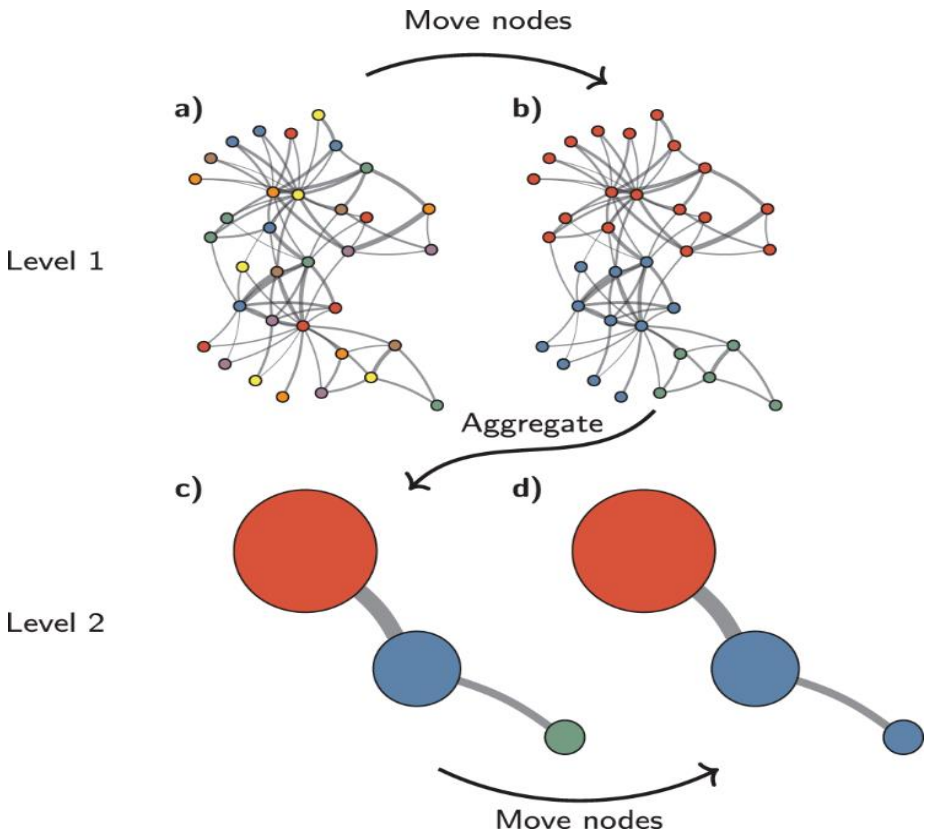


Figure 4.1 Louvain Algorithm. Two phases of Louvain algorithm.
Image from https://doi.org/10.1038/s41598-019-41695-z

4.3 Walktrap

The walktrap algorithm was created for detecting clusters in large-scale complex networks through random walks. The network nodes are divided into groups with small intra-cluster and large inter-cluster distances. The walktrap algorithm, at the worst case, in a network with n vertices and m edges, runs in time $O(mn^2)$ and space $O(n^2)$ and at the average case, in time $O(n^2 \log n)$ and space $O(n^2)$, which means that the algorithm needs a lot of space. Another disadvantage of this algorithm is that it has low accuracy. [65] [66]

The walktrap algorithm is based on random walks, but in a graph, they tend to get trapped inside dense areas. Using random walks, it is possible to create a structural similarity measurement. This measurement is the distance between vertices and clusters and using the distance the algorithm can compare how similar the nodes are. Using hierarchical algorithms and distance it is possible to create dendrograms[66] [65].

The walktrap algorithm begins with splitting the graph into groups, with one node each, so there are as many groups as the nodes and calculate all neighbor distances. Then it chooses two groups at every step k according to their distance and merges them. Thus, a new group is created and the distances refresh. After each step, a new dendrogram is created. The leaves of the dendrogram are the network nodes and the root is the cluster. The algorithm stops after n-1 steps, where n is the number of nodes [65] [66].

4.4 Edge-Betweenness

Edge Betweenness, a clustering algorithm that has been used in social and ecological networks, has the advantage that it uses properties from the whole graph, including nodes with low degrees, resulting in a clustered graph that contains more global information. An example of a graph in which it is best to use the edge-betweenness algorithm, is a network from yeast two-hybrid datasets, where it is common to find nodes with single edges [67].

The edge-betweenness algorithm, unlike other approaches which try to find which edges are the most central communities, focuses on those that are least central and between the communities. The algorithm follows the flow below:

1. Calculate the betweenness for all the edges
2. Find and remove the edge with the highest betweenness.
3. Recalculate the betweenness for the edges that are affected by the removal.
4. Repeat from step 2.

The algorithm stops when there are no remaining edges.

Edge-Betweenness uses Newman's [67] fast algorithm to calculate the betweenness. This algorithm can calculate betweenness in time O(mn) where m represents the edges and n the vertices. That means that the edge-betweenness runs, at worst case, in time $O(m^2n)$, since the Newman algorithm must be repeated for every edge[67].

4.5 Label propagation

Label propagation algorithm (LPA) has two big benefits, its almost linear running time and simple implementation, and achieves this using only the graph structure[67]. The linear time is especially important since there are increasingly larger networks. LPA has been used on the web, social networks, biological networks, etc. The clustering quality is measured with modularity, and it is acceptable between 0.3 and 0.7.

The algorithm is based on the different labels that each node has. At every step, each node decides to change its label to the one that the largest number of its neighbors has, and so, step by step the labels spread. At the end of the algorithm, the nodes that have the same label belong to the same cluster. The maximum number of nodes that a cluster can have is set by the number of nodes outside of the cluster and inside the cluster.

LPA has the disadvantage that it can return multiple results including ones with low quality. This is the result of using local minimum and it has been proven that the number of local minima is larger than the number of network nodes. One solution to the problem is to avoid unnecessary updates, since it has been shown that after 5 iterations, 95% of the nodes are at the correct cluster. So, using the boundaries of the currently existing communities, it is easy to save time.

An interesting point that makes the algorithm simple, is that the way that LPA works is analogous to an epidemic or an idea or an opinion spreading, since we assume that the node adopts the label of most of its neighbors. However, someone that adopts a new idea from a neighbor often follows the one that has more connections, and the highest number of potential

information sources. Similarly, when a node takes its neighbor label, it takes into account not only the number of nodes that have the same label (people with the same opinion), but also how the other neighbors are connected[67].

5 Transformations and graphics

In this section, we will analyze the principal 2D and 3D linear transformations. The main coordinate transformations are the translation, the scaling, and the rotation.

Let a three-dimensional coordinate system S1, and let a point P into S1, which is expressed as (x,y,z). Let a second coordinate system S2, where the same point P is expressed as (a,b,c). These coordinates can be expressed as linear transformations of the coordinates of the first coordinate system S1.

$$a = w_1x + w_2y + w_3z + t_1$$

$$b = w_4x + w_5y + w_6z + t_2$$

$$c = w_7x + w_8y + w_9z + t_3$$

The above relations define a linear transformation of S1 to S2 which can be written as:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

The vector t = (t1, t2, t3) refers to the transfer of the origin of the axes of S1, so that it coincides with that of 2. The matrix W is a constant and allows the recalculation of the basis vectors.

5.1 Translation

This transformation is about moving an object in a certain direction by a certain distance t_x, t_y for 2D or t_x, t_y, t_z for 3D. Thus, a point p1= (x, y) in 2D, can be translated by

t = (t_x, t_y) as

$$p_2 = p_1 + t = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

or in 3D can be translated by t = (t_x, t_y, t_z)

$$p_2 = p_1 + t = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \end{bmatrix}$$

A schematic representation of such a transformation is given in Figure 5.1.

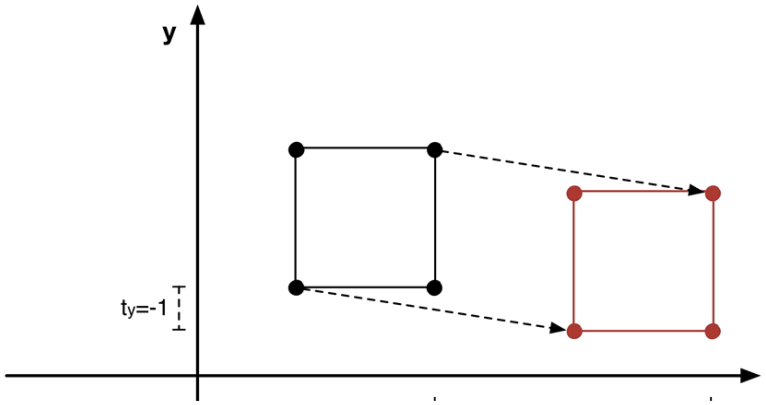


Figure 5.1 | Example of a point transfer transformation in 2D.

5.2 Scaling

This transformation is about changing the size of objects. For the implementation of the scale change transformation, the amounts of enlargement or reduction s_x, s_y, s_z for the axes x, y, z respectively are necessary. When the scaling factors s_x, s_y, s_z have a value equal to unity, then there is no change of scale, while we speak of enlargement or reduction, when they have a value greater or less than unity. In 2D, a point p1= (x, y) can be scaled using a scaling matrix S. Thus, the new point p2 is obtained as follows:

$$p_2 = S \cdot p_1 = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{vmatrix} s_x \cdot x \\ s_y \cdot y \end{vmatrix}$$

Similar, in 3D is the following:

$$p_2 = S \cdot p_1 = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x \cdot x \\ s_y \cdot y \\ s_z \cdot z \end{bmatrix}$$

A geometric example of such a transformation in 2D is represented in Figure 5.2. Of course, when it comes to a shape with many points, the transformation should be applied to each of them.

In the case of uniform scaling of a central point other than the origin of the axes, a complex transformation should be used. In other words, the object must first be moved to the beginning of the axes (translation), then the desired scaling should be performed, and finally, it should be moved back to the desired position.

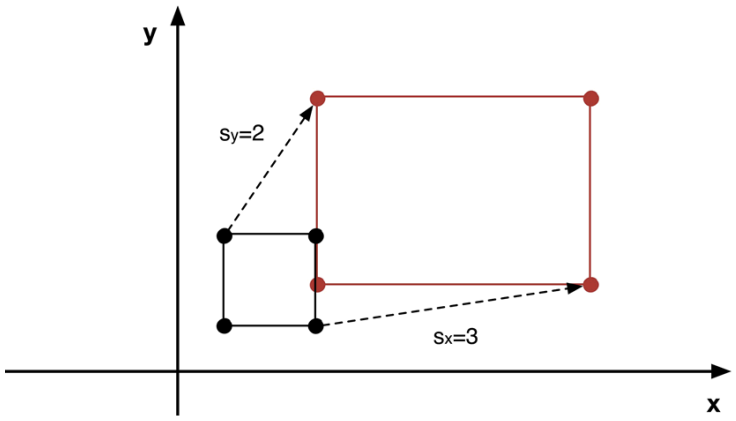


Figure 5.2 Example of object scaling transform with scaling vector (3,2)

5.3 **Rotation – 2D**

This transformation in 2D concerns the rotation of an object around the origin of the axes. Necessary to implement the rotation transformation is the rotation angle (let θ). A rotation is characterized as positive, when it becomes counterclockwise. Suppose, then, that the point p1= (x, y) must be rotated by degrees and transformed to the point p1= (a, b) (Figure 5.3).

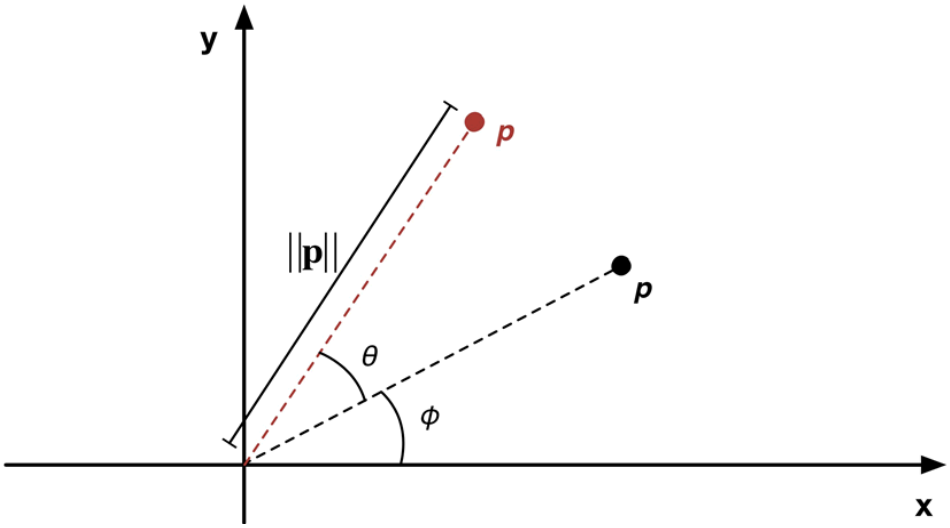


Figure 5.3 Point rotation by θ degrees

Then the linear rotation transformation will be given by the formula

$$p_2 = \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \cos\theta & -sin\theta \\ \sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R\left(\theta\right) \cdot p_1$$

where R is the transformation matrix.

As with scaling, with rotation, the coordinates of an object are changed. To rotate with a reference point other than the origin of the axes, a complex transformation should be used. That is, the object must first be moved to the origin of the axes (translation), then the desired rotation must be performed and then the point must be repositioned in the desired position.

5.4 Rotation – 3D

If we compare 2D rotation with 3D rotation, the most important difference is that in three dimensions rotation is not defined around a point, but around an axis. Thus, to implement the rotation transformation in 3D, both the rotation angle and the rotation axis are necessary.

Considering the case of rotation around an axis such as z for example, then we practically have a two-dimensional rotation, in which the x and y coordinates change, while z remains

constant. This can be modelled by expanding the 2x2 rotation matrix into a 3x3 matrix using the unit matrix for the third dimension. So, the corresponding rotation matrices R_x, R_y, R_z will be:

$$R_x\left(\theta\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -\sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix} \quad R_y\left(\theta\right) = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \quad R_z\left(\theta\right) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.5 **Complex transformations**

The order in which transformations are applied is very important and complex transformation tables can be used for this purpose. For example, it suffices to note that matrix multiplication does not carry the commutative property, so the order in which the matrices are multiplied matters. Therefore, when we want to successively apply a set of transformations M_1, M_2 .... M_n, we must calculate the complex matrix by multiplying inversely $M=M_n...M_2M_1$

6 NETWORK FILE FORMATS

A network can be described and stored in multiple human and computer-readable ways. Apart from the simple file formats such as the tab-delimited, CSV, SIF, Excel, and adjacency matrix, several others like the BioPAX [68], SBML[69], PSI-MI[70], CML[71], and CellML[72] have been introduced for biological data and semantics. For example, SBML, which stands for Systems Biology Markup Language, is an XML-like format for storing and parsing biochemical networks, as well as for describing biological processes. BioPAX stands for Biological Pathway Exchange and is made for the representation of biological pathways at the molecular and cellular levels. The PSI-MI format is used for the data exchange related to molecular interactions and CellML is used for describing mathematical models. GraphML[73] is an XML-like file format that consists of unordered sections related to a network's node and edge elements. Each node has a distinct identifier, whereas each edge is described by a source and a target node. Additional attributes, such as an edge weight, or a label, can also be included in the schema. The JavaScript Object Notation (JSON) format is a generic and widely used non-biological file format and is popular for web-based applications, or web-server asynchronous communication, and data exchange. However, it is worth mentioning that Cytoscape.js[74] accepts JSON formats for network visualization. Last but not least, the Nexus and the Newick file formats are standard ways of representing trees. While NDEx[75] is an open-source framework for the sharing of networks of many types and formats, file-format-specific parsers are available [e.g., Bioconductor[76]) rBiopaxParser[77], rsbml, RPsiXML and others]. Examples of such file formats are shown in Figure 6.1.

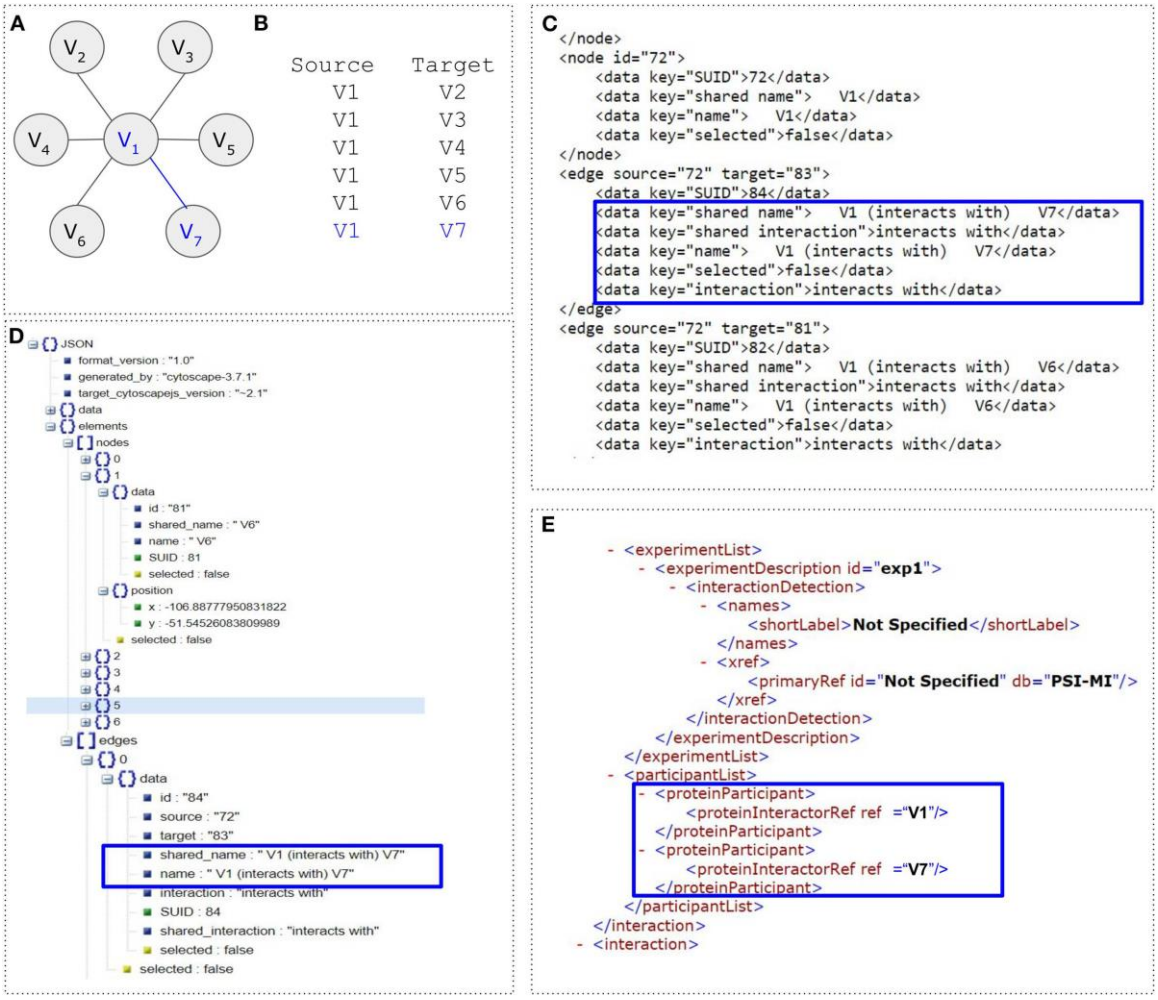


Figure 6.1 Examples of file formats. The brain network of the C. Elegans worm (Watts 1998). A) Simple undirected graph consisting of seven nodes (V = 7) and six edges (E = 6). B) Network in Tab-delimited file format. C) Network in GraphML file format. Blue box highlights the interaction between nodes V1 and V7. D) A cytoscape.js graph encoded in JSON. E) Network in PSI-MI file format.

7 NETWORK VISUALIZATION AND TOOLS

Network visualization has a key role in understanding, exploring, communicating, and identifying patterns in an interactome such as important edges, highly connected nodes, or communities. For this purpose, similar to the different network layouts, a plethora of approaches have been introduced for the visualization of networks depending on the type of network that needs to be drawn[1].

7.1 Visualization tools

There is a variety of tools that create beautiful 2D and 3D networks and each of them has its own advantages. Some of these tools are open source which means not only the tool is free to use by everyone, but their code is also accessible to the community. That means that their development process has brought together everyone that wants to be involved and because of that they constantly improved. That advantage incentivizes the users to keep working with open-source tools. [78].

7.1.1 Gephi

One open-source tool for graph and network analysis is Gephi. Gephi renders the networks using a 3D engine. This engine allows displaying large networks in real-time but also speeds up network exploration. Briefly, some of its features are spatializing, filtering, navigating, manipulating, and clustering [79].

Gephi as mentioned before utilizes a special 3D rendering engine for real-time graph rendering. In this way, the application uses the computer's graphic card (GPU) and does not require any CPU computing. Gephi team has also developed two algorithms. The first is the Force Atlas (and Force Atlas 2 [80]) algorithm, a special force-directed algorithm that allows changing speed, auto-stabilize, repulsion, inertia, and gravity in real-time. The second algorithm, Label Adjust, makes the labels avoid overlapping. It is important to note that at the same time the application can run multiple algorithms in separate workspaces. Also, it is possible to add a different and custom plugin in modules[79].

To upload a network to Gephi the user must import two spreadsheets one that contains the nodes, and one with the edges. After that, it is possible to change the size of the nodes according to their degree and use some plugins for their size gradient, color gradient, and color

clusters. One of the main features of Gephi is spatialization which gives more space to the graph using different layout algorithms such as Fruchterman Reingold, and Force Atlas 2.

After the visualization is over it is possible through this tool to display different statistics such as modularity, average degree, page rank, and betweenness centrality.

Another feature that Gephi has, is the geographical layout which allows the addition of a Latitude and Longitude to each node, and setting the Mercator and adding a map as a background. Last but not least, Gephi has also introduced the MultiViz plugin that offers multi-layer networks[81].

Gephi's latest version (0.9) is running with Java 6 and it is compatible with OS X (10.7 and further) and with Windows and can be found at https://gephi.org/. A biological network example with Gephi is shown in Figure 7.1.

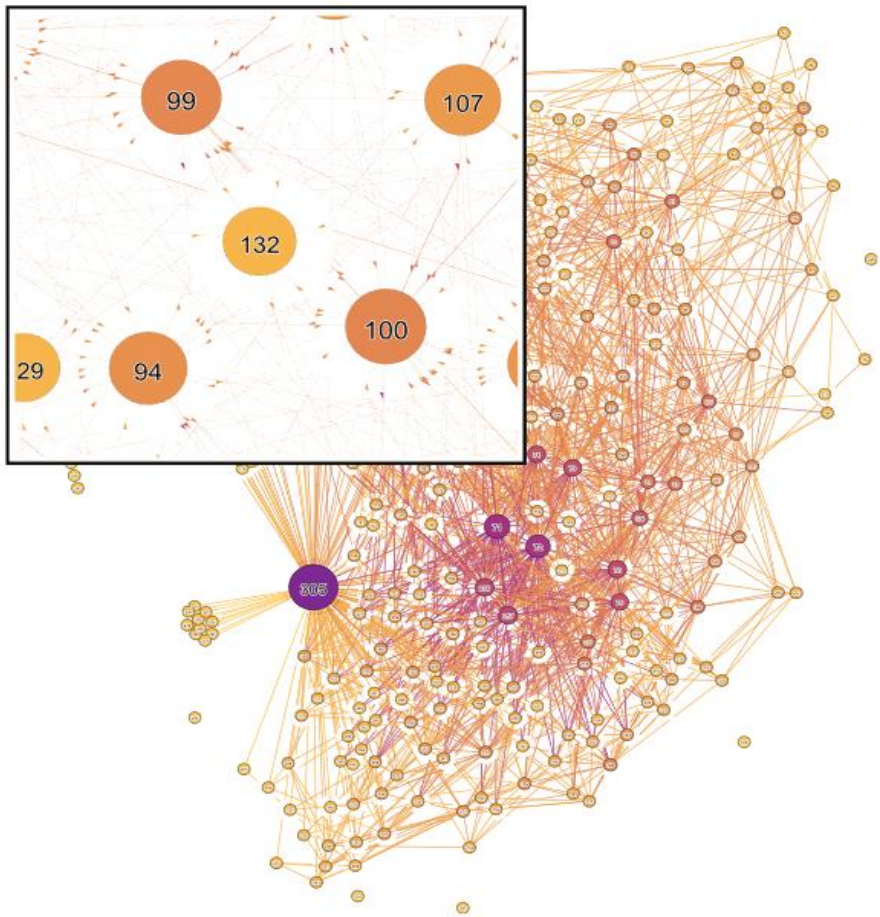


Figure 7.1 SVG File Exported from Gephi. The brain network of the C. Elegans worm (Watts 1998).

7.1.2 Pajek

One of the oldest visualization tools (the development started in 1996 by Andrej Mrvar) is Pajek (which means Spider in Slovenian[80]) which started as a visualization tool for social network analysis. Pajek can handle extra-large networks with up to 1 billion nodes and unlimited edges and the main purposes of the tool are the implementation of a powerful software that can handle large networks and efficient algorithms to analyze those and to support abstraction by decomposition of these huge networks into smaller ones[82].

Except for the main goals that are mentioned, Pajek contains a plethora of operations suitable for large networks such as extracting subnetworks, shrinking selected parts of networks, searching for connections, shortest paths, k-neighbors, maximum flow and fragments, computing centralities, clustering, generating distinct types of random networks, community detection but also some operations specifically for smaller networks such as generalized blockmodeling. Also, it is possible through the application to export data for further analysis in R, SPSS, and Excel.

Pajek supports different layout algorithms. The most used of them are Kamada-Kawai and Fruchterman Reingold optimization, VOS mapping, Pivot MDS, drawing in layers, and FishEye transformation.

It is important to note that there are two other Pajek versions in addition to the standard one, the Pajek-XXL, and Pajek3XL. Pajek-XXL has considered a special version of the classic Pajek in which memory consumption is more efficient. As a result, it needs 2-3 times less physical memory than the Pajek and it is more suitable for huge networks[82]. The capacity of nodes that Pajek-XXL can handle is 2 billion. Pajek3XL gives the user the capability to handle networks with 10 billion nodes since it uses a 64-bit integer for vertices' numbers instead of the 32-bit that the Pajek-XXL uses [48].

Even though Pajek started as a tool for social network analysis, it has been also used for citation and co-authorship networks, protein-protein interactions networks, transportation networks, and archaeological networks.

Pajek is possible to run on both Mac OS X, Linux, and Windows and downloads are available along with the documentation and supporting material available at its website: http://mrvar.fdv.uni-lj.si/pajek/. An Example of a network drawn in Pajek can be shown in Figure 7.2.

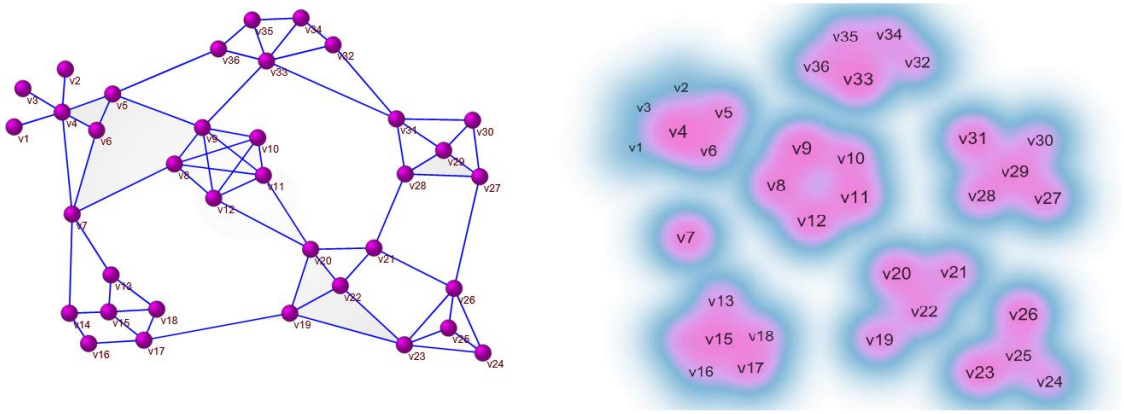


Figure 7.2 Network in Pajeck. Left: Simple network Right: The same network with VOSviewer SVG Density View

7.1.3 ONDEX

ONDEX is primarily a database system that uses graph-based analysis along with text mining and semantic database integration. As mentioned before ONDEX is a tool that can be used for visualization, large-scale database integration, sequence analysis, and text mining but it can be also used for analysis and interpretation of experimental results, a particularly useful feature for complex biological data. ONDEX is freely available for download at http://www.ondex.org/.

ONDEX has three primary features. The first is the ability to easily handle large graphs with several thousand elements both in the backend and in the front end. The second is the support for external graph libraries. This offers access to different libraries that provide a variety of algorithms such as different layout algorithms. The third feature is the graph filter. It is common at biological questions to not need all the data for answers, so the option to filter at the backend level by not importing the datasets that are not needed, but also at the front end to have more clear graphs.

ONDEX based its architecture on the Internal Graph Object, an independent data structure to represent the graphs. This object offers import and export interfaces that are needed for data exchange, a layout interface that allows access to different layout algorithms in different libraries, a filter interface for the filtering algorithms, and lastly the graph library adapter which is used to wrap different graph libraries. [83]

Except for the ONDEX software suite, there is a web-based implementation Ondex Web which offers the option to use that tool without downloading any software but also offers some new features. Ondex web is an open-source tool, written as a Java client applet and the application and its source code is available at http://sourceforge.net/projects/ondex/files/OndexWeb/.

Some of the new Ondex web features are easy-to-search functions, plugins that update the network UI, context-sensitive menus for the edges and nodes of the network, and direct support for loading data at XGMML (Cytoscape format), NWB, and Pajek formats. Besides these features, the online tool also offers two new strategies for the analysis and exploration of biological networks: the bottom-up and the top-down. Bottom-up is targeted and better used for exceptionally large datasets and top-down for small and medium-sized networks.

The Ondex Web is only limited by client resources, which means that the more memory the virtual Java machine the largest the networks can handle, but the application provides the user with a warning for larger than the usual networks to adapt if he wants the strategy to a bottom-up and use a simpler and faster layout[84].

7.1.4 Tulip

TULIP is an open-source visualization framework currently at version 5.6, and it is used for the analysis and drawing of huge graphs[85]. This C++ library is designed based on extensibility and reusability. The software is available at https://tulip.labri.fr/site/ and its code at GitHub https://github.com/Tulip-Dev/tulip.

TULIP architecture is based on five packages. The first one is the TULIP core library whose main purpose is to handle and manipulate the data sets, the entities, and relationships and the function to access those. Besides that, it contains some standard generic algorithms, for example algorithms needed for import and export, and finally plugin mechanism. Based on the previous TULIP version the current data structure is based on five requirements: property sharing, aggregation, observable data structure, state management, and alternative graph model.

The second TULIP package is the graphics one. The TULIP development team implemented the TULIP graphics library based on the OpenGL-based multilayered rendering engine[86]. There were two main reasons for not using an external engine. The first is that external engines cannot handle graphs with more than 500,000 elements in less than 256MB of memory

and the second one is that in 2000, when the project started, 3D rendering engines were not easily available and powerful enough.

The third package is the TULIP GUI. The main goal of the GUI is to allow users to go back and forth between the network and the statistics to help them understand how the metrics are represented. The architecture is built based on Model-View-Controller (MVC)[87] pattern. This means that the software is split into three independent parts, the model stores the information, the view presents the information, and the controller is responsible for the communication between the views and the model.

The fourth one is the Run-Time Environment. This package solves the difficulty that programmers may meet if they use multiple plugins. Besides the Plugin management, it also manages the model and makes the software available cross-platform.

Finally, the last one is python integration. Even though the tool is built in C++ since 3.5 TULIP offers Python binding of all its key features (Creation and manipulation of graphs, Storage of data on graph elements, Creation of interactive visualizations, The ability to write plugins in Python, Integration within the Python nebula). The bindings are available from PyPI[88] and can be installed as a standard Python package[85]. An example of a network at Tulip can be found in Figure 7.3.

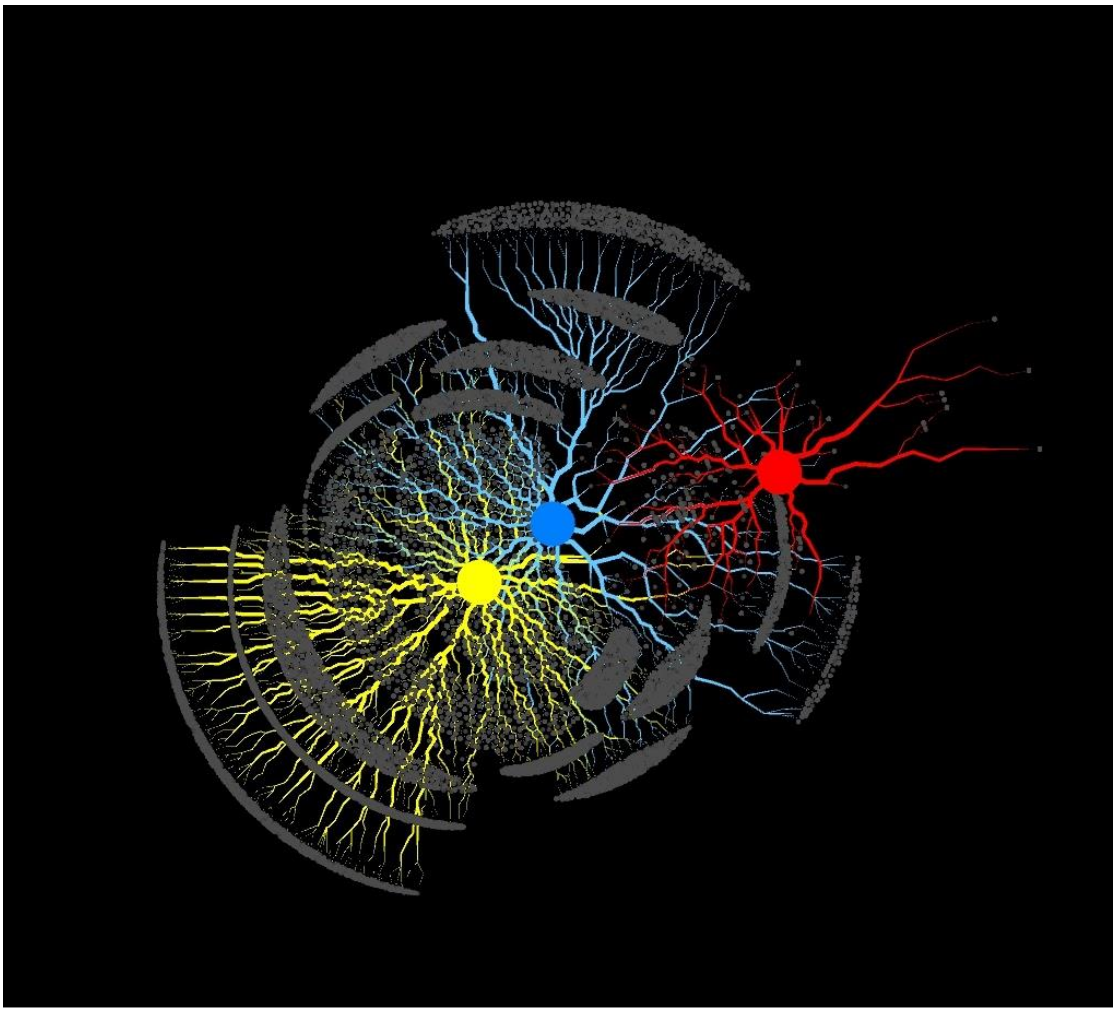


Figure 7.3 A network in Tulip

7.1.5 OmicsNet

OmicsNet is a web-based tool that is targeting multi-omics networks [89]. Through this tool, the user can build, visualize, and explore multi-omics data within the context of molecular interaction knowledge. Recently OmicsNet was updated to version 2.0 with new features and is available at www.omicsnet.ca.

Developed in 2018, the first version of the OmicsNet started as a tool that accepts a variety of biological features (also combined if needed (Figure 7.4)) such as genes, proteins, and metabolites to help the researchers to create and visualize biological 3D networks[90]. After the update, OmicsNet 2.0 include also 2D networks and has enriched its supported data types to accommodate the needs of its user.

The first step is to upload the data. The user can choose the omics type that they want (genes, proteins, transcription factors, miRNAs, metabolites, and the new additions microbial taxa, LC-MS peaks, SNPs) from the annotated table panel. Then the user must choose a database (STRING[40], InnateDB[91], IntAct[92], TRRUST[18], JASPAR[93], TarBase[94], miRTarBase[95], KEGG[20], Recon3[96],

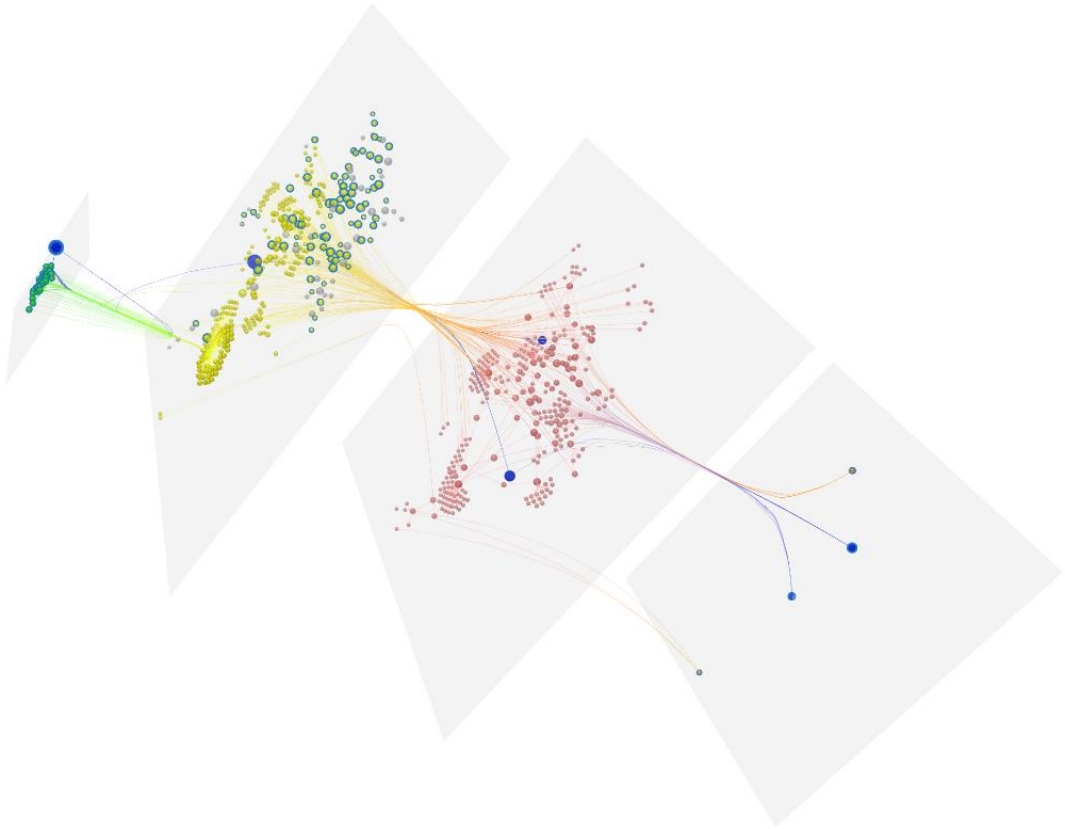


Figure 7.4 A multi-layer 3D network in OmicsNet. A multi-omics 3D network (one omics per layer)

AGORA[97], etc.) to move to the network creation and visual analytics. It is also possible to upload common graph files from the OmicsNetR package or another network tool e.g., Cytoscape. The graph layouts that are available at the application are based on the igraph package [53].

Finally, OmicsNet 2.0 offers some of its R functions through the OmicsNetR package that can be found here https://github.com/xia-lab/OmicsNetR. Among the R functions an R command history panel has been added. This feature can help the users to download locally some of the tool functions and recreate, and customize the network before uploading them to the online tool for navigation [89].

7.1.6 Graphia

Graphia is a powerful open-source visual analytics application that targets large and complex datasets[98]. Graphia was created from BioLayout Express, an application that was specifically for transcriptomic data and pathway modelling[98].

Graphia was developed focusing on the following main features: data and operating system agnostic, fast, scalable, and extensible algorithms, dynamic rendering, 3D graph visualization, correlation graphs as an essential function, attribute handling and visualization, advanced analysis capabilities, and lastly, a user interface that is easy to navigate. With these in mind, the application is built in C++, using the standard OpenGL library [86]. The architecture is separated into the core application code, plugins, shared plugin code, and the third-party libraries that are used.

Graphia can support multiple file formats, some examples are BioPAX OWL ontology, JSON graph, GraphML, Graph Modeling Language, MATLAB data file, pairwise graph formats in .txt or .layout, adjacency matrix in .csv and .tsv, as well as numeric data for correlation analyses. In the application, a layout algorithm is running frequently, and it can be stopped by the user manually. This allows the graph to change dynamically when is updated, and the nodes and edges are altered. The disadvantage of the feature is that there are cases the graph that may be transformed quickly, so the tool offers the option to slow down the transition. Another major feature that Graphia offers, is that at first the graph is rendered in 3D since this is the most advance and complex one, and then it is rendered in 2D for more closely connected nodes (Figure 7.5). Graphia is a cross-platform application that can be downloaded from https://graphia.app/ [98].

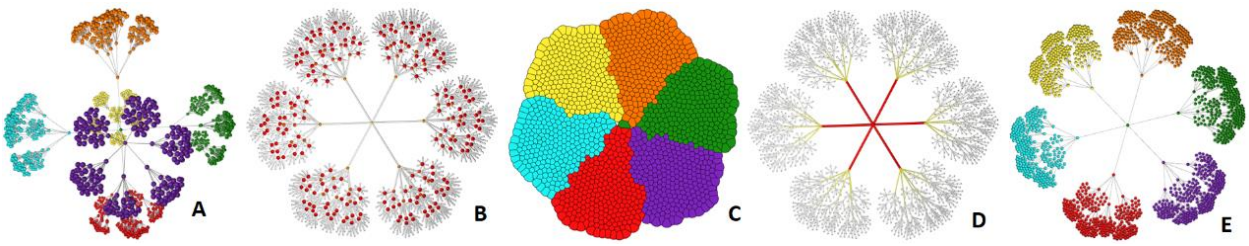


Figure 7.5 Different graph visualisation options in Graphia. A) 3D perspective view, smooth shading with visualisation of node attribute. B) PageRank values where G-I are continuous attributes, so a colour spectrum and size gradient are used for node display. C) compressed 2D layout, flat shading, showing node overlap view. D) Eccentricity values G-I are continuous attributes, so a colour spectrum and size gradient are used for node display. E) 2D view, smooth shading.

7.1.7 NORMA

The NetwORKk Makeup Artist or NORMA is an open-source web visualization tool with the ability to handle multiple networks[98]. In addition to these, it can handle multiple annotations simultaneously and provides topological analysis. NORMA is a general-purpose application that also targets non-experts users. The application is available at http://norma.pavlopouloslab.info/ and its code at https://github.com/PavlopoulosLab/NORMA.

The procedure that the user must follow to use the application is first to upload a network file that he needs from the upload tab. After that, he can also upload an annotation file or a node coloring file. It is important to note that the user can upload multiple networks and annotation files at the same time. After the files that are needed the network information are uploaded, the user can navigate to the network tab to see the generated interactive network or the automated community detection, that is calculated with Fast-Greedy, Louvain, Label-Propagation, Walktrap or Betweenness clustering algorithm. The user can also, upload an annotation file, a file that has information about predefined clusters, communities, subgraphs, etc. Then they can navigate to the annotation tab and visualize this information as convex hulls (2D and 3D), pie-chart nodes, and node diagrams. Finally, there is also the option to view different topology network analysis features in the topology tab such as the number of edges and nodes, average path length, clustering coefficient, modularity, centralization degree, and others. These are calculated through the igraph[53] library since the application is written in R and shiny[99].

After the first release of NORMA, the next version NORMA 2.0 was released with the offer of different layout strategies, that enrich the possibilities the previous version had. These layout strategies help the user to have clear annotated groups or areas of interest. These strategies are the virtual nodes, the gravity, and lastly the super nodes[100]. An example of a network in NORMA can be found in Figure 7.6.

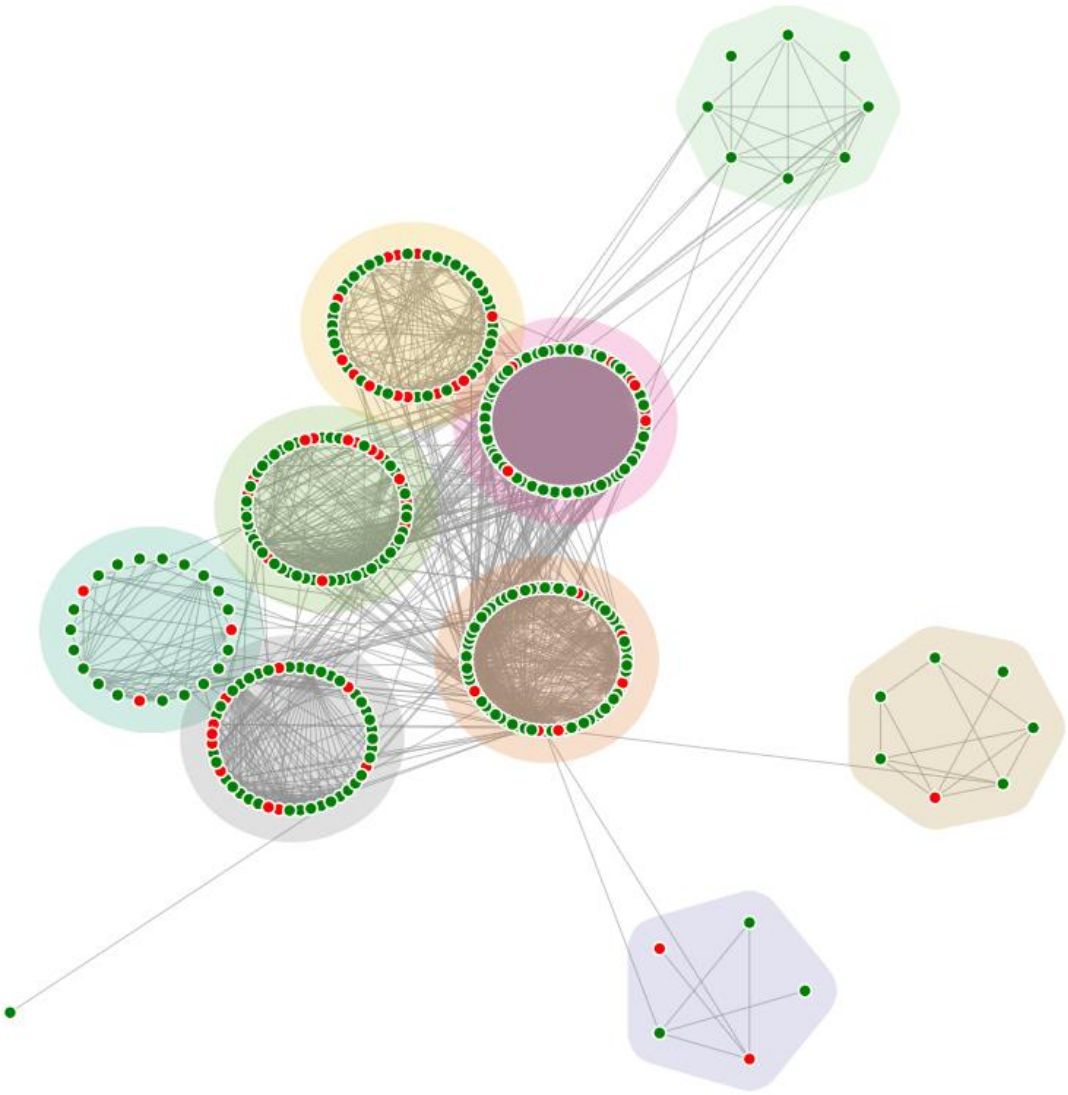


Figure 7.6 Clustered Network in Norma

7.1.8 Cytoscape

Another open-source software 2D visualization tool is Cytoscape. This software is divided into the core part and the plugins, with the core having the basic functionality and through the plugins, the core is extensible to a larger variety of abilities[54].

The core functionality includes the graph data, the representation of the graph and the integrated data, and the tools that are needed for the selection and filtering along with the UI. More specifically, Cytoscape creates pairs (name, value) to map node and edge names to specific data values. This model is called Attributes. The next feature that Cytoscape has at its core

functionality is the annotation that is specific node/ edge descriptions. This information is usually taken from a database or a repository. Like other tools, Cytoscape supports different layout algorithms such as hierarchical layout, circular layout, and spring-embedded layout, those with node and edge appearance changes (color, shape, size, thickness, and style) make the graph more visible pleasing, and easy to understand the information that is visualized. The last core functionality is the selection and filtering, which helps the users, study more easily the graph and emphasize various parts of the graph if needed. For example in Figure 7.7 there is a subset of a taxonomy tree. The available filtering is according to the name, list of names, or an attribute or more complex queries. A worth mentioning Cytoscape feature is the edge bundling that is available and with it, the graphs are clearer and easier to read.

The most important feature that Cytoscape has, is the plugins. These enrich the core functionality with new algorithms, or different network analyses, semantics, UI changes, and other visualization tool combinations. It is important to note that these plugins may have a different license agreement, so they do not have the necessary open-source code[54]. Cytoscape 3.9 is available for download at https://cytoscape.org/index.html and its code is at https://github.com/cytoscape.

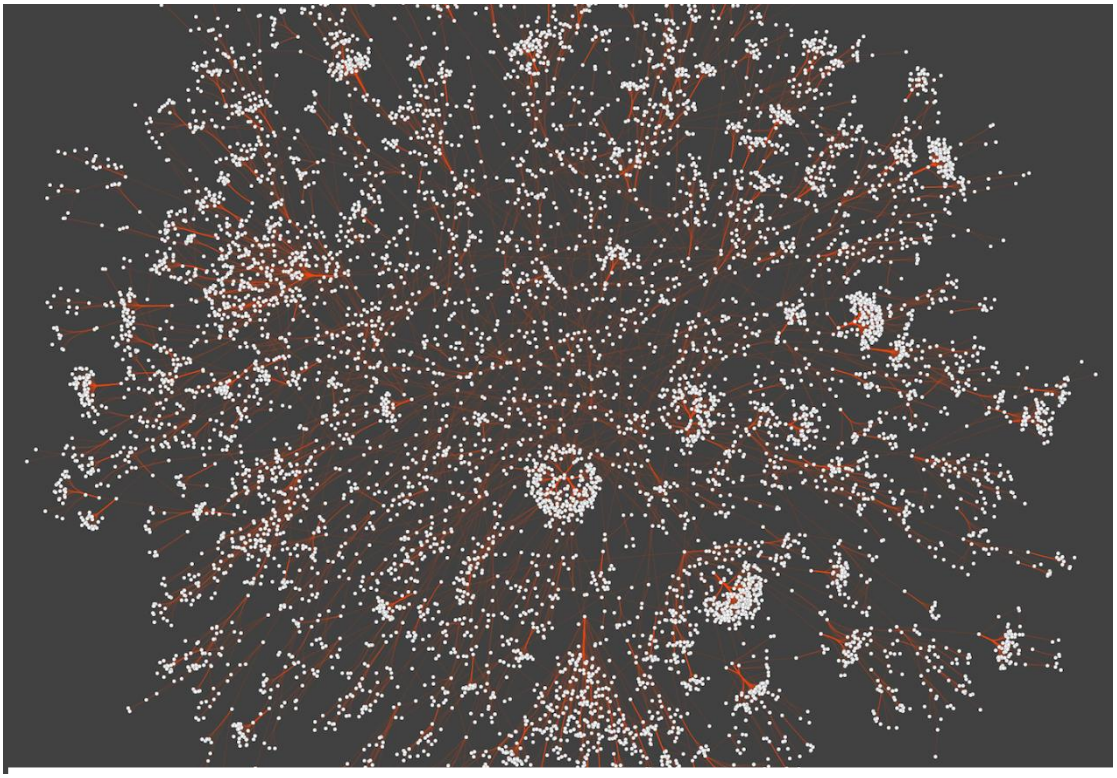


Figure 7.7 A Taxonomy Tree in Cytoscape 3

Besides the downloadable version, Cytoscape also offers a web version, Cytoscape.js [74]. Built with standard web technologies such as HTML5, CSS, and JS Cytoscape.js also provides an application programming interface (API) to offer the possibility for other developers and researchers to have easier access to their tools and easier graph integration to the application.

As previously noted, Cytoscape.js is developed in JS, but to accommodate different JS systems, it also uses hooks to different JS libraries, for example Node.js, Require.js, npm, Bower, jQuery, and Meteor. The CSS stylesheets and HTML DOM elements are accessible via the JS core API.

Cytoscape.js has a variety of features, such as multiple graph types (traditional, directed, undirected, multigraphs, etc.), the possibility to modify the graph elements, graph traversal, multiple graph theory algorithms, (for example, shortest path), ranking and centrality measures, stylesheets, built-in gesture support for mouse and touch-based devices, event binding, animations, compound nodes, the option to import and export as JSON and also export the graph as an image, different layouts such as random, circle, concentric spread, etc. and lastly, extensibility through different widgets. Figure 7.8 there is an example of using a widget for a different layout. The documentation and the project are available at http://js.cytoscape.org [74].

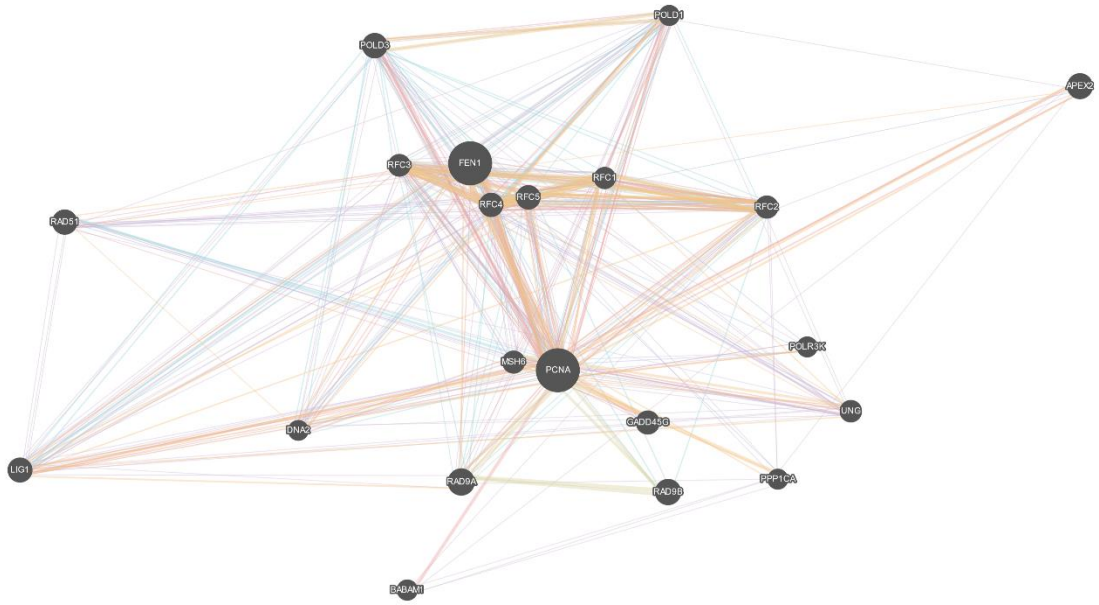


Figure 7.8 Graph in Cytoscape.js. This is a demo of a graph of gene-gene interactions that uses Cola.js for layout and Cytoscape.js for its graph model and visualization.

7.1.9 Web viewers

Except for the visualization tools, there are also some web viewers, whose main purpose is not graph visualization, but they offer them as a feature and are worth mentioning.

7.1.9.1 STRING

STRING is a relational database built by the Swiss Institute of Bioinformatics (SIB), Novo Nordisk Foundation Center Protein Research (CPR), and European Molecular Biology Laboratory (EMBL), that contains the precomputed global resource to help explore and analyze functional associations between proteins[101]. At the current version 11.5, the database contains more than 14 000 organisms and includes physical and functional interactions[40]. The sources vary from text mining of the scientific literature, databases of interaction experiments and annotated complexes/ pathways, computational interactions from co-expression and conserved genomic context, to finally systematic transfers of interactions evidence between the organisms. STRING is available at https://string-db.org/, through Cytoscape, the Bioconductor package within R, and through its REST API.

The user can search interaction by a protein name or multiple names, by one or more Amino Acid Sequences, and by proteins with values or protein families. In addition to those, the user can filter the results by organism, score, and interactors. The score is a number between zero and one and shows if the association is biologically meaningful given all the evidence that is benchmarked and scored.

STRING offers the following viewers for each result: Network, Experiments, Databases, Text mining, Cooccurrence, Coexpression, Neighborhood, and Fusion. It also offers a Legend with valuable information about the associations such as functional enrichments, exports, and other features depending on the viewer.

The main viewer is the network one and contains additional to the previous features, network statistics, K-means clustering, or MCL clustering. This network is a multi-edge network where each node is a protein and each edge is a functional link for example gene fusions, cooccurrence across genomes, a neighborhood in the genome, experimental/ Biomedical Data, etc. (Figure 7.9). The score and extra information are available on each of those links. For each protein, STRING may offer along with some information (identifier, organism) about it, the AlphaFold model, PDB structure, and homology model. Additionally, there are the following extra actions that are included in STRING's functionality for every node: "Re-center network on this node",

"Add This Node to input nodes", "Show protein sequence" and "homologs among STRING organisms"[40].

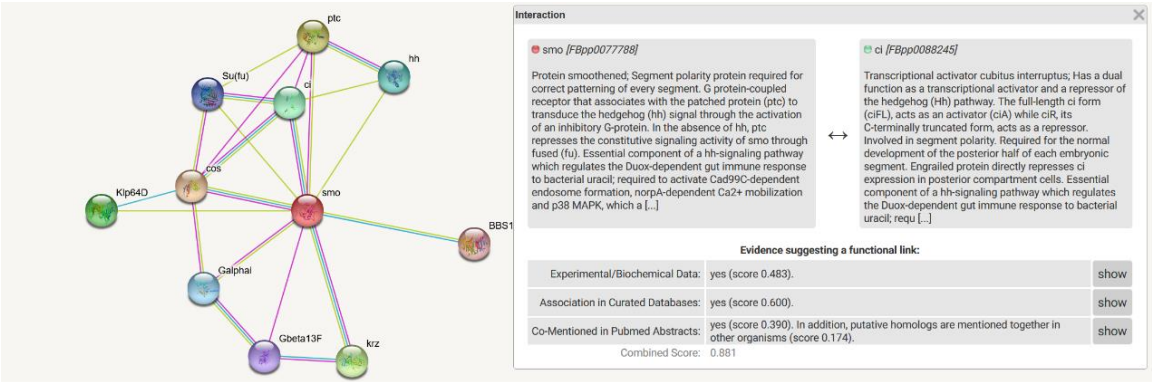


Figure 7.9 Network in STRING. smo search result and information about the smo and ci interactions.

7.1.9.2 STITCH

STITCH (Search Tool for Interacting Chemicals) is a database like STRING, created by the European Molecular Biology Laboratory (EMBL), Swiss Institute of Bioinformatics (SIB), and NNF Center for Protein Research (CPR) and contains Chemical – Protein Interaction Networks. STICH is available at http://stitch.embl.de/ and through its API. In its latest version STITCH, 5 includes more than 9600000 proteins from 2031 eukaryotic and prokaryotic genomes and 430 000 chemical compounds without including different stereoisomers[41].

THE UI looks like the one STRING has, so if a user knows how to navigate that database, it is easy to navigate to STITCH. The search is available by one or more items (chemical or proteins) or one or more protein sequences or chemical structures.

The network view contains a network where the nodes are proteins or chemical structures, and the edges represent their interactions. Protein-protein interactions are shown in gray, chemical–protein interactions in green, and interactions between chemicals in red. For each interaction (edge) there is a description of the nodes and the functional links with scores, titles, and more information about them. The weight of the edge is dependent on the number of functional links that the edge represents. For each node, there is available information about the item that it describes, some actions such as "Re-center network on this node", "Add This Node to input nodes", "Show protein sequence" in case the item is a protein, and some structure models or chemical models according to the item (Figure 7.10).

Additional to the network viewer, STITCH offers a text mining viewer, an experiments viewer, a database viewer, and a coexpression one, along with a legend about the information, statistical analysis, and data tables[41].

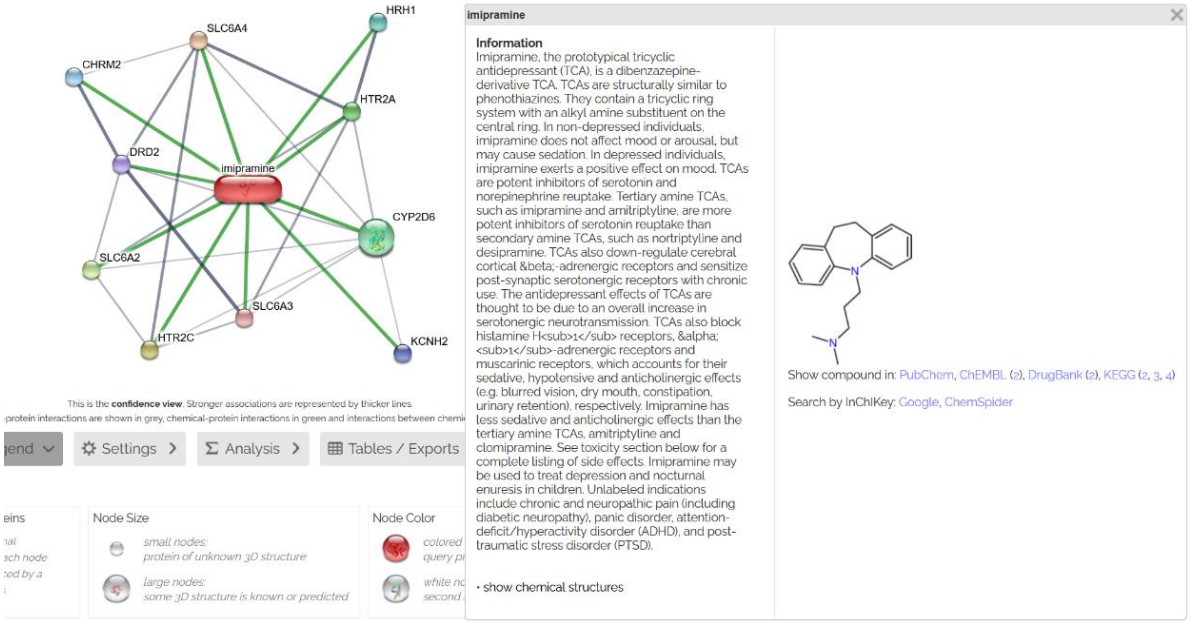


Figure 7.10 Network in STITCH. "CN(C)CCCN1C2=CC=CC=C2CCC3=CC=CC=C31" and selected only imipramine, with information about this compound.

7.1.9.3 INTACT

IntAct is a free open-source molecular interaction database that provides analysis tools for molecular interaction data[92]. The database contains entries, either curated from the literature, or direct data depositions and is supporting both IMEx- and MIMIx-level curation. IntAct is available at https://www.ebi.ac.uk/intact/home.

A user can search in IntAct by a gene name, taxon ID, UniProt ACs, Pubmed ID, protein names, Complex ACs or GO terms. If the results contain less than 1500 interactions then the database creates a 2D multi-edge network with those, and along with it some network visualization features.

For each interactor (node) with hover, the user can access some information about it, such as name, id, type species, and AC. There is also the possibility to change the layout from force-directed (default option) to circular or Bubbles. Additionally, there is the option to replace the multi-edge functionality, with one edge per link, where the weight of one of them shows the

number of interactions. Lastly, there is also the option to group some nodes per species (Figure 7.11).

The database also offers a legend that explains the color coding and the variety of the types that the network contains (different node and edge types).

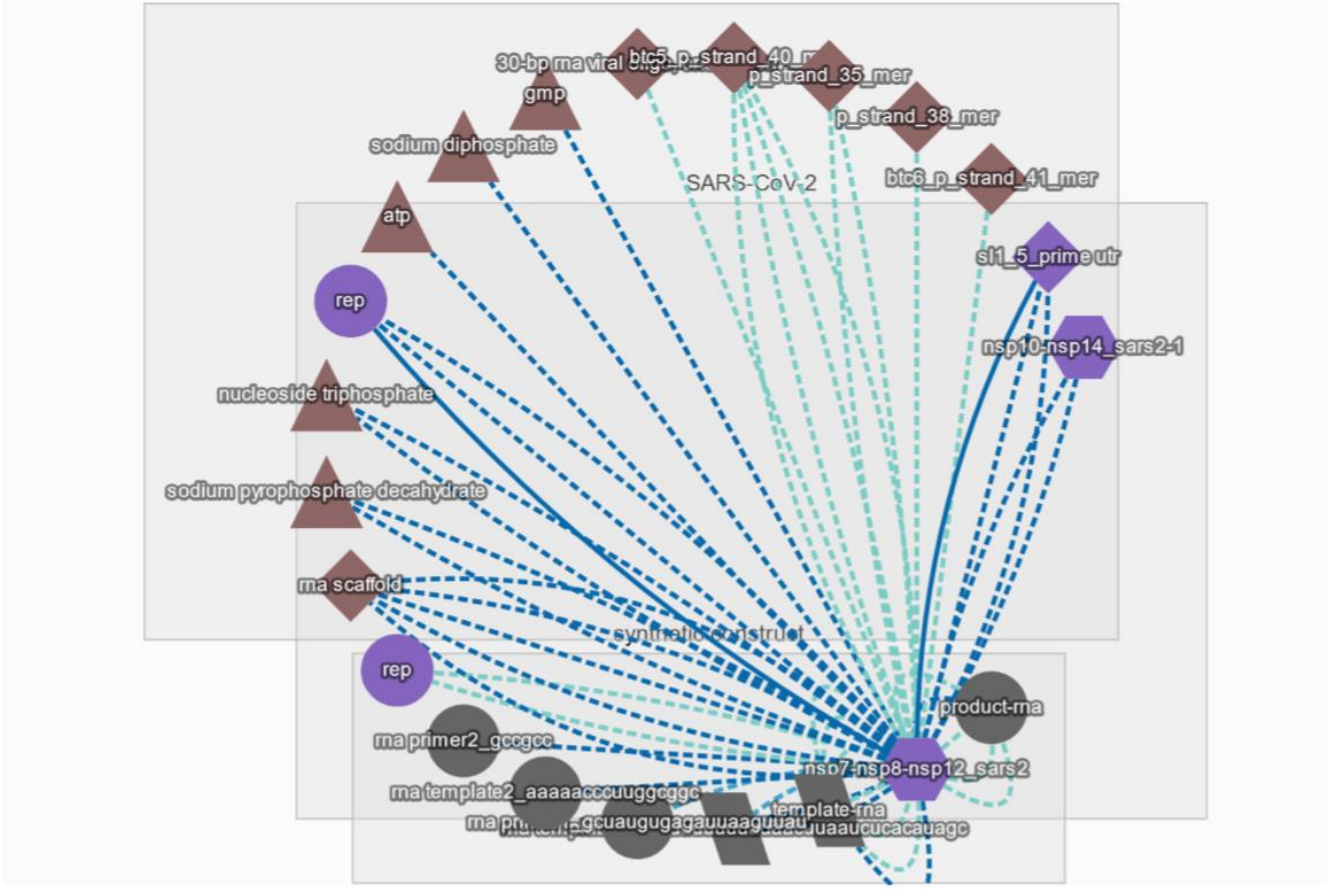


Figure 7.11 Network in IntAct. Search result of the Complex ACs "CPX-5742" with circular layout and group by species.

8 ARENA3D^{web}

In biomedical and biological studies, a common need is to capture multiple heterogeneous information for different repositories in the same view e.g., genomics, proteomics, and metabolomics. Some of the tools that are capturing this need are the Arena3D standalone version [102] [103], Py3plex[104], Mull[105], and MuxViz[106]. Even though the demand for multi-layer networks appeared in biological and biomedical studies, Arena3D^{web} is a general-purpose tool that can be used for knowledge integration, representation, transfer, and communication[107].

Arena3D^{web} is an interactive and dependency-free web visualization tool. The feature that makes Arena3D^{web} unique is the multilayered graphs in 3D space that covers the need to represent heterogenous data in networks[107]. The application has been developed in R, Shiny, and JavaScript and for the backend calculation, it uses the igraph[53] package and is accessible at http://arena3d.pavlopouloslab.info or http://arena3d.org.

8.1 UI / UX

Arena3D^{web} offers a plethora of features, that allows users to create aesthetically beautiful networks that manage to capture their heterogeneous data and explore the information through the application's menu. The application's interface consists of 9 tabs. These are:

8.1.1 Home

Home is an introduction page with some information about the application (Figure 8.1).

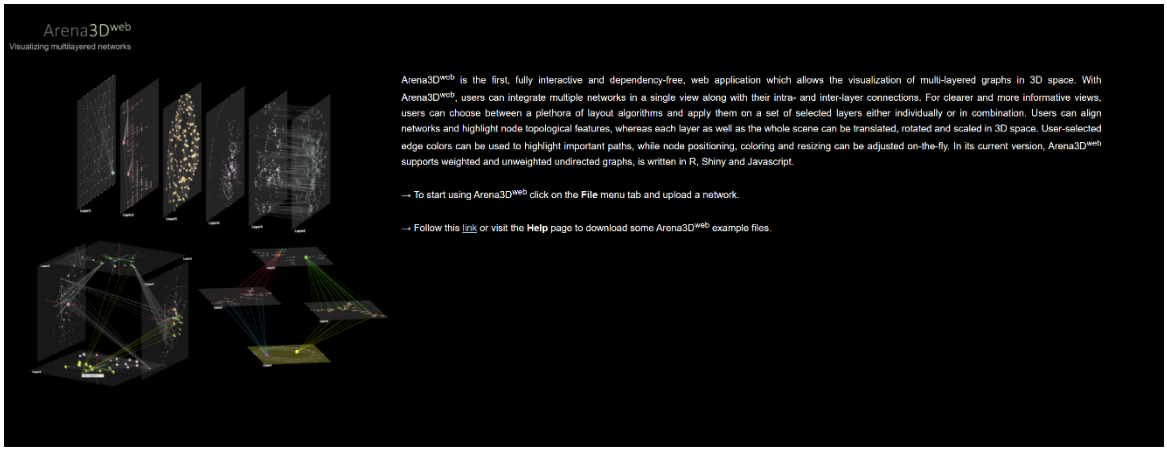


Figure 8.1 Home page in Arena3Dweb

8.1.2 Main view

This is the page that contains the network once it is uploaded. When a file has been uploaded, a side panel also appears with some useful navigation controls for the scene, layers, and nodes (Figure 8.2). The user through this panel can navigate easier in the network and modify the positions as needed. The possibilities vary from rotating the scene and the layers to mouse and keyboard navigation controls. The application includes the following hotkeys: the user can zoom in/out by mouse scrolling, the network can be translated by dragging with the mouse or by

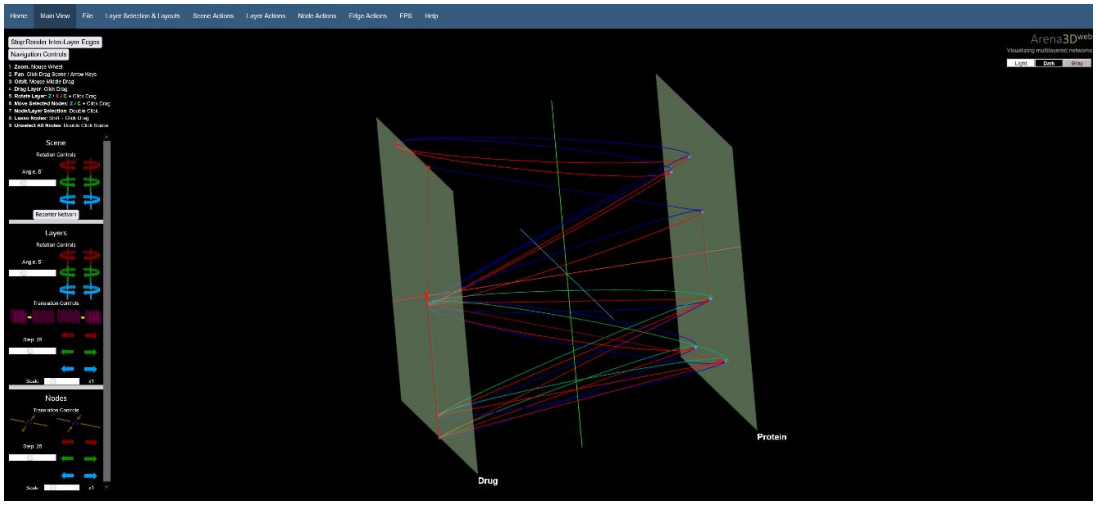


Figure 8.2 Main View in Arena3Dweb. Application's main view. At the top of the page there the main menu. At left there are the navigation control which includes the "Navigation Controls" button for hide/show the navigation panel and the "Stop:Render Inter-Layer Edges" button hides inter-layer edges to greatly improve rendering performance, General instructions on network hotkeys, and the scene, layer and node controls. At the top right there is also a menu for changing the theme. The options are light, dark, and gray.

pressing the keyboard's arrow keys, and the network view is also orbital by dragging while holding the middle-mouse button, the user can move a layer by click and dragging it, the user can rotate selected layers at X(red), Y(green) and Z(blue) axis by holding the respective hotkeys and click-dragging, the user can move selected nodes on a layer by holding the hotkeys (Y, Z) and click-dragging. Priority is given to selected nodes over layers, the user can select/deselect individual nodes or layers by double-clicking on objects. The node or layer flashes when the user hovers over it and changes color when selected, for a batch node selection, the user may hold the Shift button and click-drag to apply a lasso selection, and by double-clicking anywhere on the scene, all selected nodes and edges are deselected.

8.1.3 File

At this option, the user can upload a network in multiple formats, and optionally upload a node or edge attributes format to customize the colors (Figure 8.3). More specifically, this option includes four file upload fields, the upload Network where the user can upload the default application file, the Load Session where the user can upload a JSON file exported from a previous Arena3D[web] session and a node and edge attributes file upload fields. Finally, from this menu option, the user can also save the network with their modifications.

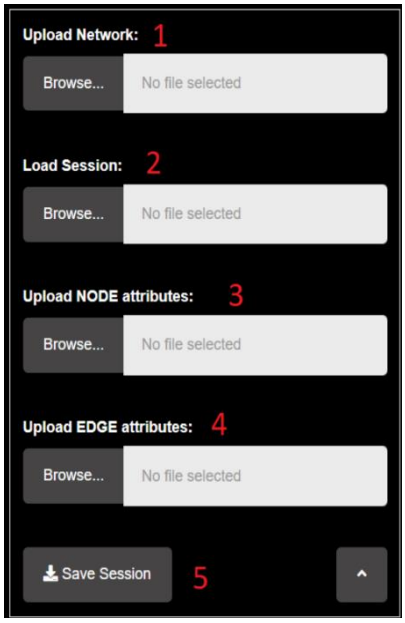


Figure 8.3 "File" menu option in Arena3Dweb. 1. The Upload Network option allows the user to upload network data in the Arena3Dweb format. **2.** The Load Session option allows the user to load network data from an exported JSON object (see 5.). **3.** The Upload NODE attributes option allows the user to upload annotation data regarding the nodes of the current network view. **4**. The Upload EDGE attributes option allows the user to upload annotation data regarding the edges. **5.** The Save Session button allows the user to save the current view in JSON format (The format is described at the API tab). The network can be restored by importing the relative saved object (see 2.).

8.1.4 Layer selection & layouts

The layer Selection & Layouts tab offers the possibilities for layer selection and all the options that are needed for the layouts and clustering algorithms. This control panel allows the user to select, deselect and hide layers, show layer-specific node labels, as well as apply layout and clustering algorithms and node scaling based on network metrics, on subgraphs of the network. More specifically, in Figure 8.4 it is possible to see the menu and the options that are offered. The layout and clustering algorithms that are offered are described further below.

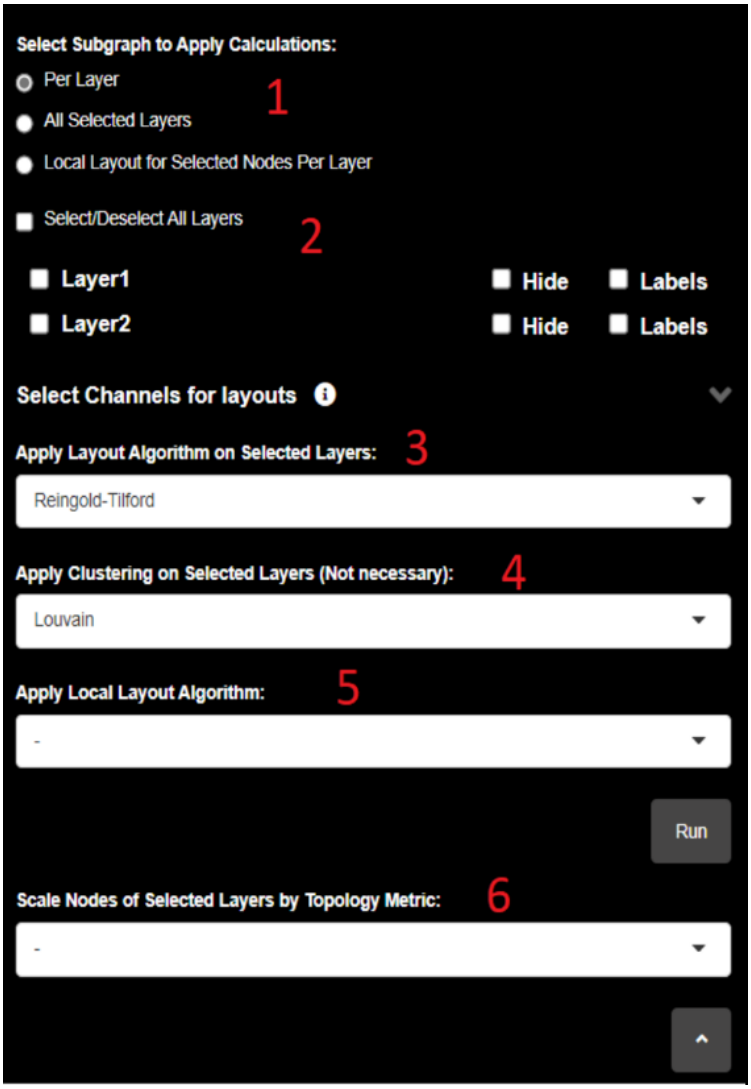


Figure 8.4 "Layer Selection & Layouts" menu option in Arena3Dweb. 1. This consists of a group of 3 exclusive options for subgraph calculations, upon which, layout algorithms (3, 5), clustering algorithms (4) and node scaling (6) is applied. The "Per Layer" choice treats each selected layer (2) as an individual network. The "All Selected Layers" choice treats all selected Layers (2) as one, combined network. After the execution of a layout or scaling algorithm, nodes are mapped back to their respective Layer. With this option, the application of force-directed layout algorithms allows network alignment among the different layers. The Local Layout option for the Selected Nodes Per Layer choice allows layout and scaling algorithms to be applied on a selected sub-group of nodes, per each selected layer respectively. **2.** The "Select/Deselect All Layers" checkbox allows the user to quickly select or deselect all available network layers. After the user uploads or imports a network, a grid of n x 3 checkboxes is created, where n is the number of network layers and 3 are the available actions for each layer; the 1st column allows the individual selection/deselection of layers, the 2nd column allows the user to hide individual layers and their inter-layer connections, and the 3rd column allows the user to show node labels per layer. **3.** A list of available layout algorithms of the igraph package, to apply on selected layers (2) based on the execution mode of (1). **4**. A list of available clustering algorithms of the igraph package, to apply on selected layers (2) based on the execution mode of (1). **5.** A list of available layout algorithms of the igraph package, to apply as local layouts on clusters (4) based on the execution mode of (1). Visible when a clustering algorithm has been selected. **6**. A list of available network metrics of the igraph package, used for node-scaling, to apply on selected layers (2) based on the execution mode of option (1).

8.1.5 Scene actions

At Scene Actions, the user can find the scene-related actions, such as showing and hiding the scene coordinates system, background, and the new features that will be analyzed further below (Figure 8.5).

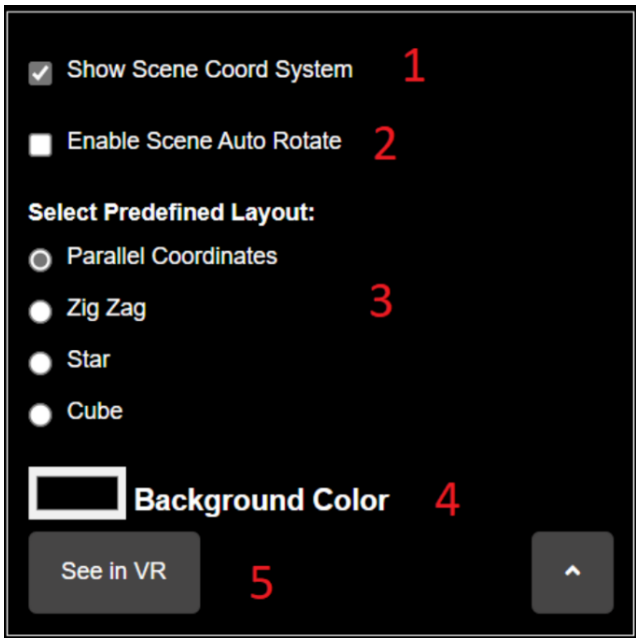
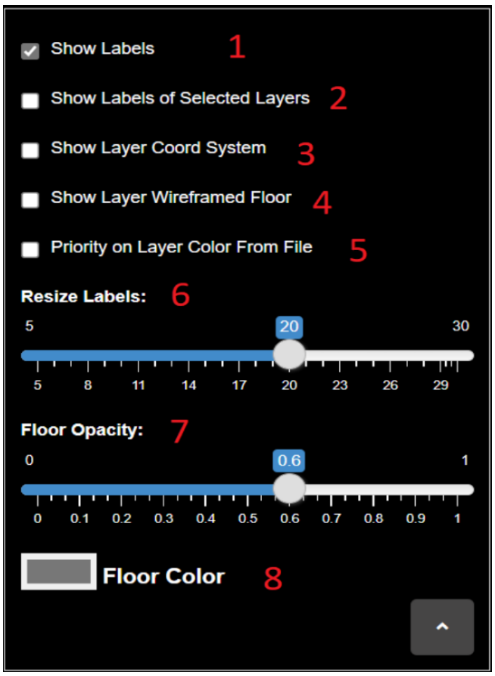


Figure 8.5 "Scene Actions" menu option in Arena3Dweb. 1. A checkbox that toggles the visibility of the scene coordinates system. **2.** A checkbox that enables scene auto rotate. (The user must enable it and then from the Navigation Controls click the arrows to rotate). **3.** A radio button with the following predefined layouts: Parallel Coordinates (default option from new files), Zig Zag, Star and Cube. **4.** A ColorPicker for the background of the network. For bright background colors, be sure to set higher opacity values for layer floors (Layer Actions tab) and edges (Edge Actions tab). **5**. A button to see the network in VR. Works only in the online version of the tool.

8.1.6 Layer actions



The layer Actions tab contains all the layer-related actions. Some examples are the actions that handle layer labels, Coordinate systems, wireframes), and layer floor color (Figure 8.6).

Figure 8.6 "File" menu option in Arena3Dweb. 1. This checkbox allows the user to show or hide all layer labels. **2.** This checkbox gives the option of showing the labels of selected layers only. Option (1) has priority over this option. **3.** This option toggles the coordinate systems -X (red), Y (green), Z (blue)- for all layers. **4.** This option allows an alternative visualization for layer floors, in wireframe mode **5.** This option gives priority to uploaded layer colors from a JSON object. **6.** This slider resizes layer labels. **7.** This slider changes layer opacities in [0-1]. **8**. This is a ColorPicker button for painting layer floors.

8.1.7 Node actions

The node Actions tab contains like the previous tabs, all the node-related actions, and from there the user can modify the network's nodes and their labels. Also, from this tab, it is possible to search and select multiple nodes by their name (Figure 8.7).

Figure 8.7 "Node Actions" menu option in Arena3Dweb. 1. This option allows the user to select/deselect all nodes. Selected nodes can then be translated in 3D space via the Navigation Controls, and via the Layer Selection & Layouts action tab can be either rearranged in a local layout or rescaled based on network metrics. **2.** This option allows the user to view every node label. This is an option that demands heavy processing power due to the constant redrawing of labels. Ensure that this is enabled only in small networks and in combination with the 15FPS option of the FPS action tab. **3.** This option allows viewing only the labels of selected nodes. Priority is given in option (2) over this option. **4**. Selected nodes are highlighted in a chartreuse color. Deselecting this option allows nodes to retain their original color (either from any uploaded node attributes or from their default layer color). Deactivating this option works in combination with activating option (3), to select certain nodes of a pre-colored path, view their labels without changing their color and extracting the respective image (either with the PrintScreen key, or by snipping or by right-clicking and then selecting the Save as Image option). **5.** This option allows resizing of node labels. **6.** This is the node search bar. The user can select multiple nodes by entering their names, without the need to specify layers, separated by commas. Any trailing and leading spaces are trimmed.

8.1.8 Edge actions

At this tab the user can access the edge actions, for example, the edge weight for inter and intra-layer edges, set the color priorities, enable and disable the direction, and edit the channel colors (Figure 8.8).

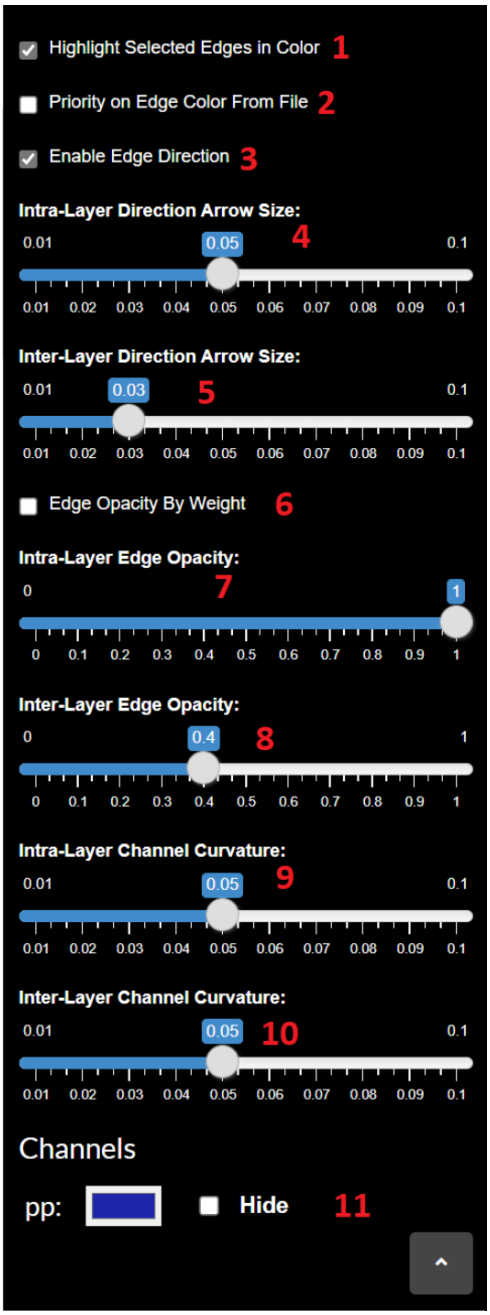


Figure 8.8 "Edge Actions" menu option in Arena3Dweb. 1. This option highlights the selected edges. **2.** This option toggles gives priority to the edge color that it is set on file. If it is not checked and their network has multiple edges, then the channel menu 11 is visible. **3**. This option toggles the graph direction from the source node to target. **4**. This slider changes the intra-layer arrow sizes. This is visible only if option (3) is enabled. **5.** This slider changes the inter-layer arrow sizes. This is visible only if option (3) is enabled. **6.** This option gives priority on any uploaded/imported values of edge Weights, which are being mapped in the [0-1] range and are assigned on edge opacities. If this option is unchecked, the edge opacity is decided through options (7) for intra-layer and (8) for inter-layer edges, respectively. **7.** If option (6) is unchecked, the intra-layer edge opacity is decided through this slider. **8**. If option (6) is unchecked, the inter-layer edge opacity is decided through this slider. **9**. If the graph is multi-edge, then this slider is visible and controls the curvature of the intra-layer edges. **10.** If the graph is multi-edge, then this slider is visible and controls the curvature of the inter-layer edges. **11**. If the graph that is uploaded is multi-edge then then the channel menu (a grid of n x 3) is created, where n is the number of channels and 3 are the available columns for each layer; the 1st column is the name of the channel, the 2nd column allows the user to change the color of each channel, the 3rd column allows the user to hide individual channels. This is visible only if option (2) is not enabled.

8.1.9 FPS

The FPS tab simply contains a radio button with different options for frame per second (Figure 8.9).

Figure 8.9 "FPS" menu option in Arena3Dweb. The available options are: 15FPS, for larger, more processing-heavy networks, 30FPS, the default option, 60FPS, for smaller networks that allow smoother rendering.

8.1.10 Help

The last tab contains all the information one needs about the Arena3D[web]. The section is separated into eleven sub-sections, including documentation for the menu options and the addition of a file format and API documentation and an example section with a variety of example files.

8.1.11 Latest update

At its latest version, Arena3D[web] contains major aesthetic improvements, in addition to the ones that already exist. The users already had the option to upload a different attribute file for edges and nodes. In those files, it is possible to change the color and size of the nodes and the edge color. Both of those files must follow the custom ". tsv" or ".txt" format that will be analyzed in the interoperability section. From the menu, as mentioned before, it was already easy to navigate to scene and layer settings to change the scene and layer background color and experiment with layer opacity. Additionally, now the tool offers three distinct color themes including a light, gray and dark mode that was already the default one (Figure 8.10 A). The light mode in particular enables users to produce publication-ready figures with a white background more easily, without the need to tweak the respective scene, layer, and edge colors manually. In addition, several predefined layer layouts are also offered. To this end, initial 3D layer setups are produced automatically and include a zig-zag, a star, and a cube layout (Figure 8.10 B). In the star layout, a virtual sphere of 360 degrees is equally divided by the total number of layers whereas, in the cube layout, each cube can contain up to 6 layers. In the case of more than 6 layers,

additional cubes are created and placed next to each other. In addition to the layouts, object manipulation has been significantly enhanced in this version. At the same option as the default layouts, the user can enable the new scene auto-rotate feature.

Raycasting via the three.js package has been implemented for easier layer and node selection with the mouse. Nodes change color on mouse hover as a visual cue, while hovered layers are highlighted in red. Layer and node selection through double-click is also offered, while layers can be dragged with the mouse on the 3D scene as an alternative to the navigation control action buttons. The scene orbit controls (middle-click drag) have also been enhanced, allowing for smoother 3D scene rotations compared to the previous version of the tool.

Finally, Arena3D^{web} now allows network exploration in VR mode (Figure 7.10C). VR views are static and can be accessed via a VR headset or a mobile phone with a gyroscope. For better clarity, layer floors are disabled while layer labels always face the user's camera. Notably, the VR view is always offered in a new tab and visualizes the running view of Arena3Dweb at any time point. For consistency, the objects' coordinate system is always adjusted in the VR view. It is important to note that the VR functionality is only accessible through the web application and not through the local version of Arena3D^{web}, since all required VR files are served internally through the API that will be analyzed in the Interoperability section.

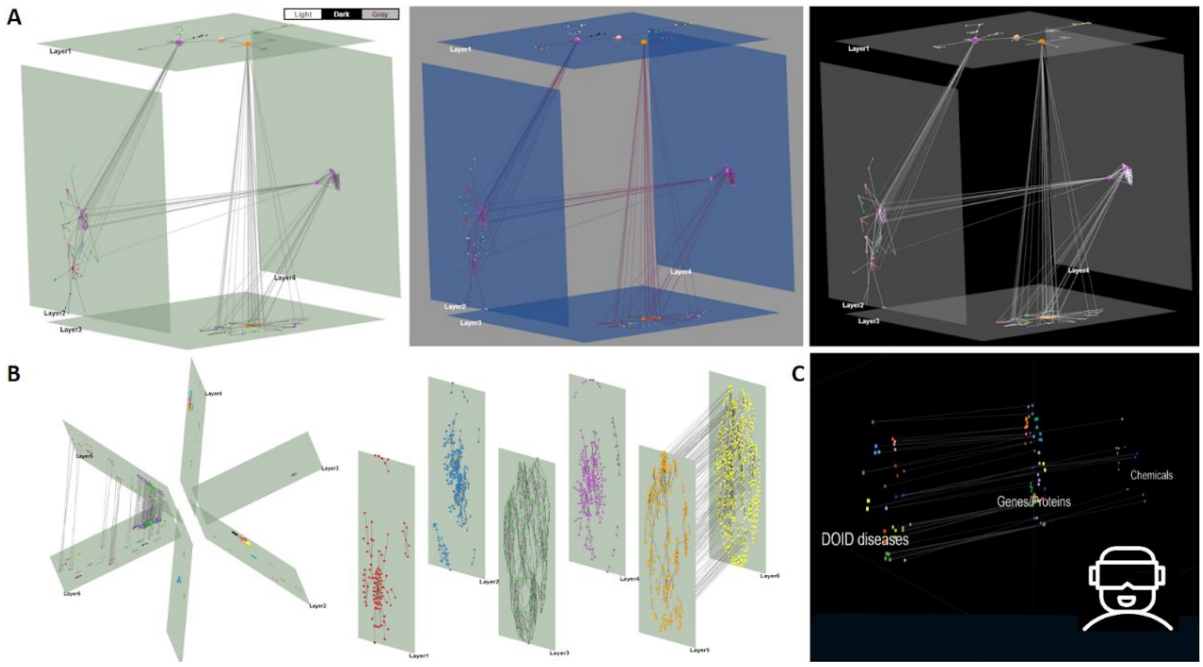


Figure 8.10 Network in Arena3Dweb. (A) Three default themes for a cube multi-layered network including (from left to right) a light, a gray and a dark mode option. **(B)** A star predefined layout (left) and a zig-zag layer layout (right) of six layers. (C) A VR view for a network with three layers (disease, proteins, and chemicals).

8.2 Directed graphs and multi-edge channels

In biology, two entities can often be linked with more than one type of connection. In the STRING database, as mentioned before, two genes can be co-expressed, co-mentioned in literature, or bind to each other. In a signal transduction network, a petri-net, or a pathway, signal tracing as well as up-and down-stream analysis can be monitored by following directed paths visualized by arrows rather than plain lines. Therefore, the support of directed and multi-edged graphs is significant.

In this version of Arena3D^{web}, both issues are addressed by allowing arrows and up to nine different types of connections between two nodes.

In the case of multi-edged networks, each information channel must be labeled in the Arena3D^{web} input file, and the corresponding edges are visualized as Bézier curves with distinct colors depending on the theme. These curves are created with *three.js* CubicBezierCurve3 objects. This object needs as input a start point, an endpoint, and two control points. The start point is the source node, and the endpoint is the target node of the edge. If the number of channels that the pair has is odd, then the middle channel is a straight line. The user can change the curvature of the channels through the Edge Actions tab. If the uploaded network contains channels, then two different slide bars are available, one for intra-layer channels and one for inter-layer channels. Both have as a default value the t factor 0.05 and their value can be from 0.01 – 0.1 with step 0.01. This t factor is multiplied by the distance between the source and endpoint this result helps calculate the distance that is needed for the extra two points that the CubicBezierCurve3 object needs. Through the Edge Action tab, the user can also change the default colors. These default colors are taken from Set3 and Set1 pallets that the RColorBrewer[107] package is offering. Set3 is used for the dark and gray theme and Set1 is used for the light theme. To change these default colors the user must disable the priority from the file (“Priority on Edge Color from File” checkbox) and the color channel menu will appear. If the user changes the theme, then the custom colors will change back to the default ones.

In the case of directed graphs, two intra- or inter-layer nodes can be connected via straight or curved arrows. To create the arrows, we used the three.js ArrowHelper Object. The object takes as input the direction from the origin that must be a unit vector, the point that the arrow starts, the length of the arrow, and as optional values a hexadecimal value to define the color, the length of the head of the arrow and the width of the head of the arrow. In Arena3D^{web} all the possible inputs are used except the width of the head of the arrow, where the default is

used which is calculated by three.js as 0.2 * head length. A problem with the use of ArrowHelper is that in its current version it is not possible to change the opacity of this object, and in Arena3D^{web} the opacity is used to set the weight of the edge. The solution to this problem is to create only the arrow's head with the ArrowHelper and keep the Line object for the rest of the arrow. The user can enable and disable the network's direction from the Edge Actions tab through the "Enable Edge Direction" checkbox. By default, in all the file formats the user must set the source and the target nodes. If this checkbox is checked then it is possible to change the size of the arrows' heads through two different slide bars, one for intra-layer edges and one for inter-layer edges.

8.3 Layouts

Once the main network has been loaded and rendered, users can choose between a variety of layout algorithms to adjust the node coordinates. By applying a layout algorithm on a selected layer or a set of selected layers individually, users can eliminate the intra-layer line crossovers without impacting the interlayer connections. On the contrary, when users decide to apply a layout algorithm on a set of selected layers in combination, then more focus is given to eliminating the interlayer crossovers. In the second case, the layout algorithm will handle all nodes from the selected layers and their connections as one unified network and will place them back on their originating layers once the layout algorithm has converged. The latter is a powerful feature for creating network pseudo-alignments as can be seen in Figure 8.11 at layers 5 and 6 and generating appealing and informative views. Finally, users are allowed to apply any of the offered layout algorithms locally on a set of selected nodes per layer. This functionality allows the user to emphasize the nodes that they need. An example of a local layout is visible in Figure 8.12. In this version, Arena3Dweb supports a variety of layout algorithms implemented as part of the R/igraph library. Briefly, these are:

- *Circle:* This layout is simple and places the vertices on a circle ordered by their vertex IDs.
- *Grid:* This simple layout places vertices on a rectangular 2D grid.
- *Random:* This function places the vertices of the graph on a 2D plane uniformly using random coordinates.
- *Fruchterman–Reingold:* It places nodes on the plane using the force-directed layout algorithm developed by Fruchterman and Reingold.

- *Distributed Recursive (Graph) Layout:* DrL is a force-directed graph layout toolbox focused on real-world large-scale graphs.
- *Multidimensional scaling:* It aims to place points from a higher dimensional space on a 2D plane so that the distance between the points is kept as much as possible.
- *Kamada–Kawai*: This force-directed layout places the vertices on a 2D plane by simulating a physical model of springs.
- *Large Graph Layout (LGL):* A force-directed layout suitable for larger graphs.
- *Graphopt:* A force-directed layout algorithm, which scales relatively well to large graphs.
- *Reingold-Tilford:* It is a tree-like layout more suitable for trees, hierarchies, and graphs without many cycles.
- *Sugiyama:* Like Reingold–Tilford, this layout algorithm is more suitable for hierarchies and layered-directed acyclic graphs. [107]

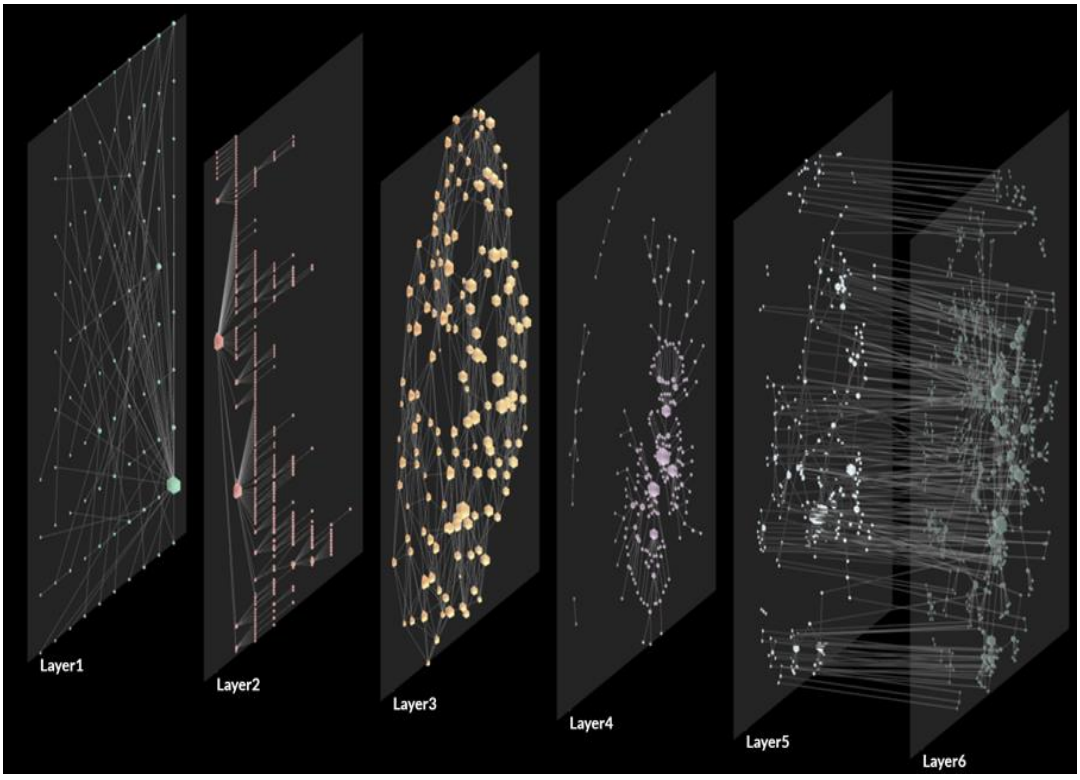


Figure 8.11 Network in Arena3Dweb Random networks with different topologies were drawn with the use of various layout algorithms supported by igraph and subsequently by Arena3Dweb. Nodes on layer 1 are placed on a grid. Nodes on layer 2 have been placed in a hierarchy using the Reingold–Tilford layout algorithm. Nodes on the first two layers are scaled based on degree. Layer 3 shows a small-world network drawn using the Kamada–Kawai layout algorithm. Layer 4 shows a scale-free network also drawn with the Kamada–Kawai layout algorithm. Nodes on layers 3 and 4 are scaled based on their clustering coefficient. Layers 5 and 6 have been combined and are drawn using the Fruchterman–Reingold layout algorithm. Both layers were handled as a unified network and nodes were placed back to their layers after the completion of the layout algorithm, thus giving a sense of a network alignment. The node scale of these two layers is relative to their degree.

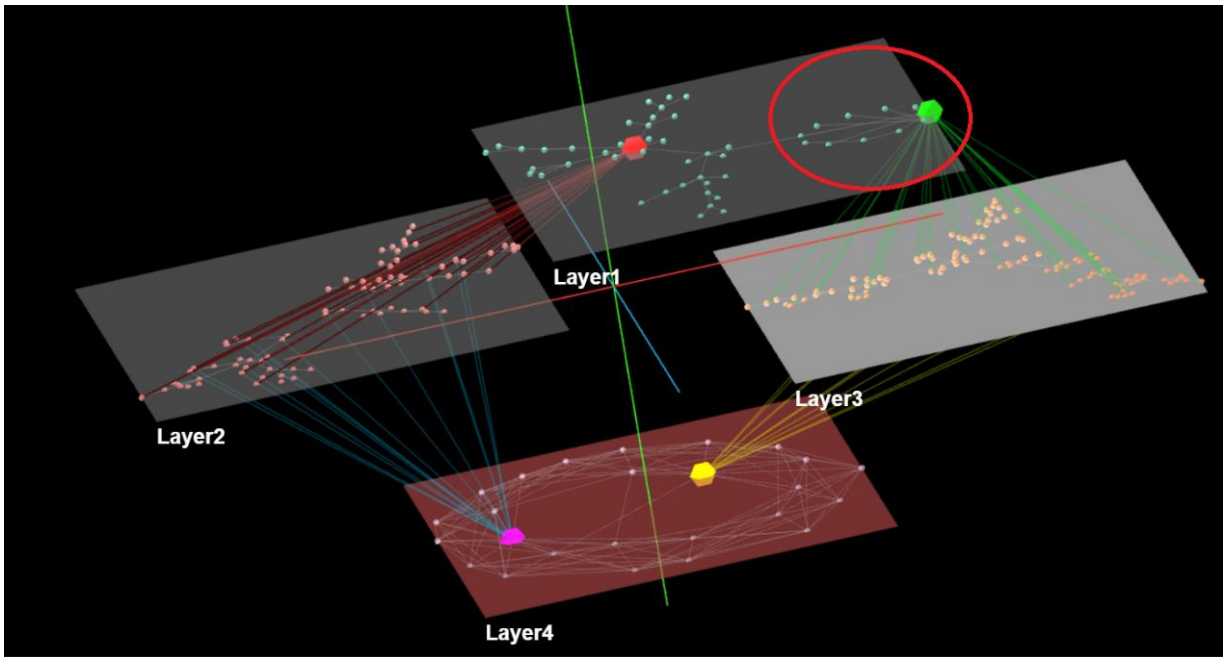


Figure 8.12 Network in Arena3Dweb Random networks with different topologies were drawn with the use of various layout algorithms supported by igraph and subsequently by Arena3Dweb and local layout. At Layer 3 Nodes inside the red circle have been positioned in a circled local layout, different from the rest of the layer.

8.4 Clustering

Arena3D[web] now supports four main clustering algorithms offered by the *igraph*[53] library in combination with various global and local layouts. These are:

- *Louvain*[62]: It is a multi-level modularity optimization algorithm for finding community structures based on the modularity measure and a hierarchical approach.
- *Walktrap*[65]: It performs random walks to detect densely connected neighborhoods. It assumes that random walks tend to restrict themselves in the same community.
- *Fast-Greedy*[24]: It optimizes a modularity score to identify densely connected communities.
- *Label propagation*[67]: It runs on a nearly linear time and tries to label the nodes with unique labels and update them by majority voting in the neighborhood of the node.

Users may apply any of these algorithms for a set of selected layers and channels to run them separately (per layer), or in combination (across layers). Upon clustering, the grouped nodes are placed on their respective layer according to the local layout selected from those offered by the *igraph* library and mentioned in the previous chapter. To minimize the cluster overlaps, Arena3D[web] follows NORMA's layout strategy named 'Super nodes' to visualize the clustered

groups. To this end, a virtual super node is created to represent each cluster, whereas the links among these super nodes (clusters) originate from initial node connections. After applying a global layout of preference, all the initial nodes will be placed around their respective virtual super-nodes according to a user-selected local layout. Figure 8.13 demonstrates a visual example of the described procedure.

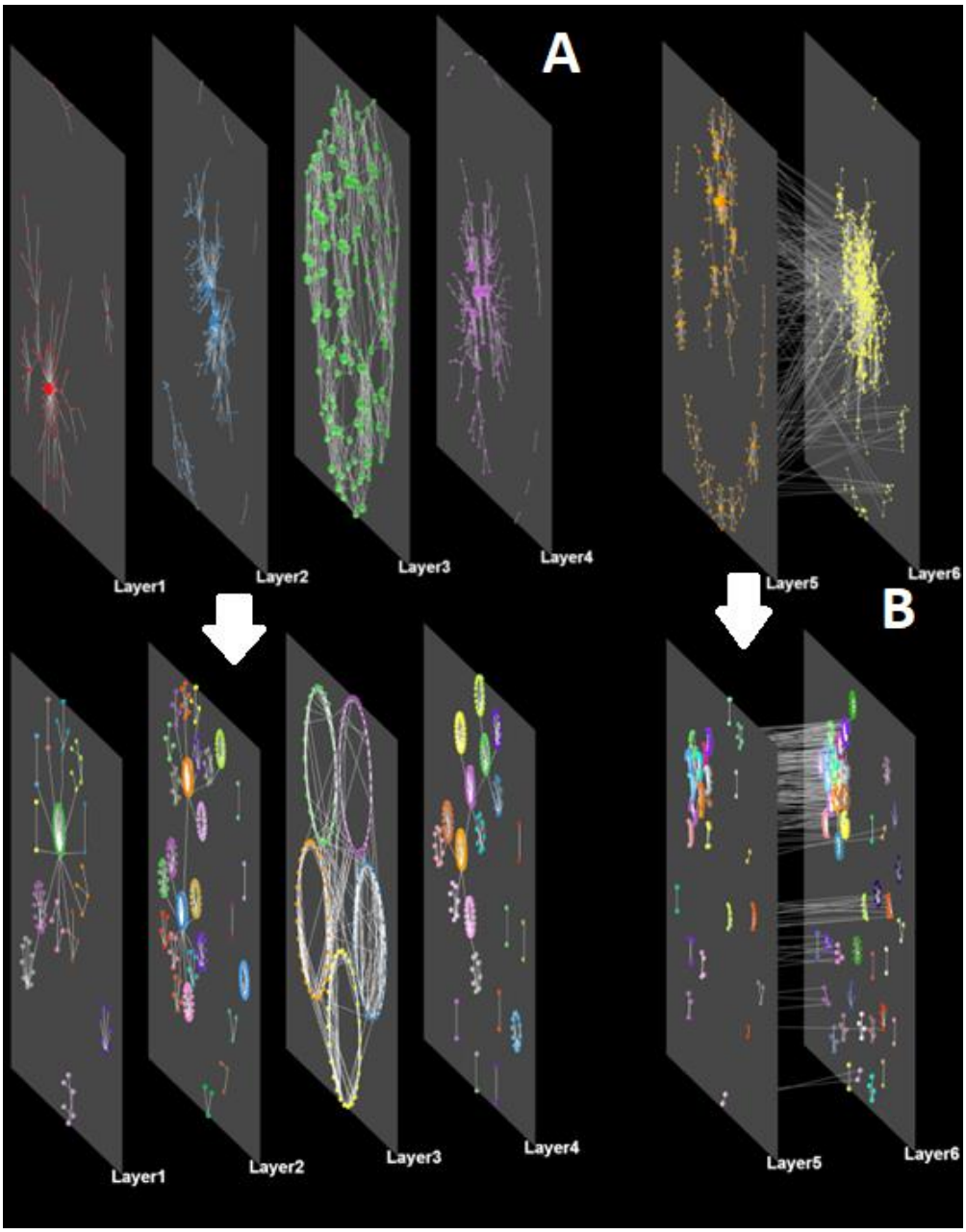


Figure 8.13 Network in Arena3Dweb. (A) A network consisting of four different layers (no inter-layer edges) before and after applying the Louvain clustering algorithm per layer. A force-directed global layout and a circular local layout have been selected. (B) A network consisting of two interconnected layers before and after clustering with the same parameters as before.

8.4.1 Benchmarking

As mentioned before, layouts and clustering algorithms' time is based on the number of nodes and edges the network has. For a better perspective table, 8.1 shows the time in seconds Arena3D^{web} application needs to run the Louvain clustering algorithm at a 1-Layer network, where the global and local layout algorithm is set to the Fruchterman algorithm. The first two columns are the number of nodes and edges each network has, and in the following columns the time in second is needed for Layout, Total Layout, Clustering, and Total Clustering whereas Total Layout and Total Clustering include the data preparation.

#		Times in seconds			
Nodes	**Edges**	**Layout**	**Total Layout**	**Clustering**	**Total Clustering**
500	5000	0.28	3.33	0.4	3.3
500	10000	0.3	9.7	0.5	9.7
500	15000	0.3	19	0.6	20
5000	5000	0.34	3.37	6.5	9.5
10000	10000	0.7	10.5	21	32
15000	15000	1	21	44	60.2

Table 8.1 Layouts Benchmarking at Arena3D

8.5 Interoperability

Arena3D^{web} comes with its API and its own updated input file format ".tsv" or ".txt", two optional attribute files for nodes and edges. and the new import/export format ".JSON" for loading and saving sessions. Starting from the first, the input file support consists of 4 mandatory columns and rows whereas each row represents an edge. The four mandatory columns *are SourceNode, TargetNode, SourceLayer,* and *TargetLayer*. Besides them, it is possible to add two extra columns, *Weight,* and the new addition *Channel* that is needed to support multi-edge networks. After the file is uploaded, the weight values are mapped in a [0-1] range and relative opacities are assigned to the respective edges. Edge transparency represents the weight. The heavier the weight, the higher the opacity. In the case of unweighted graphs, one can skip the weight column. The channel column is only used in the case of multi-edge graphs and at this, the user can add the channel name. As mentioned before Arena3D^{web} supports up to 9 different channels. The column order in the input file is irrelevant (Figure 8.14).

As mentioned, the application also offers the possibility to upload a node attributes file and/or an edge attributes file to customize even more the network. The nodes attribute file consists of two necessary columns *Node* and *Layer* and four omittable *Color, Size, URL,* and *Description*. In this file, one can add a URL and description to the nodes they want or change the size and color (Figure 8.15). The edge attributes file consists of five mandatory columns like the uploaded input file format, *SourceNode, SourceLayer, TargetNode, TargetLayer,* and *Color* (Figure 8.16).

Arena3D^{web} can save and load the running visualization in a JSON format. More specifically, the JSON object contains information regarding the: *(i)* scene object rotation, translation, scaling, and background color, *(ii)* layer names, positions, rotations, scales, colors, and widths, *(iii)* node colors, respective layers, positions, scales, colors, and optional accompanying metadata, *(iv)* edge source and target nodes, opacities, colors, optional channels and *(v)*, a global label color, a flag for enabling edge direction and last but not least a flag for enabling opacity by weight. It is important to note that a user can create and upload a file in JSON format too, with only the mandatory attributes: layer names, node names, and edge source and target nodes (example in Figure 8.17). For any missing parameters in the imported object, the tool provides default values where applicable.

The JSON format is suitable for calling Arena3D^{web} from external applications and services.

For this purpose, a dedicated API route (https://bib.fleming.gr/bib/api/arena3dweb) has been implemented in *Node.js*. This API call handles POST requests of an Arena3D^{web} JSON object, as described in the paragraph above, and opens the generated network object in an Arena3D^{web} viewer. The developer must also set the Header Content-Type to application/JSON.
The server then returns a JSON response with the URL that links to the Arena3Dweb application, having the requested network loaded:

"url": https://bib.fleming.gr:8084/app/arena3d?f=081436639JURotmRGQeFJ.json.

SourceNode	SourceLayer	TargetNode	TargetLayer	Weight	Channel
An	Group1	Cn	Group1	2	1
An	Group1	Bn	Group1	10	1
Bn	Group1	Cn	Group1	1	1
Cn	Group1	Dn	Group1	3	1
En	Group2	Fn	Group2	4	1
En	Group2	Hn	Group2	5	1
Fn	Group2	Gn	Group2	6	1
Gn	Group2	Hn	Group2	7	1
In	Group3	Jn	Group3	8	1
Bn	Group1	Fn	Group2	9	1
Dn	Group1	Hn	Group2	11	1
Dn	Group1	In	Group3	1	1
Cn	Group1	Jn	Group3	1	1
Hn	Group2	In	Group3	1	1
Hn	Group2	Kn	Group4	1	1
Kn	Group4	Ln	Group4	0.1	1
Kn	Group4	Mn	Group5	1	1
An	Group1	Nn	Group5	1	1
Kn	Group4	On	Group5	1	1
Kn	Group4	Pn	Group5	12	1
Kn	Group4	Qn	Group6	1	1
Kn	Group4	Rn	Group6	1	1
Kn	Group4	Sn	Group7	1	1
Kn	Group4	Tn	Group7	10	1

Figure 8.14 File Format in Arena3Dweb. The default input file format consists of a table with minimum 4 columns. The mandatory columns are SourceNode, SourceLayer, TargetNode and TargetLayer. The above example also contains a Weight and a Channel column.

Node	Layer	Color	Size	Url	Description
An	Group1	#6b6a4c	1		This is a node's description.
Bn	Group1	#ccccff	2		
Cn	Group1	#254284	3		
En	Group2		4	https://www.frontiersin.org/articles/10.3389/fbioe.2020.00034/full	Click the Link of this node to read our review
Fn	Group2	#7fe5f0	5	http://norma.pavlopouloslab.info/	Click the Link of this node to access our network annotation tool, NORMA.
Gn	Group2		1		
In	Group3		2		
Dn	Group1	#ffb3b3		http://nap.pavlopouloslab.info/	
Hn	Group2		3		
Kn	Group4	#e0f2f2			
Qn	Group6	#ff0067			
Rn	Group6				
Sn	Group7	#ffd8e8			
Tn	Group7		4		

Figure 8.15 Node Attributes file format in Arena3Dweb. The Node Attributes file consists of two necessary columns Node and Layer and four omittable Color, Size, URL, and Description.

SourceNode	SourceLayer	TargetNode	TargetLayer	Color
An	Group1	Cn	Group1	#4EFBE9
Bn	Group1	Fn	Group2	#D64EFB
In	Group3	Jn	Group3	
Kn	Group4	Tn	Group7	#4EFB7D

Figure 8.16 Edge Attributes file format in Arena3Dweb. The Edge Attributes file must contain the following five columns SourceNode, SourceLayer, TargetNode, TargetLayer and Color.

```
{
    "layers": [
        {
            "name": "Complex"
        },
        {
            "name": "Proteins"
        }
    ],
    "nodes": [
        {
            "name": "Pericentrin-GCP Complex",
            "layer": "Complex"
        },
        {
            "name": "TUBGCP2",
            "layer": "Proteins"
        },
        {
            "name": "TUBGCP3",
            "layer": "Proteins"
        }
    ],
    "edges": [
        {
            "src": "Pericentrin-GCP Complex_Complex",
            "trg": "TUBGCP2_Proteins"
        },
        {
            "src": "Pericentrin-GCP Complex_Complex",
            "trg": "TUBGCP3_Proteins"
        },
        {
            "src": "DNA Polymerase Alpha_Complex",
            "trg": "TUBGCP2_Proteins"
        }
    ]
}
```


Figure 8.17 JSON file format in Arena3Dweb with only the mandatory fields

8.6 Integration with Cytoscape

As mentioned before Cytoscape is an open-source software for 2D visualization. One of its main advantages is the different applications (plugins) that can be imported to the core part. Arena3D^{web} is now accessible through Cytoscape's Arena3D^{web} dedicated app called *Arena3DwebApp*. The app is implemented in Java and comes with a simple interface, where the user can configure how the Cytoscape network will be transferred to Arena3D^{web}. The most important setting is choosing which node attribute contains the information about the layers. It could be any numeric or string value that defines up to 18 different non-overlapping groups, which will be translated into layers. Furthermore, the Cytoscape app extracts the current displayed color, size, and coordinates of the nodes as well as the directionality, color, thickness, and transparency of the edges. The node label font size and the network background are also transferred. The user can choose which column, to use for the node description and URL that can be seen in Arena3D^{web} as additional node information. If there are nodes that do not participate in any named layer, they are added to a layer named "unassigned" by default, but the user can also choose not to add them to the network in Arena3D^{web}. The app generated a JSON file that was automatically sent to Arena3D^{web} and displayed in the user's default web browser. If the user wants to share the layered network or open it later, they can download the JSON file from Cytoscape and import it into Arena3D^{web}. A detailed example of this is described in the "CYTOSCAPE USE CASE" chapter.

8.7 Use cases

Following there are five use case scenarios to demonstrate the new functionalities that Arena3D^{web} 's latest version now supports.

8.7.1 STICH use case

The first use case demonstrates how Arena3D^{web} handles both directed and multi-channel connections, through STICH. As mentioned before STITCH (Search Tool for Interacting Chemicals) is a database that includes more than 9600000 proteins from 2031 eukaryotic and prokaryotic genomes and 430 000 chemical compounds without including different stereoisomers and offers multi-channels directed networks. As seen in Figure 8.18 we queried the STITCH database for 'aspirin' and filtered its interactions by only keeping the high-confidence ones (score > 0.7). We allow up to ten interactors for the channels: 'Experimental', 'Database', and 'Text Mining',

depicted in green, blue, and red, respectively. Drug and protein nodes are separated into different layers. The network consists of two layers, Drugs, and Proteins.

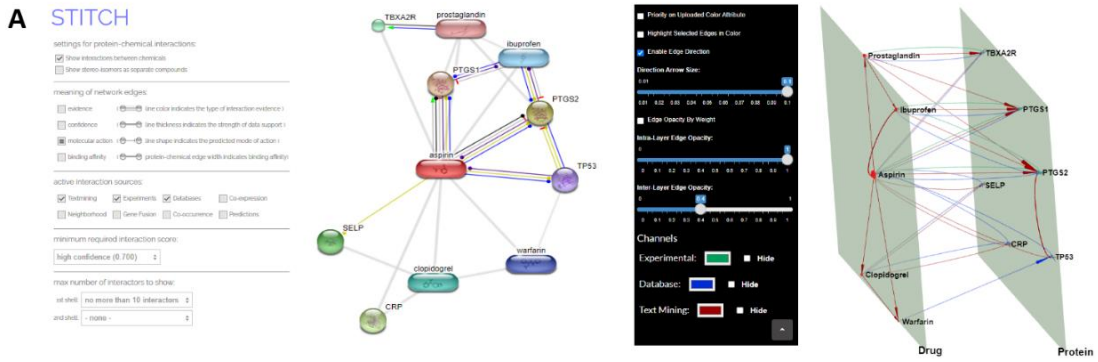


Figure 8.18 A protein-chemical interaction network generated by STITCH. Left: Aspirin compound with its top ten interactors and molecular interaction channels is shown. Right: The same network in Arena3Dweb format with three channels of information: “Experimental”, “Database” and “Text-Mining''.

8.7.2 PREGO use case

PREGO is a tool that combines text mining and data integration techniques to mine such what-where-who associations from data and metadata scattered in the scientific literature and public omics repositories. Microorganisms, biological processes, and environment types are identified and mapped to ontology terms from established community resources. Analysis of co-mentions in text and co-occurrences in metagenomics data/metadata are performed to extract associations and a level of confidence is assigned to each of them thanks to a scoring scheme. The PREGO knowledge base contains associations for 364,508 microbial taxa, 1090 environmental types, 15,091 biological processes, and 7971 molecular functions with a total of almost 58 million associations [108].

The second use case demonstrates also a multi-channel need that Arena3D[web] can now cover. We queried the PREGO knowledge base for the ‘anaerobic ammonium oxidation’ biological process (GO:0019331, anammox) and explored the associated microorganism taxa. The top 11 organisms (at the genus level) co-mentioned in the scientific literature with anammox were extracted (Figure 8.19 left). For illustration purposes (Figure 8.19 - middle and right), the genus ‘Beggiatoa’ was selected to show its associated environment types, including the top 12

environments from the Literature (text-mining) channel (edges indicated in green) along with the top 8 environments from the Environmental Sample channel (edges indicated in red).

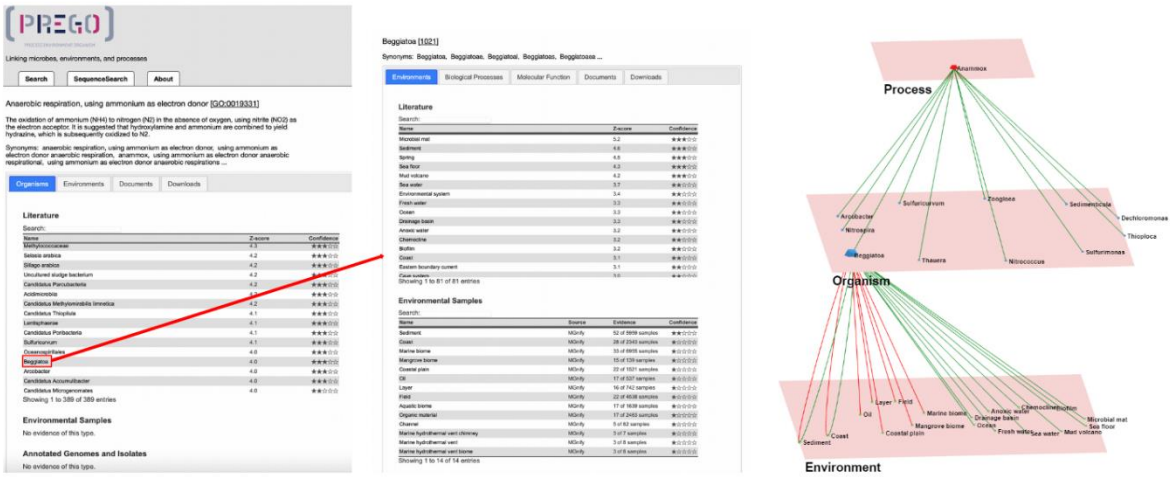


Figure 8.19 PREGO's *Process, Organism* and *Environment* association tables. ***Left****:* The associations of the '*anaerobic ammonium oxidation'* process with organisms**. Middle***:* he associations of the genus *Beggiatoa* with the corresponding *Environments* and their respective channel sources ('*Literature*' (text-mining), '*Environmental Samples*'). **Right**: The multilayer and multi-edge Arena3D[web] simultaneous visualization of these two distinct tables maintaining the source channel information ('*Literature*' (text-mining) shown via green edges, while '*Environmental Sample*' derived-ones shown in red).

8.7.3 DARLING use case

Darling, is a novel web application to query scientific publications associated with diseases, identify, and visualize bio-entities of various types and construct knowledge-based biological interaction networks. Out of a plethora of articles and available databases, the tool focuses on disease-centric repositories and generates a non-redundant set of publications, associated with entries in the OMIM, Human Phenotype Ontology (HPO), and DisGeNET databases. The abstracts of the publications are parsed through Named Entity Recognition (NER) to identify a wide range of biomedical terms (genes, chemicals, organisms, ontology terms, diseases, phenotypes, and environments). Sentence-based associations among the various biomedical entities are presented in an interactive network, as well as in searchable and sortable tables, while abstracts are shown in annotated formats[109].

Darling's use case helps demonstrate the usefulness of Arena3D^{web} 's API. Through API utilization, Arena3Dweb can be called from Darling, a text-mining disease-oriented software that produces 2D networks among tagged biomedical terms of ten different categories. In Darling, a

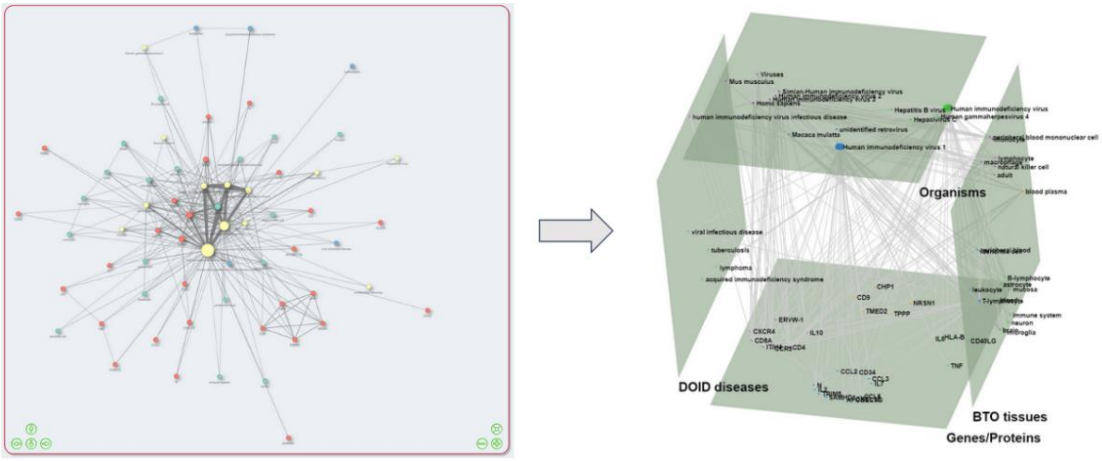


Figure 8.20 Darling API Functionality Use Case. A Darling query for HIV using the "Disease search" option to fetch literature from the DisGeNET dataset. 4 selected entity types were used ("DOID Diseases", "Genes/Proteins", "BTO tissues'' and "Organisms"; see left). On the right, the returned biological entities are projected onto their respective layers in an Arena3Dweb cube layout.

user can query for diseases, chemicals, proteins, tissues, or publications and then construct networks that link the relevant extracted biological entities based on literature co-mentions. In Arena3Dweb, the various biomedical entities will be placed on different layers according to their type (one layer per type). An example demonstrating the API is provided in Figure 8.20.

8.7.4 FLAME use case

FLAME is a web tool for combining multiple lists before enrichment analysis. Users can upload several lists and use interactive UpSet plots, as an alternative to Venn diagrams, to handle unions or intersections among the given input files. Functional and literature enrichment, along with gene conversions, are offered by g:Profiler and aGOtool applications for 197 organisms. FLAME can analyze genes/proteins for related articles, Gene Ontologies, pathways, annotations, regulatory motifs, domains, diseases, and phenotypes, and can also generate protein-protein interactions derived from STRING. We have validated FLAME by interrogating gene expression data associated with the sensitivity of the distal part of the large intestine to experimental colitis-propelled colon cancer. FLAME comes with an interactive user-friendly interface for easy list

manipulation and exploration, while results can be visualized as interactive and parameterizable heatmaps, bar charts, Manhattan plots, networks, and tables[110].

Similarly with the Darling Use Case, through the functional and literature enrichment analysis tool named Flame, two-layered Arena3D[web] networks can be produced, one layer corresponding to the terms of a user-selected enriched category and a second layer containing the related genes/proteins. An example of this use case is provided in Figure 8.21.

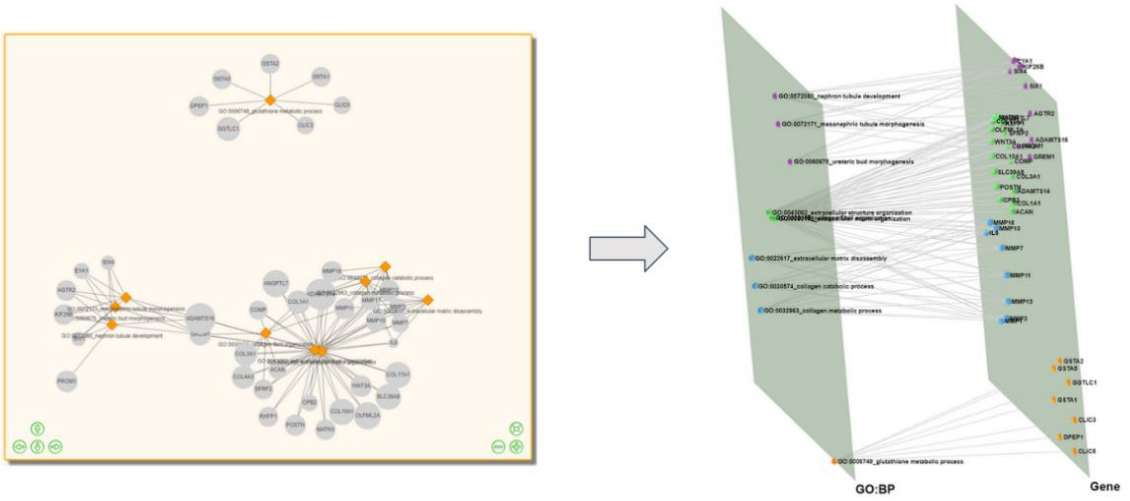


Figure 8.21 FLAME API Functionality Use Case. A Flame enriched network generated from a list of Idiopathic Pulmonary Fibrosis related differentially expressed genes and their top ten associated biological processes (see left). On the right, the genes and their related processes are projected in two distinct layers in Arena3D[web].

8.7.5 Cytoscape use case

To illustrate the interoperability between Cytoscape and Arena3D[web], we used stringApp v2.0 and Arena3DwebApp in combination with Cytoscape. Specifically, we used the "STITCH: protein/compound query" function of stringApp to search for the compound "aspirin" with a confidence score cutoff of 0.7 and up to ten additional interactors (compounds or human proteins). We then used stringApp to retrieve functional enrichment for the proteins interacting with aspirin and added all enriched diseases, tissues, and KEGG pathways to the network (two, four, and five, respectively). The resulting network in Cytoscape is shown in Figure 8.22A. To transfer the network to Arena3Dweb, we opened the Arena3DwebApp dialog box shown in Figure 8.22B. We chose to use the column "stringdb::node type" to define the layers in Arena3Dweb, which means that chemical compounds will be placed in one layer, proteins in a

second, and enriched terms in a third. We selected to not consider edges as directed, since STRING networks are undirected, and the column “stringdb::description” should be used for node descriptions. Finally, we submitted this three-layer network to Arena3Dweb for further 3D

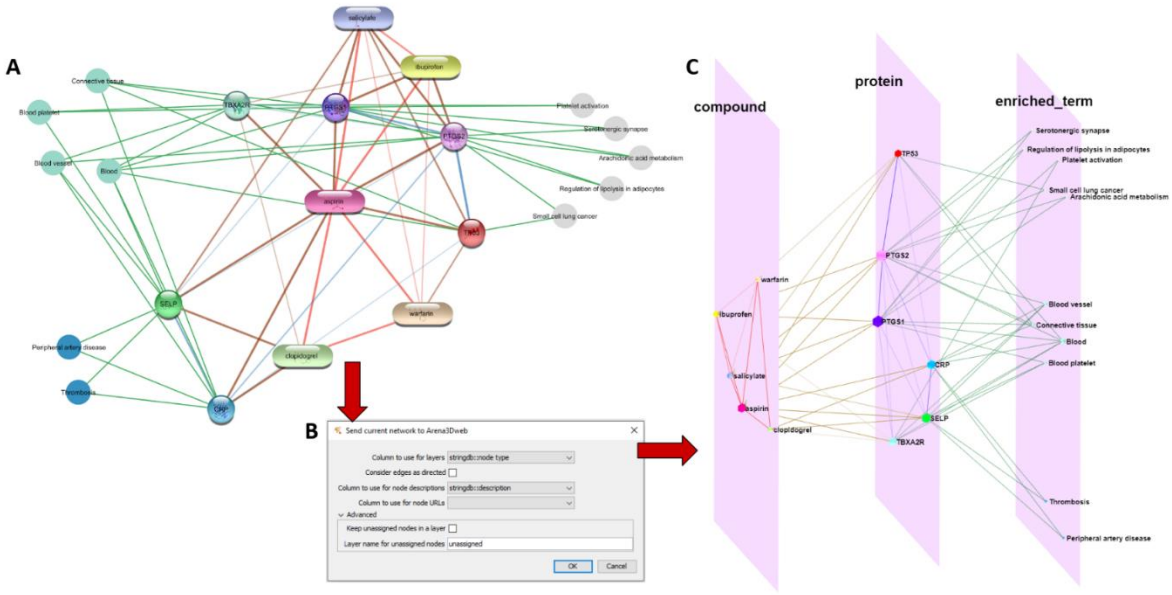


Figure 8.22 From Cytoscape to Arena3Dweb. (A) An aspirin chemical-protein interaction network along with enriched terms including diseases, tissues and KEGG pathways generated through stringApp in Cytoscape. Enriched disease nodes are colored red, tissues green and pathways gray. (B) The Arena3DwebApp dialogue window, where users are prompted to choose the node column that will indicate the various Layers in Arena3D^{web}. Other options include edge directionality, columns for node descriptions and URLs and how to handle nodes without layer information. (C) The generated Arena3D^{web} network in three layers; compounds, proteins, and enriched terms. Enriched pathways are placed on top, tissues in the middle and diseases on the bottom of the layer. The chemical compound interactions channel is colored red, compound-protein channel in brown, protein-protein in blue and protein participation in the various enriched terms in green. The node coloring scheme agrees with the initial STRING network in Cytoscape.

manipulations (Figure 8.22C). In the enriched_term layer, we show the three categories of enriched terms from STRING in three separate neighborhoods: KEGG pathways on top, tissues in the middle, and diseases on the bottom.

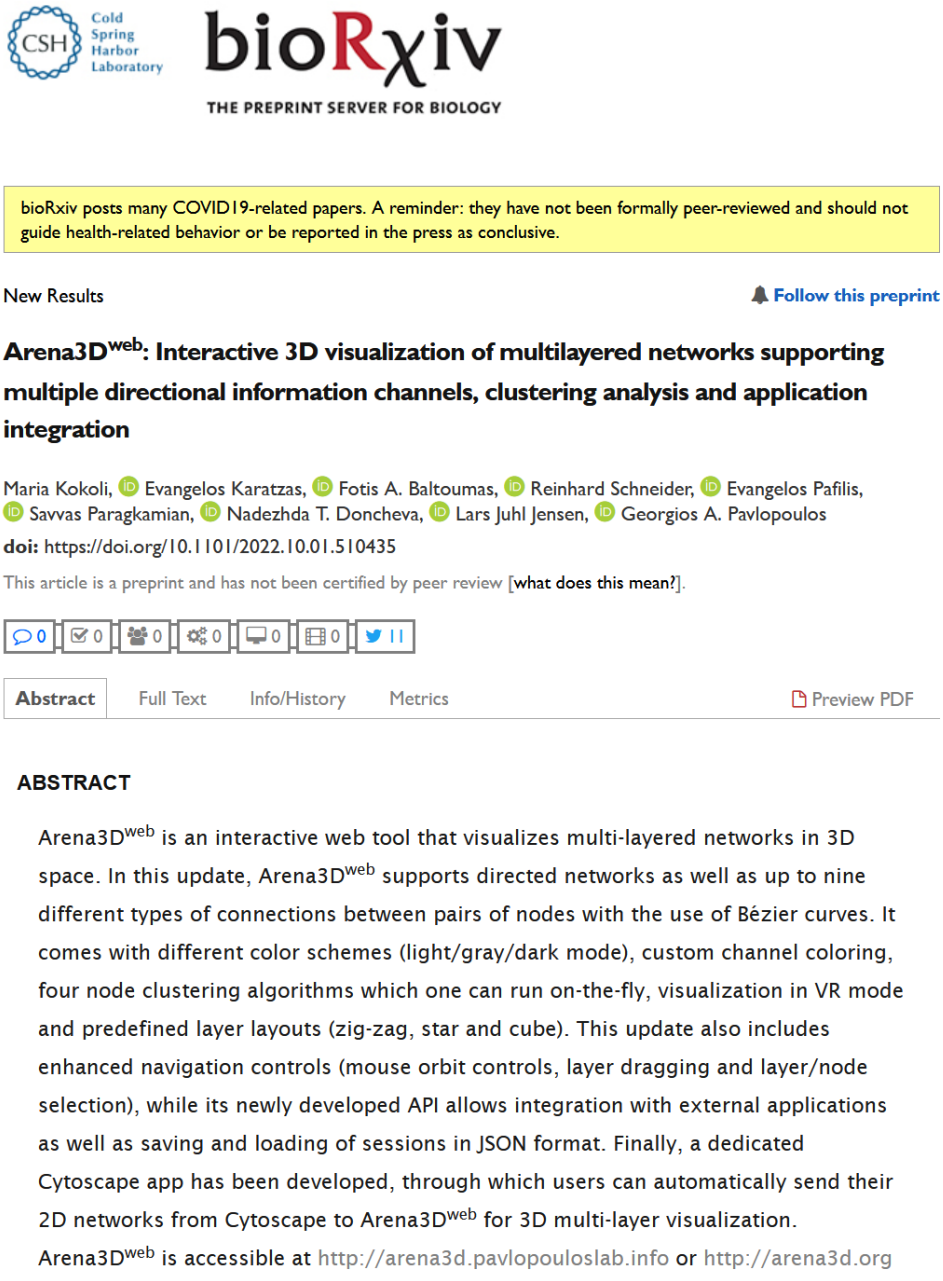
9 DISSEMINATION

Arena3D^{web} is available at http://arena3d.org or http://arena3d.pavlopouloslab.info.

The source code is available at **GitHub**: https://github.com/pavlopouloslab/arena3dweb.

Arena3DwebApp Cytoscape application is available at **Cytoscape App Store** at: https://apps.cytoscape.org/apps/arena3DwebApp.

Article preprint available through **BioRxiv**: https://www.biorxiv.org/content/10.1101/2022.10.01.510435v2.abstract.



Cold Spring Harbor Laboratory

bioRxiv

THE PREPRINT SERVER FOR BIOLOGY

bioRxiv posts many COVID19-related papers. A reminder: they have not been formally peer-reviewed and should not guide health-related behavior or be reported in the press as conclusive.

New Results

Follow this preprint

Arena3D^{web}: Interactive 3D visualization of multilayered networks supporting multiple directional information channels, clustering analysis and application integration

Maria Kokoli, Evangelos Karatzas, Fotis A. Baltoumas, Reinhard Schneider, Evangelos Pafilis, Savvas Paragkamian, Nadezhda T. Doncheva, Lars Juhl Jensen, Georgios A. Pavlopoulos

doi: https://doi.org/10.1101/2022.10.01.510435

This article is a preprint and has not been certified by peer review [what does this mean?].

0 0 0 0 0 0 11

Abstract Full Text Info/History Metrics Preview PDF

ABSTRACT

Arena3D^{web} is an interactive web tool that visualizes multi-layered networks in 3D space. In this update, Arena3D^{web} supports directed networks as well as up to nine different types of connections between pairs of nodes with the use of Bézier curves. It comes with different color schemes (light/gray/dark mode), custom channel coloring, four node clustering algorithms which one can run on-the-fly, visualization in VR mode and predefined layer layouts (zig-zag, star and cube). This update also includes enhanced navigation controls (mouse orbit controls, layer dragging and layer/node selection), while its newly developed API allows integration with external applications as well as saving and loading of sessions in JSON format. Finally, a dedicated Cytoscape app has been developed, through which users can automatically send their 2D networks from Cytoscape to Arena3D^{web} for 3D multi-layer visualization. Arena3D^{web} is accessible at http://arena3d.pavlopouloslab.info or http://arena3d.org

10 REFERENCES

1. Koutrouli, M., Karatzas, E., Paez-Espino, D. & Pavlopoulos, G. A. A Guide to Conquer the Biological Network Era Using Graph Theory. *Front. Bioeng. Biotechnol.* **8**, 1–26 (2020).
2. Jeong, H., Tombor, B., Albert, R., Oltval, Z. N. & Barabásl, A. L. The large-scale organization of metabolic networks. *Nature* **407**, 651–654 (2000).
3. Watts, Duncan J.;Strogatz, S. H. . Collective dynamics of 'small world' networks. *Nature* **1**, 143 (2017).
4. Barabási, Albert-László; Albert, R. Emergence of Scaling in Random Networks. *Netw. Sci.* **286**, 97–130 (2009).
5. Yook, S. H., Oltvai, Z. N. & Barabási, A. L. Functional and topological characterization of protein interaction networks. *Proteomics* **4**, 928–942 (2004).
6. Barabási, A. L. & Oltvai, Z. N. Network biology: Understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, 101–113 (2004).
7. Spirin, V. & Mirny, L. A. Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. U. S. A.* **100**, 12123–12128 (2003).
8. Bader, G. D., Betel, D. & Hogue, C. W. V. BIND: The Biomolecular Interaction Network Database. *Nucleic Acids Res.* **31**, 248–250 (2003).
9. Stark, C. *et al.* BioGRID: a general repository for interaction datasets. *Nucleic Acids Res.* **34**, 535–539 (2006).
10. Xenarios, I. *et al.* DIP: The Databases of Interacting Protein: 2001 update. *Nucleic Acids Res.* **29**, 239–241 (2001).
11. Ekre, A. R. & Mante, R. V. Genome sequence alignment tools: A review. *Proceeding IEEE - 2nd Int. Conf. Adv. Electr. Electron. Information, Commun. Bio-Informatics, IEEE - AEEICB 2016* 677–681 (2016) doi:10.1109/AEEICB.2016.7538378.
12. Sharan, R., Ulitsky, I. & Shamir, R. Network-based prediction of protein function. *Mol. Syst. Biol.* **3**, 1–13 (2007).
13. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, (1990).
14. Vázquez, A. *et al.* The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *Proc. Natl. Acad. Sci. U. S. A.* **101**, 17940–17945 (2004).
15. Kanehisa, M. & Goto, S. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research* vol. 28 (2000).
16. Yevshin, I., Sharipov, R., Kolmykov, S., Kondrakhin, Y. & Kolpakov, F. GTRD: A database on gene transcription regulation - 2019 update. *Nucleic Acids Res.* **47**, D100–D105 (2019).
17. Matys, V. *et al.* TRANSFAC®: Transcriptional regulation, from patterns to profiles. *Nucleic Acids Res.* **31**, 374–378 (2003).
18. Han, H. *et al.* TRRUST v2: An expanded reference database of human and mouse transcriptional regulatory interactions. *Nucleic Acids Res.* **46**, D380–D386 (2018).
19. Fabregat, A. *et al.* The Reactome Pathway Knowledgebase. *Nucleic Acids Res.* **46**, (2018).
20. Yi, Y., Fang, Y., Wu, K., Liu, Y. & Zhang, W. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Oncol. Lett.* **19**, 3316–3332 (2020).
21. Ulrich, L. E. & Zhulin, I. B. MiST: A microbial signal transduction database. *Nucleic Acids Res.* **35**, (2007).
22. Kandasamy, K. *et al.* NetPath: A public resource of curated signal transduction pathways. *Genome Biol.* **11**, (2010).
23. Gagneur, J., Jackson, D. B. & Casari, G. Hierarchical analysis of dependency in metabolic networks. *Bioinformatics* **19**, (2003).

24. Stuart, J. M., Segal, E., Koller, D. & Kim, S. K. A gene-coexpression network for global discovery of conserved genetic modules. *Science (80-.).* **302**, (2003).
25. Barrett, T. *et al.* NCBI GEO: Archive for functional genomics data sets - Update. *Nucleic Acids Res.* **41**, (2013).
26. Parkinson, H. *et al.* ArrayExpress - A public database of microarray experiments and gene expression profiles. *Nucleic Acids Res.* **35**, (2007).
27. Obayashi, T., Kagaya, Y., Aoki, Y., Tadaka, S. & Kinoshita, K. COXPRESdb v7: A gene coexpression database for 11 animal species supported by 23 coexpression platforms for technical evaluation and evolutionary inference. *Nucleic Acids Res.* **47**, (2019).
28. Huson, D. H., Rupp, R. & Scornavacca, C. *Phylogenetic networks: Concepts, algorithms and applications*. *Phylogenetic Networks: Concepts, Algorithms and Application* (2010). doi:10.1017/CBO9780511974076.
29. Arenas, M., Valiente, G. & Posada, D. Characterization of reticulate networks based on the coalescent with recombination. *Mol. Biol. Evol.* **25**, (2008).
30. Morrison, D. A. Phylogenetic networks: A new form of multivariate data summary for data mining and exploratory data analysis. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **4**, (2014).
31. Gross, E. & Long, C. Distinguishing phylogenetic networks. *SIAM J. Appl. Algebr. Geom.* **2**, (2018).
32. Yang, Z. Molecular Evolution. A Statistical Approach Preface. *Mol. Evol. A Stat. Approach* (2014).
33. Huson, D. H. & Bryant, D. Application of phylogenetic networks in evolutionary studies. *Molecular Biology and Evolution* vol. 23 (2006).
34. Huson, D. H. & Scornavacca, C. Dendroscope 3: An interactive tool for rooted phylogenetic trees and networks. *Syst. Biol.* **61**, (2012).
35. Schliep, K. P. phangorn: Phylogenetic analysis in R. *Bioinformatics* **27**, (2011).
36. Ings, T. C. *et al.* Ecological networks - Beyond food webs. *Journal of Animal Ecology* vol. 78 (2009).
37. Keeling, M. J. *et al.* Networks and the epidemiology of infectious disease. *Interdisciplinary Perspectives on Infectious Diseases* vol. 2011 (2011).
38. Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y. & Morishima, K. KEGG: New perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res.* **45**, (2017).
39. Mishra, G. R. *et al.* Human protein reference database--2006 update. *Nucleic Acids Res.* **34**, (2006).
40. Szklarczyk, D. *et al.* The STRING database in 2021: Customizable protein-protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic Acids Res.* **49**, (2021).
41. Szklarczyk, D. *et al.* STITCH 5: Augmenting protein-chemical interaction networks with tissue and affinity data. *Nucleic Acids Res.* **44**, (2016).
42. Gioutlakis, A., Klapa, M. I. & Moschonas, N. K. PICKLE 2.0: A human protein-protein interaction meta-database employing data integration via genetic information ontology. *PLoS One* **12**, (2017).
43. Nica, A. C. & Dermitzakis, E. T. Expression quantitative trait loci: Present and future. *Philosophical Transactions of the Royal Society B: Biological Sciences* vol. 368 (2013).
44. Platig, J., Castaldi, P. J., DeMeo, D. & Quackenbush, J. Bipartite Community Structure of eQTLs. *PLoS Comput. Biol.* **12**, (2016).
45. Fagny, M. *et al.* Exploring regulation in tissues with eQTL networks. *Proc. Natl. Acad. Sci. U. S. A.* **114**, (2017).
46. Sonawane, A. R., Weiss, S. T., Glass, K. & Sharma, A. Network medicine in the age of biomedical big data. *Frontiers in Genetics* vol. 10 (2019).

47. Allen, P., Matties, M. & Peterson, E. Hairball Buster: A Graph Triage Method for Viewing and Comparing Graphs. *Connections* **40**, (2020).
48. Pavlopoulos, G. A., Paez-Espino, D., Kyrpides, N. C. & Iliopoulos, I. Empirical Comparison of Visualization Tools for Larger-Scale Network Analysis. *Advances in Bioinformatics* vol. 2017 (2017).
49. Koren, Y. Drawing graphs by eigenvectors: Theory and practice. *Comput. Math. with Appl.* **49**, (2005).
50. Martin, S., Brown, W. M., Klavans, R. & Boyack, K. W. OpenOrd: an open-source toolbox for large graph layout. in *Visualization and Data Analysis 2011* vol. 7868 (2011).
51. Hu, Y. Efficient, High-Quality Force-Directed Graph Drawing. *Math. J.* **10**, (2005).
52. Schönfeld, M. & Pfeffer, J. Fruchterman/Reingold (1991): Graph Drawing by Force-Directed Placement. in (2019). doi:10.1007/978-3-658-21742-6_49.
53. Csardi, G. & Nepusz, T. The igraph software package for complex network research. *InterJournal Complex Syst.* **Complex Sy**, (2006).
54. Shannon, P. *et al.* Cytoscape: A software Environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, (2003).
55. Xu, R. & Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* vol. 16 (2005).
56. Brohée, S. & van Helden, J. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics* **7**, (2006).
57. Langfelder, P., Zhang, B. & Horvath, S. Defining clusters from a hierarchical cluster tree: The Dynamic Tree Cut package for R. *Bioinformatics* **24**, (2008).
58. Jiang, P. & Singh, M. SPICi: A fast clustering algorithm for large biological networks. *Bioinformatics* **26**, (2010).
59. van Dongen, S. M. Graph clustering by flow simulation. *Comput. Sci. Rev.* **1**, (2000).
60. Enright, A. J. & Ouzounis, C. A. GeneRAGE: A robust algorithm for sequence clustering and domain detection. *Bioinformatics* **16**, (2000).
61. Enright, A. J., Van Dongen, S. & Ouzounis, C. A. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research* vol. 30 (2002).
62. Traag, V. A., Waltman, L. & van Eck, N. J. From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.* **9**, (2019).
63. Newman, M. E. J. Fast algorithm for detecting community structure in networks. *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.* **69**, (2004).
64. Blondel, V. D., Guillaume, J. L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, (2008).
65. Hu, F. *et al.* An algorithm Walktrap-SPM for detecting overlapping community structure. *Int. J. Mod. Phys. B* **31**, (2017).
66. Pons, P. & Latapy, M. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.* **10**, (2006).
67. Xie, J. & Szymanski, B. K. Community detection using a neighborhood strength driven Label Propagation Algorithm. in *Proceedings of the 2011 IEEE 1st International Network Science Workshop, NSW 2011* (2011). doi:10.1109/NSW.2011.6004645.
68. Demir, E. *et al.* The BioPAX community standard for pathway data sharing. *Nature Biotechnology* vol. 28 (2010).
69. Hucka, M. *et al.* The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics* **19**, (2003).
70. Hermjakob, H. *et al.* The HUPO PSI's Molecular Interaction format - A community standard for the representation of protein interaction data. *Nature Biotechnology* vol. 22 (2004).
71. Murray-Rust, P., Rzepa, H. S. & Wright, M. Development of chemical markup language

(CML) as a system for handling complex chemical content. *New J. Chem.* **25**, (2001).
72. Lloyd, C. M., Halstead, M. D. B. & Nielsen, P. F. CellML: Its future, present and past. in *Progress in Biophysics and Molecular Biology* vol. 85 (2004).
73. Brandes, U., Eiglsperger, M., Lerner, J. & Pich, C. - Graph Markup Language (GraphML). in *Handbook of Graph Drawing and Visualization* 532–557 (2020). doi:10.1201/b15385-19.
74. Franz, M. *et al.* Cytoscape.js: A graph theory library for visualisation and analysis. *Bioinformatics* **32**, (2016).
75. Pillich, R. T., Chen, J., Rynkov, V., Welker, D. & Pratt, D. NDEx: A community resource for sharing and publishing of biological networks. in *Methods in Molecular Biology* vol. 1558 (2017).
76. Gentleman, R. C. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.* **5**, (2004).
77. Kramer, F., Bayerlová, M., Klemm, F., Bleckmann, A. & Beißbarth, T. RBiopaxParser-an R package to parse, modify and visualize BioPAX data. *Bioinformatics* **29**, (2013).
78. Stajich, J. E. & Lapp, H. Open source tools and toolkits for bioinformatics: Significance, and where are we? *Briefings in Bioinformatics* vol. 7 (2006).
79. Bastian, M., Heymann, S. & Jacomy, M. Gephi: An Open Source Software for Exploring and Manipulating Networks Visualization and Exploration of Large Graphs. in *Proceedings of the international AAAI conference on web and social media* (2009).
80. Jacomy, M., Venturini, T., Heymann, S. & Bastian, M. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PLoS One* **9**, (2014).
81. S., J. P. C., Chatterjee, A., M., G. & Mukherjee, A. MultiViz: A Gephi Plugin for Scalable Visualization of Multi-Layer Networks. (2022).
82. Mrvar, A. & Batagelj, V. Analysis and visualization of large networks with program package Pajek. *Complex Adaptive Systems Modeling* vol. 4 (2016).
83. Köhler, J. *et al.* Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics* **22**, (2006).
84. Taubert, J., Hassani-Pak, K., Castells-Brooke, N. & Rawlings, C. J. Ondex Web: Web-based visualization and exploration of heterogeneous biological networks. *Bioinformatics* **30**, (2014).
85. Auber, D. *et al.* Tulip 5. in *Encyclopedia of Social Network Analysis and Mining* (2017). doi:10.1007/978-1-4939-7131-2_315.
86. Vadnais, C. OpenGL. in *Instant Boris Effects* (2020). doi:10.4324/9780080522074-10.
87. Deacon, J. Model-view-controller (mvc) architecture. *Comput. Syst. Dev.* (2005).
88. PyPI. PyPI · The Python Package Index. *PyPI* (2020).
89. Zhou, G., Pang, Z., Lu, Y., Ewald, J. & Xia, J. OmicsNet 2.0: a web-based platform for multi-omics integration and network visual analytics. *Nucleic Acids Res.* (2022) doi:10.1093/nar/gkac376.
90. Zhou, G. & Xia, J. OmicsNet: A web-based tool for creation and visual analysis of biological networks in 3D space. *Nucleic Acids Res.* **46**, (2018).
91. Breuer, K. *et al.* InnateDB: Systems biology of innate immunity and beyond - Recent updates and continuing curation. *Nucleic Acids Res.* **41**, (2013).
92. Orchard, S. *et al.* The MIntAct project - IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res.* **42**, (2014).
93. Castro-Mondragon, J. A. *et al.* JASPAR 2022: The 9th release of the open-access database of transcription factor binding profiles. *Nucleic Acids Res.* **50**, (2022).
94. Karagkouni, D. *et al.* DIANA-TarBase v8: A decade-long collection of experimentally supported miRNA-gene interactions. *Nucleic Acids Res.* **46**, (2018).
95. Huang, H. Y. *et al.* MiRTarBase 2020: Updates to the experimentally validated microRNA-

target interaction database. *Nucleic Acids Res.* **48**, (2020).
96. Brunk, E. *et al.* Recon3D enables a three-dimensional view of gene variation in human metabolism. *Nat. Biotechnol.* **36**, (2018).
97. Magnúsdóttir, S. *et al.* Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota. *Nat. Biotechnol.* **35**, (2017).
98. Freeman, T. C. *et al.* Graphia: A platform for the graph-based visualisation and analysis of complex data. *bioRxiv* (2020).
99. Koutrouli, M., Karatzas, E., Papanikolopoulou, K. & Pavlopoulos, G. A. NORMA: The Network Makeup Artist — A Web Tool for Network Annotation Visualization. *Genomics. Proteomics Bioinformatics* (2021) doi:10.1016/j.gpb.2021.02.005.
100. Karatzas, E. *et al.* The network makeup artist (NORMA-2.0): distinguishing annotated groups in a network using innovative layout strategies. *Bioinforma. Adv.* **2**, (2022).
101. von Mering, C. *et al.* STRING: A database of predicted functional associations between proteins. *Nucleic Acids Research* vol. 31 (2003).
102. Pavlopoulos, G. A. *et al.* Arena3D: Visualization of biological networks in 3D. *BMC Syst. Biol.* **2**, (2008).
103. Secrier, M., Pavlopoulos, G. A., Aerts, J. & Schneider, R. Arena3D: Visualizing time-driven phenotypic differences in biological systems. *BMC Bioinformatics* **13**, (2012).
104. Škrlj, B., Kralj, J. & Lavrač, N. Py3plex toolkit for visualization and analysis of multilayer networks. *Appl. Netw. Sci.* **4**, (2019).
105. Hammoud, Z. & Kramer, F. Mully: An R package to create, modify and visualize multilayered graphs. *Genes (Basel).* **9**, (2018).
106. De Domenico, M., Porter, M. A. & Arenas, A. MuxViz: A tool for multilayer analysis and visualization of networks. *J. Complex Networks* **3**, (2015).
107. Karatzas, E., Baltoumas, F. A., Panayiotou, N. A., Schneider, R. & Pavlopoulos, G. A. Arena3Dweb: Interactive 3D visualization of multilayered networks. *Nucleic Acids Res.* **49**, W36–W45 (2021).
108. Zafeiropoulos, H. *et al.* PREGO: A Literature and Data-Mining Resource to Associate Microorganisms, Biological Processes, and Environment Types. *Microorganisms* **10**, (2022).
109. Karatzas, E. *et al.* Darling: A Web Application for Detecting Disease-Related Biomedical Entity Associations with Literature Mining. *Biomolecules* **12**, (2022).
110. Thanati, F. *et al.* Flame: A web tool for functional and literature enrichment analysis of multiple gene lists. *Biology (Basel).* **10**, (2021).